VWorks Application Programming Interface

Version 14.0

Reference Guide

Original Instructions

Notices

Manual Part Number

D0008025 Revision A

February 2021

Copyright

Contact Information

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

Web:
https://www.agilent.com

Contact page:
https://www.agilent.com/en/contact-us/page

Documentation feedback:
documentation.automation@agilent.com

Acknowledgements

Microsoft® and Windows® are either registered trademarks or trademarks of the Microsoft 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

WARNING

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.

CAUTION

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.

Contents

Preface

This preface contains the following topics:

- “About this guide” on page vi
- “Accessing Agilent Automation Solutions user guides” on page viii

About this guide

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

Software version

This guide documents the interfaces exposed by VWorks software 14.0 and later.

What this guide covers

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

This guide does not provide instructions for using the VWorks software. It is assumed that the developer is already familiar with VWorks software features and functions, including the user interface.

Table Terms used in this guide

Term	Description
VWorks Automation Control	VWorks software component that you use to create the protocols that run your automation devices.
VWorks Plus	Collection of software components required for running the VWorks software with compliance features. This edition includes Control Panel, Shared Services, Content Management, and Content Browser.
VWorks Standard	Collection of software components required for running the standard VWorks software. This edition includes Control Panel and Shared Services.
Content Management (OpenLab component)	*VWorks Plus only*. The storage repository for VWorks-related records of interest. An administrator can use the Content Browser to view the VWorks project content in Content Management.
Control Panel (OpenLab component)	The Shared Services user interface for configuring and managing user access and licenses.
Microsoft Control Panel	Part of the Microsoft Windows operating system.

Term	Description
Shared Services (OpenLab component)	Set of administrative services that control VWorks user access and file storage. Shared Services are accessed via the Control Panel.

Related guides

For information about …	See...
Computer requirements and installing the software	*VWorks Workstation Plus and VWorks Workstation Installation Guide*
• Configuring the software in Control Panel, including managing licenses and user access • Backing up and restoring the software, Shared Services database, and project content	• *VWorks Automation Control Plus Administrator Guide* • *VWorks Automation Control Standard Administrator Guide*
Setting up a specific device and operating the device using VWorks and device diagnostics software	Relevant Agilent device user guide
• *VWorks Plus only*. Managing audit trails and record states • Setting up labware definitions and liquid classes • Setting up an experiment tags database and inventory database • Migrating protocols from previous versions of VWorks software.	*VWorks Automation Control Setup Guide*
Creating and running protocols.	*VWorks Automation Control User Guide* *VWorks Quick Guide for Protocol Authors*

Accessing Agilent Automation Solutions user guides

About this topic

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

Where to find user information

The user information is available in the following locations:

- *Knowledge base*. The help system for the Automation Solutions products is available from:
  - Help menu within the VWorks software: Select **Help** > **Knowledge Base** or press F1.
  - From the Windows desktop: Select **Start (**⊞**)** > **All Apps > Agilent Technologies > VWorks Knowledge Base.**

  For guidelines on using the VWorks context-sensitive help and knowledge base features, see *Using the knowledge base*, below.
- *PDF files*. The PDF files of the user guides are installed with the VWorks software (C:\Program Files (x86)\Agilent Technologies\VWorks\UserGuides) and are available in the VWorks Knowledge Base.
- *Website*. You can search the online VWorks Knowledge Base or download the latest version of any PDF file from the Agilent website at www.agilent.com/chem/askb.

Accessing safety information

Safety information for the Agilent Automation Solutions devices appears in the *Automation Solutions Products General Safety Guide* and 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 Edge.

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.

Opening the help topic for an area in the VWorks window

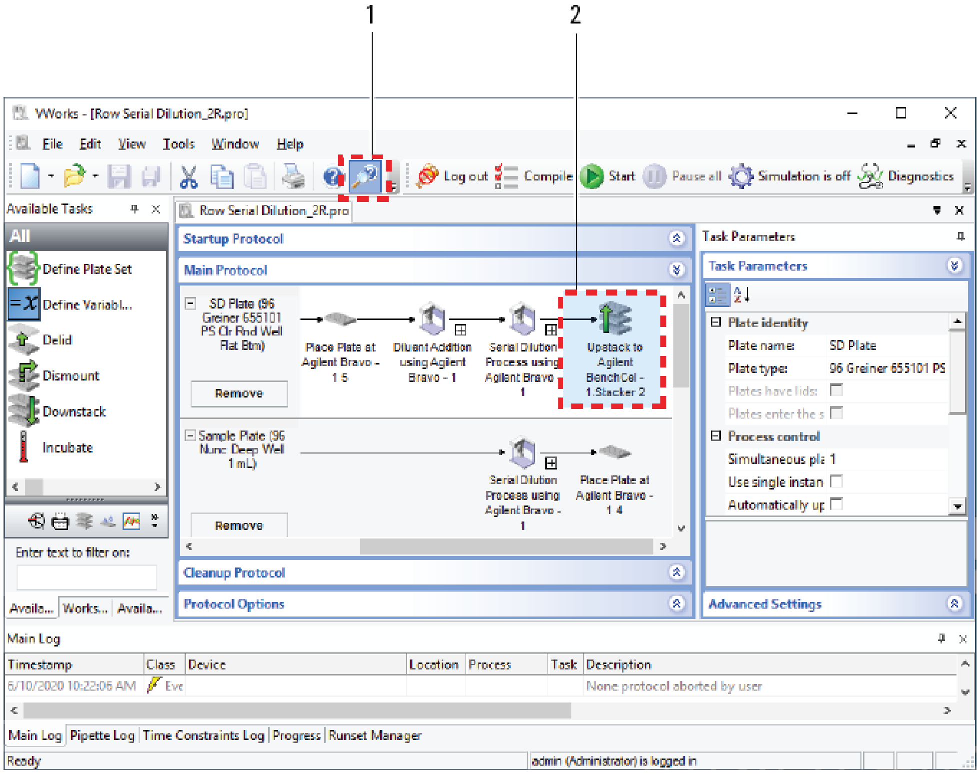

1
2
VWorks - [Row Serial Dilution_2R.pro]
File Edit View Tools Window Help
Log out
Compile
Start
Pause all
Simulation is off
Diagnostics
Available Tasks
All
Define Plate Set
Define Variabl...
Delid
Dismount
Downstack
Incubate
Enter text to filter on:
Availa... Works... Availa...
Row Serial Dilution_2R.pro
Startup Protocol
Main Protocol
SD Plate (96 Greiner 655101 PS Clr Rnd Well Flat Btm)
Remove
Place Plate at Agilent Bravo - 1 5
Diluent Addition using Agilent Bravo - 1
Serial Dilution Process using Agilent Bravo 1
Upstack to Agilent BenchCel - 1.Stacker 2
Sample Plate (96 Nunc Deep Well 1 mL)
Remove
Serial Dilution Process using Agilent Bravo - 1
Place Plate at Agilent Bravo - 1 4
Cleanup Protocol
Protocol Options
Task Parameters
Task Parameters
Advanced Settings
Main Log
Main Log
Pipette Log
Time Constraints Log
Progress
Runset Manager
Ready

Plate identity

Plate name:	SD Plate
Plate type:	96 Greiner 655101 PS
Plates have lids:	□
Plates enter the s	□

Process control

Simultaneous pl:	1
Use single instan	□
Automatically up	□

Timestamp	Class	Device	Location	Process	Task	Description
6/10/2020 10:22:06 AM	Eve					None protocol aborted by user

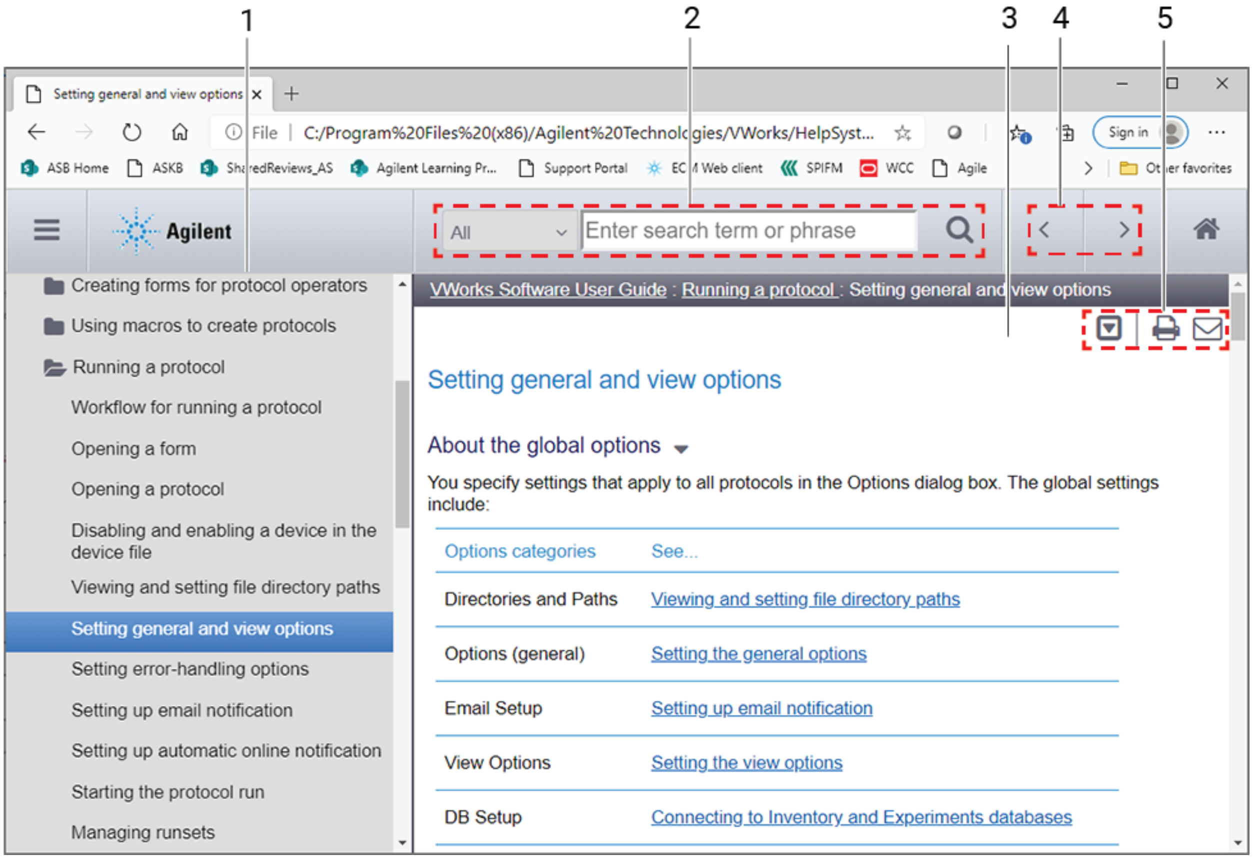
admin (Administrator) is logged in

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.

Features in the Knowledge Base window



- Creating forms for protocol operators
- Using macros to create protocols
- Running a protocol
  - Workflow for running a protocol
  - Opening a form
  - Opening a protocol
  - Disabling and enabling a device in the device file
  - Viewing and setting file directory paths
  - Setting general and view options
  - Setting error-handling options
  - Setting up email notification
  - Setting up automatic online notification
  - Starting the protocol run
  - Managing runsets

VWorks Software User Guide : Running a protocol : Setting general and view options

Setting general and view options

About the global options

You specify settings that apply to all protocols in the Options dialog box. The global settings include:

Options categories	See...
Directories and Paths	Viewing and setting file directory paths
Options (general)	Setting the general options
Email Setup	Setting up email notification
View Options	Setting the view options
DB Setup	Connecting to Inventory and Experiments databases

Step	For this task...
1	*Contents pane*. Lists all the books and the table of contents of the books.
2	*Search*. Allows you to search the Knowledge Base (all products or selected products) using keywords.
3	*Topic area*. Displays the selected online help topic.
4	*Navigation buttons*. Enable you to navigate through the next or previous topics listed in the Contents tab.
5	*Toolbar buttons*: Enable you to: • Expand or collapse all the sections in a topic that has drop-down headings. • Print the topic. • Send feedback by email for a given topic.

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 guide defines the interfaces and enumerations provided for developing applications that control VWorks software using the VWorks 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 33.	Works.exe

This page is intentionally blank.

2 Methods

The methods defined in this section are members of the IVWorks4API interface, which is included in the VWorks 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
CloseDeviceFile	Closes the specified device file.	"CloseDeviceFile" on page 6
`CloseProtocol`	Closes the specified protocol file.	"CloseProtocol method" on page 7
`CompileProtocol`	Compiles the specified protocol.	"CompileProtocol method" on page 8
`EnumerateUsers`	This method has been deprecated and is no longer used.	NA
`GetSimulationMode`	Gets the simulation mode state.	"GetSimulationMode method" on page 10
`GetTipStates`	Gets the state of the tip boxes in the specified protocol for automated tip tracking.	"GetTipStates method" on page 11
LoadDeviceFile	Loads the specified device file.	"LoadDeviceFile" on page 14
`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 in to VWorks using the specified user name and password.	"Login method" on page 18
`LoginWithDomain`	Logs in to VWorks using the specified user name, password and Domain. This method is available when Windows Domain authentication is configured in the OpenLab Control Panel.	"LoginWithDomain" on page 19

Method	Description	See...
`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
`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 diagnostics dialog box.	“ShowDiagsDialog method” on page 27
`ShowLoginDialog`	Displays the User Authentication (login) dialog box.	“ShowLoginDialog method” on page 28
`ShowManageUserDialog`	This method has been deprecated and is no longer used.	NA
`ShowOptionsDialog`	Displays the Options dialog box.	“ShowOptionsDialog method” on page 29
`ShowPlateGroupEditorDialog`	Displays the Plate Groups tab in the Inventory Editor.	“ShowPlateGroupEditorDialog method” on page 30
`ShowTipStateEditor`	Displays the Tip State Editor dialog box.	“ShowTipStateEditor method” on page 31
`ShowVWorks`	Displays or hides the VWorks main window.	“ShowVWorks method” on page 32

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 45.

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 7
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 38
ResumeProtocol method	“ResumeProtocol method” on page 23
RunProtocol method	“RunProtocol method” on page 24

CloseDeviceFile

Description

Closes the specified device file.

Syntax

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

Parameters

`device file`	[in] The path of the device file to close. The device file (.dev) must be in the OpenLab Shared Services storage.

Return value

The `CloseDeviceFile` method returns the method-call status of type `V11ReturnCode`. For possible values, see "V11ReturnCode enumerated type" on page 45.

Sample code

Visual C++

```
VWorks4Lib.V11ReturnCode retCode;
retCode = oVWorksCOM-> CloseDeviceFile("VWorks Projects/VWorks/
General/Devices/My Device File 1.dev");
```

Visual Basic .NET

```
Dim retCode as VWorks4Lib.V11ReturnCode
retCode = oVWorksCOM.CloseDeviceFile ("VWorks Projects/VWorks/
General/Devices/My Device File 1.dev");
```

Related information

For information about...	See...
LoadDeviceFile	"LoadDeviceFile" on page 14

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. The protocol file (.pro) must be in the OpenLab Shared Services storage.

Return value

The `CloseProtocol` function method returns the method-call status of type `V11ReturnCode`. For possible values, see "V11ReturnCode enumerated type" on page 45.

Sample code

Visual C++

```
VWorks4Lib.V11ReturnCode retCode;
retCode = oVWorksCOM->CloseProtocol("VWorks Projects/VWorks/
General/Protocols/My Protocol File 1.pro");
```

Visual Basic .NET

```
Dim vwRetCode As VWorks4Lib.V11ReturnCode
vwRetCode = oVWorksCOM.CloseProtocol("VWorks Projects/VWorks/
General/Protocols/My Protocol File 1.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

For information about...	See...
PauseProtocol method	"PauseProtocol method" on page 21
ResumeProtocol method	"ResumeProtocol method" on page 23
RunProtocol method	"RunProtocol method" on page 24

CompileProtocol method

Description

Compiles the specified protocol. This method is used with the `LogMessage` event. See "LogMessage event" on page 35.

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. The protocol file (.pro) must be in the OpenLab Shared Services storage.
`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 45.

Sample code

Visual C++

```
VWorks4Lib.V11ReturnCode retCode;
LONG errCount, wrnCount;
retCode = oVWorksCOM->CompileProtocol (“VWorks Projects/VWorks/General/Protocols/My
Protocol File 1.pro”, &errCount, &wrnCount);
```

Visual Basic .NET

```
Dim retCode as VWorks4Lib.V11ReturnCode
Dim errCount, wrnCount as Long
retCode = oVWorksCOM.CompileProtocol (“VWorks Projects/VWorks/General/Protocols/My
Protocol File 1.pro”, errCount, wrnCount)
```

Related information

For information about...	See...
AbortProtocol method	“AbortProtocol method” on page 5
CloseProtocol method	“CloseProtocol method” on page 7
LoadProtocol method	“LoadProtocol method” on page 15
LogMessage event	“LogMessage event” on page 35
PauseProtocol method	“PauseProtocol method” on page 21
ResumeProtocol method	“ResumeProtocol method” on page 23
RunProtocol method	“RunProtocol method” on page 24

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

GetTipStates method

Description

Gets the state of the tip boxes 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. The protocol file (.pro) must be in the OpenLab Shared Services storage.
`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>
```

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 tip box is configured labware `1` = The tip box is process labware
`ProcessOrDeviceName`	For process labware, the value is the name of the process by which the tip box enters the system. For configured labware, the value is the name of the device on which the tip box 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

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 tip box 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'
<rarr RightArrow>®ProcessOrDeviceName='Bravo - 1' >
            <TipBoxState NumWells='384' >
               <PipetteHeadMode Channels='1' ColumnCount='1' RowCount='1'
<rarr RightArrow>®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 45.

Sample code

Visual Basic .NET

```
Dim TipStateXML As String = ""
Dim retCode As VWorks4Lib.V11ReturnCode
retCode = oVWorksCOM.GetTipStates("VWorks Projects/VWorks/General/Protocols/My
Protocol File 1.pro", TipStateXML)
```

LoadDeviceFile

Description

Loads the specified device file.

Syntax

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

Parameters

`device file`	[in] The path of the device file to load. The device file (.dev) must be in the OpenLab Shared Services storage.

Return value

The `LoadDeviceFile` method returns the method-call status of type `V11ReturnCode`. For possible values, see "V11ReturnCode enumerated type" on page 45.

Sample code

Visual C++

```
VWorks4Lib.V11ReturnCode retCode;
retCode = oVWorksCOM-> LoadDeviceFile("VWorks Projects/VWorks/
General/Devices/My Device File 1.dev");
```

Visual Basic .NET

```
Dim retCode as VWorks4Lib.V11ReturnCode
retCode = oVWorksCOM.LoadDeviceFile ("VWorks Projects/VWorks/
General/Devices/My Device File 1.dev");
```

Related information

For information about...	See...
CloseDeviceFile	"CloseDeviceFile" on page 6

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. The protocol file (.pro) must be in the OpenLab Shared Services storage.

Return value

The `LoadProtocol` method returns the method-call status of type `V11ReturnCode`. For possible values, see "V11ReturnCode enumerated type" on page 45.

Sample code

Visual C++

```
VWorks4Lib.V11ReturnCode retCode;
retCode=oVWorksCOM->LoadProtocol("VWorks Projects/VWorks/General/
Protocols/My Protocol File 1.pro");
```

Visual Basic .NET

```
Dim retCode as VWorks4Lib.V11ReturnCode
retCode=oVWorksCOM.LoadProtocol ("VWorks Projects/VWorks/General/
Protocols/My Protocol File 1.pro")
```

Related information

For information about...	See...
AbortProtocol method	"AbortProtocol method" on page 5
CloseProtocol method	"CloseProtocol method" on page 7
CompileProtocol event	"CompileProtocol method" on page 8

For information about...	See...
PauseProtocol method	"PauseProtocol method" on page 21
ResumeProtocol method	"ResumeProtocol method" on page 23
RunProtocol method	"RunProtocol method" on page 24

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. The runset file (.rst) must be in the OpenLab Shared Services storage.

Return value

The `LoadRunsetFile` method returns the method-call status of type `V11ReturnCode`. For possible values, see “V11ReturnCode enumerated type” on page 45.

Sample code

Visual C++

```
VWorks4Lib.V11ReturnCode retCode;
retCode=oVWorksCOM->LoadRunsetFile (“VWorks Projects/VWorks/
General/Runsets/My Runset File 1.rst”);
```

Visual Basic .NET

```
Dim retCode as VWorks4Lib.V11ReturnCode
retCode=oVWorksCOM.LoadRunsetFile (“VWorks Projects/VWorks/
General/Runsets/My Runset File 1.rst”)
```

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

These parameters are for a user account that has been created or configured in OpenLab Control Panel.

`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 45.

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!”)
```

Related information

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

LoginWithDomain

Description

Logs in to VWorks using the specified user name, password and Domain. This method is available when Windows Domain authentication is configured in the OpenLab Control Panel.

Syntax

```
HRESULT LoginWithDomain ([in] BSTR userName,
   [in] BSTR password , [in] BSTR domain ,
   [out, retval] enum V11ReturnCode* returnCode);
```

Parameters

`userName`	The name of the user to log-in.
`password`	The password for the user account.
`domain`	The domain name to be used for authentication.

Return value

The `LoginWithDomain` method returns the login status of type `V11LoginResult`. For possible values, see "V11ReturnCode enumerated type" on page 45.

Sample code

Visual C++

```
VWorks4Lib.V11ReturnCode retCode;
retCode = oVWorksCOM-> LoginWithDomain("user1", "mypassword!",
"Agilent");
```

Visual Basic .NET

```
Dim retCode as VWorks4Lib.V11ReturnCode
retCode = oVWorksCOM.LoginWithDomain("user1", "mypassword!",
"Agilent");
```

Related information

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

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 45.

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

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 45.

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 7
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

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 45.

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 34

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 45.

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 7
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

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. The protocol file (.pro) must be in the OpenLab Shared Services storage.
`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 45.

Sample code

Visual C++

```
VWorks4Lib.V11ReturnCode retCode;
retCode=oVWorksCOM->RunProtocol (“VWorks Projects/VWorks/General/
Protocols/My Protocol File 1.pro”,2);
```

Visual Basic .NET

```
Dim retCode as VWorks4Lib.V11ReturnCode
retCode=oVWorksCOM.RunProtocol (“VWorks Projects/VWorks/General/
Protocols/My Protocol File 1.pro”,2)
```

Related information

For information about...	See...
AbortProtocol method	"AbortProtocol method" on page 5
CloseProtocol method	"CloseProtocol method" on page 7
CompileProtocol method	"CompileProtocol method" on page 8
LoadProtocol method	"LoadProtocol method" on page 15
PauseProtocol method	"PauseProtocol method" on page 21
ResumeProtocol method	"ResumeProtocol method" on page 23

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 45.

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 10

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 45.

Sample code

Visual C++

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

Visual Basic .NET

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

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 45.

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

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 45.

Sample code

Visual C++

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

Visual Basic .NET

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

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 45.

Sample code

Visual C++

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

Visual Basic .NET

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

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. The protocol file (.pro) must be in the OpenLab Shared Services storage.

Return value

The `ShowTipStateEditor` method returns the method-call status of type `V11ReturnCode`. For possible values, see “V11ReturnCode enumerated type” on page 45.

Sample code

Visual C++

```
VWorks4Lib.V11ReturnCode retCode;
retCode=oVWorksCOM->ShowTipStateEditor (“VWorks Projects/VWorks/
General/Protocols/My Protocol File 1.pro”);
```

Visual Basic .NET

```
Dim retCode as VWorks4Lib.V11ReturnCode
retCode=oVWorksCOM.ShowTipStateEditor (“VWorks Projects/VWorks/
General/Protocols/My Protocol File 1.pro”)
```

Related information

For information about...	See...
GetTipStates method	“GetTipStates method” on page 11

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 45.

Sample code

Visual C++

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

Visual Basic .NET

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

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 34
`LogMessage`	Fires when a message is posted to the Main Log.	"LogMessage event" on page 35
`MessageBoxAction`	Fires when a user response is required.	"MessageBoxAction event" on page 36
`ProtocolAborted`	Fires when the operator or automation client aborts the protocol run.	"ProtocolAborted event" on page 38
`ProtocolComplete`	Fires after the Startup Protocol, Cleanup Protocol, or Main Protocol is finished.	"ProtocolComplete event" on page 39
`RecoverableError`	Fires when an error occurs to which the user can respond with abort, retry, or ignore.	"RecoverableError event" on page 40
`UnrecoverableError`	Fires when an error occurs to which no user response is required.	"UnrecoverableError event" on page 41
`UserMessage`	Fires when a User Message task occurs.	"UserMessage event" on page 42

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

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.

Related information

For information about...	See...
CompileProtocol method	"CompileProtocol method" on page 8

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.

`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`

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

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`

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

Related information

For information about...	See...
UnrecoverableError event	"UnrecoverableError event" on page 41

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 40

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.

4 Enumerations

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

This chapter contains the following topics:

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.

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.

www.agilent.com

Febrauary 2021

D0008025   Revision A