PROSPECTOR™

The Gold Standard In Toolmaking

Prospector Object Model

Conventions

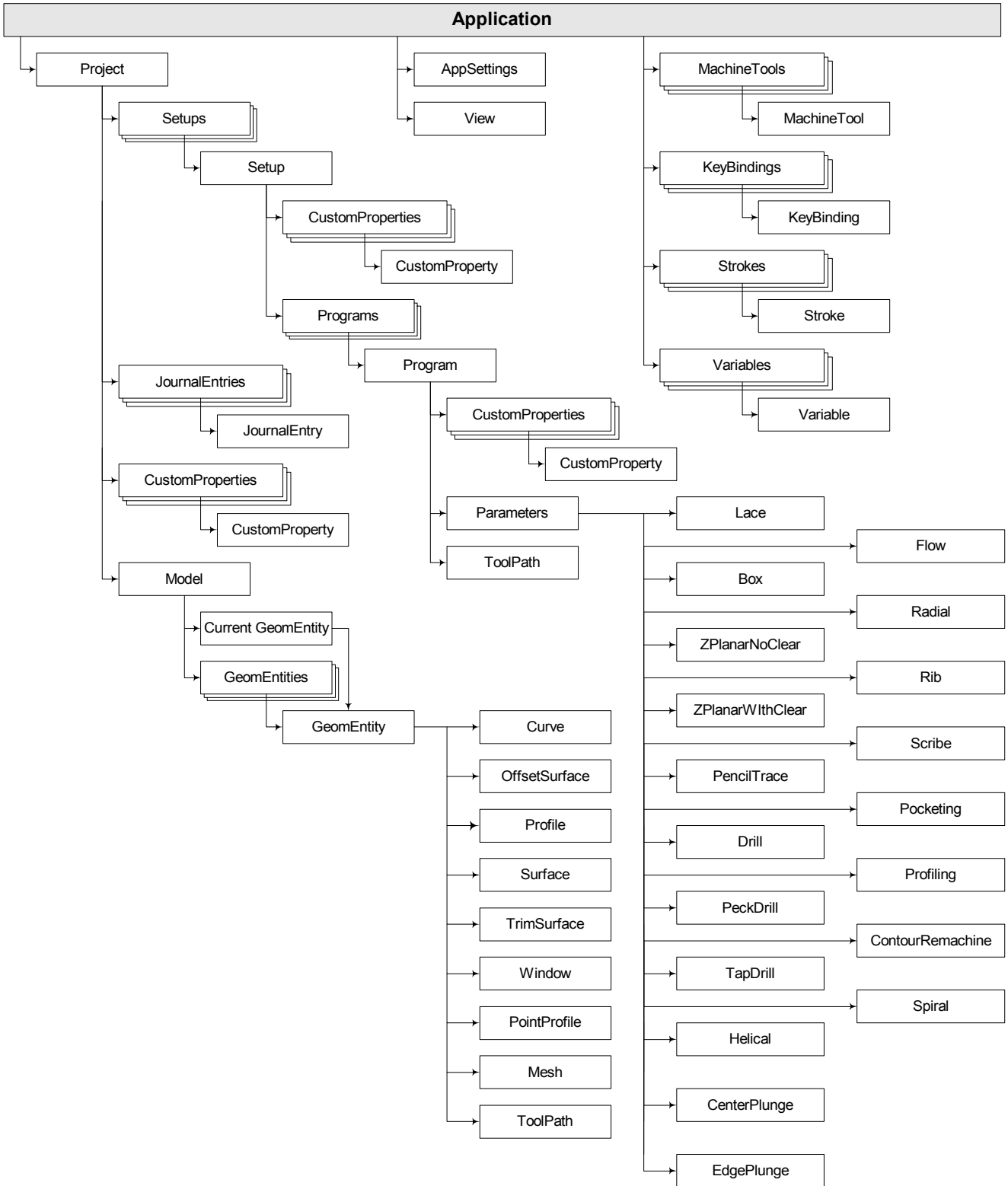
Bold text Denotes a term or character to be typed literally, such as function name.

Italic text Denotes a placeholder or variable. .

[] Encloses optional parameters.

* Denotes properties and methods that already exist in Prospector type library.

Overview


Application
Project
AppSettings
View
MachineTools
MachineTool
Setups
Setup
KeyBindings
KeyBinding
CustomProperties
CustomProperty
Strokes
Stroke
Programs
Program
Variables
Variable
JournalEntries
JournalEntry
CustomProperties
CustomProperty
CustomProperties
CustomProperty
Parameters
ToolPath
Model
Current GeomEntity
GeomEntities
GeomEntity
Curve
OffsetSurface
Profile
Surface
TrimSurface
Window
PointProfile
Mesh
ToolPath
Lace
Flow
Box
Radial
ZPlanarNoClear
Rib
ZPlanarWIthClear
Scribe
PencilTrace
Pocketing
Drill
Profiling
PeckDrill
ContourRemachine
TapDrill
Spiral
Helical
CenterPlunge
EdgePlunge

Description

Application Object

Application object represents an instance of the Prospector application. It is located at the top of the object hierarchy and is the base object through which all other objects are accessed. It is also the only object that can be retrieved directly by applications. In Visual Basic, this means that the Application object is the only object that the **CreateObject** function returns.

The **Application** object should have the following properties:

Property name	Return type	Description	Notes
ActiveProject*	VT_DISPATCH, VT_EMPTY	Returns the active **Project** object or VT_EMPTY if none; read-only.	
ActiveView	VT_DISPATCH	Returns the **View** object; read-only.	
Application*	VT_DISPATCH	Returns the **Application** object; read -only.	
AppSettings*	VT_DISPATCH	Returns the **AppSettings** object; read-only.	
Constants	VT_DISPATCH	Returns the **Constants** object that contains all of the named constants available in the Prospector type library; read-only. Needed for VBScript support.	
FullName*	VT_BSTR	Returns the file specification for the application, including path; read-only. For example, C:\Prospector\bin\Prospect.	
Height	VT_I4	Sets or returns the distance between the top and bottom edge of the main application window; read/write.	
KeyBindings	VT_DISPATCH	Returns the **KeyBindings** collection object; read-ony.	
Left	VT_I4	Sets or returns the distance between the left edge of the physical screen and the main application window; read/write.	
MachineTools	VT_DISPATCH	Returns the **MachineTools** collection object; read-only.	
Name*	VT_BSTR	Returns the name of the application ("Prospector"); read-only. The Name property is the default member (DISPID_VALUE) for the Application object.	
Parent*	VT_DISPATCH	Returns the **Application** object; read -only.	
Path	VT_BSTR	Returns the path specification for the application's executable file; read-only. For example, C:\Prospector\bin if the .exe file is C:\Prospector\bin\Prospect.exe.	
ShowProgramList	VT_BOOL	Shows or hides program list; read/write.	
ShowStatusBar	VT_BOOL	Shows or hides status bar; read/write.	
EnableWarnings	VT_BOOL	Returns or sets a value indicating whether any message or dialog boxes can be shown.; read/write.	
StatusBar	VT_BSTR	Sets or returns the text displayed in th e status bar; read/write.	
Strokes	VT_DISPATCH	Returns the **Strokes** collection object; read-only.	
Top	VT_I4	Sets or returns the distance between the top edge of the physical screen and main application window; read/write.	
UserName	VT_BSTR	Returns the user name; read-only.	
Variables	VT_DISPATCH	Returns the collection of **Variable** objects that represent the variables added to the application. Used to preserve macro settings in between Prospector sessions; read-only.	
Visible*	VT_BOOL	Sets or returns whether the application is visible to the user; read/write. The default is False when the application is started with the /Automation command-line switch; read/write.	
Width	VT_I4	Sets or returns the distance between the left and right edges of the main application window; read/write.	
WindowState	VT_I4	Returns or sets the state of the main application window. Can be one of the following prWindowState constants: prWindowStateNormal, prWindowStateMaximize, or prWindowStateMinimize; read/write.	

The **Application** object should have the following methods:

Method name	Return type	Description	Notes
CloseActiveProject*	VT_EMPTY	Closes active project.	
EnumProjectNames*	VT_ARRAY(BSTR)*	Returns a list of all project names, local and remote. Useful when writing a macro that needs to perform an operation on all projects.	
ExecuteCommand *CommandString*	VT_EMPTY	Executes a specified command or VBScript macro. ExecuteCommand method is used to run commands or macros listed on the Keyboard tab of the Customize dialog box.	
OpenRemoteProject *JobNumber, ProjectName*	VT_DISPATCH, VT_EMPTY	Opens remote project. Returns the active **Project** object or VT_EMPTY if none.	
Help [*HelpFile, HelpContextID, HelpString*]	VT_EMPTY	Displays online Help. May take three optional arguments: helpfile (VT_BSTR), helpcontextID (VT_I4), and helpstring (VT_BSTR). The helpfile argument specifies the Help file to display; if omitted, the main Help file for the application is displayed. The helpcontextID and helpstring arguments specify a Help context to display; only one of them can be supplied. If both are omitted, the default Help topic is displayed.	
InfoBox *MessageString*	VT_EMPTY	Displays a message in a scrollable window.	Hidden; for our internal/debugging use only.
LoadMacros *FileName*	VT_EMPTY	Adds macros from the specified file to the Prospector command list.	
OpenProject *JobNumber, ProjectName*	VT_DISPATCH, VT_EMPTY	Opens project. Returns the active **Project** object or VT_EMPTY if none.	
OpenProjectEx* *ProjectName*	VT_DISPATCH, VT_EMPTY	Opens project. Returns the active **Project** object or VT_EMPTY if none.	
PutActiveProject	VT_EMPTY	Closes open project and puts its compressed copy into the Remote Job location.	
Quit*	VT_EMPTY	Closes open project and exits the application.	
Redo	VT_EMPTY	Redoes the previously undone action in the user interface.	
UndefineMacro *MacroName*	VT_EMPTY	Removes previously defined macro from the Prospector command list.	
Undo	VT_EMPTY	Reverses the previous action in the user interface.	

The **Application** object should have the following events:

Event name	Description
BeforeExit	Occurs when the application starts to shut down. Returning **False** will terminate the shutdown.
Exit	Occurs when the application is about to exit.
Startup	Occurs when the application is starting up
ActiveSetupChanged	Occurs after active setup is changed.
BeforeBuildStart	Occurs before build starts. Returning **False** will abort the build.
BeforeProgramDelete	Occurs before a program is deleted. Returning **False** will prevent program from being deleted.
BeforeProgramEdit	Occurs before a program is edited. Returning **False** will prevent program from being edited.
BeforeProjectClose	Occurs before active project closes. Returning **False** will terminate the project closing.

Event name	Description
BeforeProjectEdit	Occurs before a project is edited. Returning **False** will prevent project from being edited.
BeforeProjectOpen	Occurs before a project is opened.
BeforeSetupDelete	Occurs before a setup is deleted. Returning **False** will prevent setup from being deleted.
BeforeSetupEdit	Occurs before a setup is edited. Returning **False** will prevent setup from being edited.
BuildFinish	Occurs after a build completes.
ProgramAdded	Occurs after a program is added.
ProgramChanged	Occurs after a program is changed.
ProjectChanged	Occurs after a project is changed.
ProjectCreated	Occurs after new project is created.
ProjectOpened	Occurs after a project is open.
SetupAdded	Occurs after a setup is added.
SetupChanged	Occurs after a setup is changed.

AppSettings Object

Represents application and project options in Prospector. Many of the properties for the AppSettings object correspond to items in the **Options** dialog box (**Tools** menu).

The **AppSettings** object should have the following properties:

Property name	Return type	Description	Notes
ArcTolerance*	VT_R8	Returns the arc tolerance; read/write.	
BackColor	VT_I4	Returns or sets the background color; Can be one of the following prColor constants: prColorBlack or prColorWhite; read/write.	
CacheDirectory*	VT_BSTR	Returns or sets the directory name that serves as a root of the temporary directories created by Prospector when needed; read/write.	
ComplexWindowsProximity	VT_R8	Returns or sets the complex windows proximity; read/write.	Hidden; for our internal/debugging use only
CurrentObjectColor	VT_I4	Returns or sets the current object color; Can be one of the following prColor constants: prColorDarkBlue, prColorGreen, prColorRed, prColorMagenta, prColorBrown, prColorLightBlue, prColorOrange, prColorLightOrange, prColorYellow, prColorCyan or prColorLime; read/write.	
EnableAcceleratedGraphics	VT_BOOL	Enables or disables accelerated graphics; read/write.	
EnableAdvancedCutWizard	VT_BOOL	Enables or disables advanced Cutting Wizard mode; read/write.	
EnableAdvancedCutWizardLastPage	VT_BOOL	Enables or disables advanced mode on the last page of Cutting Wizard; read/write.	
EnableBuildTiming	VT_BOOL	Enables or disables timed builds; read/write.	Hidden; for our internal/debugging use only
EnableComplexWindows	VT_BOOL	Enables or disables creation of complex windows; read/write.	
EnablePerSurfaceShading	VT_BOOL	Turns per surface shading on or off; read/write.	
GougeDetectionValue*	VT_BSTR	Sets or returns the amount by which a program can violate the surface model before being considered a gouge; read/write.	
GridSpacingXInch	VT_R8	Returns and sets the grid spacing in X in inch; read/write.	
GridSpacingXMm	VT_R8	Returns and sets the grid spacing in X in metric; read/write.	
GridSpacingYInch	VT_R8	Returns and sets the grid spacing in Y in inch; read/write.	
GridSpacingYMm	VT_R8	Returns and sets the grid spacing in Y in metric; read/write.	
GridSpacingZInch	VT_R8	Returns and sets the grid spacing in Z in inch; read/write.	
GridSpacingZMm	VT_R8	Returns and sets the grid spacing in Z in metric; read/write.	
InstallPath*	VT_BSTR	Returns the install path; read-only.	
InternalRelease	VT_BOOL	Returns the value determining whether the current release is internal; read-only.	Hidden; for our internal/debugging use only

Property name	Return type	Description	Notes
LocalJobPath*	VT_BSTR	Sets or returns the directory name for all the local projects; read/write.	
PowerSourceDatabaseName*	VT_BSTR	Sets or returns the name of the User PowerSource database; read/write.	
ProgramPrefix	VT_BSTR	Sets or returns the default base name (first 4 characters) to use for the name of newly created programs; read/write.	
RemoteJobPath*	VT_BSTR	Sets or returns the directory name for all the remote projects; read/write.	
RSMPixelsPerInch	VT_I4	Sets or returns the value determining the relationship of the RSM image to the physical size of the surface model; read/write.	
SelectedObjectColor	VT_I4	Returns or sets the selected object color; Can be one of the following prColor constants: prColorDarkBlue, prColorGreen, prColorRed, prColorMagenta, prColorBrown, prColorLightBlue, prColorOrange, prColorLightOrange, prColorYellow, prColorCyan or prColorLime; read/write.	
ShadedModelDisplayTolerance	VT_R8	Returns and sets shaded model display tolerance; read/write.	
SurfaceShadeColor	VT_I4	Returns or sets the surface shade color; Can be one of the following prColor constants: prColorDarkBlue, prColorGreen, prColorRed, prColorMagenta, prColorBrown, prColorLightBlue, prColorOrange, prColorLightOrange, prColorYellow, prColorCyan or prColorLime; read/write.	
ToolPathFeedColor	VT_I4	Returns or sets the tool path feed color; Can be one of the following prColor constants: prColorDarkBlue, prColorGreen, prColorRed, prColorMagenta, prColorBrown, prColorLightBlue, prColorOrange, prColorLightOrange, prColorYellow, prColorCyan or prColorLime; read/write.	
ToolPathRapidColor	VT_I4	Returns or sets the tool path rapid color; Can be one of the following prColor constants: prColorDarkBlue, prColorGreen, prColorRed, prColorMagenta, prColorBrown, prColorLightBlue, prColorOrange, prColorLightOrange, prColorYellow, prColorCyan or prColorLime; read/write.	

Variables Collection Object

A collection of **Variable** objects that represent the variables added to the application. Variables are used to preserve macro settings in between macro sessions.

The **Variables** object should have the following properties:

Property name	Return type	Description	Notes
Count	VT_I4	Returns the number of items in the specified collection; read-only.	

The **Variables** object should have the following methods:

Method name	Return type	Description	Notes
Add *Name, Value*	VT_EMPTY	Adds a variable to the application. *Name* is a string representing the name of the variable. *Value* is a variant representing the value of the variable.	
Item *Index/Name*	VT_DISPATCH	Returns a member of a collection by index or name. The **Item** method is the default method for collections.	

Variable Object

Represents a variable stored as part of the application. Variables are used to preserve macro settings in between macro sessions.

The **Variable** object should have the following properties:

Property name	Return type	Description	Notes
Name	VT_BSTR	Returns the variable name; read-only.	
Value	VT_VARIANT	Returns or sets the value of the variable; read/write.	
Index	VT_I4	Returns a number that indicates the position of an item in a collection; read-only.	

The **Variable** object should have the following methods:

Method name	Return type	Description	Notes
Delete	VT_EMPTY	Deletes the specified object.	

MachineTools Collection Object

MachineTools is a collection of **MachineTool** objects that defines the machine tools to be supported.

The **MachineTools** object should have the following properties:

Property name	Return type	Description	Notes
Count	VT_I4	Returns the number of items in the specified collection; read-only.	

The **MachineTools** object should have the following methods:

Method name	Return type	Description	Notes
Add *MachineName, PostProcessorName, OutputPath, OutputFileExtension, PostType,* [*PostProcessorArguments*]	VT_DISPATCH, VT_EMPTY	Adds a new **MachineTool** object. *MachineName* is a string used to describe the name of the machine tool. *PostProcessorName* is a string specifying the name of the post-processor to be used for this machine tool. *OutputPath* is a string specifying the name of the directory where post-processed files are to be written. *OutputFileExtension* is a string specifying the extension to be used when naming output files for the post-processor. *PortType* can be one of the following prPostProcessorType constants: prPostProcessorCustom or prPostProcessorSoftech. *PostProcessorArguments* is an optional string parameter specifying any optional arguments to be supplied to the post-processor when invoked. Returns a new MachineTool object or VT_EMPTY if none was created.	
Item *Index/Name*	VT_DISPATCH	Returns a member of a collection by index or by name. The **Item** method is the default method for collections.	
RemoveAll	VT_EMPTY	Removes all **MachineTool** objects from the specified collection.	

MachineTool Object

Represents a machine tool to be supported by Prospector.

The **MachineTool** object should have the following properties:

Property name	Return type	Description	Notes
MachineName	VT_BSTR	Returns or sets the machine name; read/write.	
OutputFileExtension	VT_BSTR	Returns or sets the extension of post-processed data files; read/write. For example, "fan".	
OutputPath	VT_BSTR	Returns or sets the name of a directory to put post-processed data files into; read/write.	
PostProcessorArguments	VT_BSTR	Returns or sets optional arguments to supply to the post-processor; read/write.	
PostProcessorName	VT_BSTR	Returns or sets the post-processor name; read/write.	
PostType	VT_I4	Returns or sets the post type. Can be one of the following prPostProcessorType constants: prPostProcessorSoftech or prPostProcessorCustom; read/write.	

The **MachineTool** object should have the following methods:

Method name	Return type	Description	Notes
Delete	VT_EMPTY	Removes the machine tool from the MachineTools collection.	

KeyBindings Collection Object

KeyBindings is a collection of **KeyBinding** objects that represent the custom key assignments.

The **KeyBindings** object should have the following properties:

Property name	Return type	Description	Notes
Count	VT_I4	Returns the number of items in the specified collection; read-only.	

The **KeyBindings** object should have the following methods:

Method name	Return type	Description	Notes
Add *KeyString, CommandString*	VT_DISPATCH, VT_EMPTY	Adds a new **KeyBinding** object. *KeyString* is a string specifying the keystrokes needed to execute command. This string has the format CTRL+ALT+SHIFT+A, where CTRL, ALT, and SHIFT always appear in this order. If a two-key sequence represents a command, the two keys are separated by a comma, as in CTRL+HOME, S or X, SHIFT+Y. *CommandString* is a string specifying the command that key executes. Returns new KeyBinding object or VT_EMPTY if none was created.	
ClearAll	VT_EMPTY	Clears all the customized key assignments and restores the original Prospector shortcut key assignments.	
Item *Index*	VT_DISPATCH	Returns a member of a collection by index. The **Item** method is the default method for collections.	
Key *KeyString*	VT_DISPATCH, VT_EMPTY	Returns a **KeyBinding** object that represents the specified custom key combination or VT_EMPTY if none was found.	

KeyBinding Object

Represents a custom key assignment. The **KeyBinding** object is a member of the **KeyBindings** collection. Custom key assignments are made in the **Keyboard Tab** of the **Customize** dialog box.

The **KeyBinding** object should have the following properties:

Property name	Return type	Description	Notes
Command	VT_BSTR	Returns the command assigned to the specified key combination; read-only.	
Enabled	VT_BOOL	Enables or disables the specified key combination; read-write.	
KeyString	VT_BSTR	Returns the key combination string for the specified keys (for example, CTRL+SHIFT+A); read-only.	
Protected	VT_BOOL	**True** if you cannot change the specified key binding in the **Keyboard Tab** of the **Customize** dialog box (**Tools** menu); read-only.	

The **KeyBinding** object should have the following methods:

Method name	Return type	Description	Notes
Clear	VT_EMPTY	Removes the key binding from the KeyBindings collection and resets a built-in command to its default key assignment.	
Execute	VT_EMPTY	Runs the command associated with the specified key combination.	

Strokes Collection Object

Strokes is a collection of **Stroke** objects that represent the custom mouse stroke assignments.

The **Strokes** object should have the following properties:

Property name	Return type	Description	Notes
Count	VT_I4	Returns the number of items in the specified collection; read-only.	

The **Strokes** object should have the following methods:

Method name	Return type	Description	Notes
Add *StrokeCode,* [*DescriptionString,*] *CommandString,*	VT_DISPATCH, VT_EMPTY	Adds a new **Stroke** object. *StrokeCode* is a numeric representation of the stroke; It is used for pattern recognition. *DescriptionString* is an optional string parameter specifying the stroke description. *CommandString* is a string specifying the command that stroke executes. Returns new Stroke object or VT_EMPTY if none was created.	
ClearAll	VT_EMPTY	Clears all the customized stroke assignments and restores the original Prospector stroke assignments.	
Item *Index*	VT_DISPATCH	Returns a member of a collection by index. The **Item** method is the default method for collections.	
Stroke *StrokeCode*	VT_DISPATCH, VT_EMPTY	Returns a **Stroke** object that corresponds to the specified stroke code or VT_EMPTY if none was found.	

Stroke Object

Represents a custom mouse stroke assignment. The **Stroke** object is a member of the **Strokes** collection. Custom stroke assignments are made in the **Strokes Tab** of the **Customize** dialog box.

The **Stroke** object should have the following properties:

Property name	Return type	Description	Notes
Code	VT_I4	Returns a numeric representation of a stroke that is used for pattern recognition; read-only.	
Command	VT_BSTR	Returns the command assigned to the specified stroke; read-only.	
Description	VT_BSTR	Returns a description of the specified stroke; read-only.	
Enabled	VT_BOOL	Enables or disables the specified stroke; read-write.	
Protected	VT_BOOL	**True** if you cannot change the specified stroke in the **Strokes Tab** of the **Customize** dialog box (**Tools** menu); read-only.	

The **Stroke** object should have the following methods:

Method name	Return type	Description	Notes
Clear	VT_EMPTY	Removes the stroke from the Strokes collection and resets a built-in command to its default stroke assignment.	
Execute	VT_EMPTY	Runs the command associated with the specified stroke.	

Project Object

Represents a Prospector project - an electronic model and all the associated data that describes how to machine that model.

The **Project** object should have the following properties:

Property name	Return type	Description	Notes
ActiveSetup	VT_DISPATCH	Returns the active **Setup** object or VT_EMPTY if none; read –only.	
AllJournalEntries*	VT_DISPATCH	Returns the **JournalEntries** collection object; read-only.	
BlockDimensions*	VT_DISPATCH, VT_EMPTY	Returns the **Point3d** object representing dimensions of the block or VT_EMPTY if none; read-only.	
Building	VT_BOOL	Returns **True** if project is currently being build; read-only.	
CreatedBy*	VT_BSTR	Returns the name of the project creator; read-only.	
CreatedOn*	VT_DATE	Returns the date and time of project creation; read-only.	
CustomProperties*	VT_DISPATCH	Returns the **CustomProperties** collection object; read-only.	
Name*	VT_BSTR	Returns the project name; read-only.	
JobNumber*	VT_BSTR	Returns the job number; read-only.	
Material*	VT_BSTR	Returns or sets the material name; read/write.	
Model	VT_DISPATCH	Returns the **Model** object; read-only.	
ContainsMultiCavities*	VT_BOOL	Returns or sets the value determining whether the project contains multi-cavities; read/write.	
ObjectID*	VT_BSTR	Returns the project ID; read-only.	
Path*	VT_BSTR	Returns the project path; read-only.	
ProgramPrefix*	VT_BSTR	Sets or returns the default base name (first 4 characters) to use for the name of newly created programs; read/write.	
ProjectType*	VT_I4	Returns the units of the project. Can be one of the following EProjectType constants: pr2D or pr3D; read-only.	
ProjectTypeString*	VT_BSTR	Returns the string representation of the project's type; read-only.	
PropertyEditedBy*	VT_BSTR	Returns the name of the last user who edited the project; read-only.	
PropertyEditedOn*	VT_DATE	Returns the date that the project was last modified; read-only.	
Setups*	VT_DISPATCH	Returns the **Setups** collection object; read-only.	
SurfaceFileName*	VT_BSTR	Sets or returns the surface file name; read/write.	
TopOfBlock*	VT_R8	Returns the top of block value; read-only.	
Units*	VT_I4	Returns the units of the project. Can be one of the following EUnits constants: unitEnglish or unitMetric; read-only.	
UnitsString*	VT_BSTR	Returns the string representation of the project's units; read-only.	

The **Project** object should have the following methods:

Method name	Return type	Description	Notes
AddToModel *SurfaceFileName, ReBuildPrograms*	VT_EMPTY	Imports additional part data into a project. *SurfaceFileName* is the name of the data file to be added to the model; *ReBuildPrograms* is a flag that specifies whether to regenerate all existing programs.	
Build	VT_EMPTY	Performs project build.	
CreateRSMBitmap *FileName*	VT_EMPTY	Generates bitmap file with RSM information.	Hidden; for our internal/debugging use only
Info	VT_BSTR	Returns project info string.	Hidden; for our internal/debugging use only

Method name	Return type	Description	Notes
LoadMesh *FileName*	VT_BOOL	Loads a mesh file. Returns **True** if successful, **False** otherwise.	Hidden; for our internal/debugging use only
LoadPDB *FileName, Flags*	VT_EMPTY	Loads a PDB file. Flags can be a combination of the following prLoadPDBFlags constants: prLoadPDBDisableProgressBar, prLoadPDBMerge and prLoadPDBNoUpdate. Returns **True** if successful, **False** otherwise.	Hidden; for our internal/debugging use only
Print *Style*	VT_EMPTY	Prints the entire project. *Style* can be one of the following prPrintStyle constants: prProjectTypeStyle or prProgramTypeStyle;	
RestoreModel	VT_EMPTY	Rolls back the changes made by AddToModel function.	
SetRebuildAllPrograms	VT_EMPTY	Turns *all* programs to a 'gray-ball' state. Next time a 'Build' is performed the RSM surface data will be regenerated as well as a complete regeneration of each program.	Hidden; for our internal/debugging use only
SetRebuildSelectedPrograms	VT_EMPTY	Turns *all selected* programs to a 'gray-ball' state. Next time a 'Build' is performed the RSM surface data will be regenerated as well as a complete regeneration of each selected program.	Hidden; for our internal/debugging use only
SetRecutAllPrograms	VT_EMPTY	Turns on NeedsCut flag for *all* programs.	Hidden; for our internal/debugging use only
SetRecutSelectedPrograms	VT_EMPTY	Turns on NeedsCut flag for *all selected* programs.	Hidden; for our internal/debugging use only
SetReStitchAllPrograms	VT_EMPTY	Turns on NeedsStitch flag for *all* programs.	Hidden; for our internal/debugging use only
SetReStitchSelectedPrograms	VT_EMPTY	Turns on NeedsStitch flag for *all selected* programs.	Hidden; for our internal/debugging use only
SetReRSMAllPrograms	VT_EMPTY	Turns on NeedsRSM flag for *all* programs.	Hidden; for our internal/debugging use only
SetReRSMSelectedPrograms	VT_EMPTY	Turns on NeedsRSM flag for *all selected* programs.	Hidden; for our internal/debugging use only
StopBuild	VT_EMPTY	Stops build process.	

JournalEntries Collection Object

JournalEntries is a collection of JournalEntry objects that store editing-related events.

The **JournalEntries** object has the following properties:

Property name	Return type	Description	Notes
Count*	VT_I4	Returns the number of items in the specified collection; read-only.	

The **JournalEntries** object should have the following methods:

Method name	Return type	Description	Notes
Item *Index/Name*	VT_DISPATCH	Returns a member of a collection by index or name. The **Item** method is the default method for collections.	
RemoveAll*	VT_EMPTY	Removes all **JournalEntry** objects from the specified collection.	

JournalEntry Object

Represents a record of editing-related event.

The **JournalEntry** object has the following properties:

Property name	Return type	Description	Notes
ComputerName*	VT_BSTR	Returns the computer name; read-only.	
DateTime*	VT_DATE	Returns the creation time for the journal entry; read-only.	
EventType*	VT_I4	Returns the type of event that triggered creation of this journal entry. Can be one of the following EEventType constants: prCreate, prPropertyEdit, prToolPathEdit, prDelete or prSendToControl; read-only.	
EventTypeString*	VT_BSTR	Returns the type of event that triggered creation of this journal entry as a string; read-only.	
ExtraEventInfo*	VT_BSTR	Returns the extra event info string; read-only.	
ObjectID*	VT_BSTR	Returns a unique identificator of the object that triggered creation of this journal entry; read-only.	
ObjectType*	VT_I4	Returns the type of the object that triggered creation of this journal entry. Can be one of the following EPsObjectType constants: psProject, psSetup or psProgram; read-only.	
ObjectTypeString*	VT_BSTR	Returns the type of the object that triggered creation of this journal entry as a string; read-only.	
UserName*	VT_BSTR	Returns the user name; read-only.	

The **JournalEntry** object should have the following methods:

Method name	Return type	Description	Notes
Delete	VT_EMPTY	Removes the journal entry from the **JournalEntries** collection.	

CustomProperties Collection Object

A collection of **CustomProperty** objects that represent properties added by the user to the project. Custom properties are used to preserve user-defined properties in between Prospector sessions.

The **CustomProperties** object has the following properties:

Property name	Return type	Description	Notes
Count*	VT_I4	Returns the number of items in the specified collection; read-only.	

The **CustomProperties** object should have the following methods:

Method name	Return type	Description	Notes
Item *Index/Name*	VT_DISPATCH	Returns a member of a collection by index or name. The **Item** method is the default method for collections.	
RemoveAll*	VT_EMPTY	Removes all **CustomProperty** objects from the specified collection.	

CustomProperty Object

Represents a user-defined property stored as part of the project. Custom properties are used to preserve user-defined properties in between Prospector sessions. The **CustomProperty** object is a member of the **CustomProperties** collection.

The **CustomProperty** object has the following properties:

Property name	Return type	Description	Notes
Name*	VT_BSTR	Returns the name of a custom property; read-only.	
Type*	VT_I4	Returns the type of a custom property. Can be one of the following EPsCustomPropertyType constants: psText, psDate, psNumber, psList, psYesNo or ps3dPoint; read-only.	
Value*	VT_VARIANT	Returns or sets the value of the custom property; read/write.	

The **CustomProperty** object should have the following methods:

Method name	Return type	Description	Notes
Delete	VT_EMPTY	Removes the custom property from the **CustomProperties** collection.	

Point3d Object

Represents a point in a 3d space.

The **Point3d** object has the following properties:

Property name	Return type	Description	Notes
X*	VT_R8	Returns or sets the X coordinate of a 3d point; read/write.	
Y*	VT_R8	Returns or sets the Y coordinate of a 3d point; read/write.	
X*	VT_R8	Returns or sets the Z coordinate of a 3d point; read/write.	

Setups Collection Object

Setups is a collection of Setup objects that represent a position of the part on the machine..

The **Setups** object should have the following properties:

Property name	Return type	Description	Notes
Count	VT_I4	Returns the number of items in the specified collection; read-only.	

The **Setups** object should have the following methods:

Method name	Return type	Description	Notes
Item *Index/Name*	VT_DISPATCH	Returns a member of a collection by index or by name. The **Item** method is the default method for collections.	

Setup Object

Defines a position of the part on the machine.

The **Setup** object should have the following properties:

Property name	Return type	Description	Notes
Active	VT_BOOL	Returns or sets the value determining whether the setup is active; read/write.	
ActiveWorkspace	VT_DISPATCH	Returns the active **Workspace** object or VT_EMPTY if none; read –only.	
AttachmentFileNames*	VT_ARRAY(BSTR)*	Returns or sets the attachment file names; read/write.	
CoordinateSystemOrigin*	VT_DISPATCH	Returns the **Point3d** object representing the coordinate system origin; read-only.	
CreatedBy*	VT_BSTR	Returns the name of the setup creator; read-only.	
CreatedOn*	VT_DATE	Returns the date and time of setup creation; read-only.	
CustomProperties*	VT_DISPATCH	Returns the **CustomProperties** collection object; read-only.	
Name*	VT_BSTR	Returns or sets the setup name; read/write.	
ObjectID*	VT_BSTR	Returns the setup ID; read-only.	
PowerSourceConfigurationName*	VT_BSTR	Returns or sets the name of the PowerSource Configuration used with this setup; read/write.	
ProgramPrefix*	VT_BSTR	Sets or returns the default base name (first 4 characters) to use for the name of newly created programs; read/write.	
Programs*	VT_DISPATCH	Returns the **Programs** collection object; read-only.	
PropertyEditedBy*	VT_BSTR	Returns the name of the last user who edited the setup; read-only.	
PropertyEditedOn*	VT_DATE	Returns the date that the setup was last modified; read-only.	
UndercutValue*	VT_R8	Returns or sets the undercut value; read/write.	
Workspaces	VT_DISPATCH	Returns the **Workspaces** collection object; read-only.	

The **Setup** object should have the following methods:

Method name	Return type	Description	Notes
GetAltIsoViewPictureFile* *Width, Height, Attributes, FileName*	VT_EMPTY	Generates bitmap file with model image in Alt Iso view. *Width* is an optional parameter specifying width of the picture. *Height* is an optional parameter specifying the height of the picture. *Attributes* is an optional parameter that can be a combination of any of the following EPctureAttributes constants: prBlock, prAxes, prShade. *FileName* is a string specifying the name of the bitmap file to be created.	
GetBottomViewPictureFile* *Width, Height, Attributes, FileName*	VT_EMPTY	Generates bitmap file with model image in Bottom view. *Width* is an optional parameter specifying width of the picture. *Height* is an optional parameter specifying the height of the picture. *Attributes* is an optional parameter that can be a combination of any of the following EPctureAttributes constants: prBlock, prAxes, prShade. *FileName* is a string specifying the name of the bitmap file to be created.	
GetFrontViewPictureFile* *Width, Height, Attributes, FileName*	VT_EMPTY	Generates bitmap file with model image in Front view. *Width* is an optional parameter specifying width of the picture. *Height* is an optional parameter specifying the height of the picture. *Attributes* is an optional parameter that can be a combination of any of the following EPctureAttributes constants: prBlock, prAxes, prShade. *FileName* is a string specifying the name of the bitmap file to be created.	
GetIsoViewPictureFile* *Width, Height, Attributes, FileName*	VT_EMPTY	Generates bitmap file with model image in Iso view. *Width* is an optional parameter specifying width of the picture. *Height* is an optional parameter specifying the height of the picture. *Attributes* is an optional parameter that can be a combination of any of the following EPctureAttributes constants: prBlock, prAxes, prShade. *FileName* is a string specifying the name of the bitmap file to be created.	
GetLeftViewPictureFile* *Width, Height, Attributes, FileName*	VT_EMPTY	Generates bitmap file with model image in Left view. *Width* is an optional parameter specifying width of the picture. *Height* is an optional parameter specifying the height of the picture. *Attributes* is an optional parameter that can be a combination of any of the following EPctureAttributes constants: prBlock, prAxes, prShade. *FileName* is a string specifying the name of the bitmap file to be created.	

Method name	Return type	Description	Notes
GetRightViewPictureFile* *Width, Height, Attributes, FileName*	VT_EMPTY	Generates bitmap file with model image in Right view. *Width* is an optional parameter specifying width of the picture. *Height* is an optional parameter specifying the height of the picture. *Attributes* is an optional parameter that can be a combination of any of the following EPctureAttributes constants: prBlock, prAxes, prShade. *FileName* is a string specifying the name of the bitmap file to be created.	
GetTopViewPictureFile* *Width, Height, Attributes, FileName*	VT_EMPTY	Generates bitmap file with model image in Top view. *Width* is an optional parameter specifying width of the picture. *Height* is an optional parameter specifying the height of the picture. *Attributes* is an optional parameter that can be a combination of any of the following EPctureAttributes constants: prBlock, prAxes, prShade. *FileName* is a string specifying the name of the bitmap file to be created.	
Print *Style*	VT_EMPTY	Prints the entire setup. *Style* can be one of the following prPrintStyle constants: prProjectTypeStyle or prProgramTypeStyle;	
ResetRSM	VT_EMPTY	Resets RSM composite and surface data.	Hidden; for our internal/debugging use only
Info	VT_BSTR	Returns setup info string.	Hidden; for our internal/debugging use only

Workspaces Collection Object

Workspaces is a collection of Workspace objects that orients the tool relative to the part.

The **Workspaces** object should have the following properties:

Property name	Return type	Description	Notes
Count	VT_I4	Returns the number of items in the specified collection; read-only.	

The **Workspaces** object should have the following methods:

Method name	Return type	Description	Notes
Item *Index/Name*	VT_DISPATCH	Returns a member of a collection by index or by name. The **Item** method is the default method for collections.	

Workspace Object

A workspace orients the tool relative to the part.

The **Workspace** object should have the following properties:

Property name	Return type	Description	Notes
Active	VT_BOOL	Returns or sets the value determining whether the workspace is active; read/write.	
CoordinateSystemOrigin	VT_DISPATCH	Returns the **Point3d** object representing the coordinate system origin; read-only.	
Name	VT_BSTR	Returns or sets the workspace name; read/write.	

Property name	Return type	Description	Notes
ConstantStockValue	VT_R8	Returns or sets the constant stock value; read/write.	
IsDefault	VT_BOOL	Determines whether the workspace is default; read-only.	

The **Workspace** object should have the following methods:

Method name	Return type	Description	Notes
Info	VT_BSTR	Returns setup info string.	Hidden; for our internal/debugging use only

Programs Collection Object

The **Programs** object should have the following properties:

Property name	Return type	Description	Notes
Count	VT_I4	Returns the number of items in the specified collection; read-only.	

The **Programs** object should have the following methods:

Method name	Return type	Description	Notes
Item *Index/Name*	VT_DISPATCH	Returns a member of a collection by index or by name. The **Item** method is the default method for collections.	

Program Object

The **Program** object should have the following properties:

Property name	Return type	Description	Notes
BuildBusy*	VT_BOOL	Returns the value specifying whether program is being built; read-only.	
BuildFailed*	VT_BOOL	Returns the value specifying whether program built has failed; read-only.	
BuildGouged*	VT_BOOL	Returns the value specifying whether program built has gouged; read-only.	
BuildOK*	VT_BOOL	Returns the value specifying whether program built was successful; read-only.	
Category*		Returns the program category. Can be one of the following ECategory constants: prRough, prSemiRough, prSemiFinish, prFinish; read-only.	
CategoryString*	VT_BSTR	Returns the program category as a string; read-only.	
CheckMarked*	VT_BOOL	Returns or sets the value specifying whether program is check-marked; read/write.	
CreatedBy*	VT_BSTR	Returns the name of the program creator; read-only.	
CreatedOn*	VT_DATE	Returns the date and time of program creation; read-only.	
CustomProperties*	VT_DISPATCH	Returns the **CustomProperties** collection object; read-only.	
MachineTime*	VT_BSTR	Returns the machine time; read-only.	
Name*	VT_BSTR	Returns or sets the program name; read/write.	
NeedsBuild*	VT_BOOL	Returns the value specifying whether the program needs to be build; read-only.	
ObjectID*	VT_BSTR	Returns the program ID; read-only.	
Parameters*	VT_DISPATCH	Returns the **Parameters** object; read-only.	
PostedBy*	VT_BSTR	Returns the name of the last user who posted this program; read-only.	
PostedOn*	VT_DATE	Returns the date that the program was last posted; read-only.	
PropertyEditedBy*	VT_BSTR	Returns the name of the last user who edited the program; read-only.	
PropertyEditedOn*	VT_DATE	Returns the date that the program was last modified; read-only.	
SentToController*	VT_BOOL	Returns the value specifying whether program was sent to controller; read-only.	
SetupName*	VT_BSTR	Returns the name of the parent setup; read-only.	

Property name	Return type	Description	Notes
StatusString*	VT_BSTR	Returns a string representing program state ("Gouged", "Green", etc.); read-only.	
ToolPathEdited*	VT_BOOL	Returns the value specifying whether program's toolpath was edited manually; read-only.	
ToolPathEditedBy*	VT_BSTR	Returns the name of the last user who edited the program's toolpath; read-only.	
ToolPathEditedOn*	VT_DATE	Returns the date that the program's toolpath was last edited manually; read-only.	
Workspace	VT_DISPATCH	Returns the **Workspace** object; read-only.	
WorkspaceInfo*	VT_BSTR	Returns the string describing the name and rotation info of the parent workspace; read-only.	
ToolPath	VT_DISPATCH, VT_EMPTY	Returns the **ToolPath** object or VT_EMPTY if program is not loaded; read-only.	

The **Program** object should have the following methods:

Method name	Return type	Description	Notes
GetAltIsoViewPictureFile* *Width, Height, Attributes, FileName*	VT_EMPTY	Generates bitmap file with model image and program toolpath in Alt Iso view. *Width* is an optional parameter specifying width of the picture. *Height* is an optional parameter specifying the height of the picture. *Attributes* is an optional parameter that can be a combination of any of the following EPctureAttributes constants: prBlock, prAxes, prShade. *FileName* is a string specifying the name of the bitmap file to be created.	
GetBottomViewPictureFile* *Width, Height, Attributes, FileName*	VT_EMPTY	Generates bitmap file with model image and program toolpath in Bottom view. *Width* is an optional parameter specifying width of the picture. *Height* is an optional parameter specifying the height of the picture. *Attributes* is an optional parameter that can be a combination of any of the following EPctureAttributes constants: prBlock, prAxes, prShade. *FileName* is a string specifying the name of the bitmap file to be created.	
GetFrontViewPictureFile* *Width, Height, Attributes, FileName*	VT_EMPTY	Generates bitmap file with model image and program toolpath in Front view. *Width* is an optional parameter specifying width of the picture. *Height* is an optional parameter specifying the height of the picture. *Attributes* is an optional parameter that can be a combination of any of the following EPctureAttributes constants: prBlock, prAxes, prShade. *FileName* is a string specifying the name of the bitmap file to be created.	

Method name	Return type	Description	Notes
GetIsoViewPictureFile* *Width, Height, Attributes, FileName*	VT_EMPTY	Generates bitmap file with model image and program toolpath in Iso view. *Width* is an optional parameter specifying width of the picture. *Height* is an optional parameter specifying the height of the picture. *Attributes* is an optional parameter that can be a combination of any of the following EPctureAttributes constants: prBlock, prAxes, prShade. *FileName* is a string specifying the name of the bitmap file to be created.	
GetLeftViewPictureFile* *Width, Height, Attributes, FileName*	VT_EMPTY	Generates bitmap file with model image and program toolpath in Left view. *Width* is an optional parameter specifying width of the picture. *Height* is an optional parameter specifying the height of the picture. *Attributes* is an optional parameter that can be a combination of any of the following EPctureAttributes constants: prBlock, prAxes, prShade. *FileName* is a string specifying the name of the bitmap file to be created.	
GetRightViewPictureFile* *Width, Height, Attributes, FileName*	VT_EMPTY	Generates bitmap file with model image and program toolpath in Right view. *Width* is an optional parameter specifying width of the picture. *Height* is an optional parameter specifying the height of the picture. *Attributes* is an optional parameter that can be a combination of any of the following EPctureAttributes constants: prBlock, prAxes, prShade. *FileName* is a string specifying the name of the bitmap file to be created.	
GetTopViewPictureFile* *Width, Height, Attributes, FileName*	VT_EMPTY	Generates bitmap file with model image and program toolpath in Top view. *Width* is an optional parameter specifying width of the picture. *Height* is an optional parameter specifying the height of the picture. *Attributes* is an optional parameter that can be a combination of any of the following EPctureAttributes constants: prBlock, prAxes, prShade. *FileName* is a string specifying the name of the bitmap file to be created.	
Print	VT_EMPTY	Prints the program using "Program Style" page setup.	
LoadToolPath	VT_DISPATCH, VT_EMPTY	Loads the program toolpath. Returns the **ToolPath** object or VT_EMPTY if none was loaded; read-only.	
SetNeedsBuild	VT_EMPTY	Turns the program to a 'gray-ball' state.	
Info	VT_BSTR	Returns the program info string.	Hidden; for our internal/debugging use only

Model Object

The **Model** object should have the following properties:

Property name	Return type	Description	Notes
CurrentGeomEntity	VT_DTSPATCH, VT_EMPTY	Returns or sets current geometric object; Returns VT_EMPTY if there is no current object; read/write.	
GeomEntities	VT_DISPATCH	Returns the **GeomEntities** collection; read-only.	

The **Model** object should have the following methods:

Method name	Return type	Description	Notes
AddViewPoint [*X, Y,*] *SnapFilter*	VT_BOOL	Adds a point to the current profile or creates a one-point profile and makes it current. *X, Y* are optional 2D points in a screen space; If not specified current mouse position will be used. *SnapFilter* can be one of the following prSnapFilter constants: prSnapFilterNone, prSnapFilterNearestVertex, prSnapFilterNearestObject, prSnapFilterNearestArcCenter or prSnapFilterSmart. If **prSnapFilterNone** is used, adds a 3D point that corresponds to the screen space 2D point specified. If **prSnapFilterNearestVertex** is used, *X,Y* specifies a point in screen space to begin the search for the closest vertex on a 3D object. In an ever-expanding hit radius, find a vertex (control point) on a 3D object closest to the probe point. Give up if nothing is found within 999 pixels. If a vertex is found, add a point to the current profile. If **prSnapFilterNearestObject** is used, *X,Y* specifies a point in screen space to begin the search for the closest 3D object. Finds the object in the database nearest the probe point specified. Add a point to the current profile on the nearest object. If no objects are sufficiently close to the probe point, a free point will be added. **prSnapFilterNearestArcCenter** is specified, *X,Y* specifies a point in screen space to begin the search for the closest arc. Finds the arc closest to the 2D probe point. In an ever expanding hit radius, seeks the arc nearest the probe point and adds the center point of the arc/circle to the current profile. If **prSnapFilterSmart** is used, attempt to find the closest vertex to the probe point. If none found, try to find a point on nearby geometry. If not possible, then add a free point. Returns **True** if a point was added, **False** otherwise.	
Create3PointArcProfile *Pt1, Pt2, Pt3*	VT_DISPATCH, VT_EMPTY	Creates an arc given three points. Returns **Profile** object or VT_EMPTY if none was created. *Pt1, Pt2, Pt3* are 3Dpoints.	
Create3PointCircleProfile *Pt1, Pt2, Pt3*	VT_DISPATCH, VT_EMPTY	Creates a circle given three points. Returns **Profile** object or VT_EMPTY if none was created. *Pt1, Pt2, Pt3* are 3D points.	
CreateCircleProfileByRadius *ptCenter, ptRadius*	VT_DISPATCH, VT_EMPTY	Creates a circle given the center and radius. Returns **Profile** object or VT_EMPTY if none was created. *PtCenter* and *ptRadius* are 3D points.	
CreatePointsProfile *Pt1, …PtN*	VT_DISPATCH, VT_EMPTY	Creates a profile. Returns **Profile** object or VT_EMPTY if none was created. *Pt1, …, PtN* are 3D points.	
CreateProfile *Pt1, …PtN*	VT_DISPATCH, VT_EMPTY	Creates a profile. Returns **Profile** object or VT_EMPTY if none was created. *Pt1, …, PtN* are 3D points.	
CreateSplineProfile *Pt1, …PtN*	VT_DISPATCH, VT_EMPTY	Creates a spline given control points. Returns **Profile** object or VT_EMPTY if none was created. *Pt1, …, PtN* are 3D points.	

Method name	Return type	Description	Notes
FindArcCenterPoint [*X, Y*]	VT_DISPATCH, VT_EMPTY	Returns a **Point3d** object representing an arc/circle center closest to the cursor or VT_EMPTY if none was found. *X, Y* are optional 2D points in a screen space; If not specified current mouse position will be used.	
FindGeomEntity [*X, Y*]	VT_DISPATCH, VT_EMPTY	Returns **GeomEntity** nearest to the cursor or VT_EMPTY if none is found. *X, Y* are optional 2D points in a screen space; If not specified current mouse position will be used.	
FindObjectPoint [*X, Y*]	VT_DISPATCH, VT_EMPTY	Returns a **Point3d** object representing a point on a 3D object closest to the cursor or VT_EMPTY if none was found. *X, Y* are optional 2D points in a screen space; If not specified current mouse position will be used.	
FindVertexPoint [*X, Y*]	VT_DISPATCH, VT_EMPTY	Returns a **Point3d** object representing a vertex (control point) on a 3D object closest to the cursor or VT_EMPTY if none was found. *X, Y* are optional 2D points in a screen space; If not specified current mouse position will be used.	
Info	VT_BSTR	Returns a description of this object.	Hidden; for our internal/debugging use only
JoinProfiles *Profiles*	VT_EMPTY	Joins profiles.	
OffsetProfiles *Profiles, NumOffsets, Distance*	VT_EMPTY	Offsets the profiles.	
RemoveAll	VT_EMPTY	Removes all objects from the Model.	

GeomEntities Object

The **GeomEntities** object should have the following properties:

Property name	Return type	Description	Notes
Count	VT_I4	Returns the number of objects in the specified collection; read-only.	

The **GeomEntities** object should have the following methods:

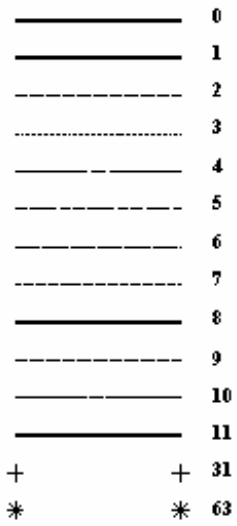
Method name	Return type	Description	Notes
Delete	VT_EMPTY	Deletes a collection from the model and destroys it.	
Info	VT_BSTR	Returns a description of the characteristics of all objects in the collection.	Hidden; for our internal/debugging use only
Restrict	VT_DISPATCH, VT_EMPTY	Takes a string containing a filter and returns a new collection containing only items that match the filter or VT_EMPTY if no items were found.	

Method name	Return type	Description	Notes
Select *eSelectionMode*	VT_EMPTY	Selects the collection objects. eSelectionMode can be one of the following prSelectionMode constants: prSelectionNormal, prSelectionToggle, prSelectionPlus, or prSelectionMinus. If prSelectionMode is used, all objects are un-selected first, then the objects in the collection are marked as selected. If the prSelectionPlus option is specified, the objects in the collection are marked selected. If the prSelectionMinus option is used, objects in the collection are de-selected. If the prSelectionToggle option is used, the state of the objects in the collection is toggled (i.e. selected objects are de-selected and un-selected objects are selected).	
SetColor *Color*	VT_EMPTY	Sets the color of all objects in the collection to the color specified. *Color c*an be one of the following prColor constants: prColorDarkBlue, prColorGreen, prColorRed, prColorMagenta, prColorBrown, prColorLightBlue, prColorOrange, prColorLightOrange, prColorYellow, prColorCyan or prColorLime;	
SetFlowLineNumber	VT_EMPTY	Sets the display characteristics of surface objects in the collection to contain the number of flow lines specified.	
SetFlowLineQuality	VT_EMPTY	Sets the quality of the flow lines of surface objects in thie collection.	
SetLayer	VT_EMPTY	Sets the layer of all collection objects to the layer specified. Layers are numbered 0 through 1024.	
SetLineStyle	VT_EMPTY	Sets the line style for all collection objects to this style. Line styles are numbered 0 through 63.	
Show	VT_EMPTY	Shows and hides all collection objects.	
WritePDB *FileName, Append*	VT_BOOL	Saves collection objects into a PDB file.	Hidden; for our internal/debugging use only

GeomEntity Object

The **GeomEntity** object should have the following properties:

Property name	Return type	Description	Notes
Color	VT_I4	Sets or returns the color of the object. Can be one of the following prColor constants: prColorDarkBlue, prColorGreen, prColorRed, prColorMagenta, prColorBrown, prColorLightBlue, prColorOrange, prColorLightOrange, prColorYellow, prColorCyan or prColorLime; read/write.	
ContainedBy *Window*	VT_BOOL	Returns the value determining whether the object is contained by the specified Window object; read-only.	
ControlPoints	VT_DISPATCH	Returns the collection of **Point3D** objects representing control points in a current workspace coordinates; read-only.	
Current	VT_BOOL	Sets or returns the value determining whether the object is current; read/write.	
EnclosedBy *Profile*	VT_BOOL	Returns the value determining whether the object is enclosed by the specified profile; read-only.	
Highlighted	VT_BOOL	Sets or returns the value determining whether the object is highlighted; read/write.	
InBox *XMin, YMin, XMax, YMax*	VT_BOOL	Returns the value determining whether the object is in a box XMin, YMin, XMax, YMax; read-only.	
Layer	VT_I4	Sets or returns the layer of the object. A layer is specified as an integer in a range from 0 to 1024; read/write.	

Property name	Return type	Description	Notes
LineStyle	VT_I4	Sets or returns the line style of the object. Line styles are specified as integers in the range from 0 through 63; read/write. Shown below are the most frequently used line styles:  	
Marked	VT_BOOL	Sets or returns the value determining whether the object is marked; read/write.	
Name	VT_BSTR	Sets or returns the name of the object; read/write.	
Near [*X, Y*]	VT_BOOL	Returns the value determining whether the object is near X, Y view point; read-only. *X, Y* are optional 2D points in a screen space; If not specified current mouse position will be used.	
Selected	VT_BOOL	Sets or returns the value determining whether the object is selected; read/write.	
ShowDirection	VT_BOOL	Sets or returns the value determining whether the object's direction is shown; read/write.	
Type	VT_I4	Returns the type of the object. Can be one of the following prGeomType constants: prGeomTypeProfile, prGeomTypePointProfile, prGeomTypeSurface, prGeomTypeWindow, prGeomTypeCurve, prGeomTypeOffsetSurface, prGeomTypeTrimSurface, prGeomTypeToolPath, or prGeomTypeMesh; read-only.	
Visible	VT_BOOL	Sets or returns the value determining whether the object is visible; read-write.	

The **GeomEntity** object should have the following methods:

Method name	Return type	Description	Notes
Delete	VT_EMPTY	Removes the object from the Model and deletes it.	

CurrentGeomEntity

The **CurrentGeomEntity** should have all the properties of the **GeomEntity** and more:

Property name	Return type	Description	Notes
CurrentPointIndex	VT_I4	Returns index of the current point; read/write.	

The **CurrentGeomEntity** should have all the methods of the **GeomEntity** object.

Profile Object

The **Profile** object has all the properties of the **GeomEntity**.

The **Profile** object should have all the methods of the **GeomEntity** object and more:

Method name	Return type	Description	Notes
Add3PointArc *Pt1, Pt2, Pt3, [Index]*	VT_EMPTY	Adds/Inserts an arc to the profile.	
AddPointAbsolute *X, Y, Z, [Index]*	VT_EMPTY	Adds/Inserts absolute point to the profile.	
AddPointIncremental *X, Y, Z,[Index]*	VT_EMPTY	Adds/Inserts incremental point to the profile.	
CloseUp	VT_EMPTY	Closes up the profile.	
DeletePoint *PointIndex*	VT_EMPTY	Deletes the specified control point from the profile.	
LastPointsToArc	VT_EMPTY	Turns the last 3 points of the profile into an arc.	
LastPointsToCircle	VT_EMPTY	Turns the 3-point profile into a circle.	
LastPointsToSpline	VT_EMPTY	Turns the last profile points into a Spline.	
ReverseDirection	VT_EMPTY	Reverses the direction of the profile.	

PointProfile Object

The **PointProfile** object has all the properties and all the methods of the **GeomEntity**.

Surface Object

The **Surface** object has all the properties and all the methods of the **GeomEntity**.

OffsetSurface Object

The **OffsetSurface** object has all the properties and all the methods of the **GeomEntity**.

TrimSurface Object

The **TrimSurface** object has all the properties and all the methods of the **GeomEntity**.

Mesh Object (hidden)

The **Mesh** object has all the properties and all the methods of the **GeomEntity**.

Window Object

The **Window** object has all the properties and all the methods of the **GeomEntity**.

ToolPath Object

The **ToolPath** object has all the properties of the **GeomEntity.**

The **ToolPath** object should have all the methods of the **GeomEntity** and more:

Method name	Return type	Description	Notes
DeleteSelectedPoints	VT_EMPTY	Deletes selected points	
MoveSelectedPoints *Vector*	VT_EMPTY	Moves selected points.	
SelectEnclosedPoints *Profile, SelectionMode*	VT_EMPTY	Selects points enclosed by a profile. *SelectionMode* can be one of the following prSelectionMode constants: prSelectionToggle, prSelectionPlus, or prSelectionMinus.	
SelectInBoxPoints *XMin, YMin, XMax, YMax, SelectionMode*	VT_EMPTY	Selects points enclosed by a box. *SelectionMode* can be one of the following prSelectionMode constants: prSelectionToggle, prSelectionPlus, or prSelectionMinus.	

Method name	Return type	Description	Notes
SelectNearestPoints [*X, Y*]	VT_EMPTY	Selects points passing through a given point. *X, Y* are optional 2D points in a screen space; If not specified current mouse position will be used.	
TrimToProfile *Profile*	VT_EMPTY	Trims ToolPath to the specified profile.	

View Object

The **View** object represents all of the application window views: 3D view, RSM view and Pencil Trace view. Some properties and methods of the **View** object work only in certain views. If you try to use a property or method that's inappropriate for a **View** object, an error occurs.

The **View** object should have the following properties:

Property name	Return type	Description	Notes
AllowPointProfileSelection	VT_BOOL	Allows point profiles to be selected; read/write.	
AllowProfileSelection	VT_BOOL	Allows profiles to be selected; read/write.	
AllowToolPathSelection	VT_BOOL	Allows toolpaths to be selected; read/write.	
Antialiasis	VT_BOOL	Turns antialiasis on or off; read/write.	
Center	VT_ARRAY (double)	Returns and sets the center point of the view; read/write.	
EnableSnapTo	VT_BOOL	Toggle snap on or off; read/write.	
Height	VT_I4	Returns the view height; read-only.	
HitTestRadius	VT_I4	Returns and sets geometry hit test radius; read/write.	Hidden; for our internal/debugging use only
MeshColorUVWBoundary	VT_I4	Returns or sets the color of mesh UVW boundary; read/write.	Hidden; for our internal/debugging use only
MeshColorUVWExterior	VT_I4	Returns or sets the color of mesh UVW exterior; read/write.	Hidden; for our internal/debugging use only
MeshInteriorLinesCount	VT_I4	Returns or sets the number of mesh interior lines; read/write.	Hidden; for our internal/debugging use only
MeshShadeLinear	VT_BOOL	Returns or sets the value specifying whether to draw linear mesh shade; read/write.	Hidden; for our internal/debugging use only
RubberBand	VT_ARRAY (int)	Returns the coordinates of a last rubber band drawn by the user; read-only.	
Shade	VT_BOOL	Turns shade on or off; read/write.	
ShowBlock	VT_BOOL	Toggle block on or off; read/write.	
ShowGrid	VT_BOOL	Toggle grid on or off; read/write.	
ShowMeshControlPoints	VT_BOOL	Returns or sets the value specifying whether to show mesh control points; read/write.	Hidden; for our internal/debugging use only
ShowMeshNegZOnlyNormals	VT_BOOL	Returns or sets the value specifying whether to show negative-z only mesh normals; read/write.	Hidden; for our internal/debugging use only
ShowMeshNormals	VT_BOOL	Returns or sets the value specifying whether to show mesh normals; read/write.	Hidden; for our internal/debugging use only
ShowShadedTool	VT_BOOL	Toggles shaded tool display on or off; read/write.	
ShowToolOutline	VT_BOOL	Toggles tool outline on or off; read/write.	

Property name	Return type	Description	Notes
ShowTrackBall	VT_BOOL	Toggle trackball on or off; read/write.	
SnapFilter	VT_I4	Returns and sets the mouse snap filter. Can be one of the following prSnapFilter constants: prSnapFilterNone, prSnapFilterNearestVertex, prSnapFilterNearestObject, prSnapFilterNearestArcCenter or prSnapFilterSmart; read/write.	
Type	VT_I4	Can be one of the following prViewType constants: pr3Dview, prRSMView, prPencilTraceView; read/write.	
Width	VT_I4	Returns the view width; read-only.	

The **View** object should have the following methods:

Method name	Return type	Description	Notes
PlaneView *Pt1, Pt2, Pt3*	VT_EMPTY	Changes the view in the direction of plane's normal vector. The plane is based on three points given by the user.	
Redraw	VT_EMPTY	Redraws the view.	
Restore *SaveState*	VT_EMPTY	Restores the view to the specified state. The view is restored by popping state information off a stack created by earlier calls to the **Save** function.	
Rotate *Ang1, Ang2, Ang3*	VT_EMPTY	Rotates the view.	
Save	VT_R8	Saves the current state of the view to a context stack. If the function succeeds, the return value identifies the saved state. If the function fails, the return value is zero. A saved state can be restored by using the **Restore** function.	
SetView *XRot, YRot, ZRot*	VT_EMPTY	Sets view rotation.	
SetViewOrientation *eViewOrientation*	VT_EMPTY	Sets view orientation. *eViewOrientation* can be one of the following prViewOrientation constants: prViewOrientationIso, prViewOrientationAltIso, prViewOrientationIsoIso, prViewOrientationAltIsoIso, prViewOrientationTop, prViewOrientationBottom, prViewOrientationLeft, prViewOrientationRight, prViewOrientationFront, or prViewOrientationBack.	
ShadeOneTime	VT_EMPTY	Performs a one-time shade.	Hidden; for our internal/debugging use only
Pan *dX, dY*	VT_EMPTY	Shifts the view in X and Y directions.	
TrackRubberBand	VT_ARRAY (int)	Does a rubber-band mouse tracking. Returns the coordinates of a box drawn by the user.	
VectorView *Pt1, Pt2*	VT_EMPTY	Changes the view in the direction of the vector. The vector is based on two points given by the user.	
Zoom *Scale*	VT_EMPTY	Zooms the view using the specified scale.	
ZoomIn	VT_EMPTY	Zooms into the view by a factor of 2.0.	
ZoomOut	VT_EMPTY	Zooms out of the view by a factor of 0.5.	
ZoomToBox *X1, Y1, X2, Y2*	VT_EMPTY	Zooms box to fit the dimensions of the view's window.	
ZoomToFit	VT_EMPTY	Zooms the view to fit the dimensions of the view's window.	

The **View** object should have the following events:

Event name	Description
MouseDown *button,key,X,Y*	Occurs when user presses the mouse button. Button Mask Constants: prLeftButton, prMiddleButton and prRightButton. Key Mask Constants: prAltMask, prCtrlMask and prShiftMask.
MouseUp *button,key,X,Y*	Occurs when user releases the mouse button. Button Mask Constants: prLeftButton, prMiddleButton and prRightButton. Key Mask Constants: prAltMask, prCtrlMask and prShiftMask.
Click *button,key,X,Y*	Occurs when user presses and then releases the mouse button. Button Mask Constants: prLeftButton, prMiddleButton and prRightButton. Key Mask Constants: prAltMask, prCtrlMask and prShiftMask.
DoubleClick *button,key,X,Y*	Occurs when user presses and releases the left mouse button twice within the double-click time limit of the system. Button Mask Constants: prLeftButton, prMiddleButton and prRightButton. Key Mask Constants: prAltMask, prCtrlMask and prShiftMask.

Global Object

The **Global** object should have the following methods:

Method name	Return type	Description	Notes
Measure3PtAngle *Pt1, Pt2, Pt3*	VT_R8	Measures three-point angle.	
Measure4PtAngle *Pt1, Pt2, Pt3, Pt4*	VT_R8	Measures four-point angle.	
MeasureDistance *Pt1, Pt2*	VT_R8	Measures the distance between two points.	
MeasureRadius *Pt1, Pt2, Pt3*	VT_R8	Measures the radius of an arc using three points.	
Max *Val1, Val2,…ValN*	VT_R8	Returns the maximum of a set of values.	
Min *Val1, Val2,…ValN*	VT_R8	Returns the minimum of a set of values.	
Fequals *Val1, Val2* [,*Epsilon*]	VT_BOOL	Checks if 2 values are nearly equal.	
Fzero *Value* [,*Epsilon*]	VT_BOOL	Checks if *Value* is zero or very close to zero.	