BenchCel ActiveX

User Guide

This guide contains the following topics:

About this guide

What this guide covers

This guide provides integrators with the ActiveX control information required to integrate another company’s lab automation device with the BenchCel Microplate Handler.

For instructions on how to set up and use the BenchCel Microplate Handler, see the *BenchCel Microplate Handler User Guide*.

Accessing Agilent Automation Solutions user guides

You can search the online knowledge base or download the latest version of any PDF file at www.agilent.com/chem/askb.

Safety information for the devices appears in the corresponding device user guide. You can also search the knowledge base or the PDF files for safety information.

Software version

This guide documents BenchCel ActiveX version 13.1.0.1366 and earlier versions.

Reporting problems

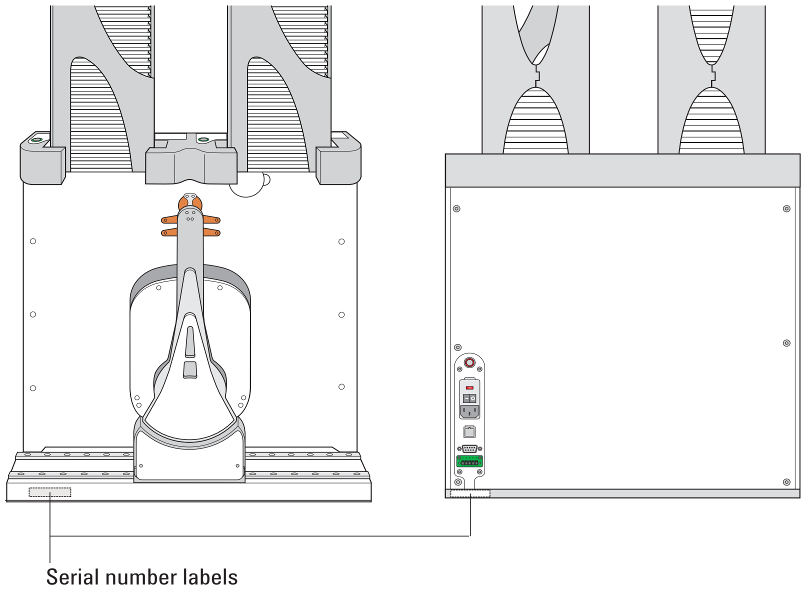
Contacting Technical Support

If you find a problem with the BenchCel Microplate Handler, contact Agilent Technical Support. For contact information, go to https://www.agilent.com/en/contact-us/page.

Reporting hardware problems

When contacting Agilent Technologies, make sure you have the serial number of the device ready. You can locate the serial number on the front and back of the BenchCel device.

Figure BenchCel device front and back views



Reporting software problems

When you contact Technical Support, make sure you provide the following:

- Short description of the problem
- Relevant software version number (for example, automation control software, diagnostics software, ActiveX control software, and firmware)
- Error message text (or screen capture of the error message dialog box)
- Relevant files, such as log files

Reporting user guide problems

If you find a problem with this user guide or have suggestions for improvement, send your comments in an email to documentation.automation@agilent.com.

About ActiveX controls

What is the BenchCel ActiveX control

The BenchCel ActiveX control is the software component that allows the BenchCel Microplate Handler to interact with a third-party lab automation system. The ActiveX has been verified to work with both Visual Studio 6 and Visual Studio .NET (v 7.1).

How the BenchCel ActiveX control is used

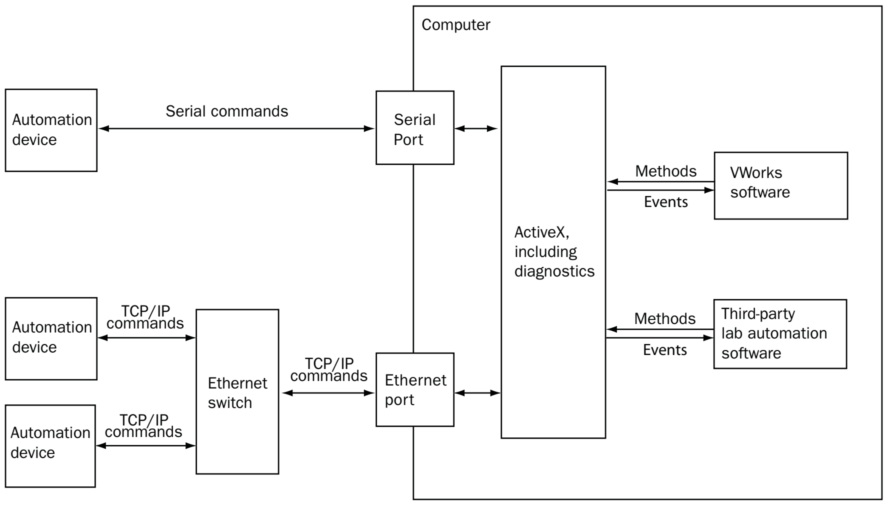
In an Agilent automation system, the VWorks software is already configured to interface with the BenchCel Microplate Handler. The operator can control the device using the software.

In a third-party lab automation system, you use ActiveX to enable the third-party software to interface with the BenchCel Microplate Handler. Each ActiveX control consists of a collection of the following:

- *Methods*. Functions that can be called to invoke individual operations
- *Properties*. Attributes or features of the BenchCel ActiveX control
- *Events*. Notifications that methods have completed or resulted in errors

When integrating the BenchCel Microplate Handler in a lab automation system, you need to know the available methods and properties for the ActiveX control.

The following figure illustrates the use of the BenchCel ActiveX control in a lab automation system environment. Actions you perform are conducted through ActiveX methods. System responses are relayed back through ActiveX events.



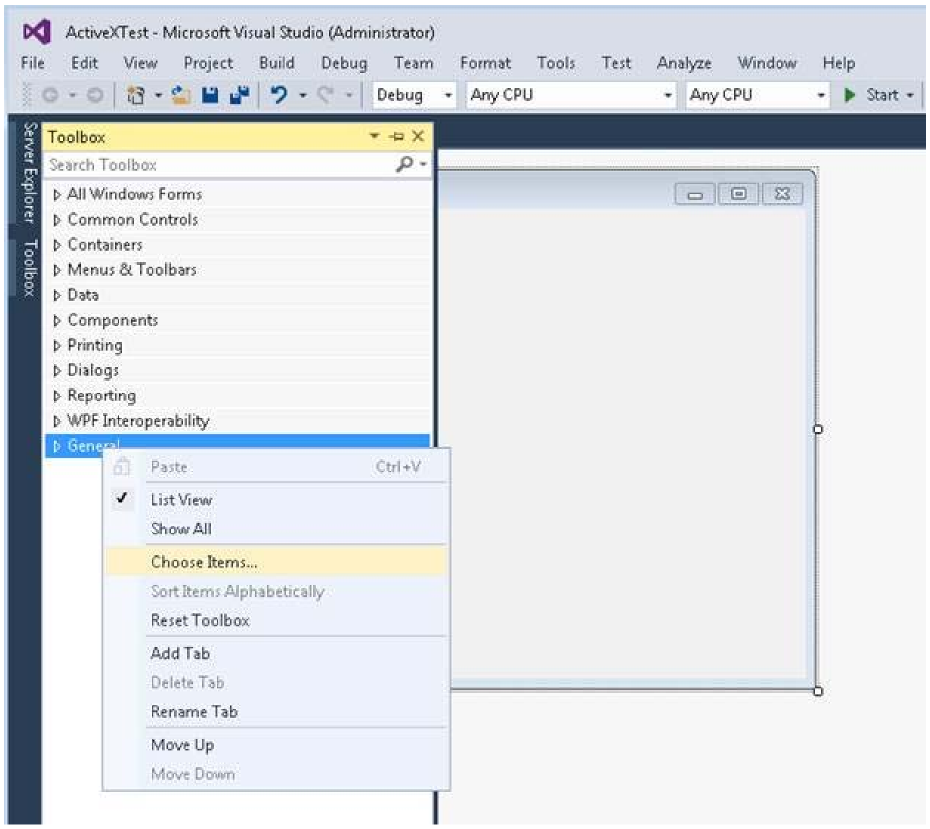
Using the ActiveX control

The Agilent ActiveX control is a Windows-based control. To use the control, you place it on a form or window. The form can be visible or hidden. The following procedure provides an example of how to set up the control using Microsoft Visual Studio.

To set up the ActiveX control using Visual Studio:

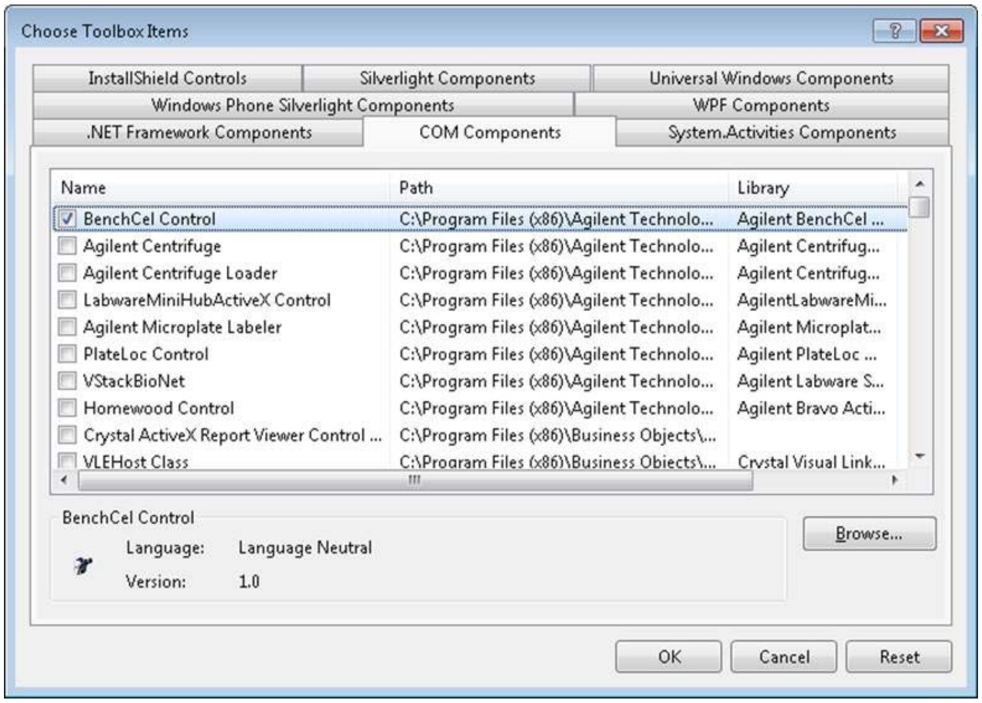
1 In Visual Studio, open the project form in the **Forms Designer**.

2 Right-click in the **Toolbox** area, and select **Choose Items** from the shortcut menu.



3 In the **Choose Toolbox Items** window, click the **COM Components** tab, and select the check box of the desired control.

For the **BenchCel ActiveX,** select the **BenchCel Control** check box.

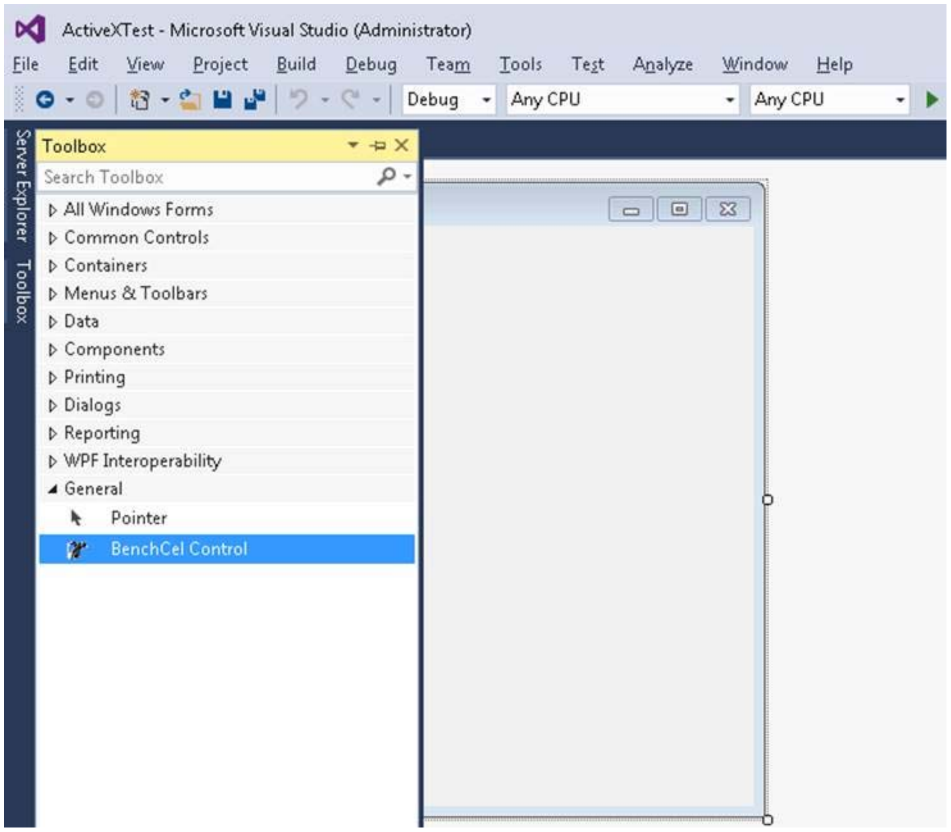

Choose Toolbox Items

Name	Path	Library
✓ BenchCel Control	C:\Program Files (x86)\Agilent Technolo...	Agilent BenchCel ...
Agilent Centrifuge	C:\Program Files (x86)\Agilent Technolo...	Agilent Centrifug...
Agilent Centrifuge Loader	C:\Program Files (x86)\Agilent Technolo...	Agilent Centrifug...
LabwareMiniHubActiveX Control	C:\Program Files (x86)\Agilent Technolo...	AgilentLabwareMi...
Agilent Microplate Labeler	C:\Program Files (x86)\Agilent Technolo...	Agilent Microplat...
PlateLoc Control	C:\Program Files (x86)\Agilent Technolo...	Agilent PlateLoc ...
VStackBioNet	C:\Program Files (x86)\Agilent Technolo...	Agilent Labware S...
Homewood Control	C:\Program Files (x86)\Agilent Technolo...	Agilent Bravo Acti...
Crystal ActiveX Report Viewer Control ...	C:\Program Files (x86)\Business Objects\...	
VLEHost Class	C:\Program Files (x86)\Business Objects\...	Crystal Visual Link...

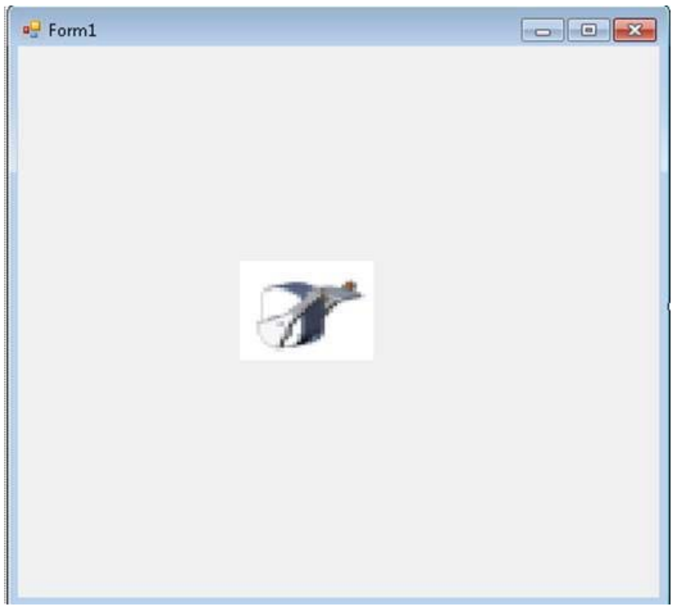
BenchCel Control
Language: Language Neutral
Version: 1.0

InstallShield Controls
Silverlight Components
Universal Windows Components
Windows Phone Silverlight Components
WPF Components
.NET Framework Components
COM Components
System.Activities Components
Browse...
OK
Cancel
Reset

4 Click **OK**. The selected control appears in the Toolbox area.



5 Drag the control onto your form.



Related topics

For information about...	See...
BenchCel ActiveX properties	"Properties" on page 8
BenchCel ActiveX methods	"Methods" on page 10
BenchCel ActiveX events	"Events" on page 25
Reporting problems	"Reporting problems" on page 3

Properties

IPictureDisp* ControlPicture

Description

Read-only property that the client can use to get an icon to represent the ActiveX control.

This example paints a BenchCel bitmap over a button.

Visual C++ example

/*The CPicture class is imported into your project when the ActiveX is installed*/

```
CButton button;
```

//Create a button

```
CPicture BenchCelPic;
BenchCelPic = m_BenchCel.GetControlPicture();
```

//Retrieve the picture

```
button.SetBitmap((HBITMAP)BenchCelPic.GetHandle());
```

/*Paint the bitmap onto the button*/

Visual Basic example

'Assume that there is a button

'named Command1 on the

'current form. You must set

'the style property of

'Command1 to Graphical

```
Command1.Picture = BenchCel.ControlPicture
```

SHORT Speed

Description

Property to specify how fast the BenchCel device should move. 0 = slow, 1 = medium, 2 = fast. This property should not be changed during an operation. Setting this property to an invalid value will have no effect (call will be ignored).

Visual C++ example

//Set the speed to fast

```
m_BenchCel.speed = 2;
```

Visual Basic example

'Set the speed to fast

```
BenchCel.Speed = 2
```

BOOL Blocking

Description

Specifies whether the ActiveX should block during an execution of a command.

If true, commands, such as PickAndPlace, will not return until the action completes or an error occurs.

If false, the command will return immediately without waiting for the action to complete and invoke an event to indicate successful completion of the command. Errors will be indicated through one of two means; 1) the return value might not be S_OK (0), in this case, no event will be fired; 2) an error event is fired. When an error occurs, the ActiveX expects a call to Abort, Retry or Ignore. ShowDiagsDialog can be called to allow the user to exercise specific diagnostic/corrective functions, but when the main execution resumes, a call to Abort, Retry or Ignore is necessary to continue the operation.

Visual C++ example

//Set the BenchCel device to block until the command completes

```
m_BenchCel.Blocking =TRUE;
```

Visual Basic example

Set the BenchCel device to block until the command completes

```
m_BenchCel.Blocking =1;
```

Related topics

For information about...	See...
BenchCel ActiveX methods	“Methods” on page 10
BenchCel ActiveX events	“Events” on page 25
Overview of the ActiveX controls	“About ActiveX controls” on page 4
Reporting problems	“Reporting problems” on page 3

Methods

Abort

```
LONG Abort()
```

Description
Method to clear an error and state information.

Parameters
None

Returns
S_OK if success; other value otherwise

Visual C++ example
```
m_BenchCel.Abort();
```

Visual Basic example
```
BenchCel.Abort
```

AboutBox

```
void AboutBox()
```

Description
Shows a small window that indicates some version information.

Parameters
None

Returns
None

Visual C++ example
```
BenchCel.AboutBox()
```

Visual Basic example
```
m_BenchCel.AboutBox();
```

Close

```
void Close()
```

Description
Method to disconnect from the BenchCel device.

Parameters
None

Returns
None

Visual C++ example

```
m_BenchCel.Close();
```

Visual Basic example

```
BenchCel.Close
```

Delid

```
LONG Delid(BSTR DelidFrom, BSTR DelidTo, LONG
nRetractionCode)
```

Description

Method used to remove a lid from a microplate. You will need to specify where the microplate is located and where to place the lid once it is removed from the microplate.

Parameters

Argument Type	Argument Name	Range	Description
BSTR	DelidFrom	Available teachpoints	Name of teachpoint where the microplate with the lid is located.
BSTR	DelidTo	Available teachpoints	*Optional*. Name of teachpoint to place the lid after it has been removed from the microplate. The BenchCel robot will hold onto the lid if the DelidTo parameter is empty.
LONG	nRetraction Code	0–3	How to position the robot arms after delidding the microplate, where 0– Do nothing 1– Retract normally 2– Not supported 3– Not supported

Returns

S_OK if successful; other value if there was an error.

EnumerateProfiles

```
VARIANT EnumerateProfiles()
```

Description

Method to retrieve a list of defined profiles. The strings in this array are the options that should be used for Initialize.

Parameters

None

Returns

An array of profile names.

Visual C++ example

```
VARIANT vPRofiles = m_BenchCel.EnumerateProfiles();
SAFEARRAY *psa = vProfiles.parray;
BSTR* bstrArray;
if
(FAILED(SafeArrayAccessData(psa,reinterpret_cast<void**>(
&bstrArray))))
{
VariantClear(&vProfiles);
return;
}
for (ULONG i = 0; i < psa->rgsabound[0].cElements; i++)
{
MessageBox(CString(bstrArray[i])));
}
SafeArrayUnaccessData(psa); VaraintClear(&vProfiles);
```

Visual Basic example

```
profileNames = BenchCel.EnumerateProfiles()
For i = LBound(profileNames) To UBound(profileNames)
MsgBox profileNames(i)
Next
```

GetFirmwareVersion

```
BSTR GetFirmwareVersion()
```

Description

Method to programmatically retrieve firmware version of the device.

Parameters

None

Returns

A version string.

Visual C++ example

```
CString strFirmVer = m_BenchCel.GetFirmwareVersion();
```

Visual Basic example

```
Version = BenchCel.GetFirmwareVersion()
```

GetLabwareNames

```
VARIANT GetLabwareNames()
```

Description

Method to retrieve a list of defined labware. The strings in this array are the options that should be used for SetLabware.

Parameters

None

Returns

An array of labware names.

Visual C++ example

```
VARIANT vLabware = m_BenchCel.GetLabwareNames();
SAFEARRAY *psa = vLabware.parray;
if
(FAILED(SafeArrayAccessData(psa, reinterpret_cast<void**>(
&bstrArray))))
{
VariantClear(&vLabware);
return;
}
for (ULONG i = 0; i < psa->rgsabound[0].cElements; i++)
{
MessageBox(CString(bstrArray[i])));
}
SafeArrayUnaccessData(psa); VariantCLear(&vLabware);
```

Visual Basic example

```
LabwareNames = BenchCel.GetLabwareNames
For i = LBound(labwareNames) To UBound(labwareNames)
MsgBox labwareNames(i)
Next
```

GetLastError

```
BSTR GetLastError()
```

Description

Method to retrieve a text message explaining the last error. This method can be called in blocking mode, after a command returns with a failure code, or in non-blocking mode, after the Error event has been fired.

Parameters

None

Returns

An error string.

Visual C++ example

```
strError = m_BenchCel.GetLastError();
```

Visual Basic example

```
strError = BenchCel.GetLastError()
```

GetStackCount

```
LONG GetStackCount(LONG *pCount)
```

Description

Method to retrieve the number of stacks on the BenchCel device. This method must be called after a successful connection in order for it to indicate the current number.

Parameters

Argument Type	Argument Name	Range	Description
LONG*	pCount	Valid pointer to receive the stack count	If successful, the value pointed to by pCount should indicate the number of stacks the device has

Returns

S_OK if successful; other value otherwise.

Visual C++ example

```
    lResult = m_BenchCel.GetStackCount(&numStacks);
```

Visual Basic example

```
    lResult = BenchCel.GetStackCount(numStacks)
```

GetTeachpointNames

```
VARIANT GetTeachpointNames()
```

Description

Method to retrieve the teachpoints known to the device. This method must be called after initialization is complete and it returns an array of available teachpoints, including the stackers.

Parameters

None

Returns

A safe array of teachpoint names.

Visual C++ example

```
VARIANT vTeachpoints = m_BenchCel.GetTeachpointNames();
SAFEARRAY *psa = vTeachpoints.parray;
BSTR* bstrArray;
if
(FAILED(SafeArrayAccessData(psa,reinterpret_cast<void**>(
&bstrArray))))
{
VariantClear(&vTeachpoints);
return;
}
for (ULONG i = 0; i < psa->rgsabound[0].cElements; i++)
{
MessageBox(CString(bstrAdday[i])));
}
SafeArrayUnaccessData(psa);VariantClear(&vTeachpoints);
```

Visual Basic example

```
TeachpointNames = BenchCel.GetTeachpointNames
For i= LBound(teachpointNames) To UBound(teachpointNames)
MsgBox teachpointNames(i)
Next
```

GetVersion

```
BSTR GetVersion()
```

Description

Method to programmatically retrieve the version of the ActiveX.

Parameters

None

Returns

A version string.

Visual C++ example

```
CString strVersion = m_BenchCel.GetVersion();
```

Visual Basic example

```
Version = BenchCel.GetVersion()
```

Ignore

```
LONG Ignore()
```

Description

Method to ignore the previously issued error. This is not a recommended course of action, as the errors are issued for a reason. However, ignoring some errors, such as “Plate is rotated”, can be appropriate if the operator understands the implications.

Parameters

None

Returns

S_OK if success; other value otherwise.

Visual C++ example

```
m_BenchCel.Ignore();
```

Visual Basic example

```
BenchCel.Ignore
```

Initialize

```
LONG Initialize(BSTR Profile)
```

Description

Method to connect to the BenchCel device. A BenchCel profile specifies how to connect to the device (serial or Ethernet; if Ethernet, which device on the network and if serial, which port to use) and which teachpoint file to use. If this is called in non-blocking mode, the client application should wait for InitializeComplete before calling other methods. This method should be called before most other methods.

Parameters

Argument Type	Argument Name	Range	Description
BSTR	Profile	Valid profile name	The name of the profile to be used for initialization

Returns

S_OK (0) on success; other value otherwise.

Visual C++ example

```
LONG1Result =
m_BenchCel.Initialize("ethernet");
```

Visual Basic example

```
LONG1Result =
BenchCel.Initialize("ethernet")
```

IsConnected

```
LONG IsConnected()
```

Description

Method used to check whether a connection to the BenchCel device is established. The BenchCel device is ready to process commands from the BenchCel Active X driver when a connection has been established (using the Initialize() method).

Parameters

None

Returns

1 if there is a connection and 0 if disconnected.

IsPlatePresent

```
LONG IsPlatePresent(SHORT sStack, [in, out] VARIANT_BOOL*
pPresent
```

Description

Method to test whether a stack has a microplate and is loaded. If the stack is not loaded, the result returned through pPresent will not be meaningful. The stack number is 0-based. This method should be called after a successful connection.

Parameters

Argument Type	Argument Name	Range	Description
SHORT	sStack	0 to n-1, where n is the number of stacks	Which stack to check
VARIANT_ BOOL*	pLoaded	Valid pointer to receive whether a microplate is present	On a successful call, the value pointed to by pPresent should indicate whether the stack is loaded and has a microplate available for downstacking

Returns

S_OK if successful, other value otherwise.

Visual C++ example

```
1Result = m_Benchcel.IsPlatePresent(1,&bPlatePresent);
```

Visual Basic example

```
1Result = BenchCel.IsPlatePresent(1,bPlatePresent)
```

IsStackLoaded

```
LONG IsStackLoaded(SHORT sStack, [in, out] VARIANT_BOOL*
pLoaded
```

Description

Method to test whether a stack has been loaded. The stack number is 0-based. This method should be called after a successful connection.

Parameters

Argument Type	Argument Name	Range	Description
SHORT	sStack	0 to n-1, where n is the number of stacks	Which stack to check
VARIANT_ BOOL*	pLoaded	Valid pointer to receive whether or not the stack is loaded	On a successful call, the value pointed to by pLoaded should indicate whether the stack is loaded or not

Returns

S_OK if successful; other value otherwise.

Visual C++ example

```
lResult = m_Benchcel.IsStackLoaded(1,&bStackLoaded);
```

Visual Basic example

```
lResult = BenchCel.IsStackLoaded(1,bStackLoaded)
```

LoadStack

```
LONG LoadStack(SHORT sStack)
```

Description

Method to release a stack. To downstack from or upstack to a stack, the stack must be loaded. A loaded stack is locked into the stacker head and cannot be freely taken from the device. The stack number is 0-based.

Parameters

Argument Type	Argument Name	Range	Description
SHORT	sStack	0 to n-1, where n is the number of stacks	The stack to be loaded

Returns

S_OK if successful; other value otherwise.

Visual C++ example

```
lResult = m_BenchCel.LoadStack(0);
```

Visual Basic example

```
lResult = BenchCel.LoadStack(0)
```

MoveToHomePosition

```
LONG MoveToHomePosition()
```

Description

Method to move the device to the origin. This method is not commonly used.

Parameters

None

Returns

S_OK if successful; other value otherwise.

Visual C++ example

```
    1Result = BenchCel.MoveToHomePosition();
```

Visual Basic example

```
    1Result = BenchCel.MoveToHomePosition()
```

OpenClamp

```
LONG OpenClamp(SHORT sStack)
```

Description

Method used to open the stacker grippers of a given stack.

Parameters

Argument Type	Argument Name	Range	Description
SHORT	sStack	0 – (number of Stacks – 1)	Specify which stack’s gripper to open

Returns

S_OK if successful; other value if there was an error.

Pause

```
LONG Pause()
```

Description

Disables the motors on the BenchCel device and pauses the motion.

See also the Unpause method.

IMPORTANT After the motors are disabled, the robot head and arms might have momentum and continue to move until they come to the end of the *x*-axis, *z*-axis, or *theta*-axis, or until they bump into an obstacle.

Parameters

None

Returns

S_OK if successful; other value if there was an error.

PickAndPlace

```
LONG PickAndPlace(BSTR PickFrom, BSTR PlaceTo, VARIANT_BOOL
bLidded, LONG nRetractionCode)
```

Description

Method to transfer a microplate. Stacker locations are called “Stacker 1”, “Stacker 2”, etc. Downstacking can be specified by using a stacker location for PickFrom and upstacking can be specified by using a stacker location for PlaceTo. bLidded indicates whether the robot should treat the microplate as if it has a lid. nRetractionCode should be 3 (reserved for future options).

Parameters

Argument Type	Argument Name	Range	Description
BSTR	PickFrom	Valid teachpoint name	Destination to pick from
BSTR	PlaceTo	Valid teachpoint name	Destination to place to
VARIANT_ BOOL	bLidded	VARIANT_TRUE, VARIANT_FALSE	Whether the microplate is lidded
LONG	nRetractioCode	0–3	0– Do nothing 1– Retract normally 2– Not supported 3– Not supported

Returns

S_OK if success; other value otherwise.

Visual C++ example

```
1Result = m_BenchCel.PickAndPlace("Stacker 1", "PlateLoc",
FALSE, 2)
```

Visual Basic example

```
1Result = BenchCel.PickAndPlace("Stacker 1", "PlateLoc",
FALSE, 2)
```

ProtocolStart

```
LONG ProtocolStart()
```

Description

Method to be called at the beginning of a run. The device is not expected to move.

Parameters

None

Returns

S_OK on success; other value on failure.

Visual C++ example

```
    1Result = m_BenchCel.ProtocolStart();
```

Visual Basic example

```
    1Result = BenchCel.ProtocolStart()
```

ProtocolFinish

```
LONG ProtocolFinish()
```

Description

Method to be called at the end of a run. The device might home during this call.

Parameters

None

Returns

S_OK if success; other value otherwise.

Visual C++ example

```
    1Result = m_BenchCel.ProtocolFinish();
```

Visual Basic example

```
    1Result = BenchCel.ProtocolFinish()
```

ReleaseStack

```
LONG ReleaseStack(SHORT sStack)
```

Description

Method to release a stack. A released stack can be freely taken from the device for the loading or unloading of microplates. However, the BenchCel Microplate Handler cannot downstack from or upstack to a released stack. The stack number is 0-based.

Note: This method can also be used to perform the close clamp function.

Parameters

Argument Type	Argument Name	Range	Description
SHORT	sStack	0 to n-1, where n is the number of stacks	The stack to be released

Returns

S_OK if successful, other value otherwise.

Visual C++ example

```
lResult = m_BenchCel.ReleaseStack(0);
```

Visual Basic example

```
lResult = BenchCel.ReleaseStack(0)
```

Relid

```
LONG Relid(BSTR RelidFrom, BSTR RelidTo, LONG
nRetractionCode)
```

Description

Method used to put a lid on a microplate. You will need to specify where the lid is located and where the microplate is located. If the RelidFrom argument is blank, it is expected that the robot is holding the lid.

Parameters

Argument Type	Argument Name	Range	Description
BSTR	RelidFrom	Available teachpoints or blank string	Name of teachpoint where the lid is located
BSTR	RelidTo	Available teachpoints	Name of teachpoint where microplate that is getting the lid is located
LONG	nRetraction Code	0–3	How to position the robot arms after relidding the microplate, where 0– Do nothing 1– Retract normally 2– Not supported 3– Not supported

Returns

S_OK if successful; other value if there was an error.

Retry

```
LONG Retry()
```

Description

Method to retry an action after an error occurred. For example, if there is insufficient air pressure during a LoadStack operation, the application can call Retry after the air pressure has been increased.

Parameters

None

Returns

S_OK if success; other value otherwise.

Visual C++ example

```
m_BenchCel.Retry();
```

Visual Basic example

```
BenchCel.Retry
```

SetLabware

```
LONG SetLabware(BSTR bstrLabware)
```

Description

Method to set the labware to use. The selection will be in effect for all operations until a different labware is set. If diagnostics are shown and the user selects a different labware, the original labware will be restored when the diagnostics window is closed. This method should not be called when any movement is in progress.

Parameters

Argument Type	Argument Name	Range	Description
BSTR	bstrLabware	Valid labware name	Labware to be used for subsequent operations

Returns

S_OK if successful; other value if there was an error.

Visual C++ example

```
1Result = m_BenchCel.SetLabware("MyPlateType");
```

Visual Basic example

```
1Result = BenchCel.SetLabware("MyPlateType")
```

ShowDiagsDialog

```
LONG ShowDiagsDialog(BOOL bModal, SHORT iSecurityLevel)
```

Description

Method to show the graphical diagnostics menu that allows the user to troubleshoot and correct problems. This method can be called before Initialize to create a profile.

Parameters

Argument Type	Argument Name	Range	Description
BOOL	bModal	TRUE,FALSE	Whether the diagnostics should be shown modally

Argument Type	Argument Name	Range	Description
SHORT	iSecurityLevel	0-3	The security level that the user has to operate the diagnostics 0 = Administrator 1 = Technician 2 = Operator 3 = Guest –1 - No access

Returns

LONG —no meaning.

Visual C++ example

```
m_BenchCel.ShowDiagsDialog(TRUE,0);
```

Visual Basic example

```
BenchCel.ShowDiagsDialog 1, 0
```

ShowLabwareEditor

```
LONG ShowLabwareEditor(BOOL bModal, BSTR bstrLabware)
```

Description

Method to display the labware editor graphical user interface. Through this interface dialog, the user can specify labware parameters that will be used by the device to handle the microplates. Parameters such as microplate height and notch information will be associated with a labware name, which can be used by SetLabware to indicate to the device how to handle the next microplate.

Parameters

Argument Type	Argument Name	Range	Description
BOOL	bModal	TRUE, FALSE	Whether to show the editor modally or not
BSTR	bstrLabware	Valid labware name	The labware to be selected when the editor is displayed

Returns

S_OK if successful; other value otherwise.

Visual C++ example

```
m_BenchCel.ShowLabwareEditor(1,"MyPlateType");
```

Visual Basic example

```
BenchCel.ShowLabwareEditor 1,"MyPlateType"
```

Unpause

```
LONG Unpause()
```

Description

Re-enables the motors on the BenchCel Microplate Handler. The BenchCel Microplate Handler will resume any remaining movements that were in process before the call to pause the device.

See also the Pause method.

Parameters

None

Return

S_OK if successful; other value if there was an error.

Related topics

For information about...	See...
BenchCel ActiveX properties	"Properties" on page 8
BenchCel ActiveX events	"Events" on page 25
Overview of the ActiveX controls	"About ActiveX controls" on page 4
Reporting problems	"Reporting problems" on page 3

Events

About events

The events listed in this topic occur only if Blocking is set to false or 0.

Error

Description

This event starts when an error occurs during any BenchCel Microplate Handler operation or initialization.

Parameters

Name	Type	Description

Number	SHORT	Not used.
Description	BSTR*	Description of the error.
Scode	LONG	Not used.
Source	BSTR	Name of the device where the error occurs, for example, BenchCel.
HelpFile	BSTR	Not used.
HelpContext	LONG	Not used.
CancelDisplay	BOOL*	Set to TRUE to disable the stock event handler behavior, which is to display a dialog box with the description in it.

Returns

None.

DelidComplete

Description

This event occurs after completing a delid operation.

Parameters

None.

Returns

None.

InitializeComplete

Description

This event occurs after the device successfully initializes.

Parameters

None.

Returns

None.

LoadStackComplete

Description

This event occurs after completing a load stack operation.

Parameters

None.

Returns

None.

MoveToHomePositionComplete

Description
This event occurs after completing a move to home position.

Parameters
None.

Returns
None.

PickComplete

Description
This event occurs after completing a pick operation.

Parameters
None.

Returns
None.

PlaceComplete

Description
This event occurs after completing a place operation.

Parameters
None.

Returns
None.

ProtocolFinishComplete

Description
This event occurs after completing a protocol finish operation.

Parameters
None.

Returns
None.

ProtocolStartComplete

Description
This event occurs after completing a protocol start operation.

Parameters
None.

Returns

None.

ReleaseStackComplete

Description

This event occurs after completing a release stack operation.

Parameters

None.

Returns

None.

RelidComplete

Description

This event occurs after completing a relid operation.

Parameters

None.

Returns

None.

Related topics

For information about...	See...
BenchCel ActiveX methods	"Methods" on page 10
BenchCel ActiveX properties	"Properties" on page 8
Overview of the ActiveX controls	"About ActiveX controls" on page 4
Reporting problems	"Reporting problems" on page 3

In This Book

This guide provides integrators with the ActiveX control information required to integrate another company's lab automation device with the BenchCel Microplate Handler.

July 2020

D0004458 Revision A

Agilent