

Contents

Overview.....2

New API Components.....3

 AlphacamObjects (*New Object*)3

 Drawing.....3

 MillData.....3

 Path4

 Spline.....4

 Surface.....4

 SolidPart4

Example Add-Ins.....5

Overview

With Alphacam 2012 R1, the API has been enhanced to allow developers to create associative and editable custom operations via VBA or C++ add-ins.

Previously, any operation that was not created with a standard Alphacam machining command could not be made to be associative and could not be edited. Meaning, if the machined geometry changed, or if the machining parameters needed to be changed, the operation would have to be deleted and recreated. Now, by using the new API members and properly constructing the add-in, custom operations can be made to act like any other Alphacam operation, including associativity, editing capability, and Machining Style support. This includes support for Solids, Surfaces, etc. as well as the following Project Manager commands...

- Edit
- Update
- Copy Operation
- Move to Own Operation
- Add Geometries
- Remove from Operation

This document lists the new API components specific to this functionality and also gives an overview of the example add-ins provided with this document. The example add-ins contain well documented source code to help understand the requirements and structure needed.

New API Components

AlphacamObjects (*New Object*)

A collection of machinable Alphacam objects, including Path, Spline, Surface, SolidPart, SolidFace

- Function **Item** (ByVal Index As Long) As Object
Returns a specific object (Path, Spline, Surface, SolidPart, or SolidFace) of the AlphacamObject collection from the given Index
- Property **Count** () As Long
Get the number of objects within the AlphacamObject collection (*read-only*)
- Sub **Add** (ByVal Dispatch As Object)
Will add the given object (Path, Spline, Surface, SolidPart, or SolidFace) to the AlphacamObject collection
- Sub **Remove** (ByVal Dispatch As Object)
Will remove the given object (Path, Spline, Surface, SolidPart, or SolidFace) from the AlphacamObject collection

Drawing

- Function **CreateAlphacamObjectsCollection** () As [AlphacamObjects](#)
Creates and returns an empty [AlphacamObjects](#) Collection

MillData

- Sub **AssociateGeometry** (ByVal Geo As Object, ByVal Flag As Long)
Will associate the given object (Path, Spline, Surface, SolidPart, or SolidFace) with the MillData

The **Flag** argument is used as a unique identifier if multiple geometries are associated with the MillData. If only one geometry is associated with the MillData, or if multiple geometries do not need to be uniquely identified, the **Flag** value can simply be set to 0.
- Sub **AssociateToolPaths** (ByVal Path As Paths)
Will associate the given Paths collection with the MillData
- Sub **SetUpdateFunction** (ByVal FunctionName As String)
Sets the function name in the Events module to be called when updating an operation that was created with this MillData. To prevent the operation from being updated, don't call this procedure, or set FunctionName to a null string.
- Sub **SetEditFunction** (ByVal FunctionName As String)
Sets the function name in the Events module to be called when editing an operation that was created with this MillData. To prevent the operation from being edited, don't call this procedure, or set FunctionName to a null string.
- Sub **SetSelectForStyleFunction** (ByVal FunctionName As String)
Sets the function name in the Events module to be called when applying a Machining Style that was created from an operation that was created with this MillData. This is only needed if special handling of geometry selection is required when applying the Machining Style (e.g., if multiple geometries).

- Sub **SetBeforeAddGeometriesFunction** (ByVal FunctionName As String)
Sets the function name in the Events module to be called when adding geometry to an operation that was created with this MillData. To disable the “Add Geometries” context menu item and prevent geometry from being added to the operation, don’t call this procedure, or set FunctionName to a null string.
- Sub **SetBeforeRemoveGeometryFunction** (ByVal FunctionName As String)
Sets the function name in the Events module to be called to enable/disable the “Remove from Operation” context menu item for geometry that reside in an operation that was created with this MillData. To always disable this menu item and prevent geometry from being removed from the operation, don’t call this procedure, or set FunctionName to a null string.
- Sub **SetBeforeMoveToOwnOpFunction** (ByVal FunctionName As String)
Sets the function name in the Events module to be called to enable/disable the “Move to Own Operation” context menu item for geometry that reside in an operation that was created with this MillData. To always disable this menu item and prevent geometry from being moved to its own operation, don’t call this procedure, or set FunctionName to a null string.
- Sub **SetBeforeChangeToolFunction** (ByVal FunctionName As String)
Sets the function name in the Events module to be called to enable/disable the “Change Tool” context menu item for an operation that was created with this MillData. To always disable this menu item, don’t call this procedure, or set FunctionName to a null string.
- Function **GetGeometries** () As [AlphacamObjects](#)
Returns a collection of geometries (Path, Spline, Surface, SolidPart, or SolidFace) that are associated with the operation.
- Property **AttributeOp** (ByVal Index As String) As Variant
This is the same as the **Attribute** property, but is attached to the Operation only and is not copied to toolpaths within the Operation.

Path

- Property **FlagForEditableOp** () As Long
Get the flag passed to [MillData.AssociateGeometry](#)

Spline

- Property **FlagForEditableOp** () As Long
Get the flag passed to [MillData.AssociateGeometry](#)

Surface

- Property **FlagForEditableOp** () As Long
Get the flag passed to [MillData.AssociateGeometry](#)

SolidPart

- Property **FlagForEditableOp** () As Long
Get the flag passed to [MillData.AssociateGeometry](#)

Example Add-Ins

Accompanying this document is a set of the following three example add-ins...

- **EditOpsExample_BoringAlong2DLine**

This add-in demonstrates how to associate the created toolpath with one geometry, while actually machining a different geometry.

Add-Ins List Name: Example - Editable Ops, Boring Along 2D Line
File Name: EditOpsExample_BoringAlong2DLine.arb
Macro Name: EditOpsExample_BoringAlong2DLine
Example Part Drawing: ExamplePart_BoringAlong2D.ard

- **EditOpsExample_ShowerBase**

This add-in demonstrates how to associate a single toolpath to multiple geometries.

Add-Ins List Name: Example - Editable Ops, Shower Base
File Name: EditOpsExample_ShowerBase.arb
Macro Name: EditOpsExample_ShowerBase
Example Part Drawing: ExamplePart_ShowerBase.ard

- **EditOpsExample_SolidRoughFinish**

This add-in demonstrates how to associate a toolpath with solid parts/faces, surfaces and splines.

Add-Ins List Name: Example - Editable Ops, Solid Rough/Finish
File Name: EditOpsExample_SolidRoughFinish.arb
Macro Name: EditOpsExample_SolidRoughFinish
Example Part Drawing: ExamplePart_SolidRoughFinish.ard

Each add-in comes with the extracted source code modules and an example part drawing that can be used for machining.

The source code has been documented throughout in attempt to highlight the areas specific to implementing editable operation support. Please search for the code comments marked with **IMPORTANT** and **INFO** to find these areas.