



Modular Vision System Division
One Vision Drive
Natick, MA 01760

Support Hotline: 508/650-4100
Fax: 508/650-3321
ts.global@cognex.com

Technical Support

Technical Bulletin

Distribution:	Application Engineers, Technical Support Engineers, Vision Solutions Engineers, Technical Documentation Group, Worldwide Sales		
Bulletin No.:	TB-08-005	☐	INTERNAL ONLY
Date:	08/20/2008	☒	Internal & Customers (as needed)
Product:	VISIONPRO	Rev:	5.0 ☐ Internal & Customers (MANDATORY)
Title:	Order of Execution in VisionPro ToolGroups		

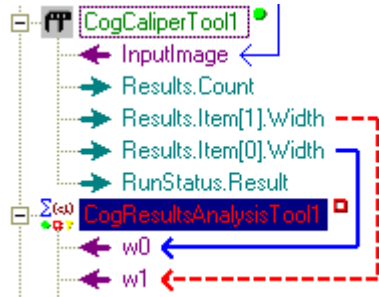
Purpose

This document is an informational bulletin on VisionPro's CogToolGroup class.
This document will:

- Explain the order of execution of CogTools inside a CogToolGroup.
- Explain how CogDataBindings pass data between CogTools inside a CogToolGroup.
- Explain the features **AbortRunOnToolFailure** and **FailOnInvalidDataBinding**, which you can use to change the order of execution of CogTools inside a CogToolGroup.
- Provide a practical example where you will use AbortRunOnToolFailure and FailOnInvalidDataBinding to alter the default order of execution of CogTools inside a CogToolGroup.

Some Useful Definitions

Term	Definition
CogTool or Tool	CogTools, or Tools are software objects that perform much of the work done by VisionPro. Examples are CogPMAAlignTool and CogCaliperTool. CogTools have a Run method which causes them to execute.
CogToolGroup or ToolGroup	CogToolGroups, or ToolGroups are software objects that group CogTools together and allow you to easily run the individual tools in sequence.
CogDataBinding or Data Binding	CogDataBindings are software objects that bind a source property from one Tool to a destination property of another Tool. DataBindings allow individual tools to easily share data. DataBindings detect when a property of the source Tool has changed and immediately update the bound property of the destination Tool. Data Bindings are represented by the graphical links that are

	displayed within QuickBuild and the ToolGroupEdit control.
Invalid Data Binding	An invalid DataBinding is a link between Tool properties in which the source property no longer contains a valid value. Invalid data bindings in QuickBuild show up as red dotted lines. In programmatic code this would be equivalent to a source property which is set to nothing in VB.NET or null in C#.  <i>Figure 1: Example of an Invalid DataBinding - CaliperTool1 has found only a single Result, hence the binding to the second Result is invalid and shown as a red dashed line.</i>
Run Status	RunStatus is a property of every Tool. After a Tool runs the RunStatus.Result indicates the success or failure of the Tool. A Tool is said to have failed if its RunStatus.Result returns "Error" after the tool is run.

Introduction - Running a ToolGroup

Figure 2 is a visual representation of a CogToolGroup in the CogToolGroupEdit control. The ToolGroup contains two CogTools and a single data binding. The data binding links Tool1's output to Tool2's input.

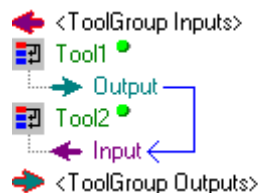


Figure 2: CogToolGroup containing two tools and a single data binding.

How to Run a ToolGroup:

You can run a ToolGroup by clicking the triangular "play" button on the CogToolGroupEdit control, or by calling its Run() method in code.

Order of Executing Tools:

When you run a ToolGroup, the ToolGroup runs of each of its enabled tools in the order that they are indexed in the Tools collection property (Or top to bottom as viewed in QuickBuild and the CogToolGroupEdit control).

When Data Is Passed Between Tools:

If a Tool is running and a data bound output value changes, that value is immediately written to the destination property by the DataBinding.

When the ToolGroup in Figure 2 is run, the sequence will be as follows:

1. Tool1 starts running.
2. Tool1's Output property changes
3. Tool1's Output property is written to Tool2's Input property.
4. Tool1 finishes running.
5. Tool2 starts running.
6. Tool2 finishes running.

Important Note: By default, a ToolGroup will not run any more of its downstream tools once any of its individual Tools has failed. This behavior can be changed using the ToolGroup.AbortRunOnToolFailure property.

AbortRunOnToolFailure

ToolGroups have a Boolean property called AbortRunOnToolFailure. This property controls the behavior of a running ToolGroup when one of its Tools fails.

If AbortRunOnToolFailure = True (DEFAULT):

A running ToolGroup will stop and cease executing downstream tools as soon as one of its individual Tools fail.

If AbortRunOnToolFailure = False:

A running ToolGroup will continue executing all of its individual tools in order, even if any of them fail.

The default value of AbortRunOnToolFailure is true. You can change the value of this property as shown in Figure 3.

To do this follow the steps below:

1. Click on the Object Browser button from the ToolGroupEdit control.
2. Find the entry for AbortRunOnToolFailure
3. Change the property from True to False.

Every ToolGroup in a QuickBuild application has its own AbortRunOnToolFailure setting. If you have multiple CogJobs, you can change this property for each one.

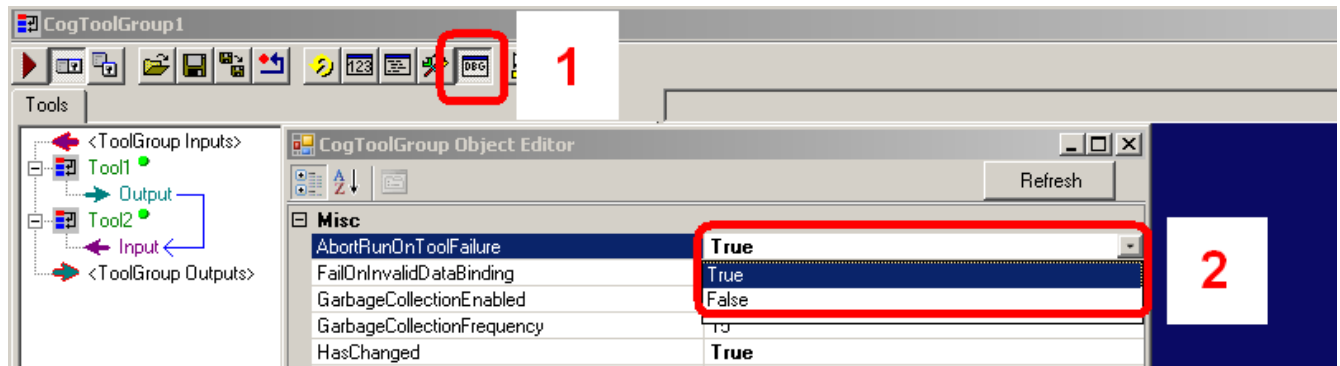


Figure 3: Object Browser showing AbortRunOnToolFailure Property.

FailOnInvalidDataBinding

All Tools have a Boolean property called FailOnInvalidDataBinding. This property controls the behavior of a Tool when one of its input terminals is connected to an invalid data binding.

When FailOnInvalidDataBinding = True (DEFAULT):

A Tool with any of its input terminals connected to an invalid data binding will fail.

When FailOnInvalidDataBinding = False:

A Tool with any of its input terminals connected to an invalid data binding will run as usual.

Important Note: When FailOnInvalidDataBinding is false, a tool with input terminal(s) connected via invalid data binding(s) runs normally except that the tool **uses the previous value of the input(s) which are still contained within the Tool.**

The default value of FailOnInvalidDataBinding is true. You can change the value of this property as shown in Figure 4.

To do this follow the steps below:

1. Click on the Object Browser button from the ToolGroupEdit control.
2. Navigate to the Tools section.
3. Identify the tool that you'd like to change.
4. Change the property from true to false.

If you have multiple Tools within this ToolGroup you can change this property for every tool.

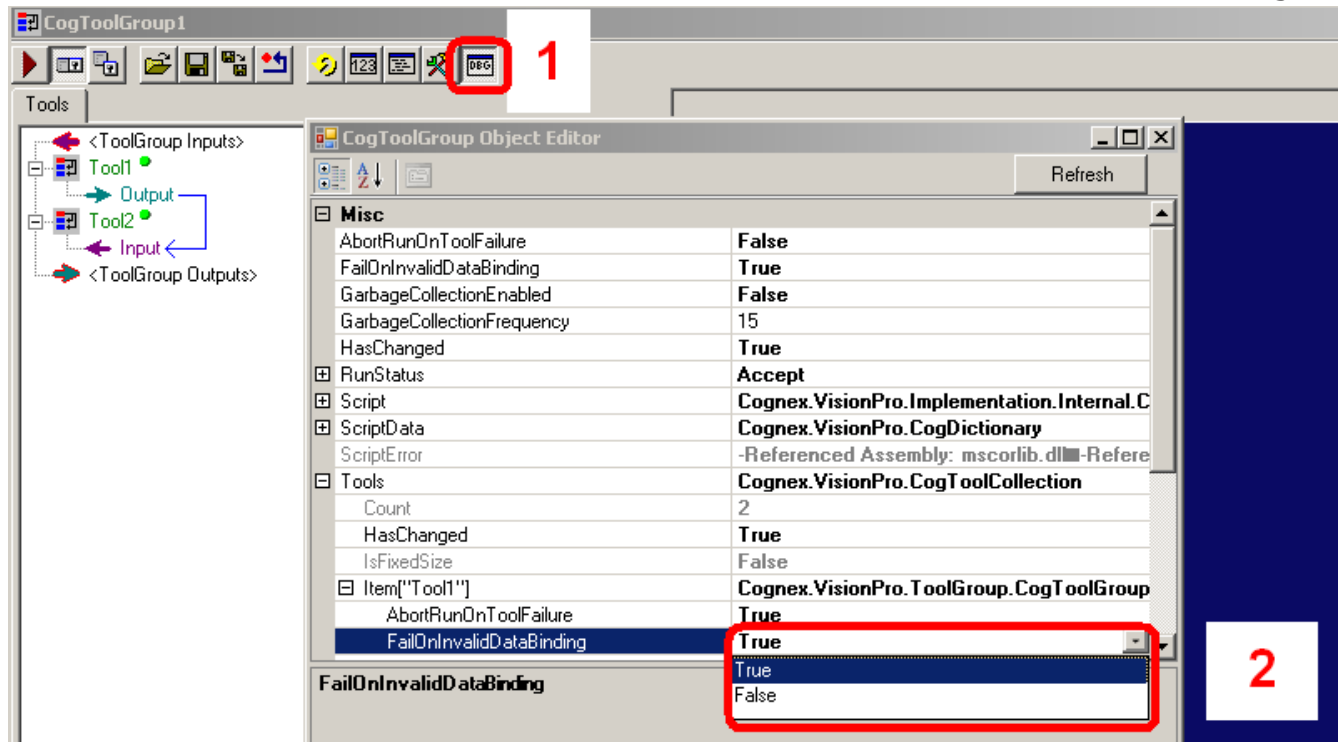


Figure 4: Object browser showing *FailOnInvalidDataBinding* property for *Tool1*.

A Practical Example

For some machine vision applications it might be useful to change the default values of the two properties described above. Consider the ToolGroup called *CheckSumOfTwoWidths* pictured in Figure 5.

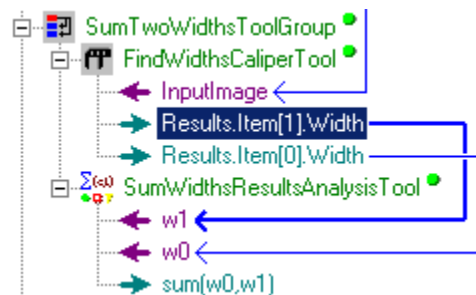


Figure 5: Sample ToolGroup that finds two widths and outputs their sum.

This ToolGroup consists of two tools. The first tool, *FindWidthsCaliperTool* (FWCT), finds two widths from features within the image. These widths are then passed via data bindings to the *SumWidthsResultsAnalysisTool* (SWRAT). SWRAT adds the two widths and then outputs the sum to a robot over a TCP/IP connection configured in the QuickBuild Communication Explorer.

A problem arises if FWCT fails. There are several possible reasons why FWCT may fail. For example, it may have been passed an invalid image, or the search region may be outside the image bounds. When a Tool fails there is a red icon placed next to the Tool name in QuickBuild. In figure 6 you can see FWCT reporting a failure, however, SWRAT still shows a passing status

(green icon). This is because the SWRAT was not run and is still reporting passing status from the last time it ran. Also, when FWCT fails, SWRAT's sum value will still contain the **old** value from the previous run. This can create a problem if the old result data is sent to the robot.

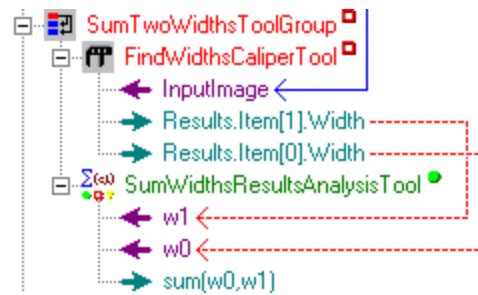


Figure 6: The FindWidthsCaliperTool has failed, the SumWidthsResultsAnalysisTool is never run and still shows a passing status from the last time it ran.

Important Note: When a tool fails a red icon is placed next to its tool name. When AbortRunOnToolFailure is true (the default), downstream tools will not run and will still reflect the pass/fail status from the previous run of the tool... Downstream tools may still be indicating a passing status even though they did not run!

A Practical Example - Resolved

To solve this problem we will change both **AbortRunOnToolFailure** and **FailOnInvalidDataBinding** as well as adjust the logic in SWRAT to force the system to output a value of 0 when the FWCT fails. We assume that our robot knows that a value of 0 indicates a bad run and this does not break the system. We will follow the following sequence of steps to resolve this problem.

Step 1:

- Set **AbortRunOnToolFailure** property of the ToolGroup to **false**. See Figure 3 on how to configure this.

This will cause the ToolGroup to run SWRAT every time, even when FWCT fails.

Step 2:

- Set the **FailOnInvalidDataBinding** property of the SWRAT to **false**. See Figure 4 on how to configure this.

When FWCT fails, its results collection is cleared and this invalidates its output data bindings. If **FailOnInvalidDataBinding** is true, SWRAT will detect its invalid input data bindings and immediately returns a failure status without actually executing its logic. If this happens SWRAT will still contain a stale result value. Setting **FailOnInvalidDataBinding** to false will force the tool to run even if its data bindings are invalid.

Step 3:

- Add an output terminal for RunStatus.Result to the FWCT. To do this:
 - Right-Click the FWCT in the QuickBuild Job Editor.
 - Select Add Terminals...
 - Click the plus sign to expand the RunStatus node of the Member Browser.
 - Select the Result node.
 - Click the Add Output button.

This new terminal will be connected to SWRAT to inform it if the FWCT has failed.

Step 4:

- Add a new input to SWRAT by:
 - Double-Click SWRAT in the QuickBuild Job Editor to open the SWRAT edit control.
 - Click the **Add Input** toolbar button in the SWRAT edit control
 - Select the Name Cell of the new Input (it will be called InputA)
 - Rename the input to "Caliper_Result".
 - Close the SWRAT edit control.

Step 5:

- Link the two new terminals together:
 - Click and hold RunStatus.Result output terminal of FWCT.
 - Drag to the Caliper_Result input terminal of SWRAT.

Step 6:

- Create a new intermediate expression in the SWRAT. See Figure 8 for details.
 - Double-Click SWRAT in the QuickBuild Job Editor to open the SWRAT edit control.
 - Click the **Add Expression** toolbar button.
 - Change the name of the new expression to "Caliper_Passed".
 - Enter -1 for Argument0.
 - Select **NotEqual** for the Operator.
 - Select **Caliper_Result** for Argument1.

The Caliper_Passed expression evaluates true when the Caliper_Result is Equal to 0 (0 is the numerical equivalent of the RunStatus.Result of Accept or Pass). This expression now allows us to tell directly if FWCT has passed.

Step 7:

- Create a new expression called "Sum*Caliper_Passed" This expression will multiply the value of Caliper_Passed and sum(w0,w1)
 - Click the **Add Expression** toolbar button.
 - Change the name of the new expression to "Sum*Caliper_Passed".
 - Select **sum(w0,w1)** as Argument0
 - Select **Mul** as the Operator.
 - Select **Caliper_Passed** as Argument1.
 - Check the Output CheckBox.
 - Reconfigure the Communication Explorer to output this new terminal value. See page 4 of the article here: [How To Configure the Communication Explorer to Output Tool Values](#).

A value of 0 will now be output to our robot when the FWCT fails to run.

After completing the outlined steps, the ToolGroup should look like Figure 7.

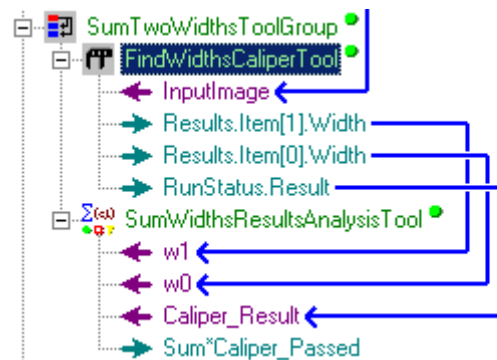


Figure 7: Incorporating a prior Tool's RunStatus.Result into a ResultsAnalysis tool.

The logic of the SWRAT is shown in Figure 8.

SumWidthsResultsAnalysisTool						
Settings Results						
Name	Argument0	Operator	Argument1	Value	Output	
w1				8.99740580453579		
w0				28.0008037140716		
Caliper_Result				Accept		
Caliper_Passed	-1	NotEqual	Caliper_Result	True		☐
sum(w0,w1)	w0	Add	w1	36.9982095186074		☐
Sum*Caliper_Passed	Caliper_Passed	Mul	sum(w0,w1)	36.9982095186074		☒
Output	Reject	If	false	Accept		☐

Figure 8: SWRAT logic to account for FindWidthsCaliperTool may fail.