SolidCAM Automation API

Public API Reference

Version 1.2

Table of Contents

Summary

SolidCAM Automation API enables third party developers to use some SolidCAM functionalities like opening a CAM part and changing it's reference model. A command line utility which uses this API is also available, enabling batch processing of SolidCAM parts without the need for developing: CAM parts can be manipulated through simple command line batch files.

There are two components, Automation and SolidCAM. SolidCAM is the main component and Automation is the proxy designed to simplify the initialization process. It is reccomended to use the Automation component.

The API is available to any language that can create and access COM objects, such as C/C++, Visual Basic 6 or JavaScript.

Some functionalities exposed by the SolidCAM Automation API:

- Start SolidCAM executable
- Start host CAD application (SolidWorks)
- Open a CAM part
- Save currently open CAM part
- Get information about currently open CAM part
- Change reference model of the currently open CAM part
- Synchronize currently open CAM part
- Calculate currently open CAM part
- Generate GCode for the currently open CAM part

Creating the automation object

DLL Registration

Before using the object it is necessary to register it. It is registered like any other ActiveX (or COM) DLL:

C:\Program Files\SolidCAM>regsvr32 scautom.dll

The registration is done automatically by the install wizard.

Project setup

C++

```
#import "scautom.dll" // full path
...
::CoInitialize(NULL);
...
SOLIDCAMLib::IAutomationPtr objSC;
HRESULT hr = objSC.CreateInstance(L"SolidCAM.Automation");
```

Visual Basic 6

Add “SolidCAM 1.0 Type Library” to project references and then create a new instance of the SolidCAM automation object:

```
Dim objSC As New SOLIDCAMLib.Automation
```

JavaScript

```
var objSC = new ActiveXObject("SolidCAM.Automation");
```

General

LastError

Type

Long

Description

Returns the error code from the last API function call.

Return

Last error code or 0 if no error occurred.

Example

C++

```
long errorCode = objSC->GetLastError();
```

VB6

```
Dim errorCode As Long
errorCode = objSC.LastError
```

JavaScript

```
var errorCode = objSC.LastError;
```

LastErrorDescription

Type

String

Description

Returns a string with the description of the last error that occurred in the last API call.

Example

C++

```
BSTR errorDescription = objSC->GetLastErrorDescription();
```

VB6

```
Dim errorDescription As String
errorDescription = objSC.LastErrorDescription
```

JavaScript

```
var errorDescription = objSC.LastErrorDescription;
```

LogFile

Type

String

Description

Gets and sets the path to the file where all API calls will be logged.

Example

C++

```
BSTR logFile = objSC->GetLogFile();
objSC->PutLogFile(_bstr_t("c:\\logs\\solidcam.txt"));
```

VB6

```
Dim logFile As String
logFile = objSC.LogFile
objSC.LogFile = “c:\logs\solidcam.txt”
```

JavaScript

```
var logFile = objSC.LogFile;
objSC.LogFile = “c:\\logs\\solidcam.txt”;
```

pid

Type

Double

Description

If this value is not set (or it is <=0) and several instances of SolidCAM are running, then the last instance will be used. Otherwise, if the value corresponds to a process id of a running instance of SolidCAM, commands will be set to that instance.
If SolidCAM has been started through the same instance of the automation object then this instance will be used throughout the life of the object.

Example

C++

```
objSC->Putpid(newPid);
double pid = objSC->Getpid();
```

VB6

```
objSC.pid = newPid
Dim pid As Double
pid = objSC.pid
```

JavaScript

```
objSC.pid = newPid;
var pid = objSC.pid;
```

IsSolidCamRunning()

Type

Boolean

Description

Checks if there is an active instance of SolidCAM Automation object.

Return

Returns True if there is an active instance of SolidCAM Automation object, otherwise returns 0.

StartApplication(path, iWaitForPluginOrSolidCAM)

Description

Starts a host CAD application or a standalone SolidCAM instance and waits for the automation API to become available. A default timeout of 60 seconds prevents the function from waiting indefinitely if the application does not exist or an error has occurred.

Return

Returns 1 if it succeeded, otherwise 0.

Arguments

path — Full path to the application which is to be started in another process. If the path is empty and the host plugin is loaded, then it will start SolidCAM.

iWaitForPluginOrSolidCAM — Integer, should be 0 if not starting SolidCAM

Example

C++

```
long result = objSC->StartApplication(_bstr_t("C:\\Program
                    Files\\SolidWorks\\SLDWORKS.exe"));
```

VB6

```
Dim result As Long
result = objSC.StartApplication ("C:\Program
                            Files\SolidWorks\SLDWORKS.exe")
```

JavaScript

StartSolidCAM()

Description

Starts SolidCAM, but only if the host CAD application is already running and the SolidCAM plugin has been loaded.

Return

Returns True if it succeeded

CAD

OpenHostFile(path)

Description

Opens a CAD file in the host (SolidWorks)

IsActiveDocCamPart

Type

Boolean, read-only

Description

Checks if the currently active document in the host CAD application is a CAM part.

Example

C++

```
BOOL bResult = objSC->GetIsActiveDocCamPart();
```

VB6

```
Dim lResult As Long
lResult = objSC.IsActiveDocCamPart
```

JavaScript

```
var bResult = objSC.IsActiveDocCamPart;
```

RenderPreview(path)

Description

Invokes the host CAD to generate a preview image (for use with PDM)

Parameters

`path` path where the resulting image will be stored

CAM

Open(path, model_path)

Description

Opens a SolidCAM part and, if the second parameter points to a valid SolidWorks part, changes the CAM part's reference model to the new one.

Return

Returns 1 if it succeeded, otherwise 0.

Arguments

`partPath`	Full path to the CAM part.
`modelPath`	Full part to the model or an empty string.

Example

C++

```
long result = objSC->Open(_bstr_t(partPath),
                          _bstr_t(modelPath));
```

VB6

```
Dim result As Long
result = objSC.Open(partPath, “”)
result = objSC.Open(partPath, modelPath)
```

JavaScript

```
objSC.Open(partPath, "");
objSC.Open(partPath, modelPath);
```

CheckSynchronization()

Description

Synchronize()

Description

Calculate(only_not_calculated)

Description

Calculates operations

Arguments

`only_not_calculated`	Boolean, whether to calculate all or only not calculated

`CalculateOperations(operations[], only_not_calculated)`

Description

`CalculateSingleOperation(number)`

Description

Calculates single operation

Arguments

`number`	Number of the operation to be calculated

`ChangePostProcessorDirectory(path)`

Description

Changes the post processor directory

`GenerateGCode()`

Description

Generates G-code

`Save(folder)`

Description

This method saves the currently open document to the specified folder. It receives one parameter which should be a fully qualified path to a folder in which the part is to be saved.

Return

If saving succeeds, the function returns the full path to the saved file, otherwise it returns an empty string.

Arguments

`folderPath`	Full path to the folder where the part will be saved.

Example

C++

```
BSTR out = objSC->Save(_bstr_t(folder));
```

VB6

```
Dim folder As String
Dim path As String
path = objSC.Save(folder)
```

JavaScript

```
var path = objSC.Save(folder);
```

SaveAs(path)

Description

Saves a part at a new location and with a new name.

Return

Returns 1 if it succeeded, otherwise 0.

Arguments

`partPath` Full path where the part will be saved.

SaveToFolder(folder)

Description

This method saves the currently open document to the specified folder. It receives one parameter which should be a fully qualified path to a folder in which the part is to be saved.

Return

If saving succeeds, the function returns the full path to the saved file, otherwise it returns an empty string.

Arguments

`folderPath` Full path to the folder where the part will be saved.

Example

C++

```
BSTR out = objSC->Save(_bstr_t(folder));
```

VB6

```
Dim folder As String
Dim path As String
path = objSC.Save(folder)
```

JavaScript

```
Var path = objSC.Save(folder);
```

Close()

Description

Closes the currently open SolidCAM part.

Example

C++

```
objSC->Close();
```

VB6

```
objSC.Close()
```

JavaScript

```
objSC.Close();
```

`Exit()`

Description

Closes SolidCAM.

Machine

MachineCount

Type

Long

Description

Returns the number of available machines.

Return

Integer representing the number of available machines.

GetMachineName(index)

Description

Returns the name of the machine.

Return

COM string containing the name of the machine with the given index.

Arguments

index Index of the machine.

CurrentMachine

Type

Long

Description

Sets and retrieves the index of the machine used for the current part.

CurrentMachineName

Type

String

Description

Retrieves the name of the machine used for the current part.

PART

Path

Type

String

Description

Returns the file path to the currently open part.

Return

COM string containing the path to the currently open part.

Example

C++

```
BSTR partPath = objSC->GetPath();
```

VB6

```
Dim partPath As String
partPath = objSC.Path
```

JavaScript

```
var partPath = objSC.Path;
```

Type

Type

Integer

Description

Returns the type of the currently open part.

Return

-2	SolidCAM not running
-1	No open documents
0	Not a CAM part
1	Milling
2	Turning
3	Turn-Mill
4	Wire Cut

Example

C++

```
short partType = objSC->GetType();
```

VB6

```
Dim partType As Integer
partType = objSC.Type
```

JavaScript

```
var partType = objSC.Type;
```

ReferenceModel

Type

String

Description

Returns a string with the path to the original reference model.

Example

C++

```
BSTR modelPath = objSC->GetReferenceModel();
```

VB6

```
Dim modelPath As String
modelPath = objSC.ReferenceModel
```

JavaScript

```
var modelPath = objSC.ReferenceModel;
```

ChangeReferenceModel(modelPath)

Description

Changes the reference model of the currently open CAM part.

Return

Returns 1 if it succeeded, otherwise 0.

Arguments

`modelPath`	Full path to the CAD part

CreateNewPart(name, path, type, machine_index, home_origin_position, inch)

Description

Creates a new part from the currently opened CAD model

Arguments

`name`	New part name
`path`	Location of the part
`part_type`	Type (0-milling, 1-turning, 2-wirecut, 3-millturn, 4-millturn_full)
`machine_index`	Index of the machine
`home_origin_position`	0-top corner, 1-top center, 2-cad origin for milling, 1 based index of the CAD coordinate system for turning
`inch`	Units

OPERATION

OperationCount

Type

Integer

Description

Returns the number of available operations.

GetOperationName(index)

Description

Returns the name of the operation with the given index.

Arguments

`index` Index of the operation.

GetOperationType(index)

Description

Returns the type of the operation.

Return

Type of the operation with the given index.

Arguments

`index` Index of the operation.

SuppressOperation(operation_name, suppress)

Description

Suppresses or unsuppresses an operation

Arguments

`operation_name`	Operation name
`suppress`	Boolean, suppress or unsurpress

NumberOfJobsOpenedWithExclamationSign

Type

Integer

`GenerateGCodeForOperation`

Description

Generates Gcode for a specified operation

Arguments

`operation_name`	Operation name
`file_name`	GCode output file name

TOOL

ToolCount

Type

Integer

Description

Returns the number of tools present in the current part tool table.

GetToolName(index)

Type

String

GetToolTag(number, position, station, turret)

Description

Returns the ID of the specified tool

Arguments

`number`	Tool number
`position`	Tool sub position
`station`	Position in turret (can be 0)
`turret`	Turret tag (can be 0)

GetToolType(index)

Return

0	TOOL_NONE
1	MILLING
2	TURNING
3	WIRE_CUT

SetOperationTool(operation_name, tool_tag)

Description

Sets the tool that is to be used in the specified operation

Arguments

`operation_name`	Operation name
`tool_tag`	Tool tag (from GetToolTag)

GetOperationToolTag(operation_name)

Description

Returns the ID of the tool used in the specified operation

Arguments

`operation_name` Operation name

GenerateToolSheet(template_name)

Description

Generates the tool sheet

Arguments

`template_name` Template name (Sheet_Full_HTML, Sheet_Full_RTF...)

ToolSheetCount

Description

Returns the number of templates

Type

Boolean

GetToolSheetName(index)

Description

Returns the template name

Type

String

Geometry

CADCoordSysCount

Description

Returns a number of coordinate systems defined in the CAD model.

Type

Integer

CreateTarget

Description

Creates target.

Returns

Boolean

Arguments

`name`	Name of the target
`iTargetDefineBy`	0-solid, 1-surface, 2-all
`b3DModelWithOutEnvelope`	
`bGenerateEnvelope`	
`bGenerateSection`	
`bGenerateMirrorEnvelope`	
`dFacetTolerance`	If left to 0 then it will be taken from settings

CreateStockBox

Description

Creates stock.

Returns

Boolean

Arguments

`name`	Name of the stock
`x_plus`	Box offset
`y_plus`	Box offset
`z_plus`	Box offset
`x_minus`	Box offset
`y_minus`	Box offset
`z_minus`	Box offset

`absolute`	
`iStockDefineBy`	0-solid, 1-surface, 2-all
`add_3d_sketch`	(bool)
`bGenerateStockEnvelope`	
`dFacetTolerance`	If left to 0 then it will be taken from settings

`CreateStockCylinder`

Description

Creates stock.

Returns

Boolean

Arguments

`name`	Name of the stock
`right`	Cylinder offset
`left`	Cylinder offset
`internal_diameter`	Cylinder offset
`external_diameter`	Cylinder offset
`absolute`	(bool)
`iStockDefineBy`	0-solid, 1-surface, 2-all
`add_3d_sketch`	(bool)
`bGenerateStockEnvelope`	
`dFacetTolerance`	If left to 0 then it will be taken from settings

`GetCADCoordSysName(index)`

Description

Returns the name of a coordinate system defined in the CAD model.

Returns

String

`CreateHomeByCAD(cad_home_name)`

Description

Creates a new coordinate system according to the one defined in the CAD model.

Returns

Boolean

Arguments

`cad_home_name`	Name of the coordinate system in the CAD

HomeCount

Description

Returns the number of homes

Type

Integer

GetHomeName(index)

Description

Returns the name of the home

Type

String

CreateHome(home_origin_position)

Description

Creates a home in the specified location

Arguments

`home_origin_position`	0-top corner, 1-top center, 2-cad origin

GeomCount

Description

Returns the number of geometries

Type

Integer

GetGeomName(index)

Description

Returns the geometry name

Returns

String

Templates

`TemplateCount`

Description

Returns the number of templates

Type

Boolean

`GetTemplateName(index)`

Description

Returns the template name

Type

String

`CreateJobFromTemplate(template_name, geometry_name, sub_machine_index, index_of_jobs_with_combinations)`

Description

Creates a new operation specified in the template

Arguments

`template_name`	Process template name
`geometry_name`	Geometry name (stock, target, ...)
`sub_machine_index`	Sub machine index, should be >=1

`ProcessTemplateCount`

Description

Returns the number of process templates

Type

Boolean

`GetProcessTemplateName(index)`

Description

Returns the name of the process template

Type

String

`CreateJobsFromProcessTemplate(process_template_name, geometry, sub_machine_index, index_of_jobs_with_combinations)`

Description

Creates a set of operations specified in the process template

Arguments

`process_template_name`	Process template name
`geometry_name`	Geometry name (stock, target, ...)
`sub_machine_index`	Sub machine index, should be >=1

Examples

Following examples do the same thing in three different programming languages: C/C++, Visual Basic 6 and JavaScript.

They all receive two command line parameters. First one, mandatory, is the path to the CAM part. Second one, optional, is the path to the reference model that will replace the model used in the CAM part.

C++

```
// Use the full path to the scautom.dll, it is located in the SolidCAM directory.
#import "D:\Program Files\SolidCAM\scautom.dll"

int main(int argc, char* argv[])
{
	HRESULT hr;
	BOOL bResult;
	SOLIDCAMLib::IAutomationPtr spSCAutomation;

	if(2 > argc) return 0;

	hr = ::CoInitialize(NULL);
	if( FAILED(hr) ) return 0;

	hr = spSCAutomation.CreateInstance(L"SolidCAM.Automation");
	if( FAILED(hr) ) return 0;

	_bstr_t bsLogPath("D:\\solidcam_automation.log");
	spSCAutomation->PutLogFile(bsLogPath);

	_bstr_t bsSolidWorks("C:\\Progra~1\\SolidWorks\\SLDWORKS.exe");
	bResult = spSCAutomation->StartApplication(bsSolidWorks);
	if( !bResult ) return 0;

	bResult = spSCAutomation->Open(_bstr_t(argv[1]), _bstr_t(""));
	if( !bResult ) return 0;

	if(argc >= 3)
	{
		bResult = spSCAutomation->ChangeReferenceModel(_bstr_t(argv[2]));
		if( bResult )
		{
			bResult = spSCAutomation->Synchronize();
			bResult = spSCAutomation->Calculate();
			bResult = spSCAutomation->GenerateGCode();
		}
	}
	spSCAutomation->Close();
	spSCAutomation.Release();
	::CoUninitialize();
	return 1;
}
```

Visual Basic 6

```
'Add "SolidCAM 1.0 Type Library" to project references

Private Sub Form_Load()
	Dim lResult As Long
	Dim strSolidWorks As String
	Dim solidCAM As New SOLIDCAMLib.Automation
	Dim commands() As String

	commands = Split(Command$, " ")

	If UBound(commands) = 0 Then Exit Sub

	solidCAM.LogFile = "D:\solidcam_automation.log"

	strSolidWorks = "C:\Program Files\SolidWorks\SLDWORKS.exe"
	lResult = solidCAM.StartApplication(strSolidWorks)
	If lResult = 0 Then Exit Sub

	lResult = solidCAM.Open(commands(0), "")
	If lResult = 0 Then Exit Sub

	If UBound(commands) >= 1 Then
		lResult = solidCAM.ChangeReferenceModel(commands(1))
		If lResult <> 0 Then
			lResult = solidCAM.Synchronize()
			lResult = solidCAM.Calculate()
			lResult = solidCAM.GenerateGCode()
		End If
	End If

	solidCAM.Close

	End
End Sub
```

JavaScript

```
// Command line: wscript.exe example.prt model.sldprt
var SolidCAM;
var SolidWorks = "C:\\Program Files\\SolidWorks\\SLDWORKS.exe";
var result;
var objArgs = WScript.Arguments;
if(objArgs.length > 0)
{
	SolidCAM = new ActiveXObject("SolidCAM.Automation");
	SolidCAM.LogFile = "D:\\solidcam_automation.log";
	result = SolidCAM.StartApplication(SolidWorks);
	if( result )
	{
		result = SolidCAM.Open(objArgs(0), "");
		if( result )
		{
			if(objArgs.length > 1)
			{
				result = SolidCAM.ChangeReferenceModel(objArgs(1));
				result = SolidCAM.Synchronize();
				result = SolidCAM.Calculate();
				result = SolidCAM.GenerateGCode();
			}
			result = SolidCAM.Close();
		}
	}
}
```

1. Operation types

-1	NC_JOB_NONE
0	NC_POCKET
1	NC_PROFILE
2	NC_CHAMFER
3	NC_DRILL_OLD
4	NC_SLOT
5	NC_RS
6	NC_TS
7	NC_SP
8	NC_VRS
9	NC_LFS
10	NC_CNS
11	NC_MSC
12	NC_THREAD
13	NC_DRILL
14	NC_DRILL_3D
15	NC_MSC_ENGRAV
16	NC_JOB_MW_5X
17	NC_JOB_MW_ELECTRODE_5X
18	NC_JOB_MW_ENGRAVING_5X
19	NC_JOB_MW_PORT_5X
20	NC_JOB_MW_SWARF_5X
21	NC_JOB_MW_TURBINE_5X
22	NC_JOB_MW_CONVERTED_5X
23	NC_JOB_MW_3X
24	NC_JOB_MW_4X
25	NC_JOB_HSM_CONST_STEP
26	NC_JOB_HSM_WATERLINE
27	NC_JOB_HSM_AREA_CLEARANCE
28	NC_JOB_HSM_RASTER_CLEARANCE
29	NC_JOB_HSM_HORIZ_AREA
30	NC_JOB_HSM_RASTER
31	NC_JOB_HSM_RADIAL
32	NC_JOB_HSM_SPIRAL
33	NC_JOB_HSM_MORPHED
34	NC_JOB_HSM_BOUNDARY

```
35  NC_JOB_HSM_PENCIL
36  NC_JOB_HSM_PAR_PENCIL
37  NC_JOB_HSM_REST_MACH
38  NC_JOB_HSM_REST_ROUGHING
39  NC_JOB_HSM_CORNER_OFFSET
40  NC_POCKET_3D
41  NC_PROFILE_3D
42  NC_TSLOT
43  NC_JOB_HSM_COMBINE_CONST_Z_HORIZONTAL
44  NC_JOB_HSM_COMBINE_CONST_Z_LINEAR
45  NC_JOB_HSM_COMBINE_CONST_Z_CONST_STEP
46  NC_JOB_MW_SWARF_4X
47  NC_JOB_MW_CONVERTED_3X
48  NC_JOB_MW_CONVERTED_4X
49  NC_JOB_MW_IMPELER_ROUGH_5X
50  NC_JOB_MW_IMPELLER_FLOOR_TC_5X
51  NC_JOB_MW_IMPELLER_FLOOR_NO_TC_5X
52  NC_JOB_MW_IMPELLER_BLADE_5X
53  NC_JOB_MW_DRILL_5X
54  NC_JOB_MW_CAVITY_WITH_GOUGE_5X
55  NC_FACE_MILLING
56  NC_JOB_HSM_HELICAL
57  NC_JOB_HSM_OFFSET_CUTTING
58  NC_JOB_HSM_MIXED_CONST_Z_CONST_STEP
59  NC_DRILL_HR
60  NC_SPIRAL_POCKET
61  NC_SPIRAL_MSC
62  NC_JOB_MW_DRILL_4X
63  NC_FIRST_TURN_JOB
63  NC_TURN_TURN
64  NC_TURN_DRILL
65  NC_TURN_THREAD
66  NC_TURN_GROOVE
67  NC_TURN_GROOVE_CUT
68  NC_JOB_MW_DEBURRING
69  NC_JOB_HSS_CONSTANT_Z
70  NC_JOB_HSS_LINEAR
71  NC_JOB_HSS_PERP_TOCURVE
72  NC_JOB_HSS_MORPH_BETWEEN_CURVES
```

```
73  NC_JOB_HSS_PARALLEL_TO_CURVE
74  NC_JOB_HSS_MORPH_BETWEEN_SURF
75  NC_JOB_HSS_PROJECTION
76  NC_JOB_HSS_PARALLEL_TO_SURF
77  NC_JOB_HSS_HATCH
78  NC_JOB_MW_CONSTANT_Z_4X
79  NC_JOB_MW_LINEAR_4X
80  NC_JOB_MW_PERP_TOCURVE_4X
81  NC_JOB_MW_MORPH_BETWEEN_CURVES_4X
82  NC_JOB_MW_PARALLEL_TO_CURVE_4X
83  NC_JOB_MW_MORPH_BETWEEN_SURF_4X
84  NC_JOB_MW_PROJECTION_4X
85  NC_JOB_MW_PARALLEL_TO_SURF_4X
86  NC_JOB_MW_HATCH_4X
87  NC_JOB_MW_CONSTANT_Z_5X
88  NC_JOB_MW_LINEAR_5X
89  NC_JOB_MW_PERP_TOCURVE_5X
90  NC_JOB_MW_MORPH_BETWEEN_CURVES_5X
91  NC_JOB_MW_PARALLEL_TO_CURVE_5X
92  NC_JOB_MW_MORPH_BETWEEN_SURF_5X
93  NC_JOB_MW_PROJECTION_5X
94  NC_JOB_MW_PARALLEL_TO_SURF_5X
95  NC_JOB_MW_HATCH_5X
96  NC_JOB_MW_NAVIROBO_DEBURRING
97  NC_JOB_MW_PARALLEL_CUTS_4X
98  NC_JOB_MW_PARALLEL_CUTS_5X
99  NC_JOB_HSM_COMBINE_CONST_Z_CORNER_OFFSET
100 NC_TOOL_BOX
124 NC_BACK_SPINDLE_OPERATION
125 NC_FIXTURE
126 NC_G_SPLIT
127 NC_EXTERN_FILE
```

Wire-cut

```
1   WC_PROFILE
2   WC_4X
3   WC_ANGLE
4   WC_POS_JOB
```