Contents

[ArcGIS.Desktop.Core.Geoprocessing 1](#_Toc413674994)

[Overview 1](#_Toc413674995)

[How to execute a tool 1](#_Toc413674996)

[Example 1 1](#_Toc413674997)

[Example 2 2](#_Toc413674998)

[Setting geoprocessing environment 2](#_Toc413674999)

[Open a geoprocessing tool dialog. 3](#_Toc413675000)

[Example 1 3](#_Toc413675001)

[Example 2 3](#_Toc413675002)

[Using the delegate GPToolExecuteEventHandler 4](#_Toc413675003)

[Still to do 5](#_Toc413675004)

[ShowMessageBox 5](#_Toc413675005)

The geoprocessing public APIs are exposed through two namespaces in the ArcGIS Pro API Reference Guide.

The core useful methods, enumerators and interfaces are in ArcGIS.Desktop.Core.Geoprocessing namespace. Some properties and methods from the ArcGIS.Desktop.GeoProcessing namespace may also be exposed.

# ArcGIS.Desktop.Core.Geoprocessing

## Overview

ExecuteTool is the main method to be used.

## How to execute a tool

There are two ways to execute a tool – in both cases the method name is ExecuteTool – but the parameter signatures are different. In Pro, the execute method cannot be used in a stand-alone .Net application. All codes are executed from within an Add In, which, after installation, will show up as a button in Addin ribbon of the app.

Example 1: Simplest tool execution

private async Task<IGPResult> ExecuteGetCount()

{

string[] args = new string[] {

@"C:\data\Hydrants.shp"

};

var gp\_result = await Geoprocessing.ExecuteTool("GetCount\_management", args);

if (!gp\_result.IsFailed)

{

int count = Convert.ToInt32(gp\_result.Values[0]);

System.Windows.Forms.MessageBox.Show("Result : " + gp\_result.Values[0]);

}

var messages = gp\_result.Messages;

string msgStr = "";

foreach (IGPMessage msg in messages)

{

msgStr += msg.Text + "\n";

}

System.Windows.Forms.MessageBox.Show(msgStr);

return gp\_result;

}

Example 2: Progress dialog

protected async override void OnClick()

{

await ExecuteBuffer2();

}

public async Task<IGPResult> ExecuteBuffer2()

{

// in ArcGIS.Framework….Threading.Tasks.

ProgressDialog progDlg = new ProgressDialog("Buffer\_management", "Cancel", 100);

progDlg.Show();

// Threading.Tasks.

var progSrc = new CancelableProgressorSource(progDlg);

string[] paramList = new string[3];

paramList[0] = @"C:\data\Hydrants.shp";

paramList[1] = @"C:\temp\testout19.shp";

paramList[2] = "50 meters";

KeyValuePair<string, string>[] env = new KeyValuePair<string, string>[]{

new KeyValuePair<string, string>("extent", "-170.5 0.0 0.5 38.4")

};

var gp\_result = await Geoprocessing.ExecuteTool("Buffer\_analysis", paramList, env, progSrc.Progressor);

progDlg.Hide();

return gp\_result;

}

## Setting geoprocessing environment

The third parameter of ExecuteTool method is a KyeValuePair of strings. You pass pair(s) of values as argument to set the environment:

System.Collections.Generic.KeyValuePair<string, string>[] envs = new System.Collections.Generic.KeyValuePair<string, string>[]

{

new System.Collections.Generic.KeyValuePair<string, string>("Output Workspace", @"c:\temp"),

new System.Collections.Generic.KeyValuePair<string, string>("Extent", "0 0 12 14")

};

## Open a geoprocessing tool dialog.

Example 1: Open a geoprocessing tool dialog with parameter values filled in. User needs to click on ‘Run’ to execute the tool:

public class OpenToolButton : Button

{

protected override void OnClick()

{

OpenGPTool();

}

private void OpenGPTool()

{

string[] paramList = new string[3];

paramList[0] = @"C:\data\Hydrants.shp";

paramList[1] = @"c:\temp\fgdb.gdb\testout1";

paramList[2] = "50 meters";

KeyValuePair<string, string>[] envs = {

new KeyValuePair<string, string>("Output Workspace", @"c:\temp") };

Geoprocessing.OpenToolDialog(@"Buffer\_analysis", paramList, envs);

}

}

Example 2: Open a script tool dialog.

public class OpenToolButton2 : Button

{

protected override void OnClick()

{

OpenScriptTool();

}

private void OpenScriptTool()

{

string[] paramList = new string[3];

paramList[0] = @"C:\data\Hydrants.shp";

paramList[1] = @"c:\temp\testout12.shp";

string toolPath = @"C:\data\ScriptTool.tbx/MyScriptTool";

System.Collections.Generic.KeyValuePair<string, string>[] envs = { new System.Collections.Generic.KeyValuePair<string, string>("Output Workspace", @"c:\temp") };

Geoprocessing.OpenToolDialog(toolPath, paramList, envs);

}

}

## Using the delegate GPToolExecuteEventHandler

Use a long running process (such as Empirical Bayesian Kriging) to show implementation of callback delegate. Collect all output messages while the tool is running. Call cancel\_test3() to stop execution of the process.

static CancellationTokenSource \_cts;

internal async static void test3()

{

string[] args = { "ca\_ozone\_pts", "OZONE", "", "in\_memory\\raster", "4040", "NONE", "100", "1", "100",

"NBRTYPE=StandardCircular RADIUS=310833.272442914 ANGLE=0 NBR\_MAX=15 NBR\_MIN=10 SECTOR\_TYPE=ONE\_SECTOR",

"PREDICTION", "0.5", "EXCEED", "" };

string tool\_path = "ga.EmpiricalBayesianKriging"; // "EmpiricalBayesianKriging\_ga"

System.Diagnostics.Debug.Assert(\_cts == null);// call cancel\_test3()

\_cts = new CancellationTokenSource();

var gp = FrameworkApplication.FindModule("esri\_geoprocessing\_module") as IGeoprocessing;

var t = gp.ExecuteTool(tool\_path, args, null, \_cts.Token,

(event\_name, o) => //implement deligate and handle events

{

switch (event\_name)

{

case "OnValidate": //stop execute if any warnings

if ((o as IGPMessage[]).Any(it => it.Type == GPMessageType.Warning))

\_cts.Cancel();

break;

case "OnProgressMessage": System.Diagnostics.Debug.WriteLine("{0}: {1}", new object[] { event\_name, (string)o }); break;

case "OnProgressPos": System.Diagnostics.Debug.WriteLine("{0}: {1} %", new object[] { event\_name, (int)o }); break;

default: break;

}

});

var ret = await t;

\_cts = null;

}

/// <summary>

/// cancel test3

/// </summary>

internal static void cancel\_test3()

{

if (\_cts != null)

\_cts.Cancel();

\_cts = null;

}

Still to do:

### ShowMessageBox