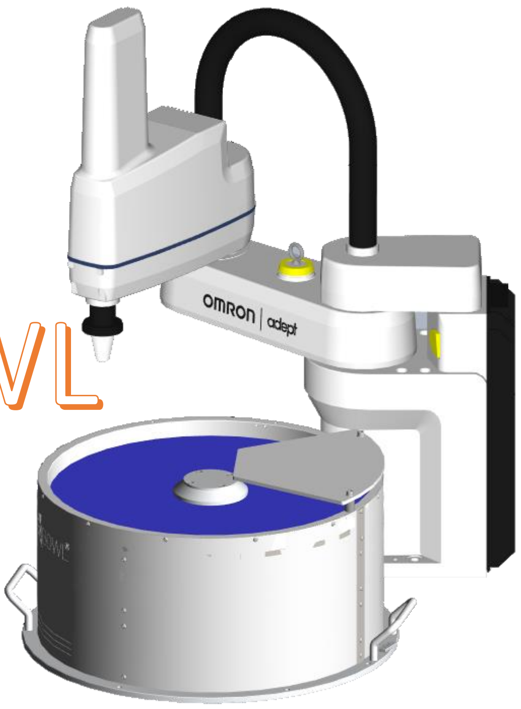
OMRON FLEXIBOWL PLUGIN



This Plugin was developed with the idea of communicating quickly and safely with the flexibowl through Omron robots, using version 4 or version 3 of the Omron Ace software.
The Plugin does not require additional Omron licenses.

ACE

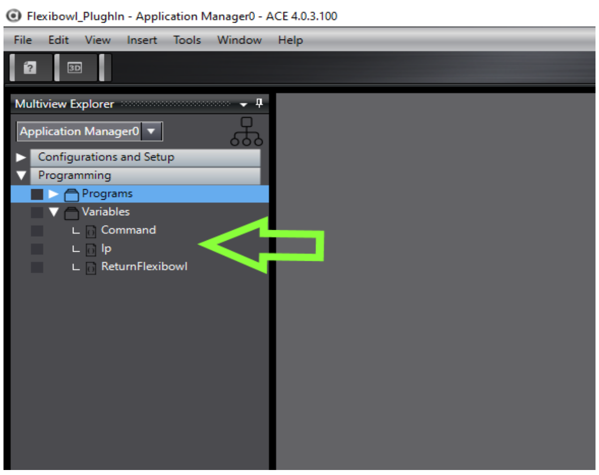
Automation Control Environment

4.0.3.100

(c) 2017-2019 Omron Robotics and Safety Technologies, Inc. All rights reserved.

OMRON

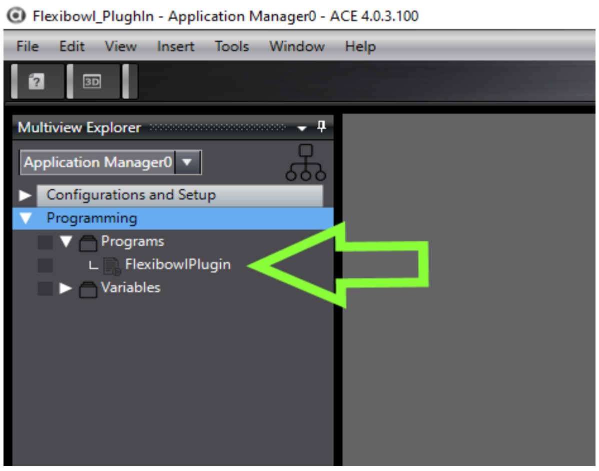
STEP 1:



By selecting the desired *Application Manager* in the *Multiview Explorer*, by right clicking under the *Variables* tree you can add three *string variables* named as follows:

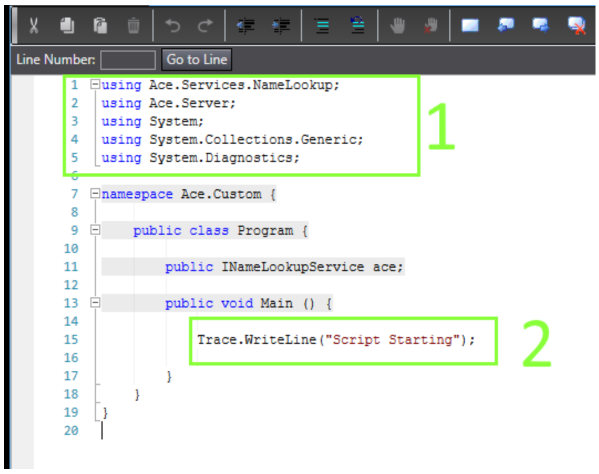
- **Command**
- **Ip**
- **ReturnFlexibowl**

STEP 2:



By again selecting the desired *Application Manager*, a new c# task can be added under the *Programs* tree by right clicking.
This task can be launched from V+ to move the flexibowl.
Rename the task as “Flexibowl Plugin”.

STEP 3:

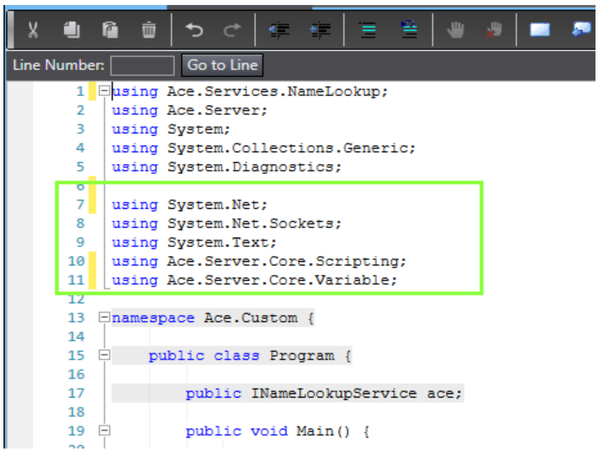


```
using Ace.Services.NameLookup;
using Ace.Server;
using System;
using System.Collections.Generic;
using System.Diagnostics;

namespace Ace.Custom {

    public class Program {

        public INameLookupService ace;

        public void Main () {

            Trace.WriteLine("Script Starting");

        }
    }
}
```

Double click the program just created to edit it.
The declarations to be used are in zone 1, while the body of the script is in part 2.
We will now edit said script.

ACE
versione 4

STEP 4:



Line Number: Go to Line

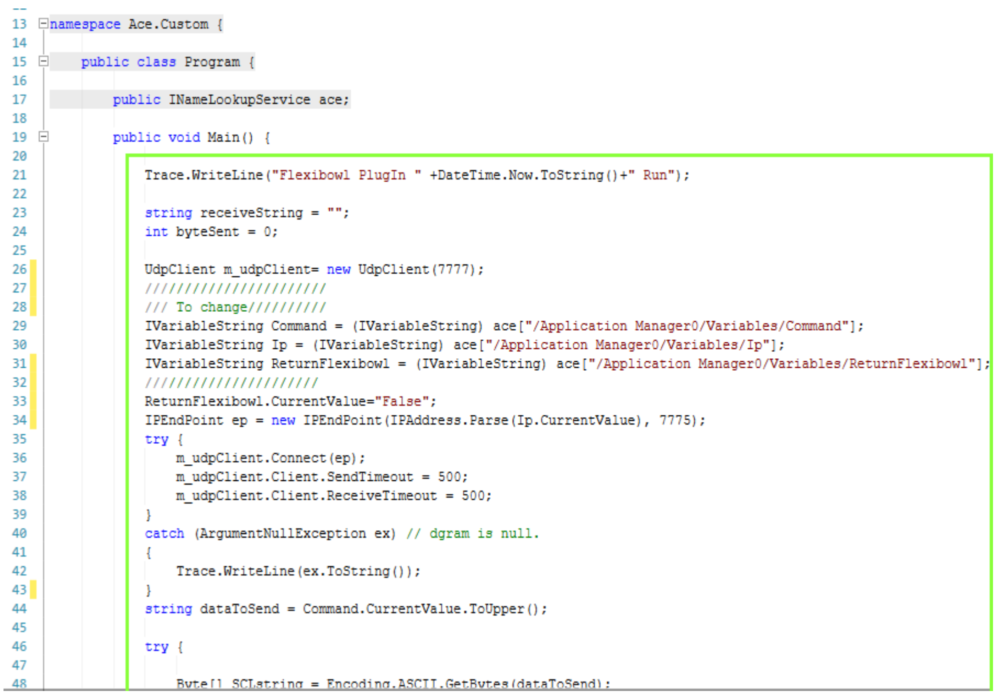
```
using Ace.Services.NameLookup;
using Ace.Server;
using System;
using System.Collections.Generic;
using System.Diagnostics;

using System.Net;
using System.Net.Sockets;
using System.Text;
using Ace.Server.Core.Scripting;
using Ace.Server.Core.Variable;

namespace Ace.Custom {

    public class Program {

        public INameLookupService ace;

        public void Main() {
```

Add the following dependencies in section 1 of the code:

using System.Net;
using System.Net.Sockets;
using System.Text;
using Ace.Server.Core.Scripting;
using Ace.Server.Core.Variable;

STEP 5:

In the section of code 2 instead delete and replace with all the code on the following page. The image below shows a preview of the final result.



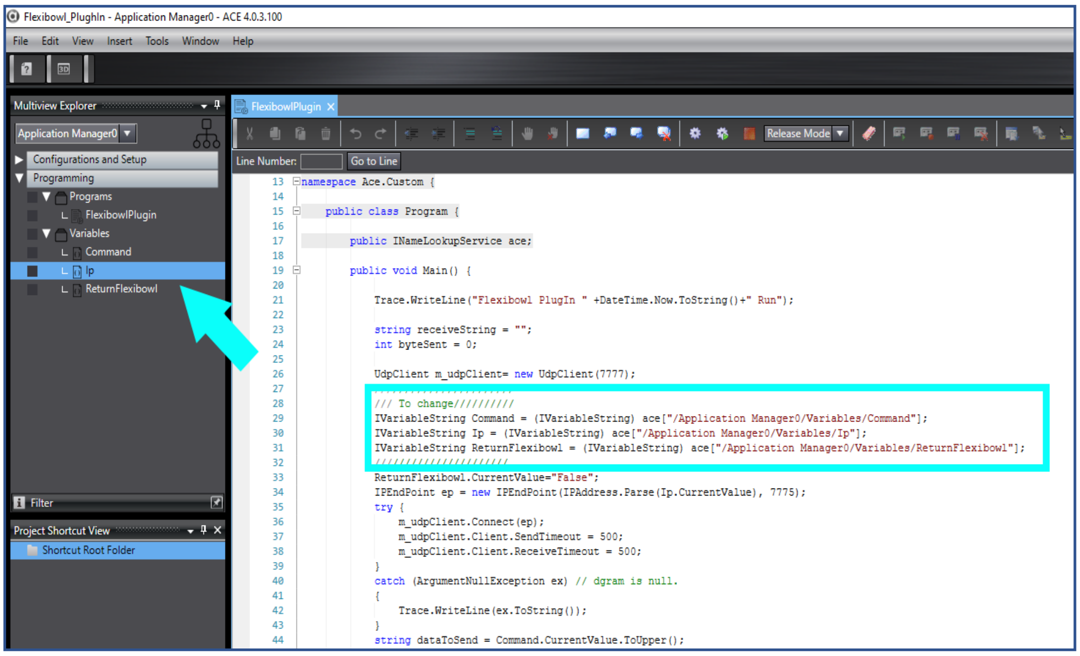
```
namespace Ace.Custom {

    public class Program {

        public INameLookupService ace;

        public void Main() {

            Trace.WriteLine("Flexibowl PlugIn " +DateTime.Now.ToString()+" Run");

            string receiveString = "";
            int byteSent = 0;

            UdpClient m_udpClient= new UdpClient(7777);
            ///////////////////////
            /// To change/////////
            IVariableString Command = (IVariableString) ace["/Application Manager0/Variables/Command"];
            IVariableString Ip = (IVariableString) ace["/Application Manager0/Variables/Ip"];
            IVariableString ReturnFlexibowl = (IVariableString) ace["/Application Manager0/Variables/ReturnFlexibowl"];
            //////////////////////
            ReturnFlexibowl.CurrentValue="False";
            IPEndPoint ep = new IPEndPoint(IPAddress.Parse(Ip.CurrentValue), 7775);
            try {
                m_udpClient.Connect(ep);
                m_udpClient.Client.SendTimeout = 500;
                m_udpClient.Client.ReceiveTimeout = 500;
            }
            catch (ArgumentNullException ex) // dgram is null.
            {
                Trace.WriteLine(ex.ToString());
            }
            string dataToSend = Command.CurrentValue.ToUpper();

            try {

                Byte[] SCLstring = Encoding.ASCII.GetBytes(dataToSend);
```

```
Trace.WriteLine("Flexibowl PlugIn " +DateTime.Now.ToString()+" Run");

string receiveString = "";
int byteSent = 0;

UdpClient m_udpClient= new UdpClient(7777);
///////////////////////
/// To change//////////
IVariableString Command = (IVariableString) ace["/Application Manager0/Variables/Command"];
IVariableString Ip = (IVariableString) ace["/Application Manager0/Variables/Ip"];
IVariableString ReturnFlexibowl = (IVariableString) ace["/Application Manager0/Variables/ReturnFlexibowl"];
///////////////////////
ReturnFlexibowl.CurrentValue="False";
IPEndPoint ep = new IPEndPoint(IPAddress.Parse(Ip.CurrentValue), 7775);

try {
        m_udpClient.Connect(ep);
        m_udpClient.Client.SendTimeout = 500;
        m_udpClient.Client.ReceiveTimeout = 500;
}
catch (ArgumentNullException ex)
{
        Trace.WriteLine(ex.ToString());
}

string dataToSend = Command.CurrentValue.ToUpper();

try {
        Byte[] SCLstring = Encoding.ASCII.GetBytes(dataToSend);
        Byte[] sendBytes = new Byte[SCLstring.Length + 3];
        sendBytes[0] = 0;
        sendBytes[1] = 7;
        System.Array.Copy(SCLstring, 0, sendBytes, 2, SCLstring.Length);
        sendBytes[sendBytes.Length - 1] = 13; // CR
        byteSent = m_udpClient.Send(sendBytes, sendBytes.Length);
        var receivedData = m_udpClient.Receive(ref ep);
        receiveString = Encoding.ASCII.GetString(receivedData);
        if ((receiveString.Contains("%")) && (dataToSend.Contains("Q"))) {
                bool moving = true;
                while (moving == true) {
                        SCLstring = Encoding.ASCII.GetBytes("RS");
                        sendBytes = new Byte[SCLstring.Length + 3];
                        sendBytes[0] = 0;
                        sendBytes[1] = 7;
                        System.Array.Copy(SCLstring, 0, sendBytes, 2, SCLstring.Length);
                        sendBytes[sendBytes.Length - 1] = 13; // CR
                        byteSent = m_udpClient.Send(sendBytes, sendBytes.Length);
                        receivedData = m_udpClient.Receive(ref ep);
                        receiveString = Encoding.ASCII.GetString(receivedData);
                        if (receiveString.Contains("F"))
                                moving = true;
                        else
                                moving = false;
                        System.Threading.Thread.Sleep(20);
                }
                ReturnFlexibowl.CurrentValue = "Done";
        }
        else {
                SCLstring = new Byte[receivedData.Length - 3];
                System.Array.Copy(receivedData, 2, SCLstring, 0, SCLstring.Length);
                receiveString = Encoding.ASCII.GetString(SCLstring);
                ReturnFlexibowl.CurrentValue = receiveString;
        }
        m_udpClient.Dispose();
}
catch (ArgumentNullException ex)
{
        Trace.WriteLine(ex.ToString());
}
```

STEP 6:

After copying and pasting the code, check that the paths of the variables previously created are correct.
To verify this, check the box highlighted in the image to make sure the paths of the three variables are correct.
Select one of the three variables previously created, drag&drop on the page with the code.
In this case you have created a reference to your variable; check the correct path and delete the line created.
Ensure the paths of the three variables in the code are correct.

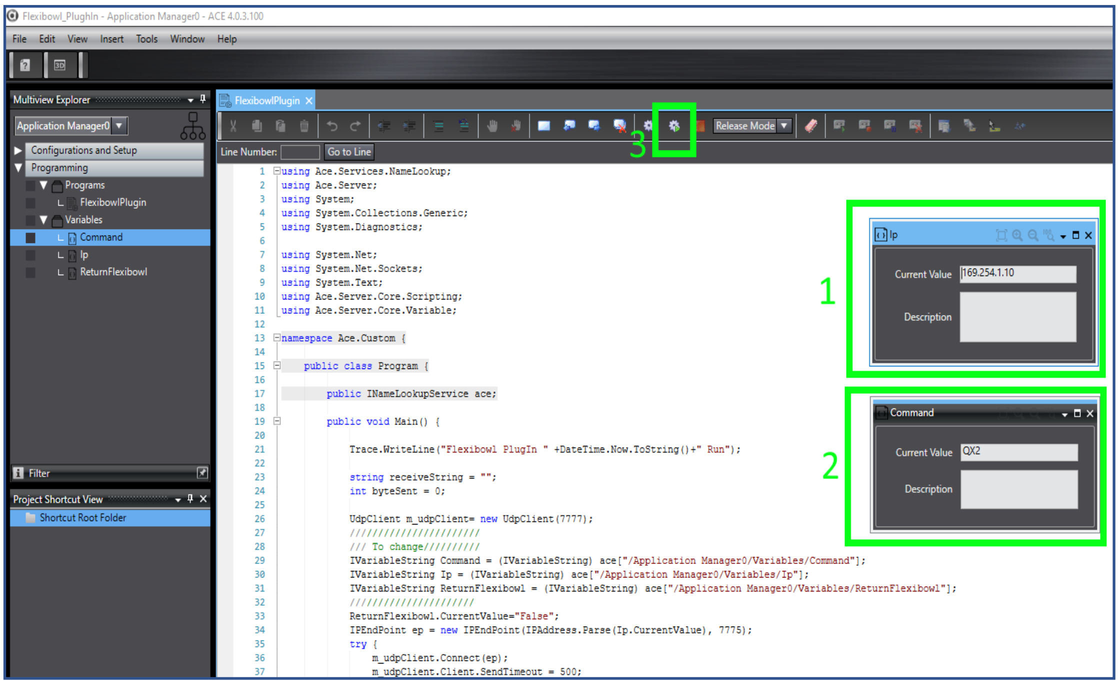
Example:
Original
IVariableString Command = (IVariableString) ace["/Application Manager0/Variables/Command"];
Edited
IVariableString Command = (IVariableString) ace["/Application Manager4/Variables/Command"];



```
namespace Ace.Custom {

    public class Program {

        public INameLookupService ace;

        public void Main() {

            Trace.WriteLine("Flexibowl PlugIn " +DateTime.Now.ToString()+" Run");

            string receiveString = "";
            int byteSent = 0;

            UdpClient m_udpClient= new UdpClient(7777);
            //////////////////////
            /// To change//////////
            IVariableString Command = (IVariableString) ace["/Application Manager0/Variables/Command"];
            IVariableString Ip = (IVariableString) ace["/Application Manager0/Variables/Ip"];
            IVariableString ReturnFlexibowl = (IVariableString) ace["/Application Manager0/Variables/ReturnFlexibowl"];
            //////////////////////
            ReturnFlexibowl.CurrentValue="False";
            IPEndPoint ep = new IPEndPoint(IPAddress.Parse(Ip.CurrentValue), 7775);
            try {
                m_udpClient.Connect(ep);
                m_udpClient.Client.SendTimeout = 500;
                m_udpClient.Client.ReceiveTimeout = 500;
            }
            catch (ArgumentNullException ex) // dgram is null.
            {
                Trace.WriteLine(ex.ToString());
            }
            string dataToSend = Command.CurrentValue.ToUpper();
```

STEP 7:

Once here, the movement of the Flexibowl can be tested.
By setting the Ip in the IP variable (ref. 1) and the command to be run in the Command variable (ref. 2), click the Run button (ref. 3) to send the command to the Flexibowl with the set Ip.



```
using Ace.Services.NameLookup;
using Ace.Server;
using System;
using System.Collections.Generic;
using System.Diagnostics;

using System.Net;
using System.Net.Sockets;
using System.Text;
using Ace.Server.Core.Scripting;
using Ace.Server.Core.Variable;

namespace Ace.Custom {

    public class Program {

        public INameLookupService ace;

        public void Main() {

            Trace.WriteLine("Flexibowl PlugIn " +DateTime.Now.ToString()+" Run");

            string receiveString = "";
            int byteSent = 0;

            UdpClient m_udpClient= new UdpClient(7777);
            ///////////////////////
            /// To change/////////
            IVariableString Command = (IVariableString) ace["/Application Manager0/Variables/Command"];
            IVariableString Ip = (IVariableString) ace["/Application Manager0/Variables/Ip"];
            IVariableString ReturnFlexibowl = (IVariableString) ace["/Application Manager0/Variables/ReturnFlexibowl"];
            //////////////////////
            ReturnFlexibowl.CurrentValue="False";
            IPEndPoint ep = new IPEndPoint(IPAddress.Parse(Ip.CurrentValue), 7775);
            try {
                m_udpClient.Connect(ep);
                m_udpClient.Client.SendTimeout = 500;
```

STEP 8:

We will now see how to set the variables and run the script from V+
Let's create a V+ program with the code on the next page.
Copy the code and check that the paths of the variables are correct, e.g.:

$object = "/Application Manager0/Variables/Ip"

After setting the Ip and the command, by running the V+ script the flexibowl will carry out the command

```
;insert the data
;///////////////////////
$ip="169.254.1.10"
$command="QX3"
;//////////////////////
```

At the moment the Ip, Command and return.flexibow variables in V+ are local (AUTO). To set them from external programs, make these variables Global, therefore not Auto. Running the V+ script will execute the C# script, which will operate the flexibowl

```
.PROGRAM flbplugin()
        AUTO $object, $variable, $ip, $command , $return.flexibow , $method, $args[0]
        AUTO REAL status, is.alive
         ;insert the data
        ;//////////////////////
        $ip="169.254.1.10"
        $command="QX3"
        ;/////////////////////
        ;Set the data on c#
        ;IP
        $object = "/Application Manager0/Variables/Ip"
        $variable = "CurrentValue"
        CALL rm.write.str($object, $variable, 1, $ip, status)
        IF (status < 0) THEN
            TYPE "Unable To Write Value: ", status
            PAUSE
        END
        ;COMMAND
        $object = "/Application Manager0/Variables/Command"
        $variable = "CurrentValue"
        CALL rm.write.str($object, $variable, 1, $command, status)
        IF (status < 0) THEN
            TYPE "Unable To Write Value: ", status
            PAUSE
        END
        ;Execute the c#
        CALL rm.chk.server(is.alive)
        IF (is.alive == FALSE) THEN
            TYPE "Not Communicating"
            PAUSE
        END     ; Execute a script on the server and wait for 3 seconds for it to complete
        $object = "/Application Manager0/Programs/FlexibowlPlugin"
        $method = "Execute"
        CALL rm.execute($object, $method, 0, $args[], 5, status)
        IF (status < 0) THEN
            TYPE "Problem executing script: ", status
            PAUSE
        END
        ;Read the Answer
        $object = "/Application Manager0/Variables/ReturnFlexibowl"
        $variable = "CurrentValue"
        ;Read the answer from flexibowl
        CALL rm.read.str($object, $variable, 1, $return.flexibow, status)
        IF (status < 0) THEN
            TYPE "Unable To Read the Value: ", status
            PAUSE
        END
.END
```

STEP 9:

List of commands and descriptions to be sent to the Flexibowl:

Action	Description
MOVE	Moves the feeder the current parameters.
MOVE-FLIP	Moves the feeder and activates Flip simultaneously
MOVE-BLOW-FLIP	Moves the feeder and activates Flip and blow simultaneously
MOVE-BLOW	Moves the feeder and activates Flip simultaneously
SHAKE	Shakes the feeder with the current parameters
LIGHT ON	Light on
LIGHT OFF	Light off
FLIP	Flip
BLOW	Blow
QUICK_EMPTING	Quick Emptying Option
RESET_ALARM	Reset Alarm and enable the motor

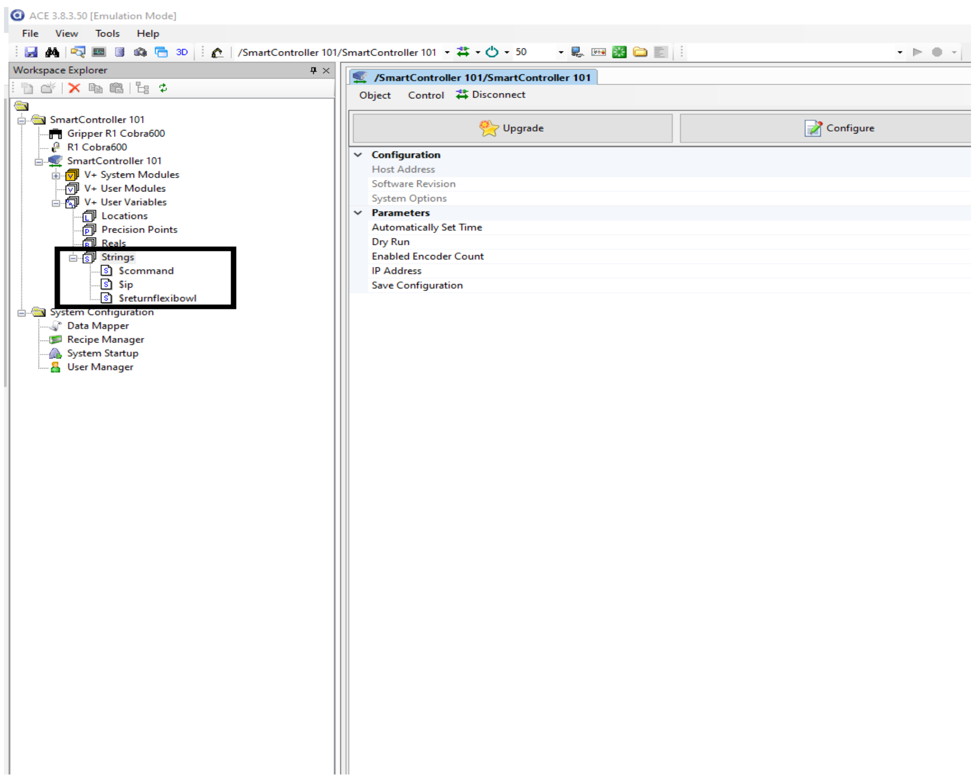
Command	Description
QX2	Move
QX3	Move - Flip
QX4	Move - Blow - Flip
QX5	Move - Blow
QX5	Shake
QX7	Light on
QX8	Light off
QX9	Blow
QX10	Flip
QX11	Quick Emptying Option
QX12	Reset Alarm

ACE versione 3

From here on we will see how to integrate the Flexibowl Plugin into Ace 3.X or earlier versions.

STEP 1:

Create THREE String V+ variables.
-Ip
-Command
-RerturnFlexibowl


ACE 3.8.3.50 [Emulation Mode]

File View Tools Help

/SmartController 101/SmartController 101 50

Workspace Explorer

- SmartController 101
  - Gripper R1 Cobra600
  - R1 Cobra600
  - SmartController 101
    - V+ System Modules
    - V+ User Modules
    - V+ User Variables
      - Locations
      - Precision Points
      - Reals
      - Strings
        - $command
        - $ip
        - $returnflexibowl
- System Configuration
  - Data Mapper
  - Recipe Manager
  - System Startup
  - User Manager

/SmartController 101/SmartController 101

Object Control Disconnect

Upgrade Configure

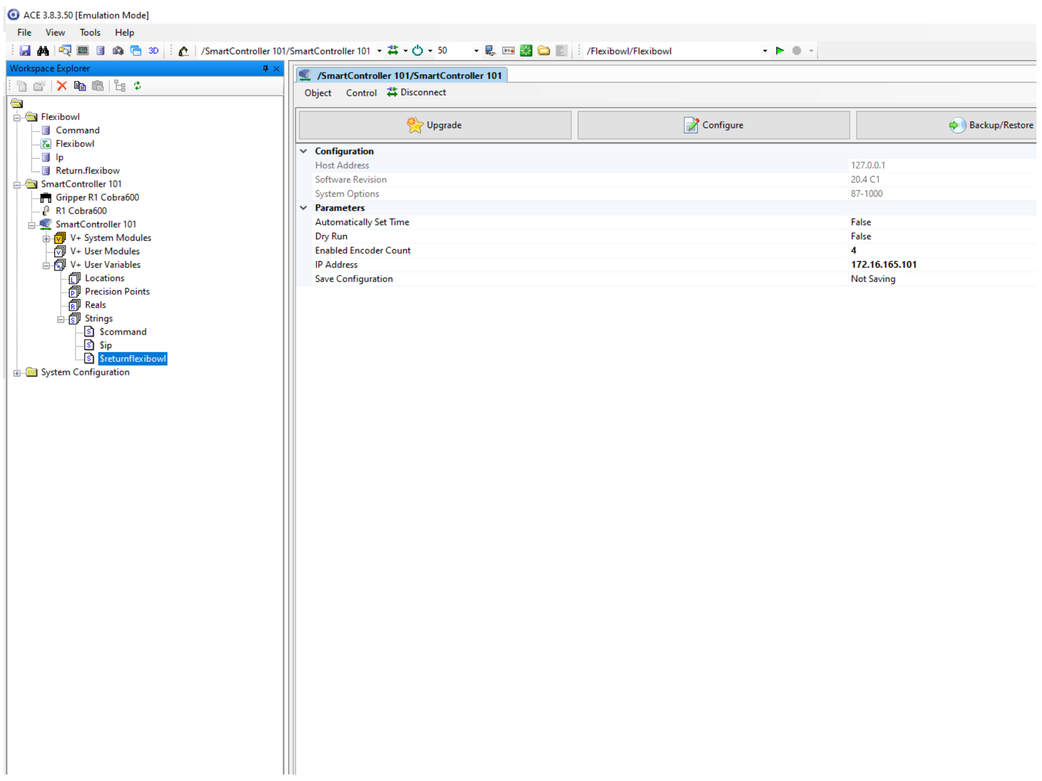
Configuration
- Host Address
- Software Revision
- System Options

Parameters
- Automatically Set Time
- Dry Run
- Enabled Encoder Count
- IP Address
- Save Configuration

ACE
versione 3

STEP 2:

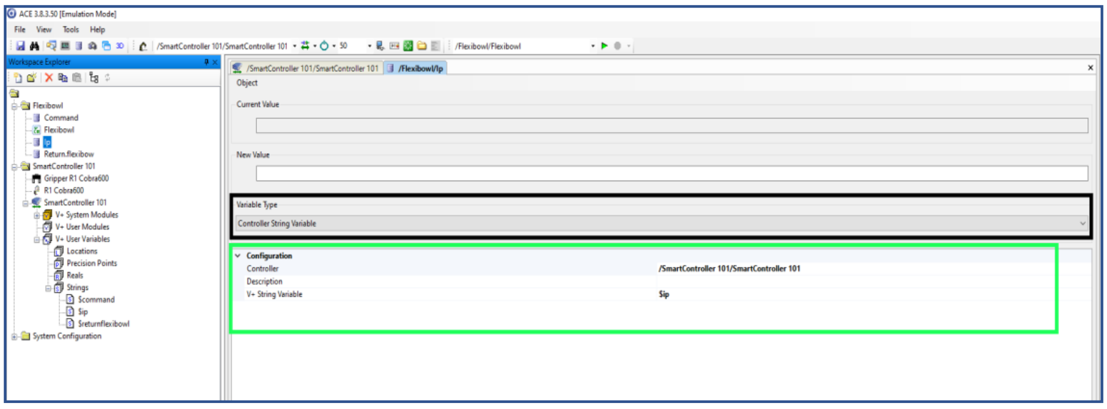
Create a folder in the WorkspaceExplorer and call it Flexibowl.
Now right click the folder just created, ImportWorkspaceFile, and upload the FlexibowlPlugin.awp file provided by us.



Configuration	
Host Address	127.0.0.1
Software Revision	20.4 C1
System Options	87-1000
Parameters	
Automatically Set Time	False
Dry Run	False
Enabled Encoder Count	4
IP Address	172.16.165.101
Save Configuration	Not Saving

STEP 3:

Now the V+ variables need to be indexed with the C# variables.
For example, double click the C# Ip variable. By setting this variable as a *ControllerStringVariable* (black box), it can be associated with our V+ Ip variable (green box). Do this for all three variables.



Variable Type: Controller String Variable

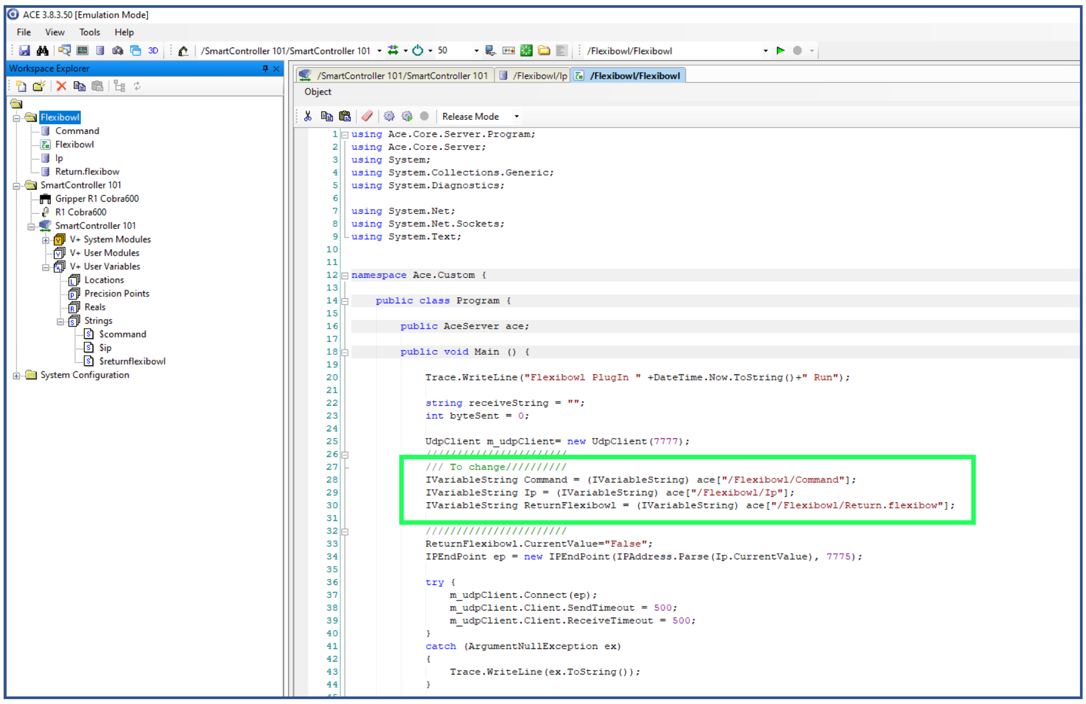
Configuration	
Controller	/SmartController 101/SmartController 101
Description	
V+ String Variable	$ip

ACE
versione 3

STEP 4:

You need to check that the paths of the C# variables are correct.
To verify this, check the box highlighted in the image to make sure the paths of the three variables are correct.
To do this, select one of the three C# variables, drag&drop on the code page and check that the path is correct.
In this case you have created a reference to your variable; check the correct path and delete the line created.
Ensure the paths of the three variables in the code are correct.

Example:
Original
IVariableString Command = (IVariableString) ace["/Application Manager0/Variables/Command"];
Edited
IVariableString Command = (IVariableString) ace["/Application Manager4/Variables/Command"];



```
using Ace.Core.Server.Program;
using Ace.Core.Server;
using System;
using System.Collections.Generic;
using System.Diagnostics;

using System.Net;
using System.Net.Sockets;
using System.Text;


namespace Ace.Custom {

    public class Program {

        public AceServer ace;

        public void Main () {

            Trace.WriteLine("Flexibowl PlugIn " +DateTime.Now.ToString()+" Run");

            string receiveString = "";
            int byteSent = 0;

            UdpClient m_udpClient= new UdpClient(7777);
            ///////////////////////
            /// To change//////////
            IVariableString Command = (IVariableString) ace["/Flexibowl/Command"];
            IVariableString Ip = (IVariableString) ace["/Flexibowl/Ip"];
            IVariableString ReturnFlexibowl = (IVariableString) ace["/Flexibowl/Return.flexibow"];

            ///////////////////////
            ReturnFlexibowl.CurrentValue="False";
            IPEndPoint ep = new IPEndPoint(IPAddress.Parse(Ip.CurrentValue), 7775);

            try {
                m_udpClient.Connect(ep);
                m_udpClient.Client.SendTimeout = 500;
                m_udpClient.Client.ReceiveTimeout = 500;
            }
            catch (ArgumentNullException ex)
            {
                Trace.WriteLine(ex.ToString());
            }
```

STEP 5:

We will now see how to set the variables and run the script from V+
Let's create a V+ program with the code on the next page.
Copy the code and check that the paths of the variables are correct, e.g.:

$object = "/Application Manager0/Variables/Ip"

After setting the Ip and the command, by running the V+ script the flexibowl will carry out the command

;insert the data
;///////////////////////
$ip="169.254.1.10"
$command="QX3"

```
.PROGRAM flexibowlplg()

        AUTO $object, $variable, $ip, $command , $return.flexibow
, $method, $args[0]
        AUTO REAL status, is.alive

        ;insert the data
        ;////////////////////////
        $ip="169.254.1.10"
        $command="QX3"
        ;///////////////////////

        ;Execute the c#
        CALL rm.chk.server(is.alive)
        IF (is.alive == FALSE) THEN
            TYPE "Not Communicating"
            PAUSE
        END     ; Execute a script on the server and wait for 3
seconds for it to complete
        $object = "/Flexibowl/Flexibowl"
        $method = "Execute"
        CALL rm.execute($object, $method, 0, $args[], 5, status)
        IF (status < 0) THEN
            TYPE "Problem executing script: ", status
            PAUSE
        END

        ;the Answer
        ;$returnflexibowl

.END
```

At the moment the Ip, Command and return.flexibow variables in V+ are local (AUTO). To set them from external programs, make these variables Global, therefore not Auto.
Running the V+ script will execute the C# script, which will operate the flexibowl