VWorks Application Programming Interface

Reference Guide

Agilent Technologies

Notices

User Guide Part Number

G5415-90064

December 2010

Contact Information

Agilent Technologies Inc.
Automation Solutions
5301 Stevens Creek Blvd.
Santa Clara, CA 95051
USA

Technical Support: 1.800.979.4811
or +1.408.345.8011
service.automation@agilent.com

Customer Service: 1.866.428.9811
or +1.408.345.8356
orders.automation@agilent.com

European Service: +44 (0)1763853638
euroservice.automation@agilent.com

Documentation feedback:
documentation.automation@agilent.com

Web:
www.agilent.com/lifesciences/automation

Acknowledgements

Adobe® and Acrobat® are trademarks of Adobe Systems Incorporated.

Microsoft® and Windows® are either registered trademarks or trademarks of the Microsoft Corporation in the United States and other countries.

Pentium® is a trademark of Intel Corporation in the United States and other countries

Warranty

The material contained in this document is provided "as is," and is subject to being changed, without notice, in future editions. Further, to the maximum extent permitted by applicable law, Agilent disclaims all warranties, either express or implied, with regard to this manual and any information contained herein, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. Agilent shall not be liable for errors or for incidental or consequential damages in connection with the furnishing, use, or performance of this document or of any information contained herein. Should Agilent and the user have a separate written agreement with warranty terms covering the material in this document that conflict with these terms, the warranty terms in the separate agreement shall control.

Technology Licenses

The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license.

Restricted Rights Legend

If software is for use in the performance of a U.S. Government prime contract or subcontract, Software is delivered and licensed as "Commercial computer software" as defined in DFAR 252.227-7014 (June 1995), or as a "commercial item" as defined in FAR 2.101(a) or as "Restricted computer software" as defined in FAR 52.227-19 (June 1987) or any equivalent agency regulation or contract clause. Use, duplication or disclosure of Software is subject to Agilent Technologies' standard commercial license terms, and non-DOD Departments and Agencies of the U.S. Government will receive no greater than Restricted Rights as defined in FAR 52.227-19(c)(1-2) (June 1987). U.S. Government users will receive no greater than Limited Rights as defined in FAR 52.227-14 (June1987) or DFAR 252.227-7015 (b)(2) (November 1995), as applicable in any technical data.

Safety Notices

A WARNING notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in personal injury or death. Do not proceed beyond a WARNING notice until the indicated conditions are fully understood and met.

A **CAUTION** notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in damage to the product or loss of important data. Do not proceed beyond a **CAUTION** notice until the indicated conditions are fully understood and met.

<Go Back

Contents

<Go Back

Preface

This preface contains the following topics:

<Go Back

Who should read this guide

This guide is for experienced software developers and integrators who have the following requisite skills and knowledge:

- Experience creating and using COM objects in any COM-enabled programming language and implementing COM interfaces
- Familiarity with VWorks software features and functions
- Experience creating client applications, server applications, or both for Microsoft Windows

<Go Back

What this guide covers

What is covered

This guide defines the VWorks Application Programming Interface interfaces, methods, and enumerated types needed to programmatically initialize devices, run protocols, respond to errors, and monitor the status of VWorks.

What is not covered

This guide does not provide instructions for using VWorks software. It is assumed that the developer is already familiar with VWorks software features and functions, including the user interface. More information is available in the *VWorks Automation Control Setup Guide* and the *VWorks Automation Control User Guide*.

Software version

This guide documents the interfaces exposed by VWorks software version 11.

<Go Back

Accessing Automation Solutions user guides

About this topic

This topic describes the different formats of Automation Solutions user information and explains how to access the user information.

Where to find user information

The Automation Solutions user information is available in the following locations:

- *Knowledge base*. The help system that contains information about all of the Automation Solutions products is available from the Help menu within the VWorks software.
- *PDF files*. The PDF files of the user guides are installed with the VWorks software and are on the software CD that is supplied with the product. A PDF viewer is required to open a user guide in PDF format. You can download a free PDF viewer from the internet. For information about using PDF documents, see the user documentation for the PDF viewer.
- *Agilent Technologies website*. You can search the online knowledge base or download the latest version of any PDF file from the Agilent Technologies website at www.agilent.com/lifesciences/automation.

Accessing safety information

Safety information for the Agilent Technologies devices appears in the corresponding device safety guide or user guide.

You can also search the knowledge base or the PDF files for safety information.

Using the knowledge base

Knowledge base topics are displayed using web browser software such as Microsoft Internet Explorer and Mozilla Firefox.

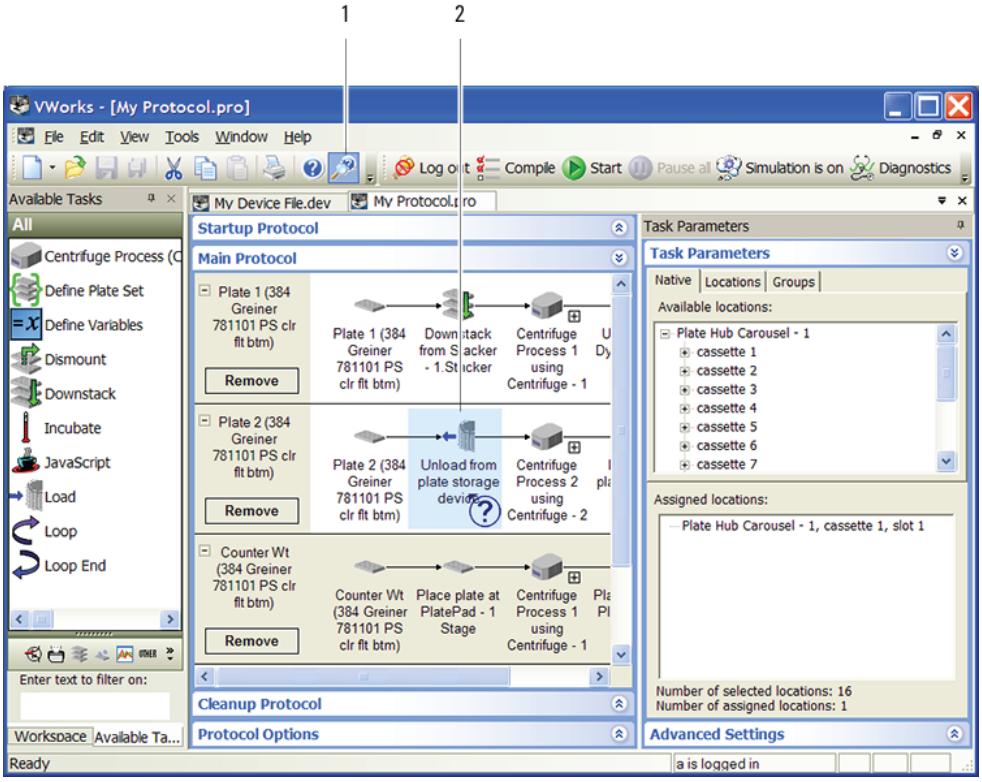
Note: If you want to use Internet Explorer to display the topics, you might have to allow local files to run active content (scripts and ActiveX controls). To do this, in Internet Explorer, open the **Internet Options** dialog box. Click the **Advanced** tab, locate the **Security** section, and select **Allow active content to run in files on my computer.**

To open the knowledge base, do one of the following:

- From within VWorks software, select **Help** > **Knowledge Base** or press F1.
- From the Windows desktop, select **Start** > **All Programs > Agilent Technologies > VWorks** > **User Guides** > **Knowledge Base.**

<Go Back

Opening the help topic for an area in the VWorks window



To access the context-sensitive help feature:

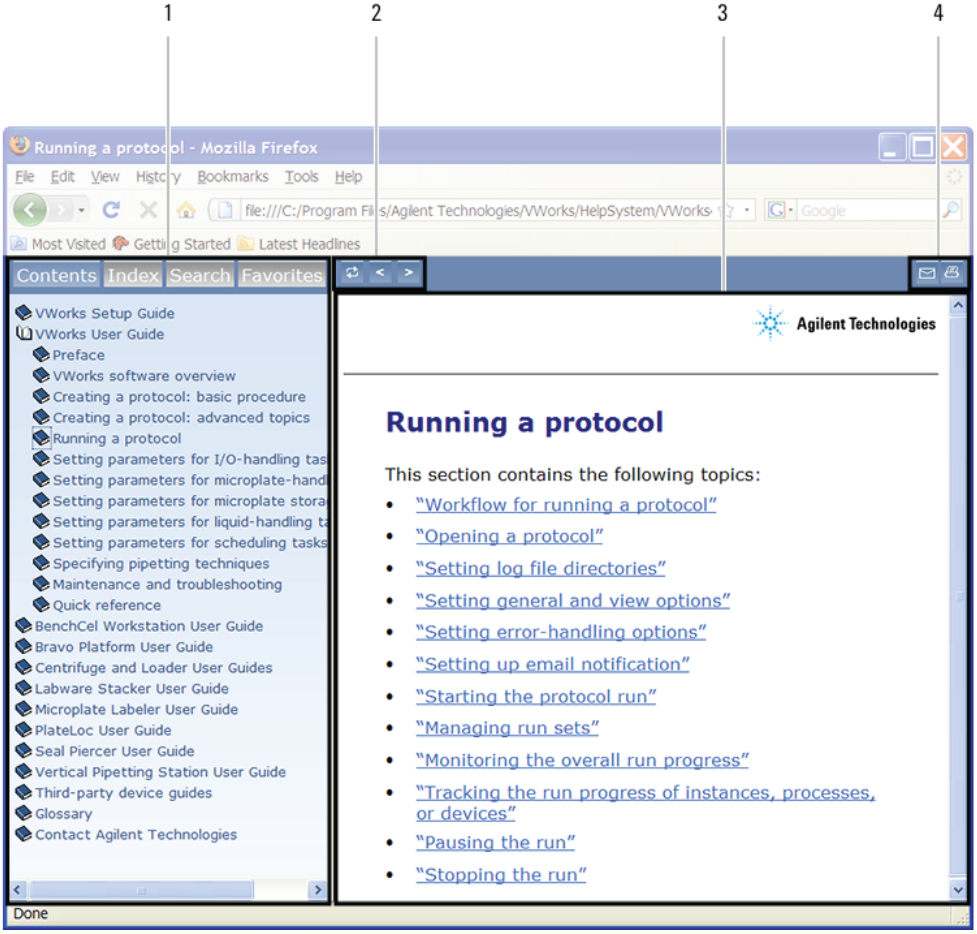
1. In the main window of the VWorks software, click the help button  .

   The pointer changes to  . Notice that the different icons or areas are highlighted as you move the pointer over them.

2. Click an icon or area of interest. The relevant topic or document opens.

<Go Back

Features in the Knowledge Base window



Item	Feature
1	*Navigation area*. Consists of four tabs: • *Contents*. Lists all the books and the table of contents of the books. • *Index*. Displays the index entries of all of the books. • *Search*. Allows you to search the Knowledge Base (all products) using keywords. You can narrow the search by product. • *Favorites*. Contains bookmarks you have created.
2	*Navigation buttons*. Enable you to navigate through the next or previous topics listed in the Contents tab.
3	*Content area*. Displays the selected online help topic.
4	*Toolbar buttons*. Enable you to print the topic or send documentation feedback by email.

<Go Back

1 Introduction

VWorks software includes optional features that allow customers to extend its core functionality to meet specific process or application needs.

The VWorks Application Programming Interface is an optional feature of VWorks software that developers can use to write their own applications to control VWorks software programmatically.

This reference guide defines the interfaces and enumerations provided for developing applications that control VWorks software using VWorks software's Component Object Model (COM) Application Programming Interface (API).

The following interfaces are included in the COM implementation:

Interface name	Purpose	Type library
IVWorks4API	The IVWorksAPI interface is used by an application to control VWorks software programmatically. IVWorks4API interface member methods are defined in "Methods" on page 3.	Works.exe
_IVWorks4APIEvent	The _IVWorks4APIEvent interface designates an event sink interface that an application must implement to receive event notifications from VWorks software. _IVWorks4APIEvent interface member events are defined in "Events" on page 35.	Works.exe

<Go Back

<Go Back

2 Methods

The methods defined in this section are members of the IVWorks4API interface, which is included in the VWorks software COM implementation.

You can use the following table to quickly locate a VWorks Application Programming Interface method by name, by description, or by page number.

Method	Description	See...
AbortProtocol	Aborts the protocol run that is in progress.	“AbortProtocol method” on page 5
CloseProtocol	Closes the specified protocol file.	“CloseProtocol method” on page 6
CompileProtocol	Compiles the specified protocol.	“CompileProtocol method” on page 8
EnumerateUsers	Returns a list of all users who have VWorks accounts.	“EnumerateUsers method” on page 10
GetSimulationMode	Gets the simulation mode state.	“GetSimulationMode method” on page 11
GetTipStates	Gets the state of the tipboxes in the specified protocol for automated tip tracking.	“GetTipStates method” on page 12
LoadProtocol	Loads the specified protocol for a run.	“LoadProtocol method” on page 15
LoadRunsetFile	Loads the specified runset file.	“LoadRunsetFile method” on page 17
Login	Logs the specified user in to VWorks using the specified user name and password.	“Login method” on page 18
Logout	Ends the current user session when the user logs out.	“Logout method” on page 20
PauseProtocol	Pauses the protocol run that is in progress.	“PauseProtocol method” on page 21
ReinitializeDevices	Reinitializes all devices in the active device file.	“ReinitializeDevices method” on page 22
ResumeProtocol	Resumes the protocol run.	“ResumeProtocol method” on page 23

<Go Back

Method	Description	See...
RunProtocol	Runs the specified protocol the specified number of times.	"RunProtocol method" on page 24
SetSimulationMode	Turns simulation mode on or off.	"SetSimulationMode method" on page 26
ShowDiagsDialog	Displays the Diagnostics window, which contains a list of active devices. The user can select a device from the list and then click a button to display the device's diagnostics dialog box.	"ShowDiagsDialog method" on page 27
ShowLoginDialog	Displays the User Authentication (login) dialog box.	"ShowLoginDialog method" on page 28
ShowManageUserDialog	Displays the User Management dialog box.	"ShowManageUserDialog method" on page 29
ShowOptionsDialog	Displays the Options dialog box.	"ShowOptionsDialog method" on page 30
ShowPlateGroupEditorDialog	Displays the Plate Groups tab in the Inventory Editor.	"ShowPlateGroupEditorDialog method" on page 31
ShowTipStateEditor	Displays the Tip State Editor dialog box.	"ShowTipStateEditor method" on page 32
ShowVWorks	Displays or hides the VWorks main window.	"ShowVWorks method" on page 33

<Go Back

AbortProtocol method

Description

Aborts the protocol run that is in progress.

Syntax

```
HRESULT AbortProtocol(
   [out,retval] enum V11ReturnCode* returnCode
);
```

Return value

The `AbortProtocol` method returns the method-call status of type `V11ReturnCode`. For possible values, see "V11ReturnCode enumerated type" on page 49.

Sample code

Visual C++

```
VWorks4Lib.V11ReturnCode retCode;
retCode = oVWorksCOM->AbortProtocol();
```

Visual Basic .NET

```
Dim retCode as VWorks4Lib.V11ReturnCode
retCode = oVWorksCOM.AbortProtocol()
```

Related information

For information about...	See...
CloseProtocol method	"CloseProtocol method" on page 6
CompileProtocol method	"CompileProtocol method" on page 8
LoadProtocol method	"LoadProtocol method" on page 15
PauseProtocol method	"PauseProtocol method" on page 21
ProtocolAborted event	"ProtocolAborted event" on page 41
ResumeProtocol method	"ResumeProtocol method" on page 23
RunProtocol method	"RunProtocol method" on page 24

<Go Back

CloseProtocol method

Description

Closes the specified protocol file.

Syntax

```
HRESULT CloseProtocol(
   [in] BSTR protocol,
   [out,retval] enum V11ReturnCode* returnCode
);
```

Parameters

protocol	[in] The file path of the protocol.

Return value

The `CloseProtocol` function method returns the method-call status of type `V11ReturnCode`. For possible values, see "V11ReturnCode enumerated type" on page 49.

Sample code

Visual C++

```
VWorks4Lib.V11ReturnCode retCode;
retCode = oVWorksCOM->CloseProtocol("myprotocol.pro");
```

Visual Basic .NET

```
Dim vwRetCode As VWorks4Lib.V11ReturnCode
vwRetCode = oVWorksCOM.CloseProtocol("myprotocol.pro")
```

Related information

For information about...	See...
AbortProtocol method	"AbortProtocol method" on page 5
CompileProtocol method	"CompileProtocol method" on page 8
LoadProtocol method	"LoadProtocol method" on page 15
PauseProtocol method	"PauseProtocol method" on page 21

<Go Back

For information about...	See...
ResumeProtocol method	“ResumeProtocol method” on page 23
RunProtocol method	“RunProtocol method” on page 24

<Go Back

CompileProtocol method

Description

Compiles the specified protocol. This method is used with the `LogMessage` event. See "LogMessage event" on page 37.

Syntax

```
HRESULT CompileProtocol(
   [in] BSTR protocol,
   [out] LONG* errorCount,
   [out] LONG* warningCount,
   [out,retval] enum V11ReturnCode* returnCode
);
```

Parameters

`protocol`	[in] The file path of the protocol.
`errorCount`	[out] Pointer to a variable that receives the number of errors found.
`warningCount`	[out] Pointer to a variable that receives the number of warnings found.

Return value

The `CompileProtocol` method returns the method-call status of type `V11ReturnCode`. For possible values, see "V11ReturnCode enumerated type" on page 49.

Sample code

Visual C++

```
VWorks4Lib.V11ReturnCode retCode;
LONG errCount, wrnCount;
retCode = oVWorksCOM->CompileProtocol ("c:\\myprotocol.pro", &errCount, &wrnCount);
```

Visual Basic .NET

```
Dim retCode as VWorks4Lib.V11ReturnCode
Dim errCount, wrnCount as Long
retCode = oVWorksCOM.CompileProtocol ("c:\myprotocol.pro", errCount, wrnCount)
```

<Go Back

Related information

For information about...	See...
AbortProtocol method	“AbortProtocol method” on page 5
CloseProtocol method	“CloseProtocol method” on page 6
LoadProtocol method	“LoadProtocol method” on page 15
LogMessage event	“LogMessage event” on page 37
PauseProtocol method	“PauseProtocol method” on page 21
ResumeProtocol method	“ResumeProtocol method” on page 23
RunProtocol method	“RunProtocol method” on page 24

<Go Back

EnumerateUsers method

Description

Returns a list of all users with VWorks accounts.

Syntax

```
HRESULT EnumerateUsers(
   [out] VARIANT* user,
   [out,retval] enum V11ReturnCode* returnCode
);
```

Parameters

`user`	[out] Pointer to a variable that receives an array of user names.

Return value

The `EnumerateUsers` method returns the method-call status of type `V11ReturnCode`. For possible values, see "V11ReturnCode enumerated type" on page 49.

Sample code

Visual Basic .NET

```
Dim oEnumerateUsers As Object = Nothing
Dim retCode As VWorks4Lib.V11ReturnCode
Dim sUsers As String = ""

retCode = oVWorksCOM.EnumerateUsers(oEnumerateUsers)

If Not (oEnumerateUsers Is Nothing) Then
Dim i As Integer
For i = 0 To oEnumerateUsers.GetLength(0) - 1
sUsers = sUsers & oEnumerateUsers(i) & " , "
Next
End If
```

Related information

For information about...	See...
ShowManageUserDialog method	"ShowManageUserDialog method" on page 29

<Go Back

GetSimulationMode method

Description

Gets the simulation mode state.

Syntax

```
HRESULT GetSimulationMode(
   [out, retval] VARIANT_BOOL *mode
);
```

Parameters

None.

Return value

The `GetSimulationMode` method returns the simulation mode state.

Possible values:

`TRUE` = Simulation mode is on

`FALSE` = Simulation mode is off

Sample code

Visual C++

```
VARIANT_BOOL bSimMode;
bSimMode= oVWorksCOM->GetSimulationMode();
```

Visual Basic .NET

```
Dim bSimMode as Boolean
bSimMode= oVWorksCOM.GetSimulationMode()
```

Related information

For information about...	See...
SetSimulationMode method	"SetSimulationMode method" on page 26

<Go Back

GetTipStates method

Description

Gets the state of the tipboxes in the specified protocol for automated tip tracking.

Syntax

```
HRESULT GetTipStates(
   [in] BSTR protocol,
   [out] BSTR* TipStateXML,
   [out,retval] enum V11ReturnCode* returnCode
);
```

Parameters

protocol	[in] The file path of the protocol.
TipStateXML	[out] Pointer to a variable that receives the current status of the tipboxes.

GetTipStates method output

The GetTipStates method returns an XML metadata string in the TipStateXML parameter.

XML structure

```
<Velocity11>
   <AllTipBoxStateQuery>
      <TipBoxStateQuery>
         <SingleTipBoxStateQuery>
            <TipBoxState>
               <PipetteHeadMode />
                  <TipPositions>
                     <TipPosition State='0' >
                        <Wells >
                           <Well />
                           ...
                        </Wells>
                     </TipPosition>
                  <TipPosition />
               </TipPositions>
            </TipBoxState>
         </SingleTipBoxStateQuery>
      </TipBoxStateQuery>
   </AllTipBoxStateQuery>
</Velocity11>
```

<Go Back

XML elements and attributes

The elements and attributes of interest to this method are described in this section. `Velocity11` is the root element, and all other elements except `PipetteHeadMode` are container elements.

SingleTipBoxStateQuery element

The `SingleTipBoxStateQuery` element has the following attributes:

Attribute name	Value
`InstanceOrLocationName`	For process labware, the value is the labware instance of the tipbox. For configured labware, the value is the name of the location on the device where the tipbox has been placed.
`ProcessLabware`	Indicates the type of labware. Possible value: `0` = The tipbox is configured labware `1` = The tipbox is process labware
`ProcessOrDeviceName`	For process labware, the value is the name of the process by which the tipbox enters the system. For configured labware, the value is the name of the device on which the tipbox has been placed.

TipBoxState element

The `TipBoxState` element has the following attribute:

Attribute name	Description
`NumWells`	The number of wells (tips) in the tipbox.

TipPosition element

The `TipPosition` element has the following attribute:

Attribute name	Description
`State`	Indicates whether the tips have been used. Possible values: `0` = The tips have not been used `1` = The tips have been used

<Go Back

Well element

The `Well` element has the following attributes:

Attribute name	Description
`Column`	The column index of the tip, where `0` indicates the leftmost column.
`Row`	The row index of the tip, where `0` indicates the topmost row.

Example of GetTipStates method output

The following sample code is a truncated example of an XML metadata string that is returned by the `GetTipStates` method in the `TipStateXML` parameter. In this example, the tipbox contains only unused tips. The wells listed under `<TipPosition State='0'>` contain tips that have not been used. The wells listed under `<TipPosition State='1'>` contain tips that have been used.

```
<Velocity11 file='MetaData' md5sum='50a7e93353a1993ae7a53db21dd3a948' version='1.0' >
   <AllTipBoxStateQuery >
      <TipBoxStateQuery >
         <SingleTipBoxStateQuery InstanceOrLocationName='1' ProcessLabware='0'
→ProcessOrDeviceName='Bravo - 1' >
            <TipBoxState NumWells='384' >
               <PipetteHeadMode Channels='1' ColumnCount='1' RowCount='1'
→SubsetConfig='0' SubsetType='4' TipType='0' />
                  <TipPositions >
                     <TipPosition State='0' >
                        <Wells >
                           <Well Column='0' Row='0' />
                           <Well Column='1' Row='0' />
...
                           <Well Column='22' Row='15' />
                           <Well Column='23' Row='15' />
                        </Wells>
                     </TipPosition>
                  <TipPosition State='1' />
               </TipPositions>
            </TipBoxState>
         </SingleTipBoxStateQuery>
      </TipBoxStateQuery>
   </AllTipBoxStateQuery>
</Velocity11>
```

Return value

The `GetTipStates` method returns the method-call status of type `V11ReturnCode`. For possible values, see "V11ReturnCode enumerated type" on page 49.

Sample code

Visual Basic .NET

```
Dim TipStateXML As String = ""
Dim retCode As VWorks4Lib.V11ReturnCode
retCode = oVWorksCOM.GetTipStates("c:\myprotocol.pro", TipStateXML)
```

<Go Back

LoadProtocol method

Description

Loads the specified protocol for a run.

Syntax

```
HRESULT LoadProtocol(
   [in] BSTR protocol,
   [out,retval] enum V11ReturnCode* returnCode
);
```

Parameters

`protocol`	[in] The file path of the protocol.

Return value

The `LoadProtocol` method returns the method-call status of type `V11ReturnCode`. For possible values, see "V11ReturnCode enumerated type" on page 49.

Sample code

Visual C++

```
VWorks4Lib.V11ReturnCode retCode;
retCode=oVWorksCOM->LoadProtocol("c:\\myprotocol.pro");
```

Visual Basic .NET

```
Dim retCode as VWorks4Lib.V11ReturnCode
retCode=oVWorksCOM.LoadProtocol ("c:\myprotocol.pro")
```

Related information

For information about...	See...
AbortProtocol method	"AbortProtocol method" on page 5
CloseProtocol method	"CloseProtocol method" on page 6
CompileProtocol event	"CompileProtocol method" on page 8
PauseProtocol method	"PauseProtocol method" on page 21

<Go Back

For information about...	See...
ResumeProtocol method	“ResumeProtocol method” on page 23
RunProtocol method	“RunProtocol method” on page 24

<Go Back

LoadRunsetFile method

Description

Loads the specified runset file.

Syntax

```
HRESULT LoadRunsetFile(
   [in] BSTR runset,
   [out,retval] enum V11ReturnCode* returnCode
);
```

Parameters

runset	[in] The file path of the runset.

Return value

The `LoadRunsetFile` method returns the method-call status of type `V11ReturnCode`. For possible values, see "V11ReturnCode enumerated type" on page 49.

Sample code

Visual C++

```
VWorks4Lib.V11ReturnCode retCode;
retCode=oVWorksCOM->LoadRunsetFile ("c:\\myrunset.rst");
```

Visual Basic .NET

```
Dim retCode as VWorks4Lib.V11ReturnCode
retCode=oVWorksCOM.LoadRunsetFile ("c:\myrunset.rst")
```

<Go Back

Login method

Description

Logs the specified user in to VWorks using the specified user name and password.

Syntax

```
HRESULT Login(
   [in] BSTR userName,
   [in] BSTR password,
   [out,retval] enum V11LoginResult* loginResult
);
```

Parameters

`userName`	[in] The name of the user.
`password`	[in] The user's password.

Returns

The `Login` method returns the login status of type `V11LoginResult`. For possible values, see ""V11ReturnCode enumerated type" on page 49" on page 47.

Return value

The `EnumerateUsers` method returns the method-call status of type `V11ReturnCode`. For possible values, see "V11ReturnCode enumerated type" on page 49.

Sample code

Visual C++

```
VWorks4Lib.V11LoginResult retCode;
loginResult= oVWorksCOM->Login("user1","mypassword!");
```

Visual Basic .NET

```
Dim loginResult as VWorks4Lib.V11LoginResult
loginResult= oVWorksCOM.Login("user1","mypassword!")
```

<Go Back

Related information

For information about...	See...
Logout method	"Logout method" on page 20
ShowLoginDialog method	"ShowLoginDialog method" on page 28

<Go Back

Logout method

Description

Ends the current user session when the user logs out.

Syntax

```
HRESULT Logout(
   [out,retval] enum V11ReturnCode* returnCode
);
```

Parameters

None.

Return value

The `Logout` method returns the method-call status of type `V11ReturnCode`. For possible values, see "V11ReturnCode enumerated type" on page 49.

Sample code

Visual C++

```
VWorks4Lib.V11ReturnCode retCode;
retCode = oVWorksCOM->Logout();
```

Visual Basic .NET

```
Dim retCode as VWorks4Lib.V11ReturnCode
retCode = oVWorksCOM.Logout()
```

Related information

For information about...	See...
Login method	"Login method" on page 18
ShowLoginDialog method	"ShowLoginDialog method" on page 28

<Go Back

PauseProtocol method

Description

Pauses the protocol run that is in progress.

Syntax

```
HRESULT PauseProtocol(
   [out,retval] enum V11ReturnCode* returnCode
);
```

Parameters

None.

Return value

The `PauseProtocol` method returns the method-call status of type `V11ReturnCode`. For possible values, see "V11ReturnCode enumerated type" on page 49.

Sample code

Visual C++

```
VWorks4Lib.V11ReturnCode retCode;
retCode = oVWorksCOM->PauseProtocol();
```

Visual Basic .NET

```
Dim retCode as VWorks4Lib.V11ReturnCode
retCode = oVWorksCOM.PauseProtocol()
```

Related information

For information about...	See...
AbortProtocol method	"AbortProtocol method" on page 5
CloseProtocol method	"CloseProtocol method" on page 6
CompileProtocol method	"CompileProtocol method" on page 8
LoadProtocol method	"LoadProtocol method" on page 15
ResumeProtocol method	"ResumeProtocol method" on page 23
RunProtocol method	"RunProtocol method" on page 24

<Go Back

ReinitializeDevices method

Description

Reinitializes all devices in the active device file.

Syntax

```
HRESULT ReinitializeDevices(
   [out,retval] enum V11ReturnCode* returnCode
);
```

Parameters

None.

Return value

The `ReinitializeDevices` method returns the method-call status of type `V11ReturnCode`. For possible values, see "V11ReturnCode enumerated type" on page 49.

Sample code

Visual C++

```
VWorks4Lib.V11ReturnCode retCode;
retCode=oVWorksCOM->ReinitializeDevices();
```

Visual Basic .NET

```
Dim retCode as VWorks4Lib.V11ReturnCode
retCode=oVWorksCOM.ReinitializeDevices()
```

Related information

For information about...	See...
InitializationComplete event	"InitializationComplete event" on page 36

<Go Back

ResumeProtocol method

Description

Resumes the protocol run.

Syntax

```
HRESULT ResumeProtocol(
   [out,retval] enum V11ReturnCode* returnCode
);
```

Parameters

None.

Return value

The `ResumeProtocol` method returns the method-call status of type `V11ReturnCode`. For possible values, see "V11ReturnCode enumerated type" on page 49.

Sample code

Visual C++

```
VWorks4Lib.V11ReturnCode retCode;
retCode = oVWorksCOM->ResumeProtocol();
```

Visual Basic .NET

```
VWorks4Lib.V11ReturnCode retCode;
retCode = oVWorksCOM.ResumeProtocol();
```

Related information

For information about...	See...
AbortProtocol method	"AbortProtocol method" on page 5
CloseProtocol method	"CloseProtocol method" on page 6
CompileProtocol method	"CompileProtocol method" on page 8
LoadProtocol method	"LoadProtocol method" on page 15
PauseProtocol method	"PauseProtocol method" on page 21
RunProtocol method	"RunProtocol method" on page 24

<Go Back

RunProtocol method

Description

Runs the specified protocol the specified number of times.

Syntax

```
HRESULT RunProtocol(
   [in] BSTR protocol,
   [in] LONG runCount,
   [out,retval] enum V11ReturnCode* returnCode
);
```

Parameters

protocol	[in] The file path of the protocol.
runCount	[in] The number of times to run the protocol.

Return value

The RunProtocol method returns the method-call status of type V11ReturnCode. For possible values, see "V11ReturnCode enumerated type" on page 49.

Sample code

Visual C++

```
VWorks4Lib.V11ReturnCode retCode;
retCode=oVWorksCOM->RunProtocol ("c:\\myprotocol.pro",2);
```

Visual Basic .NET

```
Dim retCode as VWorks4Lib.V11ReturnCode
retCode=oVWorksCOM.RunProtocol ("c:\myprotocol.pro",2)
```

Related information

For information about...	See...
AbortProtocol method	"AbortProtocol method" on page 5
CloseProtocol method	"CloseProtocol method" on page 6
CompileProtocol method	"CompileProtocol method" on page 8

<Go Back

For information about...	See...
LoadProtocol method	“LoadProtocol method” on page 15
PauseProtocol method	“PauseProtocol method” on page 21
ResumeProtocol method	“ResumeProtocol method” on page 23

<Go Back

SetSimulationMode method

Description

Turns simulation mode on or off.

Syntax

```
HRESULT SetSimulationMode(
   [in] VARIANT_BOOL mode,
   [out, retval] enum V11ReturnCode* returnCode
);
```

Parameters

`mode`	[in] The simulation mode state to set. Possible values: `TRUE` = Turn on simulation mode `FALSE` = Turn off simulation mode

Return value

The `SetSimulationMode` method returns the method-call status of type `V11ReturnCode`. For possible values, see "V11ReturnCode enumerated type" on page 49.

Sample code

Visual C++

```
VWorks4Lib.V11ReturnCod retCode;
retCode = oVWorksCOM->SetSimulationMode(VARIANT_TRUE);
retCode = oVWorksCOM->SetSimulationMode(VARIANT_FALSE);
```

Visual Basic .NET

```
Dim retCode as VWorks4Lib.V11ReturnCode
retCode=oVWorksCOM.SetSimulationMode(True)
retCode=oVWorksCOM.SetSimulationMode(False)
```

Related information

For information about...	See...
GetSimulationMode method	"GetSimulationMode method" on page 11

<Go Back

ShowDiagsDialog method

Description

Displays the Diagnostics window in front of the VWorks main window. In this window, the user can select a device from the list of active devices and then click the Device diagnostics button to display the device's diagnostics dialog box.

Syntax

```
HRESULT ShowDiagsDialog(
   [out,retval] enum V11ReturnCode* returnCode
);
```

Parameters

None.

Return value

The `ShowDiagsDialog` method returns the method-call status of type `V11ReturnCode`. For possible values, see "V11ReturnCode enumerated type" on page 49.

Sample code

Visual C++

```
VWorks4Lib.V11ReturnCode retCode;
retCode=oVWorksCOM->ShowDiagsDialog();
```

Visual Basic .NET

```
Dim retCode as VWorks4Lib.V11ReturnCode
retCode=oVWorksCOM.ShowDiagsDialog()
```

<Go Back

ShowLoginDialog method

Description

Displays the User Authentication (login) dialog box.

Syntax

```
HRESULT ShowLoginDialog(
   [out,retval] enum V11ReturnCode* returnCode
);
```

Parameters

None.

Return value

The `ShowLoginDialog` method returns the method-call status of type `V11ReturnCode`. For possible values, see "V11ReturnCode enumerated type" on page 49.

Sample code

Visual C++

```
VWorks4Lib.V11ReturnCode retCode;
retCode=oVWorksCOM->ShowLoginDialog();
```

Visual Basic .NET

```
Dim retCode as VWorks4Lib.V11ReturnCode
retCode=oVWorksCOM.ShowLoginDialog()
```

Related information

For information about...	See...
Login method	"Login method" on page 18
Logout method	"Logout method" on page 20

<Go Back

ShowManageUserDialog method

Description

Displays the User Management dialog box.

Syntax

```
HRESULT ShowManageUserDialog(
   [out,retval] enum V11ReturnCode* returnCode
);
```

Parameters

None.

Return value

The `ShowManageUserDialog` method returns the method-call status of type `V11ReturnCode`. For possible values, see "V11ReturnCode enumerated type" on page 49.

Sample code

Visual C++

```
VWorks4Lib.V11ReturnCode retCode;
retCode=oVWorksCOM->ShowManageUserDialog();
```

Visual Basic .NET

```
Dim retCode as VWorks4Lib.V11ReturnCode
retCode=oVWorksCOM.ShowManageUserDialog()
```

Related information

For information about...	See...
EnumerateUsers method	"EnumerateUsers method" on page 10

<Go Back

ShowOptionsDialog method

Description

Displays the Options dialog box.

Syntax

```
HRESULT ShowOptionsDialog(
   [out,retval] enum V11ReturnCode* returnCode
);
```

Parameters

None.

Return value

The `ShowOptionsDialog` method returns the method-call status of type `V11ReturnCode`. For possible values, see "V11ReturnCode enumerated type" on page 49.

Sample code

Visual C++

```
VWorks4Lib.V11ReturnCode retCode;
retCode=oVWorksCOM->ShowOptionsDialog();
```

Visual Basic .NET

```
Dim retCode as VWorks4Lib.V11ReturnCode
retCode=oVWorksCOM.ShowOptionsDialog()
```

<Go Back

ShowPlateGroupEditorDialog method

Description

Displays the Plate Groups tab in the Inventory Editor.

Syntax

```
HRESULT ShowPlateGroupEditorDialog(
   [out,retval] enum V11ReturnCode* returnCode
);
```

Parameters

None.

Return value

The `ShowPlateGroupEditorDialog` method returns the method-call status of type `V11ReturnCode`. For possible values, see "V11ReturnCode enumerated type" on page 49.

Sample code

Visual C++

```
VWorks4Lib.V11ReturnCode retCode;
retCode=oVWorksCOM->ShowPlateGroupEditorDialog();
```

Visual Basic .NET

```
Dim retCode as VWorks4Lib.V11ReturnCode
retCode=oVWorksCOM.ShowPlateGroupEditorDialog()
```

<Go Back

ShowTipStateEditor method

Description

Displays the Tip State Editor.

Syntax

```
HRESULT ShowTipStateEditor(
   [in] BSTR protocol,
   [out,retval] enum V11ReturnCode* returnCode
);
```

Parameters

`protocol`	[in] The file path of the protocol.

Return value

The `ShowTipStateEditor` method returns the method-call status of type `V11ReturnCode`. For possible values, see "V11ReturnCode enumerated type" on page 49.

Sample code

Visual C++

```
VWorks4Lib.V11ReturnCode retCode;
retCode=oVWorksCOM->ShowTipStateEditor ("c:\\myprotocol.pro");
```

Visual Basic .NET

```
Dim retCode as VWorks4Lib.V11ReturnCode
retCode=oVWorksCOM.ShowTipStateEditor ("c:\myprotocol.pro")
```

Related information

For information about...	See...
GetTipStates method	"GetTipStates method" on page 12

<Go Back

ShowVWorks method

Description

Displays or hides the VWorks main window.

Syntax

```
HRESULT ShowVWorks(
   [in] VARIANT_BOOL showOrHide,
   [out, retval] enum V11ReturnCode* returnCode
);
```

Parameters

`showOrHide`	[in] Indicates whether to display or hide the VWorks main window. Possible values: `TRUE` = Display the VWorks main window `FALSE` = Hide the VWorks main window

Return value

The `ShowVWorks` method returns the method-call status of type `V11ReturnCode`. For possible values, see "V11ReturnCode enumerated type" on page 49.

Sample code

Visual C++

```
VWorks4Lib.V11ReturnCode retCode;
retCode=oVWorksCOM->ShowVWorks (VARIANT_TRUE);
```

Visual Basic .NET

```
Dim retCode as VWorks4Lib.V11ReturnCode
retCode=oVWorksCOM.ShowVWorks (True)
```

<Go Back

<Go Back

3 Events

The events defined in this chapter are members of the _IVWorks4APIEvent interface, which is included in the VWorks COM implementation. To receive event notifications from a VWorks object, an automation client must implement this interface.

You can use the following table to quickly locate a VWorks Application Programming Interface event by name, by description, or by page number.

Event	Description	See...
`InitializationComplete`	Fires when the device initialization is finished.	"InitializationComplete event" on page 36
`LogMessage`	Fires when a message is posted to the Main Log.	"LogMessage event" on page 37
`MessageBoxAction`	Fires when a user response is required.	"MessageBoxAction event" on page 39
`ProtocolAborted`	Fires when the operator or automation client aborts the protocol run.	"ProtocolAborted event" on page 41
`ProtocolComplete`	Fires after the Startup Protocol, Cleanup Protocol, or Main Protocol is finished.	"ProtocolComplete event" on page 42
`RecoverableError`	Fires when an error occurs to which the user can respond with abort, retry, or ignore.	"RecoverableError event" on page 43
`UnrecoverableError`	Fires when an error occurs to which no user response is required.	"UnrecoverableError event" on page 45
`UserMessage`	Fires when a User Message task occurs.	"UserMessage event" on page 46

<Go Back

InitializationComplete event

Description

Fires when device initialization is finished.

Syntax

```
void InitializationComplete(
   [in] LONG session
);
```

Parameters

`session`	[in] The session ID.

Related information

For information about...	See...
ReinitializeDevices method	"ReinitializeDevices method" on page 22

<Go Back

LogMessage event

Description

Fires when a message is posted to the Main Log.

Syntax

```
HRESULT LogMessage(
   [in] LONG session,
   [in] LONG logClass,
   [in] BSTR timeStamp,
   [in] BSTR device,
   [in] BSTR location,
   [in] BSTR process,
   [in] BSTR task,
   [in] BSTR fileName,
   [in] BSTR message
);
```

Parameters

`session`	[in] The session ID.

`logClass`	[in] A value that represents the message type.
	Possible values:
	`0 = Info`
	`1 = Warning`
	`2 = Error`
	`3 = Event`
	`4 = PipetteTransfer`
	`5 = Script`
	`6 = UserDefined`

`timeStamp`	[in] The time at which the event occurred.
`device`	[in] The device name, or an empty string.
`location`	[in] The location, or an empty string.
`process`	[in] The labware name, or an empty string.
`task`	[in] The task name, or an empty string.
`fileName`	[in] The protocol file name, the device file name, or an empty string.
`message`	[in] The text that describes the error.

<Go Back

Related information

For information about...	See...
CompileProtocol method	“CompileProtocol method” on page 8

<Go Back

MessageBoxAction event

Description

Fires when a user response is required.

Syntax

```
HRESULT MessageBoxAction(
   [in] LONG session,
   [in]LONG type,
   [in] BSTR message,
   [in] BSTR caption,
   [out] LONG* actionToTake
);
```

Parameters

session	[in] The session ID.
type	[in] A value that represents the message-box type. Possible values: 0 = The message box contains one push button: OK (MB_OK) 1 = The message box contains three push buttons: Yes, No, and Cancel (MB_OKCANCEL) 2 = The message box contains three push buttons: Abort, Retry, and Ignore (MB_ABORTRETRYIGNORE) 3 = The message box contains three push buttons: Yes, No, and Cancel (MB_YESNOCANCEL) 4 = The message box contains two push buttons: Yes and No (MB_YESNO) 5 = The message box contains two push buttons: Retry and Cancel (MB_RETRYCANCEL)
message	[in] The message text.
caption	[in] The title bar text.

<Go Back

`actionToTake`	[out] Pointer to a variable that receives a value that represents the action to take.
	Possible values:
	`1 = OK`
	`2 = CANCEL`
	`3 = ABORT`
	`4 = RETRY`
	`5 = IGNORE`
	`6 = YES`
	`7 = NO`

<Go Back

ProtocolAborted event

Description

Fires when the operator or automation client aborts the protocol run.

Syntax

```
HRESULT ProtocolAborted(
   [in] LONG session,
   [in] BSTR protocol,
   [in] BSTR protocol_type
);
```

Parameters

session	[in] The session ID.
protocol	[in] The file path of the protocol.
protocol_type	[in] A value that represents the protocol type. Possible values: • Startup • Main • Cleanup

Related information

For information about...	See...
AbortProtocol method	"AbortProtocol method" on page 5

<Go Back

ProtocolComplete event

Description

Fires after each of the following protocols is finished:

- Startup Protocol
- Cleanup Protocol
- Main Protocol

Syntax

```
HRESULT ProtocolComplete(
   [in] LONG session,
   [in] BSTR protocol,
   [in] BSTR protocol_type
);
```

Parameters

session	[in] The session ID.
protocol	[in] The file path of the protocol.
protocol_type	[in] A value that represents the protocol type. Possible values: • Startup • Main • Cleanup

<Go Back

RecoverableError event

Description

Fires whenever an error occurs to which the user can respond with abort, retry, or ignore.

Syntax

```
HRESULT RecoverableError(
   [in] LONG session,
   [in] BSTR device,
   [in] BSTR location,
   [in] BSTR description,
   [out] LONG* actionToTake,
   [out] VARIANT_BOOL* vworksHandlesError
);
```

Parameters

session	[in] The session ID.
device	[in] The name of the device on which the error occurred, or an empty string.
location	[in] The location on the device where the error occurred, or an empty string.
description	[in] The text that describes the error.
actionToTake	[out] Pointer to a variable that receives a value that represents the action to take. Possible values: 0 = Abort 1 = Retry 2 = Ignore
vworksHandlesError	[out] A variable that receives a value that indicates whether VWorks software should handle the error. Possible values: TRUE = Allow VWorks software to handle the error FALSE = Do not allow VWorks software to handle the error

<Go Back

Related information

For information about...	See...
UnrecoverableError event	"UnrecoverableError event" on page 45

<Go Back

UnrecoverableError event

Description

Fires when an error occurs to which no user response is required.

Syntax

```
HRESULT UnrecoverableError(
   [in] LONG session,
   [in] BSTR description
);
```

Parameters

`session`	[in] The session ID.
`description`	[in] The message text.

Related information

For information about...	See...
RecoverableError event	"RecoverableError event" on page 43

<Go Back

UserMessage event

Description

Fires when a User Message task occurs.

Syntax

```
HRESULT UserMessage(
   [in] LONG session,
   [in] BSTR caption,
   [in] BSTR message,
   [in] VARIANT_BOOL wantsData,
   [out] BSTR* userData
);
```

Parameters

session	[in] The session ID.
caption	[in] The title bar text.
message	[in] The message (body) text.
wantsData	[in] A value that indicates whether data can be returned in the userData parameter to update a JavaScript variable specified in the User Message task. Possible values: TRUE = Data can be returned FALSE = Data cannot be returned
userData	[out] Pointer to a variable that receives the variable value. The value of the wantsData parameter must be TRUE. The data returned is used to set the JavaScript variable specified in the User Message task.

<Go Back

4 Enumerations

This chapter defines the enumerated types used in VWorks Application Programming Interface methods.

This chapter contains the following topics:

<Go Back

V11LoginResult enumerated type

Description

Indicates the login status.

Syntax

```
enum V11LoginResult {
   LOGIN_SUCCESS = 0,
   LOGIN_FAIL = 1,
   LOGIN_DISABLED = 2,
   LOGIN_EXPIRED = 3,
   LOGIN_TOO_MANY_FAILED_ATTEMPTS = 4,
   LOGIN_NOT_ENOUGH_AUTHORIZATION = 5,
};
```

Constants

Name	Value	Description
LOGIN_SUCCESS	0	The login succeeded.
LOGIN_FAIL	1	The login failed.
LOGIN_DISABLED	2	The login was disabled.
LOGIN_EXPIRED	3	The login period passed.
LOGIN_TOO_MANY_FAILED_ATTEMPTS	4	Too many login attempts were made and failed.
LOGIN_NOT_ENOUGH_AUTHORIZATION	5	Higher access privileges are required to perform the requested action.

<Go Back

V11ReturnCode enumerated type

Description

Indicates the method-call status.

Syntax

```
enum V11ReturnCode {
   RETURN_SUCCESS = 0,
   RETURN_BAD_ARGS = 1,
   RETURN_FAIL = 2,
};
```

Constants

Name	Value	Description
RETURN_SUCCESS	0	The method-call succeeded.
RETURN_BAD_ARGS	1	The method returned bad arguments.
RETURN_FAIL	2	The method-call failed.

<Go Back

<Go Back

Glossary

cassette The column of shelves or slots in a Labware MiniHub or the Plate Hub Carousel.

clamps (BenchCel) The components inside of the stacker head that close and open the stacker grippers during the loading, unloading, downstacking, and upstacking processes.

controlling computer The lab automation system computer that controls the devices in the system.

cycle See seal cycle.

deadlock An error that occurs when the number of locations available in the system is less than the number of microplates in the system. Because the microplates cannot move to the expected locations, the protocol pauses.

device An item on your lab automation system that can have an entry in the device file. A device can be a robot, an instrument, or a location on the lab automation system that can hold a piece of labware.

device file A file that contains the configuration information for a device. The device file has the .dev file name extension and is stored in the folder that you specify when saving the file.

downstack The process in which a microplate is moved out of the stack.

error handler The set of conditions that define a specific recovery response to an error.

home position The position where all robot axes are at the 0 position (the robot head is approximately at the center of the x-axis and at 0 of the z-axis, and the robot arms are perpendicular to the x-axis).

homing The process in which the robot is sent to the factory-defined home position for each axis of motion.

hot plate (PlateLoc) A heated metal plate inside the sealing chamber that descends and presses the seal onto the plate.

insert A pad placed under the plate to support the bottom of the wells for uniform sealing.

location group A list of labware that can be moved into or out of particular slots in a storage device.

plate group A list of specific labware that can be moved into or out of a storage device without regard for the slot locations.

plate instance A single labware in a labware group that is represented by the process plate icon.

plate stage The removable metal platform on which you load a plate.

plate-stage support (Centrifuge) The structure on which you load a plate stage. The plate-stage support extends when the door opens.

profile The Microsoft Windows registry entry that contains the communication settings required for communication between a device and the VWorks software.

process A sequence of tasks that are performed on a particular labware or a group of labware.

protocol A schedule of tasks to be performed by a standalone device, or devices in the lab automation system.

regrip station A location that enables the robot to change its grip orientation (landscape or portrait), or adjust its grip at the specified gripping height. Grip height adjustment might be necessary after a robot picks up a labware higher than the specified gripping height because of physical restrictions at a teachpoint.

robot grippers The components that the robot uses to hold labware.

<Go Back

run A process in which one or more microplates are processed. In a standalone device, the run consists of one cycle. In a lab automation system, a run can consist of multiple cycles that are automated.

safe zone The boundary within which the robot is allowed to move without colliding with external devices.

seal cycle The process in which a single plate is sealed on the PlateLoc Sealer.

seal entry slot The narrow entry on the back of the PlateLoc Sealer where the seal is inserted into the device.

seal-loading card A rectangular card that is used to facilitate the seal loading process on the PlateLoc Sealer.

seal-roll support The triangular structures at the top of the PlateLoc Sealer where a roll of seal is mounted.

sealing chamber The area inside of the PlateLoc Sealer where the seal is applied to a plate.

shelves (BenchCel) The components inside of the stacker head that provide leveling surfaces for the microplates, thus ensuring accurate robot gripping, during the downstacking process.

stacker grippers The padding at the bottom of the stacker racks that hold microplates when a microplate is loaded, downstacked, or upstacked.

subprocess A sequence of tasks performed as a subroutine within a protocol. Typically the subprocess is performed by a single device type, such as the Bravo device.

task An operation performed on one or more labware.

task parameters The parameters associated with each task in a protocol. For example, in a labeling task, the parameters include the label value.

teachpoint A set of coordinates that define where the robot can pick up or place labware and the location of a known object.

teachpoint file The XML file that contains the settings for one or more device teachpoints.

touch screen The interface on the front of the PlateLoc Sealer where sealing parameters are set, the seal cycle can be started or stopped, and the seal cycle can be monitored.

upstack The process in which a microplate is moved back into the stack.

waypoint A set of coordinates that define a location the robot passes through on its way to a teachpoint.

workspace The boundary within which the robot can move without limitations.

<Go Back

Index

<Go Back

Agilent Technologies

Reference Guide

G5415-90064