VWorks Plugin

Developer Guide

Agilent Technologies

Notices

User Guide Part Number

G5415-90065

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®, Windows®, and Visual Studio®, 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.

Contents

Preface iii
Who should read this guide? iv
What this guide covers v
Accessing Automation Solutions user guides vii

1. Introduction 1
VWorks plugin fundamentals 2
XML metadata terminology 5

2. Writing a plugin 9
Creating a new plugin project 10

3. IWorksDriver interface 15
IWorksDriver methods overview 17
Abort method 19
Close method 20
Command method 21
Compile method 25
ControllerQuery method 29
Get32x32Bitmap method 34
GetDescription method 36
GetErrorInfo method 40
GetLayoutBitmap method 42
GetMetaData method 45
Ignore method 59
Initialize method 61
IsLocationAvailable method 64
MakeLocationAvailable method 67
LocationAvailable XML block components 70
PlateDroppedOff method 75
PlatePickedUp method 77
PlateTransferAborted method 79
Plates XML block components 81
PrepareForRun method 83
Retry method 85

4. IControllerClient interface 87
SetController method 88

5. IWorksDiags interface 91
IWorksDiags methods overview 92
CloseDiagsDialog method 93
ShowDiagsDialog method 94

6. IBCRDriver interface . . . 97
Strobe method . . . 98

7. IIODriver interface . . . 101
IIODriver methods overview . . . 102
EnumPoints method . . . 103
Read method . . . 105
Set method . . . 108

8. ILabelerDriver interface . . . 111
ILabelerDriver methods overview . . . 112
EnumerateFormats method . . . 113
Print method . . . 115
PrintAndApply method . . . 117
PrinterMetaData XML block components . . . 119

9. ILiddingDriver interface . . . 121
ILiddingDriver methods overview . . . 122
LidIsRetained method . . . 123
OnDelidMoveComplete method . . . 124
OnRelidMoveComplete method . . . 127
RobotEndsUpHoldingPlate method . . . 130

10. IMeasurementDriver interface . . . 131
IMeasurementDriver methods overview . . . 132
GetMeasurement method . . . 133
GetMeasurementTypes method . . . 134

11. IPipetteDriver interface . . . 139

12. IRobotDriver interface . . . 141
IRobotDriver methods overview . . . 142
CheckPlatePresent method . . . 143
DelidRelid method . . . 146
GetPlatePresentResult method . . . 150
GetSimulationTimes method . . . 153
GetTeachPoints method . . . 156
Move method . . . 159
SetSpeed method . . . 162

13. ISpinDriver interface . . . 165
SpinCycle method . . . 166

14. IStackerDriver interface . . . 169
IStackerDriver methods overview . . . 170
IsStackEmpty method . . . 171
IsStackFull method . . . 172
LoadStack method . . . 173

ScanStack method . . . 175
SinkPlate method . . . 176
SourcePlate method . . . 178
UnloadStack method . . . 180

15. IStorageDriver interface . . . 183
IStorageDriver methods overview . . . 184
LoadPlate method . . . 185
LookupLocations method . . . 188
QueryStorageLocations method . . . 192
UnloadPlate method . . . 194

16. IVHooks interface . . . 197
IVHooks methods overview . . . 199
IVHooks interface methods output . . . 201
Aborted method . . . 203
BarCodeMisread method . . . 205
BarCodeRead method . . . 210
CompileComplete method . . . 214
CustomHook method . . . 216
Deadlock method . . . 218
Error method . . . 220
FileOpened method . . . 222
FileSaved method . . . 224
GetUserInterface method . . . 226
LiquidTransferComplete method . . . 227
ProcessFinished method . . . 231
ProcessStarting method . . . 234
ProtocolFinished method . . . 237
ProtocolPaused method . . . 239
ProtocolStarted method . . . 241
RobotMove method . . . 243
RobotPickComplete method . . . 246
RobotPlaceComplete method . . . 248
ScriptPlateError method . . . 250
TaskFinished method . . . 252
TaskStarting method . . . 255
UserLoggedIn method . . . 258
UserLoggedOut method . . . 260

17. IWorksAsyncDriver interface . . . 263
IWorksAsyncDriver methods overview . . . 264
Abort method . . . 265
GetListOfAsyncTasks method . . . 269
Ignore method . . . 271
Retry method . . . 274
Asynchronous Task Command XML block components . . . 277

18. IWorksController interface 281
IWorksController methods overview 283
NotifyDataChanged method 285
NotifyTipOperation method 287
OnCloseDiagsDialog method 290
PrintToLog method 291
Query method 292
Update method 347

19. IWorksProfiles interface 373
IWorksProfiles methods overview 374
ExportXML method 375

20. Enumerations 379
CompileType enumerated type 380
MetaDataType enumerated type 381
PauseType enumerated type 382
PlateFlagsType enumerated type 383
ReturnCode enumerated type 384
SecurityLevel enumerated type 386
TIMER_MODES enumerated type 387

21. Testing and debugging 389
Testing your VWorks plugin 390
Testing plugins that implement the IVHooks interface 394

A. Common elements and attributes 397
Command element 399
Device element 403
Location element 407
MetaData element 409
Parameter element 409
PipetteHeadMode element 414
Range element 415
Ranges element 416
Velocity11 element 416
Well element 417
Wells element 418
WellSelection element 418

B. Methods Terminology 421

C. Obsolete, deprecated, and reserved for internal use methods 423
Obsolete methods 423
Deprecated methods 424
Reserved for internal use methods 425

<Go Back **Glossary** 427

Index 429

<Go Back

Preface

This preface contains the following topics:

- "Who should read this guide?" on page iv
- "What this guide covers" on page v
- "Accessing Automation Solutions user guides" on page vii

<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
- Working knowledge of how to define, validate, read, and process Extensible Markup Language (XML) documents
- Working knowledge of integrating with Laboratory Information Systems (LIMS)
- Familiarity with VWorks software features and functionality
- Expert knowledge of how to control their devices

What this guide covers

What is covered

This guide defines the VWorks software interfaces, methods, and enumerated types needed to create VWorks plugins for third-party devices. This guide also explains how to use the IWorksTest utility for debugging and testing plugins.

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 functionality, including the user interface. More information about VWorks software is available in the *VWorks Automation Control Setup Guide* and the *VWorks Automation Control User Guide*.

Software version

This guide documents the interfaces used to develop plugins for VWorks software version 11.

What's new in this revision

Feature and description	See
The descriptions of IWorks Device Driver and VWorks Hooks interfaces were combined into a single guide.	
The "Writing XML metadata for IWorks software" chapter was deleted and most of its contents were moved to the "GetMetaData method" section of the "IWorksDriver interface" chapter. The title of the "VHooks software interface reference" chapter was changed to "IVHooks interface," and the document was otherwise reorganized.	• IWorksDriver "GetMetaData method" on page 45 • "IVHooks interface" on page 197
A summary of all VWorks Plugin interfaces was added to the "Introduction" chapter.	"Introduction" on page 1
Some terminology was changed and other existing terminology has been defined.	• "XML metadata terminology" on page 5 • "IVHooks interface methods output" on page 201 • "Methods Terminology" on page 421

<Go Back

Feature and description	See
Eight interfaces were added.	• "IBCRDriver interface" on page 97 • "IIODriver interface" on page 101 • "ILabelerDriver interface" on page 111 • "ILiddingDriver interface" on page 121 • "IMeasurementDriver interface" on page 131 • "ISpinDriver interface" on page 165 • "IWorksAsyncDriver interface" on page 263 • "IWorksDiags interface" on page 91
One existing interface has been reserved for internal use.	"IPipetteDriver interface" on page 139
One method was added to the IWorksDriver interface.	"IWorksDriver interface" on page 15
Seven methods were added to the IRobotDriver interface.	"IRobotDriver interface" on page 141
One method was added to the IStackerDriver interface.	"IStackerDriver interface" on page 169
Three methods were added to the IWorksController interface, seven categories were added for the Query method and one is obsolete, and six categories were added for the Update method and one is reserved for internal use.	"IWorksController interface" on page 281
One enumerated type was added.	"Enumerations" on page 379
Twelve methods are obsolete, two methods are deprecated, and one method is reserved for internal use.	"Obsolete, deprecated, and reserved for internal use methods" on page 423

Related guides

The VWorks Plugin Developer Guide should be used in conjunction with:

- *VWorks Automation Control Setup Guide*
- *VWorks Automation Control User Guide*
- Agilent Technologies device user guides
- Third-party device user documents

<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.

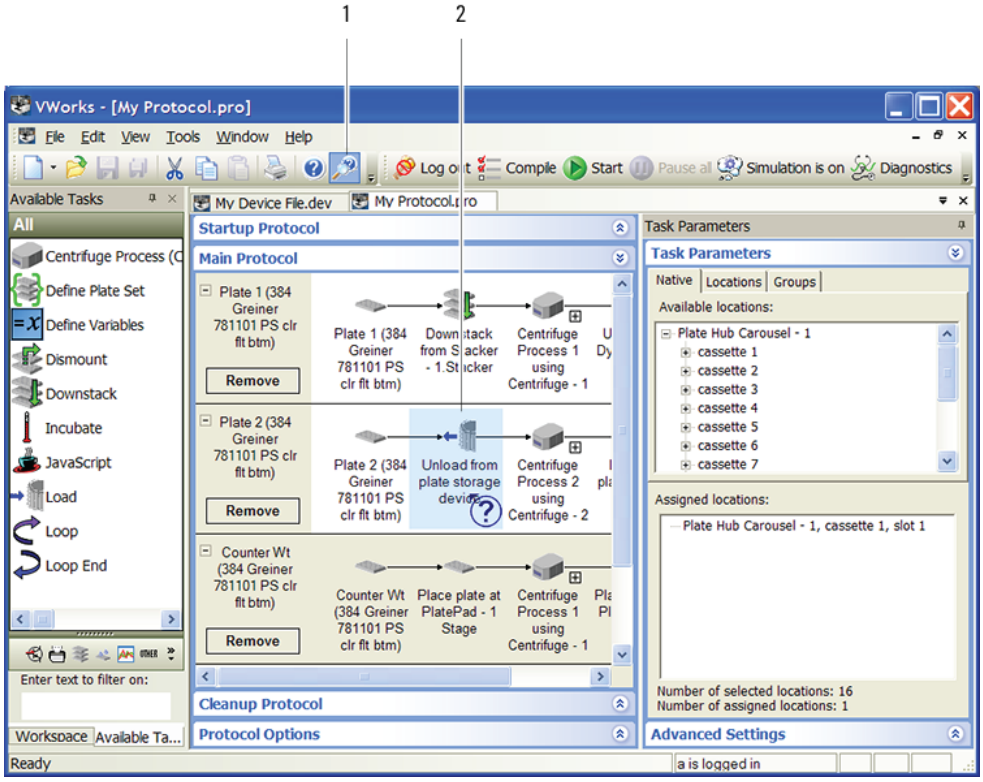
Note: If you want to use Internet Explorer to display the topics, you might have to allow local files to run active content (scripts and ActiveX controls). To do this, in Internet Explorer, open the **Internet Options** dialog box. Click the **Advanced** tab, locate the **Security** section, and select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

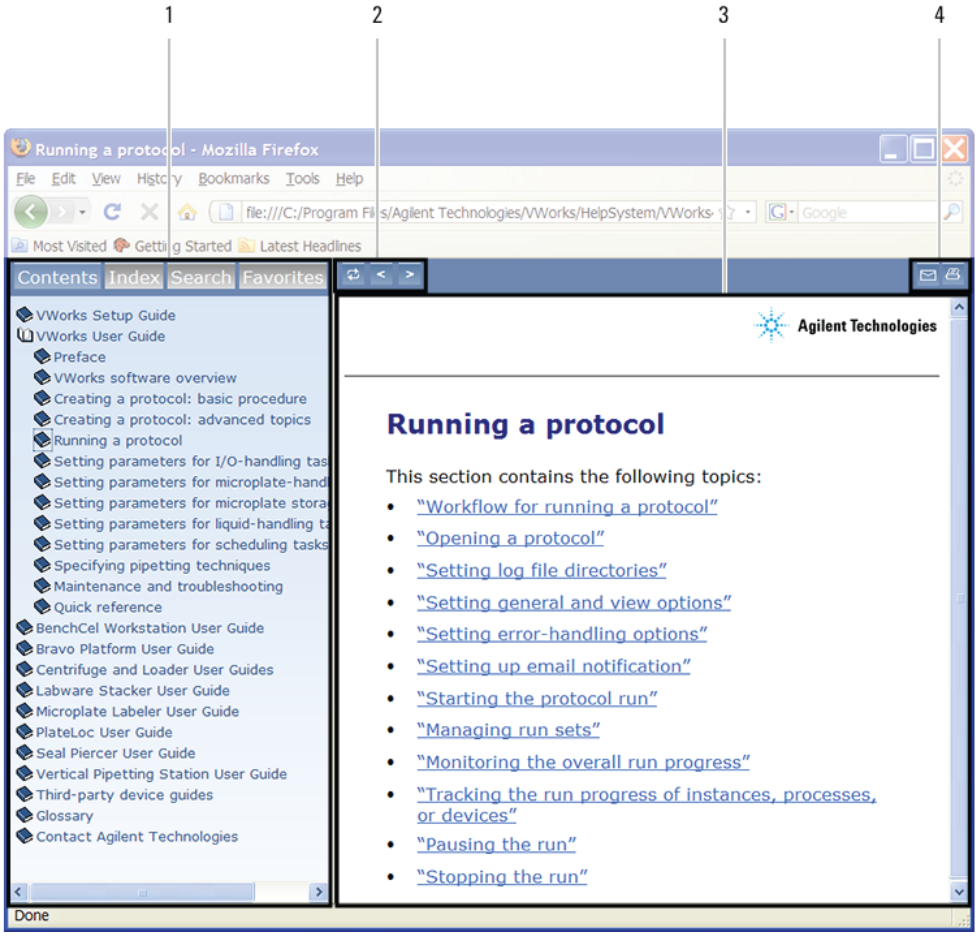


To access the context-sensitive help feature:

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.

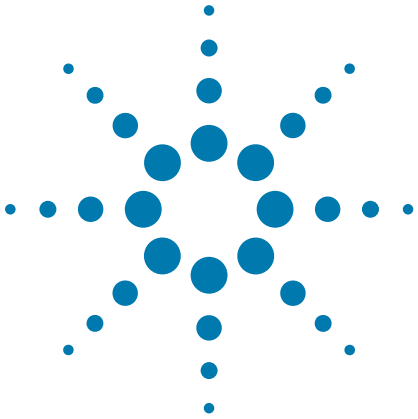
<Go Back

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.

VWorks software allows developers to create plugins in any programming language capable of generating COM components for controlling devices, interacting with data sources, and monitoring activity within VWorks software. These custom VWorks plugins enable VWorks software to do the following:

- Communicate with third-party devices
- Integrate with backend data systems such as Laboratory Information Management Systems (LIMS), sample management, and workflow management

VWorks software also includes the IWorksTest utility, which is a diagnostics tool designed to debug and test the basic functionality of VWorks plugins.

This developer guide describes the required and optional COM interfaces that a plugin developer would implement to extend VWorks software.

<Go Back

VWorks plugin fundamentals

The rest of this chapter provides the following basic information:

- An explanation of how information is exchanged between VWorks software and the plugin
- Definitions of the COM interfaces used for creating VWorks plugins

Information exchange

Information is exchanged between VWorks software and a plugin by passing XML metadata strings as parameters in COM interface methods.

To work with VWorks software, a plugin must do the following:

- Parse XML strings that VWorks software provides as input parameters in the COM interface methods
- Construct well-formed XML strings as output parameters that are returned to VWorks software by the COM interface methods

You can write your own utilities to build the XML strings, or you can use the Microsoft COM implementation of the XML Document Object Model (DOM) classes, which enables you to construct XML documents in memory. For more information about DOM, visit the Microsoft Developer Center at http://windowssdk.msdn.microsoft.com/en-us/library/ms766487.aspx.

IMPORTANT All XML strings must be well-formed, or the plugin will fail to load when VWorks software is started.

Interfaces overview

VWorks software defines the COM interfaces that are listed in the following table. All VWorks device driver plugins must implement the IWorksDriver, IControllerClient, and IWorksDiags interfaces. The remaining interfaces are optional and can be implemented by the plugins that require them.

Interface name	Purpose	Type library	Req'd
"IBCRDriver interface" on page 97	VWorks plugins that control barcode readers must implement the IBCRDriver interface.	IWorksDriver.dll	No
"IControllerClient interface" on page 87	All VWorks plugins must implement the IControllerClient interface to get a pointer from VWorks software to the IWorksController interface.	IWorksDriver.dll	Yes
"IIODriver interface" on page 101	VWorks plugins that manage I/O signals must implement the IIODriver interface.	IWorksDriver.dll	No
"ILabelerDriver interface" on page 111	VWorks plugins that perform labware-labeling tasks must implement the ILabelerDriver interface.	IWorksDriver.dll	No

<Go Back

Interface name	Purpose	Type library	Req'd
"ILiddingDriver interface" on page 121	VWorks plugins that perform delidding and relidding operations must implement the ILiddingDriver interface.	IWorksDriver.dll	No
"IMeasurementDriver interface" on page 131	VWorks plugins that return measurement values and monitor device measurements must implement the IMeasurementDriver interface.	IWorksDriver.dll	No
"IPipetteDriver interface" on page 139	*Reserved for internal use.* VWorks plugins should not implement the IPipetteDriver interface.	IWorksDriver.dll	Not used
"IRobotDriver interface" on page 141	VWorks plugins that use robots to move labware must implement the IRobotDriver interface.	IWorksDriver.dll	No
"ISpinDriver interface" on page 165	VWorks plugins that perform Centrifuge tasks must implement the ISpinDriver interface.	IWorksDriver.dll	No
"IStackerDriver interface" on page 169	VWorks plugins that control labware stacker devices must implement the IStackerDriver interface.	IWorksDriver.dll	No
"IStorageDriver interface" on page 183	VWorks plugins that control labware storage devices must implement the IStorageDriver interface.	IWorksDriver.dll	No
"IVHooks interface" on page 197	VWorks plugins that want to act on events in VWorks software must implement the IVHooks interface.	VHooksInterface.dll	No
"IWorksAsyncDriver interface" on page 263	VWorks plugins that perform simultaneous tasks must implement the IWorksAsyncDriver interface.	IWorksDriver.dll	No
"IWorksController interface" on page 281	VWorks software implements the IWorksController interface. Plugins must call IWorksController methods to do the following: • Notify VWorks software about user activities • Write messages to the Main Log • Communicate with another plugin using VWorks software as the intermediary • Tell VWorks software to perform certain actions • Request information from VWorks software • Return information to VWorks software	IWorksDriver.dll	No

Interface name	Purpose	Type library	Req'd
"IWorksDiags interface" on page 91	To open diagnostics dialog boxes in VWorks software, all VWorks device driver plugins must implement the IWorksDiags interface.	IWorksDriver.dll	Yes
"IWorksDriver interface" on page 15	All VWorks device driver plugins must implement the IWorksDriver interface. VWorks software calls IWorksDriver methods to do the following: • Tell the plugin to initialize or close a device • Notify the plugin of a user action • Tell the plugin to execute a task • Get an icon or description of a device or task from the plugin • Manage labware-handling processes • Get the description of the error that occurred when the plugin returned an error code • Log errors and warnings reported by the plugin during protocol compilation • Provide the means for two plugins to communicate with each other	IWorksDriver.dll	Yes
"IWorksProfiles interface" on page 373	To export nonstandard registry and file data associated with a profile, VWorks plugins must implement the IWorksProfiles interface.	IWorksDriver.dll	No

XML metadata terminology

This section defines the terminology used in this guide to describe the XML metadata structures and values used by VWorks software.

XML blocks and XML elements

The following terms describe the structure of the XML metadata that is passed in the input and output parameters of VWorks Plugin methods.

Note: Normally, non-XML data is not documented in the methods sections.

XML structures

VWorks software uses the following XML structures, which are passed as strings to VWorks Plugin methods:

- XML block

  An XML block is a portion of an XML byte string that contains a parent element and all its children. All XML blocks include the XML declaration and most contain the `Velocity11` root element. In addition, some XML blocks contain the `MetaData` element.

  The following code is an example of a CompilerErrors XML block.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData'
→md5sum='849392019ca47102839e845113d11840' version='1.0'>
   <MetaData>
      <CompilerErrors>
         <CompilerError Value='Warning: xyz' ErrorType='1'>
         <CompilerError Value='Warning: xyzabc' ErrorType='1'>
      </CompilerErrors>
   </MetaData>
</Velocity11>
```

- XML element

  An XML element is a portion of an XML byte string that contains an element that has no children. All XML elements include the XML declaration and most contain the `Velocity11` root element.

  The following code is an example of an analogInput XML element.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Velocity11'
→md5sum='cc239a8f99203e73b3de0ed8a058f77e' version='1.1' >
   <analogInput name='Humidity' value='70' />
</Velocity11>
```

Escaped XML blocks

Whenever an XML block is used as the value of an XML attribute, the following five special characters that are used in XML markup should be escaped: <, >, &, ', and ".

Using escaped XML blocks in `Parameter` elements instead of specifying additional attributes is recommended. This simplifies the user interface and allows for future expansion.

<Go Back

For sample code that shows an escaped XML block, see "Example of an AllDeviceInfo query response" on page 299.

Empty XML blocks and elements

This section contains examples of empty XML structures.

XML block

If an XML block is empty, only its empty parent element is specified.

The following code is an example of an empty DeviceLocationTeachpoints XML block.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='a16eb3001ae9fcd2ff3fd34654eba2d5' version='1.0' >
   <DeviceLocationTeachpoints />
</Velocity11>'
```

Query and Update XML blocks

An empty Query or Update XML block includes the parent element's `Category` attribute.

The following code is an example of an empty Query XML block.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Query' md5sum='caa52e1e5fae3c9dbea6cdc245eb3485' version='1.0' >
   <Query Category='GetDeviceName' />
</Velocity11>
```

The following code is an example of an empty Update XML block.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Update' md5sum='0b8eb2ad5df221c13e336af98ffec528' version='1.0' >
    <Update Category='InventoryPlateBarcodes' />
</Velocity11>
```

XML element

If an XML element is empty, none of its element's attributes is specified.

The following code is an example of an empty RobotPickComplete XML element.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='3c2d69b02b8e7505682cfb59c06a0105' version='1.0' >
   <RobotPickComplete />
</Velocity11>
```

XML attribute values

In this guide, some XML attribute values are identified as *not specified* or as having *no value*.

Not specified

If an attribute is not specified, it is not present in the code.

<Go Back

In the following code, the `Labware` element is not specified for the third and fourth `Layout` elements.

```
<Layout Labware='Labware Type 1' Name='Location 1' />
<Layout Labware='Labware Type 2' Name='Location 2' />
<Layout Name='Location 3' />
<Layout Name='Location 4' />
```

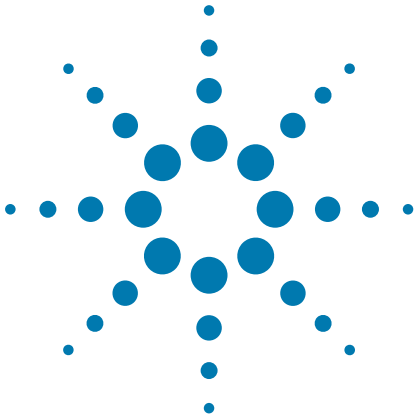
No value

If an attribute has no value, its value is empty.

In the following code, the `OptionalDevice` attribute has no value.

```
<VolumeUpdates Location='Stage 1' OptionalDevice='' ResetAbsolute='0' >
```

<Go Back



2 Writing a plugin

Agilent Technologies recommends that you develop your plugin using Microsoft Visual Studio. This chapter contains instructions for starting a plugin project in Visual Basic.NET and in C# using Visual Studio 2008. If you are using a different version of Visual Studio, refer to the documentation for that version.

<Go Back

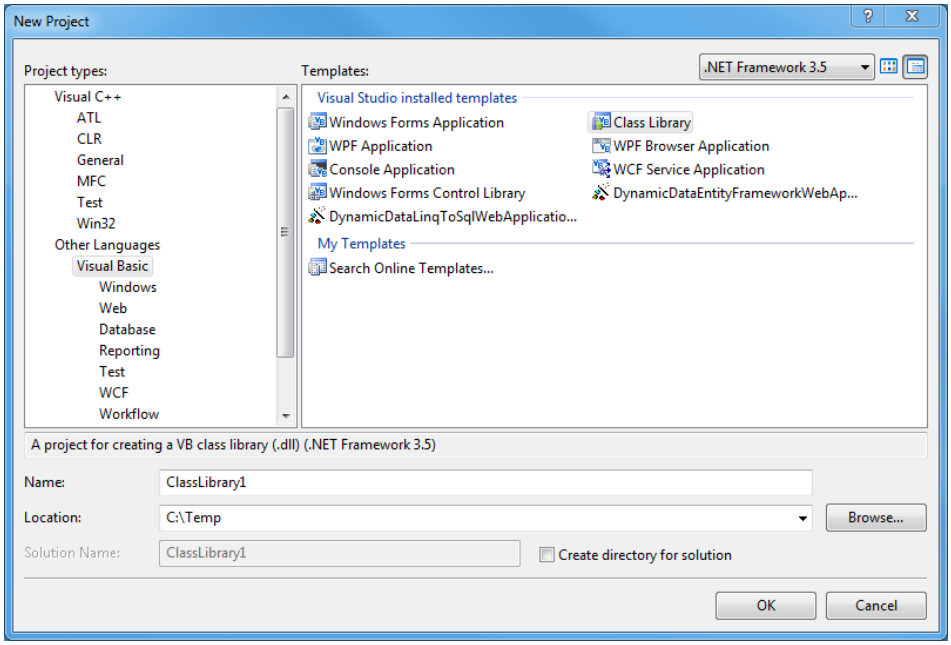
Creating a new plugin project

Starting a project in Visual Basic .NET

To start a plugin project in Visual Basic .NET:

1 Start Visual Studio.

2 To create a new project, select **File** > **New** > **Project**.



3 For a Visual Basic project type, select **Class Library** from the templates, enter a **Name** for your project, and then click **OK**.

4 To successfully interact with VWorks Plugin, you must tell the compiler to generate a type library (TLB) and register it with COM as follows:

   a Edit the properties for your project by selecting **Project** > ***projectname*** **Properties**.

   b Click the **Compile** tab, and then select the **Register for COM interop** check box.

   c Save your changes.

5 Add a reference to the COM interface in your project as follows:

   a Select **Project** > **Add Reference.**

   b Click the **COM** tab.

   c Select the IWorksDriver 1.0 Type Library (IWorksDriver.dll).

   d If you want to implement the IVHooks interface, select the VHooksInterface 1.0 Type Library (VHooksInterface.dll).

   Note: If you do not see the type library on the list, click the **Browse** button and then navigate to the ...\Agilent Technologies\VWorks folder.

   e Click **OK**.

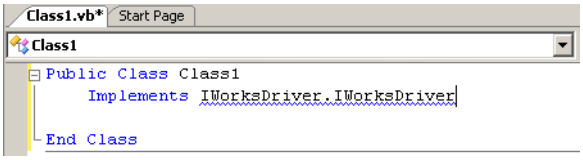
<Go Back

6 In the code window, add an Implements statement on the line after the `Class` statement in the following format:

```
Implements Interfacename.Interfacemember
```

Refer to the table in "Interfaces overview" on page 2 for a list of VWorks Plugin interfaces. For all interfaces but IVHooks, use the `IWorksDriver` Namespace. For the IVHooks interface, use `VHookInterfaceLib`.

The following example shows the Implements statement for the IWorksDriver interface:



7 Press **Enter**.

Visual Studio automatically creates empty method bodies for each method defined in the interface.

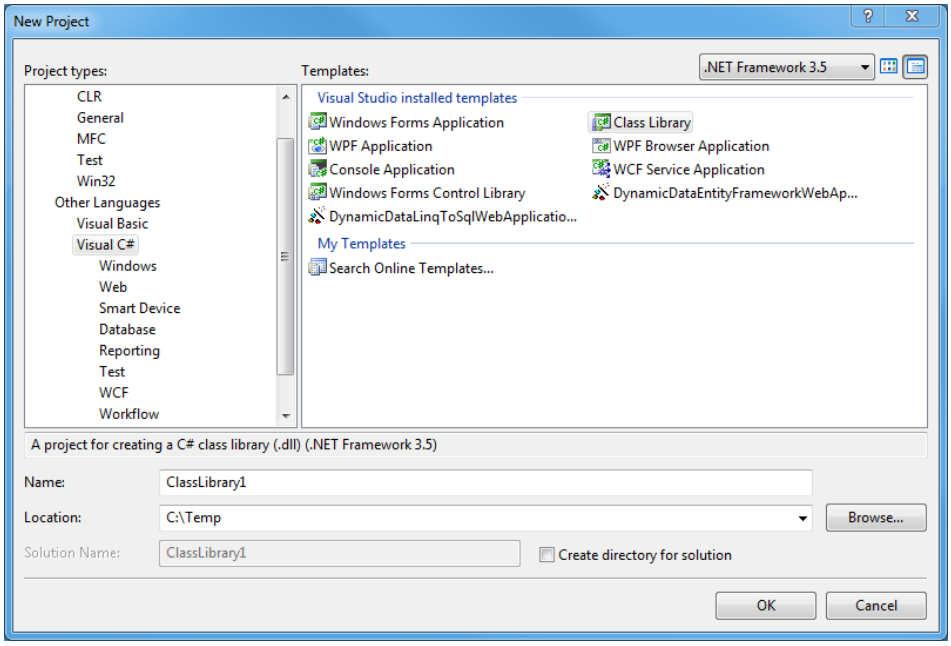
8 Add your code to implement the interface's methods.

Starting a project in C#

To start your project in C#:

1 Start Visual Studio.

2 To create a new file, select **File** > **New** > **Project**.



3 For a C# project type, select **Class Library** from the templates, enter a **Name** for your project, and then click **OK**.

4 To successfully interact with VWorks Plugin, you must tell the compiler to additionally generate a type library (TLB) and register it for COM interop as follows:

<Go Back

a Edit the properties for your project by selecting **Project** > ***projectname*** **Properties**.

b Click the **Build** tab, and then select the **Register for COM interop** check box.

c Click the **Application** tab, and then click **Assembly Information**.

d Select the **Make assembly COM-Visible** check box.

e Save your changes.

5 Add a reference to the COM interface in your project as follows:

a Select **Project** > **Add Reference.**

b Click the **COM** tab.

c Select the IWorksDriver 1.0 Type Library (IWorksDriver.dll).

d If you want to implement the IVHooks interface, select the VHooksInterface 1.0 Type Library (VHooksInterface.dll).

Note: If you do not see the type library on the list, click the **Browse** button and then navigate to the ...\Agilent Technologies\VWorks folder.

e Click **OK**.

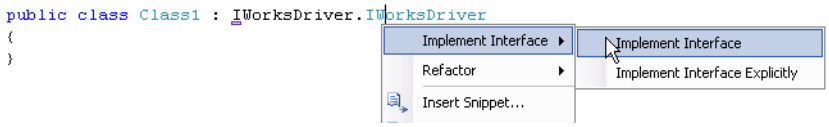
6 In the code window, add the interface declaration to the class at the end of the class declaration in the following format:

```
: Interfacename.Interfacemember
```

Right-click the Interfacemember and select **Implement Interface** > **Implement Interface**.

Visual Studio automatically creates empty method bodies for each method defined in the interface.

The following example shows the interface declaration for the IWorksDriver interface:



7 Add your code to implement the interface's methods.

Loading your plugin into VWorks software

To load your plugin into VWorks software for testing:

1 Compile your plugin and copy the appropriate *.dll files into the ...\Agilent Technologies\VWorks\Plugins folder as follows:

- For Visual Basic, copy the project *.dll and TLB files
- For C#, copy the project *.dll, TLB, and interop *.dll files

2 Now you can test your plugin using the IWorksTest utility and VWorks software. For more information about the IWorksTest utility, see "Testing and debugging" on page 389.

<Go Back

Related information

For information about...	See...
Defining the XML metadata for your plugin	“Information exchange” on page 2
Testing your VWorks software plugin	“Testing and debugging” on page 389

<Go Back

3 IWorksDriver interface

All VWorks device driver plugins must implement the IWorksDriver interface. VWorks software calls IWorksDriver methods to do the following:

- Tell the plugin to initialize or close a device
- Notify the plugin of a user action
- Tell the plugin to execute a task
- Get an image or description of a device or task from the plugin
- Manage labware-handling processes
- Get the description of the error that occurred when the plugin returned an error code
- Log errors and warnings reported by the plugin during protocol compilation
- Provide the means for two plugins to communicate with each other

This chapter defines the IWorksDriver methods.

IMPORTANT All VWorks device driver plugins must implement the IWorksDriver, IControllerClient, and IWorksDiags interfaces.

This chapter contains the following topics:

- "IWorksDriver methods overview" on page 17
- "Abort method" on page 19
- "Close method" on page 20
- "Command method" on page 21
- "Compile method" on page 25
- "ControllerQuery method" on page 29
- "Get32x32Bitmap method" on page 34
- "GetDescription method" on page 36
- "GetErrorInfo method" on page 40
- "GetLayoutBitmap method" on page 42
- "GetMetaData method" on page 45
- "Ignore method" on page 59
- "Initialize method" on page 61
- "IsLocationAvailable method" on page 64
- "MakeLocationAvailable method" on page 67
- "LocationAvailable XML block components" on page 70
- "PlateDroppedOff method" on page 75
- "PlatePickedUp method" on page 77
- "PlateTransferAborted method" on page 79

<Go Back

- "Plates XML block components" on page 81
- "PrepareForRun method" on page 83
- "Retry method" on page 85

<Go Back

IWorksDriver methods overview

Use the following table to quickly locate an IWorksDriver method by name, by description, or by page number.

Method	Description	See...
`Abort`	Tells the plugin to terminate a task or to terminate all currently executing tasks.	“Abort method” on page 19
`Close`	Tells the plugin to terminate the connection to a device.	“Close method” on page 20
`Command`	Tells the plugin to execute a task.	“Command method” on page 21
`Compile`	Notifies the plugin of the state of a protocol’s compile sequence.	“Compile method” on page 25
`ControllerQuery`	Provides the means for plugin-to-plugin communication.	“ControllerQuery method” on page 29
`Get32x32Bitmap`	Gets a 32x32 bitmap image from the plugin to display for a device or task in the protocol editor.	“Get32x32Bitmap method” on page 34
`GetDescription`	Gets a dynamic description or a detailed description for a task from the plugin.	“GetDescription method” on page 36
`GetErrorInfo`	Gets a text string describing the error that occurred when the plugin returned an error code; writes the string to the Main Log; and displays the standard error dialog box, which contains the string.	“GetErrorInfo method” on page 40
`GetLayoutBitmap`	Gets a dynamic bitmap from the plugin that shows the labware located on a device.	“GetLayoutBitmap method” on page 42
`GetMetaData`	Gets information about the plugin as XML metadata, or notifies the plugin of changes that the user made to device or task parameters.	“GetMetaData method” on page 45
`Ignore`	Tells the plugin to ignore an error that occurred during a task.	“Ignore method” on page 59
`Initialize`	Tells the plugin to initialize a device.	“Initialize method” on page 61
`IsLocationAvailable`	Asks the plugin whether a location is available for a labware-handling process.	“IsLocationAvailable method” on page 64

<Go Back

Method	Description	See...
`MakeLocationAvailable`	Tells the plugin to make a location available for a labware-handling process.	"MakeLocationAvailable method" on page 67
`PlateDroppedOff`	Notifies the plugin that a labware was dropped off.	"PlateDroppedOff method" on page 75
`PlatePickedUp`	Notifies the plugin that a labware was picked up.	"PlatePickedUp method" on page 77
`PlateTransferAborted`	Notifies the plugin that a labware-transfer process was aborted.	"PlateTransferAborted method" on page 79
`PrepareForRun`	Notifies the plugin that the user started a protocol.	"PrepareForRun method" on page 83
`Retry`	Tells the plugin to retry a task.	"Retry method" on page 85
`ShowDiagsDialog`	*Obsolete.* This method should be implemented as `return E_NOTIMPL (0x80004001)`. Instead of this method, VWorks software calls the IWorksDiags `ShowDiagsDialog` method to display a diagnostics dialog box. See IWorksDiags "ShowDiagsDialog method" on page 94.	

Abort method

Description

VWorks software calls the `Abort` method to terminate a specific task or to terminate all currently executing tasks.

To terminate a specific task

1. The plugin notifies VWorks software that an error occurred during a task.
2. VWorks software does the following:
   a. Calls the `GetErrorInfo` method to get a text string from the plugin that describes the error. See "GetErrorInfo method" on page 40.
   b. Writes the string to the Main Log.
   c. Displays the standard error dialog box, which includes the following components:
      - The error text string.
      - The Abort, Retry, and Ignore and Continue... buttons.

      The figure on page 40 shows a standard error dialog box.
3. If the user clicks the Abort button, VWorks software calls the `Abort` method to tell the plugin to terminate the task.

To terminate all currently executing tasks

When the user aborts a run in the Runset Manager, VWorks software calls the `Abort` method to tell the plugin to terminate all currently executing tasks.

Note: This is not an emergency stop.

Syntax

```
HRESULT Abort(void);
```

Parameters

None.

Related information

For information about...	See...
GetErrorInfo method	"GetErrorInfo method" on page 40
Ignore method	"Ignore method" on page 59
IWorksDiags ShowDiagsDialog method	"ShowDiagsDialog method" on page 94
PrepareForRun method	"PrepareForRun method" on page 83
Retry method	"Retry method" on page 85

<Go Back

Close method

Description

VWorks software calls the `Close` method to tell the plugin to terminate the connection to a device.

IMPORTANT The `Close` method is a synchronous blocking call. It should not return until the device is completely closed.

Note: To initialize the device, VWorks software calls the `Initialize` method. See "Initialize method" on page 61.

Syntax

```
HRESULT Close(void);
```

Parameters

None.

Related information

For information about...	See...
Initialize method	"Initialize method" on page 61

Command method

Note: Throughout this guide, when the word *command* refers to a task in the user interface, the word *task* is used instead of *command*.

Description

VWorks software calls the `Command` method to tell the plugin to execute the specified task.

IMPORTANT Plugins must implement the `Command` method if the device has any associated tasks. The plugin should not return until the task is completed.

Syntax

```
HRESULT Command(
   [in] BSTR CommandXML,
   [out,retval] enum ReturnCode *retVal
);
```

Parameters

`CommandXML`	[in] A Command XML block that describes the task to be executed.
`retVal`	[out, retval] Returns an error code. Possible values: `0` = The request was completed (`RETURN_SUCCESS`) `1` = Something was wrong with the input, so the request was not completed (`RETURN_BAD_ARGS`) `2` = The request was not completed (`RETURN_FAIL`) For more information, see "ReturnCode enumerated type" on page 384.

Command method input

VWorks software passes a Command XML block into the `CommandXML` parameter of the `Command` method.

Command XML block

The Command XML block contains the `Command` element and all its children. This XML block describes the task to be executed.

When the plugin receives the Command XML block, it only needs to check the following attributes:

- `Name` attribute of the `Command` element
- `Name` and `Value` attributes of each `Parameter` element
- `Value` attribute of each `Locations` element's child `Value` element

<Go Back

These attributes are designated by bold text in the following XML structure and input example.

Although VWorks software passes other XML metadata in the Command XML block, this information is of no interest to the plugin.

XML structure

The value of the `file` attribute for the `Velocity11` element is `MetaData`. See "Velocity11 element" on page 46.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Command ... Name='' ...>
      <Parameters>
         <Parameter ... Name='' ... Value='' >
            <Ranges>
               <Range />
               ...
            </Ranges>
         </Parameter>
         ...
      </Parameters>
      <Locations>
         <Value Value='' />
         ...
      </Locations>
   </Command>
</Velocity11>
```

Command element

The `Command` element has two children: `Parameters` and `Locations`. The `Command` element has the following attributes:

Name	Value
`Compiler`	See "Compiler attribute" on page 399.
`Description`	The description of the task.
`Editor`	See "Editor attribute" on page 400.
`Name`	The name of the task.
`NextTaskToExecute`	See "NextTaskToExecute attribute" on page 400.
`ProtocolName`	The name of the protocol that contains the task. If the protocol has been saved, the value of this attribute is the protocol's file path. If the protocol has not been saved, the value is the default protocol name.
`RequiresRefresh`	See "RequiresRefresh attribute" on page 402.
`TaskRequiresLocation`	See "TaskRequiresLocation attribute" on page 402.
`VisibleAvailability`	See "VisibleAvailability attribute" on page 403.

Parameters element

The `Parameters` element contains one or more `Parameter` elements.

<Go Back

Parameter element

The `Parameter` element contains one `Ranges` element and has the following attributes:

Name	Value
`Description`	The description of the parameter.
`Name`	The name of the parameter.
`Scriptable`	See "Scriptable attribute" on page 410.
`Style`	See "Style attribute" on page 411.
`Type`	See "Type attribute" on page 411.
`Value`	See "Value attribute" on page 413.

Locations element

The `Locations` element contains one of more `Value` elements.

Value element

Each `Value` element contains the name of a location that can be used by the task. This element has the following attribute:

Name	Value
`Value`	The name of a location that can be used by the task.

Example of Command method input

The following sample code is a Command XML block that is received by the plugin from VWorks software as a string in the `CommandXML` parameter of the `Command` method. VWorks software tells the plugin to execute the `Execute method` task at the location named `Location`.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='9e0e9dcbdbb460a7d444cba2a2d0a474' version='1.0' >
   <Command Compiler='0' Description='Execute a method' Editor='2'
→Name='Execute method' NextTaskToExecute='1' ProtocolName='Protocol File - 1'
→RequiresRefresh='0' TaskRequiresLocation='1' VisibleAvailability='1' >
      <Parameters >
         <Parameter Description='Method name' Name='Method name' Scriptable='1'
→Style='0' Type='2' Value='a.lmeth' >
            <Ranges >
               <Range Value='a.lmeth' />
            </Ranges>
         </Parameter>
      </Parameters>
      <Locations >
         <Value Value='Location' />
      </Locations>
   </Command>
</Velocity11>
```

<Go Back

Related information

For information about...	See...
GetErrorInfo method	“GetErrorInfo method” on page 40
ReturnCode enumerated type	“ReturnCode enumerated type” on page 384

<Go Back

Compile method

Description

VWorks software calls the `Compile` method to notify the plugin of the state of a protocol's compile sequence. With every call to this method, VWorks software passes a value in the `iCompileType` parameter that represents the current compilation stage. During the compile sequence, the plugin should do the following:

- Accumulate information about changes in the compile state for each stage in the compile sequence
- Report to VWorks software any compiler errors or warnings that occur so VWorks software can log them to the Main Log

Syntax

```
HRESULT Compile(
   [in] enum CompileType iCompileType,
   [in] BSTR MetaDataXML,
   [out,retval] BSTR *CompileResultXML
);
```

Parameters

`iCompileType`	[in] Represents the current stage of the compile sequence. Possible values: `0` = Compiling begins (`COMPILE_BEGIN`) `1` = Compiling the task in a process in the Main Protocol (`COMPILE_TASK_PROCESS`) `2` = Compiling the task in a subprocess that uses this device (`COMPILE_TASK_SUBPROCESS`) `3` = Compiling the task in a process in the Startup Protocol (`COMPILE_TASK_PREPROCESS`) `4` = Compiling the task in the Cleanup Protocol (`COMPILE_TASK_POSTPROCESS`) `5` = Compiling a subprocess begins (`COMPILE_BEGIN_SUBPROCESS`) `6` = Compiling a subprocess ends (`COMPILE_END_SUBPROCESS`) `7` = Compiling ends (`COMPILE_END`) `8` = Compiling a Loop task in a subprocess that uses this device (`COMPILE_LOOP_BEGIN`) `9` = Compiling a Loop End task in a subprocess that uses this device (`COMPILE_LOOP_END`)

<Go Back

MetaDataXML	[in] A Command XML block. The contents vary for different values of the iCompileType parameter.
CompileResultXML	[out, retval] An empty string or a CompilerErrors XML block. If there are no errors or warnings to report to VWorks software, this parameter contains an empty string; otherwise, this parameter contains a CompilerErrors XML block. The CompilerErrors XML block describes any errors that occurred while the protocol was compiling and classifies them as errors or warnings.

Compile method input

VWorks software passes a Command XML block into the MetaDataXML parameter of the Compile method.

Command XML block

The contents of the Command XML block vary for different values of the iCompileType parameter, as shown in the following table:

iCompileType parameter	Contents of Command XML block
COMPILE_BEGIN COMPILE_END	Data for the currently selected profile and the file path of the protocol that is compiling.
COMPILE_BEGIN_SUBPROCESS COMPILE_END_SUBPROCESS	An empty Command element.
All other values	The metadata for the task that is compiling. See "Command XML block" on page 53.

Compile method output (no errors occurred)

If no errors occurred during protocol compilation, the plugin returns an empty CompileResult XML element in the CompileResultXML parameter of the Compile method.

Example of Compile method output (no errors occurred)

The following sample code is an empty CompileResult XML element that is returned by the plugin to VWorks software as a string in the CompileResultXML parameter of the Compile method. The plugin tells VWorks software that no errors occurred during the protocol compilation.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='ce4aaae5bf2c4067123c379b6b697345' version='1.0' >
   <CompileResult />
</Velocity11>
```

<Go Back

Compile method output (errors occurred)

If one or more errors occurred during protocol compilation, the plugin returns a CompileErrors XML block in the `CompileResultXML` parameter of the `Compile` method.

CompilerErrors XML block

The CompilerErrors XML block contains the `CompilerErrors` element and all its children. This XML block describes any errors that occurred during protocol compilation, which the plugin must report to VWorks software.

XML structure

The value of the `file` attribute for the `Velocity11` element is `MetaData`. See "Velocity11 element" on page 46.

```
<?xml version='1.0' encoding='ASCII'>
<Velocity11>
   <MetaData>
      <CompilerErrors>
         <CompilerError />
         ...
      </CompilerErrors>
   </MetaData>
</Velocity11>
```

CompilerErrors element

The `CompilerErrors` element contains one or more `CompilerError` elements.

CompilerError element

The `CompilerError` element has the following attributes:

Name	Value
`Value`	A string that represents the compiler error. Required: No Default value: An empty string
`ErrorType`	Represents the type of error. Possible values: `0` = Compiler error `1` = Compiler warning Required: No Default value: `0`

<Go Back

Example of Compile method output (errors occurred)

The following sample code is a CompilerErrors XML block that is returned by the plugin to VWorks software as a string in the `CompileResultXML` parameter of the `Compile` method. The plugin tells VWorks software that the following two warning-type compiler errors occurred during the protocol compilation: `Warning: xyz` and `Warning: xyzabc`.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='849392019ca47102839e845113d11840' version='1.0'>
   <MetaData>
      <CompilerErrors>
         <CompilerError Value='Warning: xyz' ErrorType='1'>
         <CompilerError Value='Warning: xyzabc' ErrorType='1'>
      </CompilerErrors>
   </MetaData>
</Velocity11>
```

Related information

For information about...	See...
Command XML block	"Command XML block" on page 53
CompileType enumerated type	"CompileType enumerated type" on page 380

<Go Back

ControllerQuery method

Description

VWorks software uses the `ControllerQuery` method in conjunction with the IWorksController `Query` method to provide the means for two plugins to communicate with each other. (See "Query method" on page 292.) VWorks software acts as the intermediary between the source plugin and the destination plugin by receiving and forwarding the plugins' queries and responses in the input and output parameters of these two methods.

Interplugin communication

The following diagram shows how interplugin communication is done using the IWorksController `Query` and IWorksDriver `ControllerQuery` methods.

Figure Interplugin communication using the Query and ControllerQuery methods

1. To send a query to Plugin B, Plugin A calls the IWorksController `Query` method and passes the query to VWorks software in the input parameter. Then the plugin waits for a reply.
2. VWorks software forwards the query to Plugin B by calling the IWorksDriver `ControllerQuery` method and passing the query in the input parameter.
3. Plugin B returns the query response to VWorks software in the output parameter of the IWorksDriver `ControllerQuery` method.
4. VWorks software forwards the query response to Plugin A in the output parameter of the IWorksController `Query` method.

Interplugin queries and query responses

Interplugin communication enables two plugins to work together as a team. For plugins to exchange information, they must be able to understand the queries they receive from each other and respond appropriately. The developers of the two plugins are responsible for defining and parsing the contents of the XML documents that are passed between the plugins (with VWorks software as the intermediary). The developers must define their own mutually agreed-upon values for the `Parameter` elements' `Name` and `Value` attributes that are contained in the following XML blocks:

- IWorksDriver ControllerQuery and ControllerResponse XML blocks, which are explained in this section
- IWorksController Query and Response XML blocks, which are explained in "Query method" on page 292

<Go Back

Syntax

```
HRESULT ControllerQuery(
   [in] BSTR Query,
   [out,retval] BSTR *QueryResult
);
```

Parameters

`Query`	[in] A ControllerQuery XML block that contains the query from the source plugin, which VWorks software provides in this parameter.
`QueryResult`	[out, retval] A ControllerResponse XML block that contains the query response from the destination plugin, which VWorks software receives in this parameter.

ControllerQuery method input

The plugin passes a ControllerQuery XML block into the `Query` parameter of the `ControllerQuery` method.

ControllerQuery XML block

The ControllerQuery XML block contains the `Query` element and all its children. This XML block includes the query from the source plugin.

XML structure

The value of the `file` attribute for the `Velocity11` element is `Query`. See "Velocity11 element" on page 46.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Query Category='InterPlugin' ... >
      <Parameters>
         <Parameter />
      </Parameters>
   </Query>
</Velocity11>
```

Query element

The `Query` element contains one `Parameters` element and has the following attributes:

Name	Value
`Category`	The value `InterPlugin`. Required: Yes
`Destination`	The name of the plugin that is to receive the query. Required: Yes
`Source`	The name of the plugin that is sending the query. Required: Yes

<Go Back

Parameters element

The `Parameters` element contains one `Parameter` element.

Parameter element

The `Parameter` element has the following attributes:

Name	Value
`Name`	The developer-defined name for the interplugin query parameter. Required: Determined by the plugin developer
`Style`	See "Style attribute" on page 411. Required: No Default value: `0`
`Type`	See "Type attribute" on page 411. Required: Yes
`Value`	The developer-defined interplugin query. Required: Determined by the plugin developer

Example of a ControllerQuery method query

The following sample code is a query from Plugin A for Plugin B that is received by VWorks software as a string in the `Query` parameter of the IWorksController `Query` method. (See "Query method" on page 292.) VWorks software then passes the query to Plugin B in the IWorksController `Query` parameter of the `ControllerQuery` method.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Query' md5sum='77406e9e4b59a02f7f058223bc1a81ce' version='1.0' >
   <Query Category='InterPlugin' Destination='Plugin A'' Source='Plugin B' >
      <Parameters>
         <Parameter Name='InterPlugin Param'  Style='0' Type='1' Value='Query test'/>
      </Parameters>
   </Query>
</Velocity11>
```

ControllerQuery method output

The plugin passes a ControllerResponse XML block into the `QueryResult` parameter of the `ControllerQuery` method.

ControllerResponse XML block

The ControllerResponse XML block contains the `Response` element and all its children. This XML block includes the query response from the destination plugin.

<Go Back

XML structure

The value of the `file` attribute for the `Velocity11` element is `QueryResponse`. See "Velocity11 element" on page 46.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Response Category='InterPlugin' ... >
      <Parameters>
         <Parameter />
      </Parameters>
   </Response>
</Velocity11>
```

Response element

The `Response` element contains one `Parameters` element and has the following attributes:

Name	Value
`Category`	The value `InterPlugin`. Required: Yes
`Destination`	The name of the plugin that is to receive the query response. Required: Yes
`Source`	The name of the plugin that is sending the query response. Required: Yes

Parameters element

The `Parameters` element contains one `Parameter` element.

Parameter element

The ControllerResponse XML block's `Parameter` element must have the same attributes as the ControllerQuery XML block's `Parameter` element. See "Parameter element" on page 31.

Name	Value
`Name`	The developer-defined name for the interplugin query response parameter. Required: Determined by the plugin developer
`Style`	See "Style attribute" on page 411. Required: No Default value: `0`
`Type`	See "Type attribute" on page 411. Required: Yes
`Value`	The developer-defined interplugin query response. Required: Determined by the plugin developer

<Go Back

Example of a ControllerQuery method query response

The following sample code is the query response from Plugin B that is received by VWorks software as a string in the `QueryResult` parameter of the `ControllerQuery` method. VWorks software then passes the query response to Plugin A in the `QueryResult` parameter of the IWorksController `Query` method. See "Query method" on page 292.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='QueryResponse' md5sum='77406e9e4b59a02f7f058223bc1a81ce'
→version='1.0' >
   <Response Category='InterPlugin' Destination='Plugin A'' Source='Plugin B' >
      <Parameters>
         <Parameter Name='InterPlugin Param'  Style='0' Type='1'
→Value='Receive test'/>
      </Parameters>
   </Response>
</Velocity11>
```

Related information

For information about...	See...
IWorksController Query method	"Query method" on page 292

<Go Back

Get32x32Bitmap method

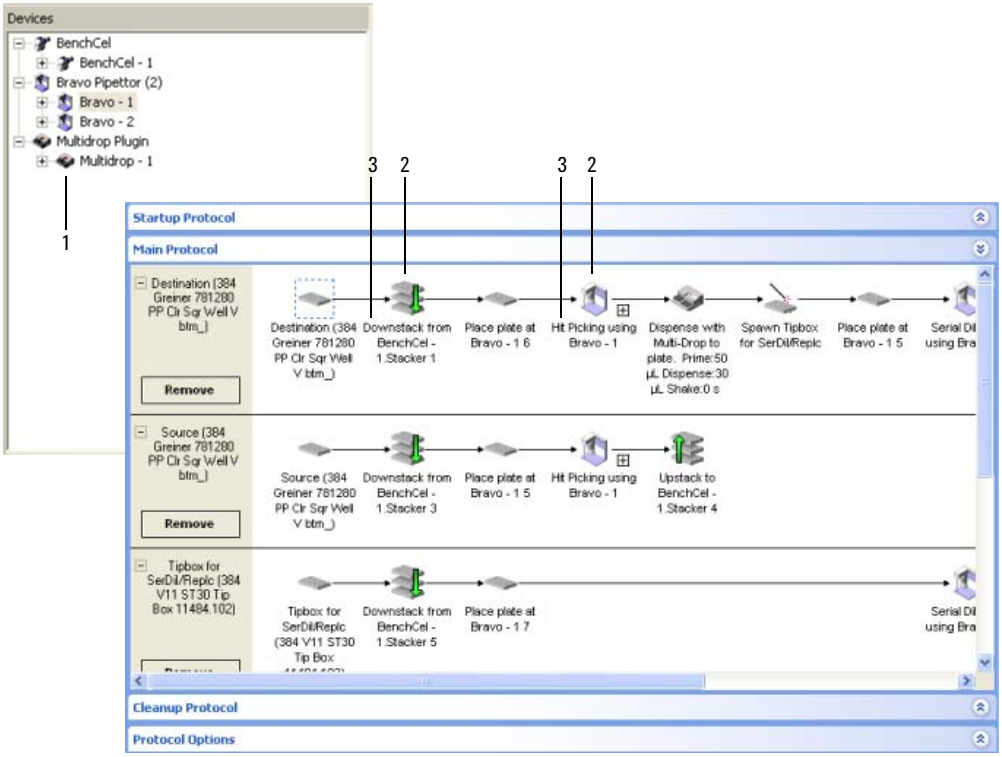
Description

VWorks software calls the `Get32x32Bitmap` method to get a bitmap image from the plugin that represents a device or the specified task.

The following figure shows two bitmap images:

- Device icon in the Device file area that represents the Multidrop
- Task icon in the protocol editor that represents a Downstack task

Figure Device and task icons



Item	Description	Purpose
1	Device icon (bitmap image)	Represents a device
2	Task icon (bitmap image)	Represents a task
3	Task description	Describes the task associated with the task icon The task description is returned with a call to the `GetDescription` method. See "GetDescription method" on page 36.

<Go Back

Syntax

```
HRESULT Get32x32Bitmap(
   [in] BSTR CommandName,
   [out,retval] IPictureDisp **ppPicture
);
```

Parameters

`CommandName`	[in] An empty string, or the name of a task. If the `CommandName` parameter contains an empty string, VWorks software is asking for a bitmap image that represents a device. If the parameter contains a string (the name of a task), VWorks software is asking for a bitmap image to represent a task.
`ppPicture`	[out, retval] An IPictureDisp object that contains the handle to the bitmap for the image. This bitmap must be 32 pixels wide by 32 pixels high. Refer to the Microsoft SDK documentation for more details about the standard COM IPictureDisp interface at http://windowssdk.msdn.microsoft.com/en-us/library.

Related information

For information about...	See...
GetDescription method	"GetDescription method" on page 36

<Go Back

GetDescription method

Description

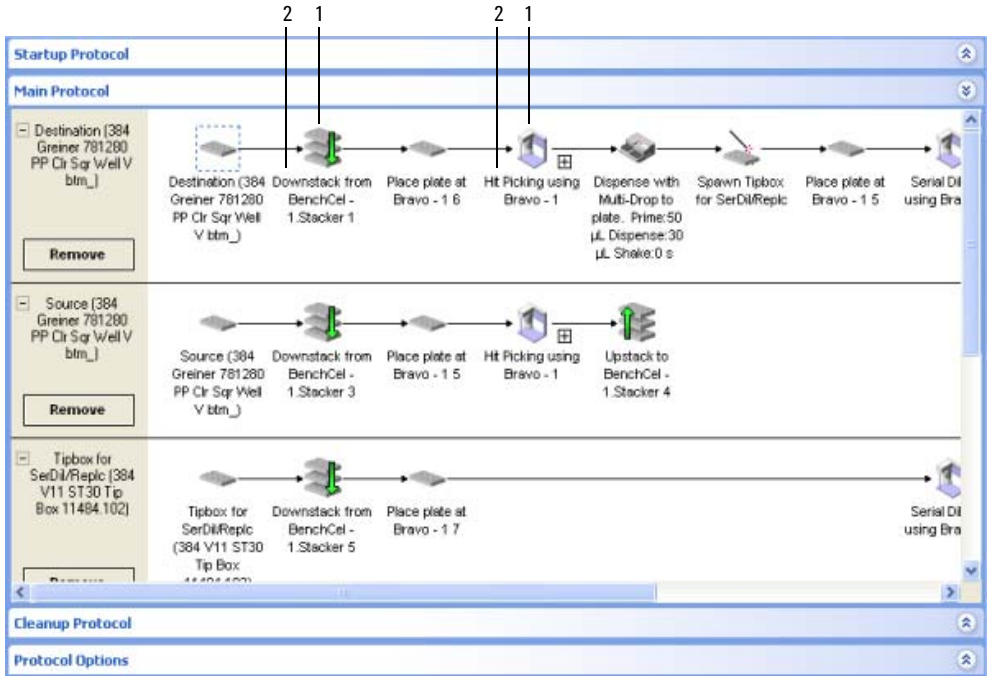
VWorks software calls the `GetDescription` method to get the description for the specified task from the plugin. Depending on the value of the `Verbose` parameter, the plugin returns one of following:

- A short description of the task to display in the protocol editor under the icon associated with the task
- A full, dynamic description of the task to enter in the Main Log

VWorks software calls the `GetDescription` method whenever it needs to render the task description. This enables the description to change dynamically. However, if the `GetDescription` method is not implemented, VWorks software gets the task description by calling the `GetMetaData` method. The description returned with this method call is static. See "GetMetaData method" on page 45.

The following figure shows a task icon, along with its associated task description, in the protocol editor:

Figure Task icon and its associated description



Item	Description	Purpose
1	Task icon	Represents a task The task icon is returned with a call to the `Get32x32Bitmap` method. See "Get32x32Bitmap method" on page 34.
2	Task description	Describes the task associated with the icon

<Go Back

Syntax

```
HRESULT GetDescription(
   [in] BSTR CommandXML,
   [in] VARIANT_BOOL Verbose,
   [out,retval] BSTR *Description
);
```

Parameters

`CommandXML`	[in] A Command XML block that describes the task.
`Verbose`	[in] Indicates whether VWorks software is asking for a short description or a full, dynamic description. Possible values: `VARIANT_TRUE` = VWorks software is requesting a full, dynamic description of the task `VARIANT_FALSE` = VWorks software is requesting a short description of the task
`Description`	[out, retval] The description of the specified task. If the value of the `Verbose` parameter is `VARIANT_TRUE`, this parameter contains a full, dynamic description of the specified task. If the value is `VARIANT_FALSE`, this parameter contains a short description of the specified task.

GetDescription method input

VWorks software passes a Boolean value into the `Verbose` parameter of the `GetDescription` method to tell the plugin which type of description to send: dynamic or short. VWorks software also passes a Command XML block into the `CommandXML` parameter.

Command XML block

The Command XML block contains the `Command` element and all its children. This XML block describes the specified task.

When the plugin receives the Command XML block, it only needs to check the following attributes:

- `Name` attribute of the `Command` element
- `Name` and `Value` attributes of each `Parameter` element

These attributes are designated by bold text in the following XML structure and input example.

Although VWorks software passes other XML metadata in the Command XML block, this information is of no interest to the plugin.

<Go Back

XML structure

The value of the `file` attribute for the `Velocity11` element is `MetaData`. See "Velocity11 element" on page 46.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Command ... Name='' ... >
      <Parameters>
         <Parameter ... Name='' ... Value=''>
            <Ranges>
               <Range>
               ...
            </Ranges>
         </Parameter>
         ...
      </Parameters>
   </Command>
</Velocity11>
```

Command element

The `Command` element contains one `Parameters` element and has the following attributes:

Name	Value
`Compiler`	See "Compiler attribute" on page 399.
`Description`	The description of the task.
`Editor`	See "Editor attribute" on page 400.
`Name`	The name of the task.
`NextTaskToExecute`	See "NextTaskToExecute attribute" on page 400.
`RequiresRefresh`	See "RequiresRefresh attribute" on page 402.
`TaskRequiresLocation`	See "TaskRequiresLocation attribute" on page 402.
`VisibleAvailability`	See "VisibleAvailability attribute" on page 403.

Parameters element

The `Parameters` element contains one or more `Parameter` elements.

Parameter element

The `Parameter` element can contain one `Ranges` elements and has the following attributes:

Name	Value
`Description`	The description of the parameter.
`Name`	The name of the parameter.
`Scriptable`	See "Scriptable attribute" on page 410.
`Style`	See "Style attribute" on page 411.

<Go Back

Name	Value
Type	See "Type attribute" on page 411.
Value	See "Value attribute" on page 413.

Example of GetDescription method input

The following code is a Command XML block that is received by the plugin from VWorks software as a string in the `CommandXML` parameter of the `GetDescription` method. VWorks software asks the plugin for a description of the `Execute a method` task. If the value of the `Verbose` parameter is `VARIANT_TRUE`, VWorks software is requesting a full, dynamic description of the task. If the value is `VARIANT_FALSE`, VWorks software is requesting a short description of the task.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='4122f65abc7c2fae10b8ba5b02ba18c1' version='1.0' >
   <Command Compiler='0' Description='Execute a method' Editor='2'
→Name='Execute a method' NextTaskToExecute='1' RequiresRefresh='0'
→TaskRequiresLocation='1' VisibleAvailability='1' >
      <Parameters >
         <Parameter Description='Method name' Name='Method name' Scriptable='1'
→Style='0' Type='2' Value='a.lmeth' >
            <Ranges >
               <Range Value='a.lmeth' />
            </Ranges>
         </Parameter>
      </Parameters>
   </Command>
</Velocity11>
```

Related information

For information about...	See...
Command XML block	"Command XML block" on page 53
Get32x32Bitmap method	"Get32x32Bitmap method" on page 34
GetMetaData method	"GetMetaData method" on page 45

<Go Back

GetErrorInfo method

Description

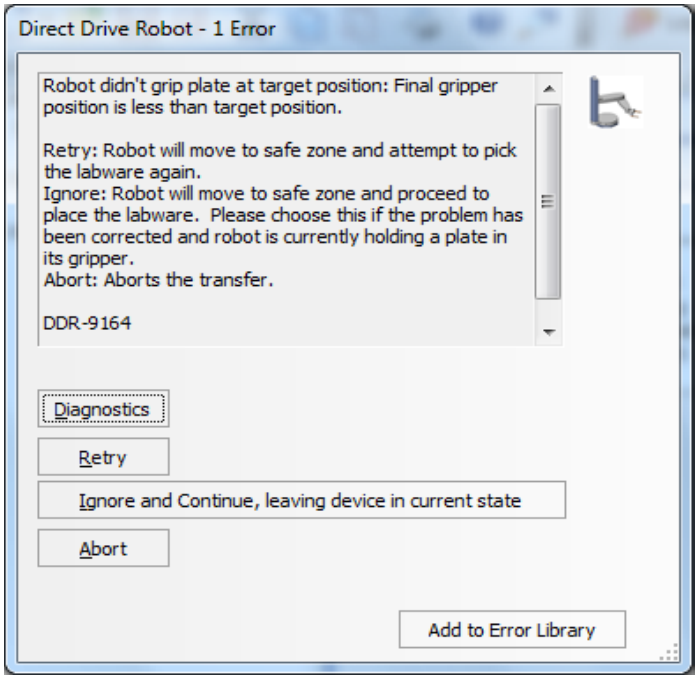
VWorks software calls the `GetErrorInfo` method when the plugin returns a value other than `RETURN_SUCCESS` for a method with a `retVal` output parameter of type `ReturnCode`. See "ReturnCode enumerated type" on page 384.

When the plugin notifies VWorks software that an error occurred during a task, VWorks software does the following:

1. Calls the `GetErrorInfo` method to get a text string from the plugin that describes the error.
2. Writes the string to the Main Log.
3. Displays the standard error dialog box, which includes the following components:
   - The error text string
   - The Abort, Retry, and Ignore and Continue... buttons.

   The following figure shows a standard error dialog box for the 3-Axis Robot.

Figure Standard error dialog box



Syntax

```
HRESULT GetErrorInfo(
   [out,retval] BSTR *ErrorInfoXML
);
```

<Go Back

Parameter

`ErrorInfoXML`	[out, retval] A text string that describes the error. Agilent Technologies does not currently support error codes. The plugin developer can embed error codes in this parameter.

Related information

For information about...	See...
Compile method	"Compile method" on page 25
CompileType enumerated type	"CompileType enumerated type" on page 380
IWorksDiags ShowDiagsDialog method	"ShowDiagsDialog method" on page 94
IWorksDriver Abort method	"Abort method" on page 19
IWorksDriver Ignore method	"Ignore method" on page 59
IWorksDriver Retry method	"Retry method" on page 85
ReturnCode enumerated type	"ReturnCode enumerated type" on page 384

<Go Back

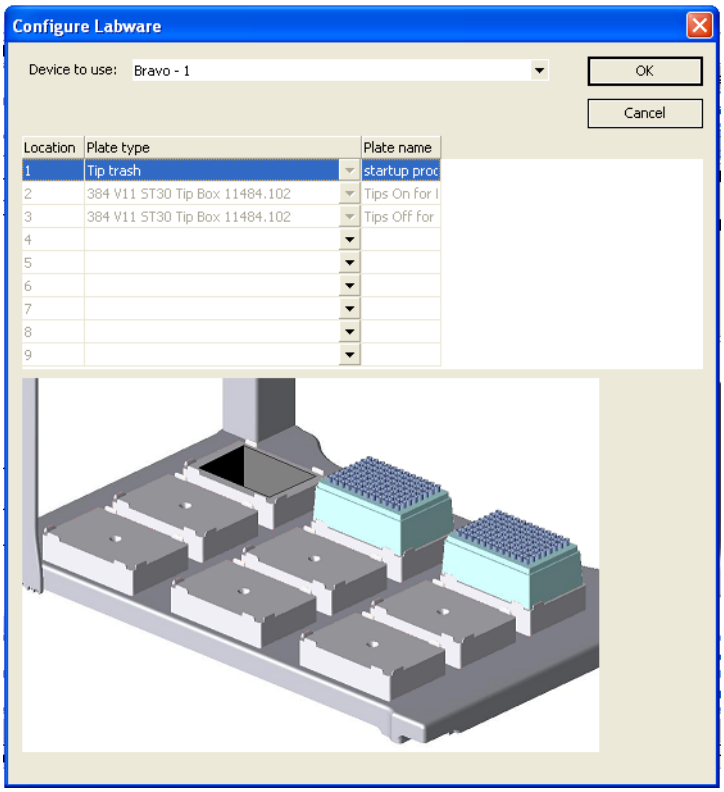
GetLayoutBitmap method

Description

VWorks software calls the `GetLayoutBitmap` method to get a dynamic bitmap from the plugin. The bitmap displays the specified labware and its location on a device.

The following figure shows a Configure Labware dialog box that contains the dynamic bitmap of a Bravo deck with two tip boxes and a tip trash.

Figure Configure Labware dialog box showing a dynamic bitmap



Syntax

```
HRESULT GetLayoutBitmap(
   [in] BSTR LayoutInfoXML,
   [out,retval] IPictureDisp **ppPicture
);
```

Parameters

`LayoutInfoXML`	[in] A LayoutVector XML block containing the names and locations of the labware that VWorks software wants to represent on a device.

<Go Back

ppPicture	[out, retval] An IPictureDisp object that contains the handle to the bitmap for the image. The suggested size for this bitmap is 480 pixels by 320 pixels or smaller. Refer to the Microsoft SDK documentation for more details about the standard COM IPictureDisp interface at http://windowssdk.msdn.microsoft.com/en-us/library.

GetLayoutBitmap method input

VWorks software passes a LayoutVector XML block into the `LayoutInfoXML` parameter of the `GetLayoutBitmap` method.

LayoutVector XML block

The LayoutVector XML block contains the `LayoutVector` element and all its children. This XML block provides the names and locations of the specified labware. The plugin should parse the LayoutVector XML block and render a dynamic bitmap that shows the labware and its locations on the device.

XML structure

The value of the `file` attribute for the `Velocity11` element is `MetaData`. See "Velocity11 element" on page 46.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <LayoutVector>
      <Layouts>
         <Layout />
         ...
      </Layouts>
   </LayoutVector>
</Velocity11>
```

LayoutVector element

The `LayoutVector` element contains one `Layouts` element.

Layouts element

The `Layouts` element contains one or more `Layout` elements.

Layout element

Each `Layout` element contains the type of a labware to show on the device and has the following attributes:

Name	Value
Labware	The labware type. If a labware is not configured at a location, this attribute is not specified.
Name	The name of the location on the device.

<Go Back

Example of GetLayoutBitmap method input

The following sample code is a LayoutVector XML block that is received by the plugin from VWorks software as a string in the `LayoutInfoXML` parameter of the `GetLayoutBitmap` method. VWorks software asks the plugin for a dynamic bitmap.

The plugin uses the information in the LayoutVector XML block to generate a dynamic bitmap that includes two labware at the locations named `1` and `2`. No labware is present at locations `3` though `9`, so the `Labware` attribute for these `Layout` elements is not specified.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='623442a0db37a6ec5c093d6666e9f1bd' version='1.0' >
   <LayoutVector >
      <Layouts >
         <Layout Labware='! Novartis 1536' Name='1' />
         <Layout Labware='384 V11 Tip Box ST70 19133.002' Name='2' />
         <Layout Name='3' />
         <Layout Name='4' />
         <Layout Name='5' />
         <Layout Name='6' />
         <Layout Name='7' />
         <Layout Name='8' />
         <Layout Name='9' />
      </Layouts>
   </LayoutVector>
</Velocity11>
```

GetMetaData method

Description

VWorks software calls the `GetMetaData` method for the following reasons:

- To get all metadata for a plugin

  When the plugin is first loaded, VWorks software calls the `GetMetaData` method, where `iDataType` is `METADATA_ALL`, to get all XML metadata for the plugin. The plugin returns a MetaData XML block in the `MetaData` parameter. This XML block identifies the device, provides version information, and describes all the tasks that the device can perform. See "Metadata XML block" on page 46.

  Subsequent calls to the `GetMetaData` method return device, command (task), or version XML metadata, depending on the value of the `iDataType` parameter. See "Device XML block" on page 48, "Versions XML block" on page 52, and "Command XML block" on page 53.

- To notify the plugin when the user changes device parameters and task parameters

Syntax

```
HRESULT GetMetaData(
   [in] enum MetaDataType iDataType,
   [in] BSTR current_metadata,
   [out,retval] BSTR *MetaData
);
```

Parameters

`iDataType`	[in] Represents the type of metadata. The first time the `GetMetaData` method is called, the value of this parameter is `0`. Possible values: `0` = Requests all XML metadata (`METADATA_ALL`). See "Metadata XML block" on page 46. `1` = Requests device metadata only (`METADATA_DEVICE`). See "Device XML block" on page 48. `2` = Requests command metadata only (`METADATA_COMMAND`). See "Command XML block" on page 53. `3` = Requests version metadata only (`METADATA_VERSION`). See "Versions XML block" on page 52.

<Go Back

current_metadata	[in] A Device XML block or a Command XML block reflecting the current state of user-specified parameters. When the plugin is first loaded and VWorks software calls the GetMetaData method, this parameter is empty.
MetaData	[out, retval] Depending on the value of iDataType, this parameter can contain one of the following: • MetaData XML block • Device XML block • Versions XML block • Command XML block The plugin can modify the metadata using the updated user settings received in the current_metadata parameter before it returns to VWorks software.

Velocity11 element

All XML metadata used in the GetMetaData method contains the Velocity11 element. The Velocity11 element is defined in "Velocity11 element" on page 416.

GetMetaData method output (iDataType is METADATA_ALL)

IMPORTANT The GetMetaData method is the only method that uses the Metadata XML block. All device driver plugins must implement this method and provide the Metadata XML block to VWorks software.

When the plugin is first loaded and VWorks software calls the GetMetaData method, the plugin returns the Metadata XML block to VWorks software in the MetaData parameter.

Metadata XML block

The Metadata XML block contains the MetaData element and all its children. This XML block identifies the device, provides version information, and describes all the tasks that the device can perform.

XML structure

The following code shows the high-level structure of the Metadata XML block. The value of the file attribute for the Velocity11 element is MetaData. See "Velocity11 element" on page 416.

```
<?XML version='1.0' encoding='ASCII' ?>
<Velocity11>
   <MetaData>
      <Device />
      <Versions />
      <Commands />
   </MetaData>
</Velocity11>
```

<Go Back

MetaData element

The `MetaData` element has three children: `Device`, `Versions`, and `Commands`. The `MetaData` element has no attributes. See "Velocity11 element" on page 416.

Device element

The `Device` element is defined in "Device element" on page 403.

Versions element

The `Versions` element is defined in "Versions XML block" on page 52.

Commands element

The `Commands` element contains one `Command` element for every task that the device can perform. The `Commands` element has no attributes. The `Command` element is defined in "Command XML block" on page 53.

Example of a MetaData XML block

The following sample code is the MetaData XML block for a PlateLoc Sealer.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='849392019ca47102839e845113d11840' version='1.0' >
   <MetaData >
      <Device Description='Velocity11 PlateLoc sealer' MiscAttributes='0'
→Name='PlateLoc' PreferredTab='Plate Handling'
         <Parameters >
            <Parameter Name='Profile' Style='0' Type='2' />
         </Parameters>
         <Locations >
            <Location Group='0' Name='Stage' Offset='0' Type='1' />
         </Locations>
         <StorageDimensions DirectStorageAccess='0' />
      </Device>
      <Versions>
         <Version Author='Joe Smith' Company='ABC Company' Date='April 3, 2006'
→Name='PlateLoc' Version='3.0.0' />
      </Versions>
      <Commands>
         <Command Compiler='0' Description='Seal a plate' Editor='2' Name='Seal'>
            <Parameters>
               <Parameter Name='Seal time' Style='0' Type='12' Units='s' Value='1.2'>
                  <Ranges>
                     <Range Value='0.5' />
                     <Range Value='12' />
                  </Ranges>
               </Parameter>
               <Parameter Name='Seal temperature' Style='0' Type='8' Units='°C'
→Value='170'>
                  <Ranges>
                     <Range Value='20' />
                     <Range Value='235' />
                  </Ranges>
               </Parameter>
            </Parameters>
         </Command>
      </Commands>
   </MetaData>
</Velocity11>
```

GetMetaData method input and output (iDataType is METADATA_DEVICE)

When VWorks software calls the `GetMetaData` method and passes the value `METADATA_DEVICE` into the `iDataType` parameter:

<Go Back

- VWorks software passes a Device XML block into the `current_metadata` parameter.
- The plugin returns a Device XML block to VWorks software in the `MetaData` parameter.

Note: When VWorks software first calls the `GetMetaData` method (`iDataType` is `METADATA_ALL`), the plugin returns the MetaData XML block, which contains the `Device` element and all its children. See "Metadata XML block" on page 46.

Device XML block

The Device XML block contains the `Device` element and all its children.

XML structure

The value of the `file` attribute for the `Velocity11` element is `MetaData`. See "Velocity11 element" on page 46.

```
<?xml version='1.0' encoding='ASCII'>
<Velocity11>
   <MetaData>
      <Device>
         <Parameters>
            <Parameter>
               <Ranges>
                  <Range />
                  ...
               </Ranges>
            ...
         </Parameters>
         <Locations>
            <Location />
            ...
         </Locations>
         <StorageDimensions>
            <Dimensions>
               <StorageDimension />
               ...
            </Dimensions>
         </StorageDimensions>
         <RobotMetaData />
      </Device>
   /<MetaData>
</Velocity11>
```

Device element

The `Device` element has four children: `Parameters`, `Locations`, `StorageDimensions`, and `RobotMetaData`. The `Device` element is defined in "Device element" on page 403.

Parameters element

The `Parameters` element contains one or more `Parameter` elements.

<Go Back

Parameter element

The `Parameter` element can contain one `Ranges` element and has the following attributes:

Name	Value
`Name`	The name of the parameter. Required: Yes
`Scriptable`	See "Scriptable attribute" on page 410. Required: No Default value: `0`
`Style`	See "Style attribute" on page 411. Required: No Default value: `0`
`Type`	See "Type attribute" on page 411. Required: Yes

Ranges element

If the plugin has one or more profiles, the `Ranges` element and its child `Range` elements are specified in the Device XML block. The `Ranges` element contains one or more `Range` elements.

Range element

The `Range` element has the following attribute:

Name	Value
`Value`	See "Value attribute" on page 415. Required: Yes

Locations element

The `Locations` element contains one or more `Location` elements.

Location element

Each `Location` element contains the name of a location on the device where labware can be placed. The `Location` element is defined in "Location element" on page 407.

StorageDimensions element

The StorageDimensions XML block is used for storage devices only. VWorks software ignores the `StorageDimensions` element for all non-storage devices.

<Go Back

The StorageDimensions element has the following attributes:

Name	Value
Name0	The name of the first dimension, or the cassette. This attribute is not specified for non-storage devices. Required: Yes
Name1	The name of the second dimension, or whatever holds each labware. This attribute is not specified for non-storage devices. Required: Yes
DirectStorageAccess	Indicates whether a robot accesses labware in a device's storage area or on an external staging area. For example, a robot accesses labware on a multi-sided storage carousel by reaching into the device. However, a robot accesses labware on the external staging area of an STX incubator or a Cytomat storage device. Possible values: 0 = The robot accesses labware on an external staging area 1 = The robot accesses labware directly in the device The value of this attribute is 0 for non-storage devices. Required: No Default value: 0

The contents of the StorageDimensions element are different for storage devices and non-storage devices, as shown in the following table:

Device type	Contents of the StorageDimensions element
Storage device	One StorageDimensions element, where the Name0, Name1, and DirectStorageAccess attributes are specified.
Non-storage device	One empty StorageDimensions element. The Name0 and Name1 attributes are not specified, and the value of the DirectStorageAccess attribute is 0, as shown in the following sample code: `<StorageDimensions DirectStorageAccess='0' />`

<Go Back

Dimensions element

The `Dimensions` element contains one or more `StorageDimension` elements.

StorageDimension element

Each `StorageDimension` element contains the capacity of one cassette, which is the number of slots per cassette. This element has the following attribute:

Name	Value
`Size`	The capacity of one cassette. Required: Yes

RobotMetaData element

The `RobotMetaData` element contains information about the robot. If the device is not a robot, this element is ignored. The `RobotMetaData` element has the following attribute:

Name	Value
`ReachesExternalLocations`	Indicates whether the robot can reach external locations. Possible values: `0` = The robot cannot reach external locations `1` = The robot can reach external locations Required: No Default value: `1`

Example of a Device XML block

The following sample code is a Device XML block for the Lid Hotel Station.

Because this device is not a storage device:

- For the `StorageDimensions` element's attributes:
  - The value of `DirectStorageAccess` is `0`
  - The `Name0` and `Name1` attributes are not specified
- VWorks software ignores the `StorageDimensions` and `RobotMetaData` elements.

<Go Back

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='fe1fd59caa05b0f363e9ed4fbd82fffd' version='1.0' >
   <MetaData>
      <Device Description='Lid Hotel Station' DynamicLocations='0'
→HardwareManufacturer='Agilent Technologies' HasBarcodeReader='0' MiscAttributes='0'
→Name='Lid Hotel Station' PreferredTab='Other'
→RegistryName='Lid Hotel Plugin\Profiles'
         <Parameters >
            <Parameter Name='Profile' Scriptable='1' Style='0' Type='2' >
               <Ranges >
                  <Range Value='ActiveX_LidHotel' />
               </Ranges>
            </Parameter>
         </Parameters>
         <Locations >
            <Location Group='0' MaxStackHeight='460' Name='Location' Offset='0'
→Type='1' />
         </Locations>
         <StorageDimensions DirectStorageAccess='0' />
         <RobotMetaData ReachesExternalLocations='1' />
      </Device>
   </MetaData>
</Velocity11>
```

Versions XML block

The Versions XML block contains the `Versions` element and all its children. This XML block describes the plugin version information to VWorks software.

When VWorks software calls the `GetMetaData` method and the value of the `iDataType` parameter is `METADATA_VERSION`:

- VWorks software passes a Versions XML block into the `current_metadata` parameter.
- The plugin returns a Versions XML block to VWorks software in the `MetaData` parameter.

Note: When VWorks software first calls the `GetMetaData` method (`iDataType` is `METADATA_ALL`), the plugin returns the MetaData XML block, which contains the `Versions` element and all its children. See "Metadata XML block" on page 46.

XML structure

The value of the `file` attribute for the `Velocity11` element is `MetaData`. See "Velocity11 element" on page 46.

```
<?xml version='1.0' encoding='ASCII'>
<Velocity11>
   <MetaData>
      <Versions>
         <Version />
         ...
      </Versions>
   </MetaData>
</Velocity11>
```

Versions element

The `Versions` element contains one or more `Version` elements.

<Go Back

Version element

The `Version` element describes the version information for the plugin. Multiple `Version` elements can be used to keep a revision history for the plugin. However, only the information for the first `Version` element is displayed in the About VWorks dialog box. The `Version` element has the following attributes:

Name	Value
`Author`	The name of the author of the plugin. Required: No Default value: None
`Company`	The company name of the plugin's author. Required: No Default value: None
`Date`	The date the plugin was completed. Required: No Default value: None
`Name`	The name of the plugin. Required: Yes
`Version`	The version number for the plugin. Required: Yes

Example of a Versions XML block

The following sample code is a Versions XML block for the Lid Hotel Station.

```
<?xml version='1.0' encoding='ASCII'>
<Velocity11 file='MetaData' md5sum='07885f2690dba0d44ca74c70133981d7' version='1.0' >
   <MetaData>
      <Versions >
         <Version Author='Agilent Technologies' Company='Agilent Technologies'
→Date='Mar 19th, 2010' Name='Lid Hotel Station' Version='5.0.1' />
      </Versions>
   </MetaData>
</Velocity11>
```

Command XML block

The Command XML block contains the `Command` element and all its children. This XML block describes a single task that the device can perform.

When VWorks software calls the `GetMetaData` method and the value of the `iDataType` parameter is `METADATA_COMMAND`:

- VWorks software passes a Command XML block into the `current_metadata` parameter
- The plugin returns a Command XML block to VWorks software in the `MetaData` parameter

<Go Back

Note: When VWorks software first calls the `GetMetaData` method (`iDataType` is `METADATA_ALL`), the plugin returns the MetaData XML block, which contains the `Commands` element and one or more child `Command` elements and all their children. See "Metadata XML block" on page 46.

XML structure

The value of the `file` attribute for the `Velocity11` element is `MetaData`. See "Velocity11 element" on page 46.

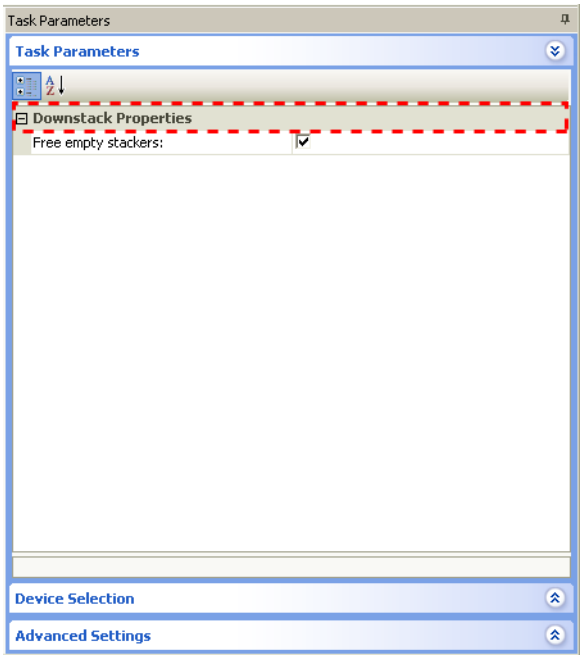
```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Command>
      <Parameters>
         <Parameter>
            <Ranges>
               <Range>
            </Ranges>
            ...
         </Parameter>
         ...
      </Parameters>
      <Locations>
         <Value />
         ...
      </Locations>
   </Command>
</Velocity11>
```

Command element

The `Command` element has the following attributes:

Name	Value
`Compiler`	See "Compiler attribute" on page 399. Required: No Default value: `0`
`Description`	The description displayed under the task icon in the protocol editor, as illustrated in the following figure.  Required: No Default value: None
`Editor`	See "Editor attribute" on page 400. Required: No Default value: `0`

<Go Back

Name	Value
Name	The name of the task. The task name, followed by the word *Properties*, is displayed in the Task Parameters area, as illustrated in the following figure.  Required: No Default value: None
NextTaskToExecute	See "NextTaskToExecute attribute" on page 400. Required: No Default value: 1
PreferredTab	See "PreferredTab attribute" on page 401. Required: No Default value: None
ProtocolName	The name of the protocol that contains the task. If the protocol has been saved, the value of this attribute is the protocol's file path. If the protocol has not been saved, the value is the default protocol name. Required: Yes
RequiresRefresh	See "RequiresRefresh attribute" on page 402. Required: No Default value: 0

<Go Back

Name	Value
TaskRequiresLocation	See "TaskRequiresLocation attribute" on page 402. Required: No Default value: 1
VisibleAvailability	See "VisibleAvailability attribute" on page 403. Required: No Default value: 1
DisplayName	The task name that is displayed in the user interface. If the DisplayName attribute is not specified, the value of the Name attribute is displayed. Required: No Default value: None

Parameters element

The Parameters element contains one or more Parameter elements.

Parameter element

The Parameter element describes device parameters and task parameters. Each Parameter element contains three required attributes (Description, Name, and Type) and might contain one or more of the optional attributes that are listed in the following table. A Parameter element can also contain one Ranges element.

Name	Value
Category	A name used to group two or more Parameter elements into a category. Required: No Default value: None
Hide_if	See "Hide_if attribute" on page 409. Required: No Default value: None
Description	The description of the parameter. Required: Yes
Name	The name of the parameter. Required: Yes
Scriptable	See "Scriptable attribute" on page 410. Required: No Default value: 0

<Go Back

Name	Value
Style	See "Style attribute" on page 411. Required: No Default value: 0
Type	See "Type attribute" on page 411. Required: Yes
Units	See "Units attribute" on page 412. Required: No Default value: None (No units are displayed)
Value	See "Value attribute" on page 413. Required: No Default value: None

Ranges element

The Ranges element contains one or more Range elements.

Range element

The Range element has the following attribute:

Name	Value
Value	See "Value attribute" on page 415. Required: Yes

Locations element

The Locations element contains one of more Value elements.

Value element

Each Value element contains the name of a location used by the task. The Value element has the following attribute:

Name	Value
Value	The name of a location used by the task. Required: No Default value: None

<Go Back

Example of a Command XML block

The following sample code is a Command XML block that describes the `Seal` task.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='b9f9dff80bf8d380efb03cda921220e4' version='1.0' >
   <Command Compiler='20' Description='Seal a plate' Editor='2' Name='Seal'
→NextTaskToExecute='1' ProtocolName='Protocol File - 1' RequiresRefresh='0'
→TaskRequiresLocation='1' VisibleAvailability='1' >
      <Parameters >
         <Parameter Name='Seal time' Scriptable='1' Style='0' Type='12' Units='s'
→Value='1.2' >
            <Ranges >
               <Range Value='0.5' />
               <Range Value='12' />
            </Ranges>
         </Parameter>
         <Parameter Name='Seal temperature' Scriptable='1' Style='0' Type='8'
→Units='C' Value='170' >
            <Ranges >
               <Range Value='20' />
               <Range Value='235' />
            </Ranges>
         </Parameter>
      </Parameters>
      <Locations >
         <Value Value='Stage' />
      </Locations>
   </Command>
</Velocity11>
```

<Go Back

Ignore method

Description

VWorks software calls the `Ignore` method as follows:

1. The plugin notifies VWorks software that an error occurred during a task.
2. VWorks software does the following:
   a. Calls the `GetErrorInfo` method to get a text string from the plugin that describes the error. See "GetErrorInfo method" on page 40.
   b. Writes the string to the Main Log.
   c. Displays the standard error dialog box, which includes the following components:
      - The error text string.
      - The Abort, Retry, and Ignore and Continue... buttons.

      The figure on page 40 shows a standard error dialog box.
3. If the user clicks the Ignore and Continue... button, VWorks software calls the `Ignore` method to tell the plugin to ignore the error.

   The plugin should continue the task, if possible, or exit the task if continuing would be dangerous or impossible. A call to the `Ignore` method should not cause an unsafe condition, which could result in a new error or a premature completion of the task.

   If all errors encountered during a task are ignored, the system should be able to continue as if the failing step was skipped.

Syntax

```
HRESULT Ignore(
   [out,retval] enum ReturnCode *retVal
);
```

Parameter

`retVal`	[out, retval] Returns an error code. Possible values: `0` = The request was completed (`RETURN_SUCCESS`) `1` = Something was wrong with the input, so the request was not completed (`RETURN_BAD_ARGS`) `2` = The request was not completed (`RETURN_FAIL`) For more information, see "ReturnCode enumerated type" on page 384.

<Go Back

Related information

For information about...	See...
Abort method	“Abort method” on page 19
GetErrorInfo method	“GetErrorInfo method” on page 40
IWorksDiags ShowDiagsDialog method	“ShowDiagsDialog method” on page 94
Retry method	“Retry method” on page 85
ReturnCode enumerated type	“ReturnCode enumerated type” on page 384

<Go Back

Initialize method

Description

VWorks software calls the `Initialize` method to tell the plugin to initialize a device. The plugin is expected to do everything necessary to bring the device into a state that allows it to accept commands for activities such as opening a serial port, setting the baud rate, or homing a motor.

Note: VWorks software calls the `Close` method to terminate the connection to the device. See "Close method" on page 20.

IMPORTANT The `Initialize` method should execute synchronously, that is, it should not return until the device initialization is complete.

Syntax

```
HRESULT Initialize(
   [in] BSTR CommandXML,
   [out,retval] enum ReturnCode *retVal
);
```

Parameters

`CommandXML`	[in] An Initialize XML block that contains initialization parameters for the device.
`retVal`	[out, retval] Returns an error code. Possible values: `0` = The request was completed (`RETURN_SUCCESS`) `1` = Something was wrong with the input, so the request was not completed (`RETURN_BAD_ARGS`) `2` = The request was not completed (`RETURN_FAIL`) For more information, see "ReturnCode enumerated type" on page 384.

Initialize method input

VWorks software passes an Initialize XML block into the `CommandXML` parameter of the `Initialize` method.

Initialize XML block

VWorks software creates the Initialize XML block as follows:

- Extracts the device initialization parameters from the Device XML block that was received with a previous call to the `GetMetaData` method. See "GetMetaData method" on page 45.
- Modifies any parameters that the user changed.

<Go Back

VWorks software then passes the Initialize XML block to the plugin in the `CommandXML` parameter of the `Initialize` method.

When the plugin receives the Initialize XML block, it only needs to check the `Name` and `Value` attributes of the `Parameter` element.

These attributes are designated by bold text in the following XML structure and input example.

Although VWorks software passes other XML metadata in the Command XML block, this information is of no interest to the plugin.

XML structure

The value of the `file` attribute for the `Velocity11` element is `MetaData`. See "Velocity11 element" on page 46.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Command>
      <Parameters>
         <Parameter Name='Profile' ... Value=''>
            <Ranges>
               <Range />
               ...
            </Ranges>
         </Parameter>
      </Parameters>
   </Command>
</Velocity11>
```

Parameter element

The `Parameter` element contains one `Ranges` element and has the following attributes:

Name	Value
`Name`	The value `Profile`.
`Scriptable`	See "Scriptable attribute" on page 410.
`Style`	See "Style attribute" on page 411.
`Type`	See "Type attribute" on page 411.
`Value`	The name of the profile to be used to initialize the plugin.

<Go Back

Example of Initialize method input

The following sample code is an Initialize XML block received by the plugin from VWorks software as a string in the `CommandXML` parameter of the `Initialize` method. VWorks software tells the plugin to initialize the device using the profile named `My profile`.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='674489a50c869a3df216bf989803b350' version='1.0' >
   <Command Compiler='0' Editor='0' Name='Initialize' NextTaskToExecute='1'
→RequiresRefresh='0' TaskRequiresLocation='1' VisibleAvailability='1' >
      <Parameters >
         <Parameter Name='Profile' Scriptable='1' Style='0' Type='2'
→Value='My profile' >
            <Ranges >
               <Range Value='A test profile' />
               <Range Value='My profile B' />
               <Range Value='My profile C' />
            </Ranges>
         </Parameter>
      </Parameters>
   </Command>
</Velocity11>
```

Related information

For information about...	See...
Close method	“Close method” on page 20
Command XML block	“Command XML block” on page 53
GetErrorInfo method	“GetErrorInfo method” on page 40
GetMetaData method	“GetMetaData method” on page 45
ReturnCode enumerated type	“ReturnCode enumerated type” on page 384

<Go Back

IsLocationAvailable method

Description

VWorks software calls the `IsLocationAvailable` method (repeatedly) during task scheduling to ask the plugin whether the target location is available for a labware-handling process.

This method was created for complicated multi-robot systems to prevent one robot from reaching into the envelope of another robot, which could cause a robot crash.

The `IsLocationAvailable` method prevents VWorks software from scheduling tasks on devices with complicated geometry or very long processes until preconditions are met. This method is not meant to preempt the scheduler or to wait for a precondition to occur.

IMPORTANT The `IsLocationAvailable` method should take as little time as possible, as it is called repeatedly during scheduling.

Syntax

```
HRESULT IsLocationAvailable(
   [in] BSTR LocationAvailableXML,
   [out,retval] VARIANT_BOOL *Available
);
```

Parameters

`LocationAvailableXML`	[in] A LocationAvailable XML block that contains information about the target location.
`Available`	[out, retval] Indicates whether the target location is available for a labware-handling process. Possible values: `VARIANT_TRUE` = The target location is available for a labware-handling process `VARIANT_FALSE` = The target location is not available for a labware-handling process

IsLocationAvailable method input

VWorks software passes a LocationAvailable XML block into the `LocationAvailableXML` parameter of the `IsLocationAvailable` method.

<Go Back

LocationAvailable XML block

The LocationAvailable XML block contains the `LocationAvailable` element and all its children. This XML block provides information about the target location. The contents of the LocationAvailable XML block are different for storage and non-storage devices.

The plugin uses the information from the LocationAvailable XML block to determine whether the target location is available for a labware-handling process. Although most plugins return `TRUE` directly, some plugins determine availability according to the task name and the location name.

XML structure

The value of the `file` attribute for the `Velocity11` element is `MetaData`. See "Velocity11 element" on page 416.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <LocationAvailable>
      <StorageLocation>
         <Location />
      </StorageLocation>
      <Command />
   </LocationAvailable>
</Velocity11>
```

XML elements and attributes

See "LocationAvailable XML block components" on page 70.

Example of IsLocationAvailable method input

The following sample code is a LocationAvailable XML block that is received by the plugin from VWorks software as a string in the `LocationAvailableXML` parameter of the `IsLocationAvailable` method. VWorks software asks the plugin if the target location named `Cassette 1, Slot 1` on the storage device named `Plate Hub Carousel - 1` is available for a labware-handling process.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='cddc9ccb381e2307da078cd2efb92d4f' version='1.0' >
   <LocationAvailable Device='Plate Hub Carousel - 1' IsRelidding='0'
→IsSimulating='0' IsSourceLocation='0' Labware='1536 Black Greiner'
→Location='Cassette 1, Slot 1' PlateDevice='Plate Hub Carousel - 1'
→PlateLocation='Cassette 1, Slot 1' PlateName='process - 1' >
      <StorageLocation >
         <Location Group='0' MaxStackHeight='460' Offset='0' Type='1' />
      </StorageLocation>
      <Command Compiler='0' Editor='0' NextTaskToExecute='1' RequiresRefresh='0'
→TaskRequiresLocation='1' VisibleAvailability='1' />
   </LocationAvailable>
</Velocity11>
```

IsLocationAvailable method output

The plugin returns `VARIANT_TRUE` or `VARIANT_FALSE` in the `Available` parameter of the IsLocationAvailable method.

VARIANT_TRUE

When the plugin returns `VARIANT_TRUE` in the `Available` parameter, the scheduler does one of the following:

<Go Back

- Calls the `MakeLocationAvailable` method and begins a command cycle on the device (see "MakeLocationAvailable method" on page 67)
- Calls the `IsLocationAvailable` method again at a later time

Even if the plugin returns the value `VALIANT_TRUE`, VWorks software might not use the location, as explained in the next section.

VARIANT_FALSE

When a task is executing at a device location, VWorks software knows that the location is unavailable, or busy. Therefore, VWorks software will *not* do either of the following:

- Schedule another task until the currently executing task completes
- Call the `IsLocationAvailable` method

Most plugins return `VARIANT_TRUE` in response to this method; however, if a task is not executing, but the location should still be considered unavailable, the plugin should return `VARIANT_FALSE`.

Example of an unavailable location

When the gantry on the Bravo moves around, it blocks access to deck locations that are not occupied. This interference might last awhile. Returning `VARIANT_FALSE` allows the scheduler to move on and see what else it can do in the system.

Related information

For information about...	See...
LocationAvailable XML block components	"LocationAvailable XML block components" on page 70
MakeLocationAvailable method	"MakeLocationAvailable method" on page 67

MakeLocationAvailable method

Description

VWorks software calls the `MakeLocationAvailable` method when a labware is scheduled for delivery to a specified location. The plugin should perform all actions necessary to ensure that the target location is available for a labware-handling process. For example, a device might need to open a door and extend a labware stage at this point.

After VWorks software calls this method, it always calls the `PlateDroppedOff`, `PlatePickedUp`, or `PlateTransferAborted` method before making any other calls to the plugin. See "PlateDroppedOff method" on page 75, "PlatePickedUp method" on page 77, "PlateTransferAborted method" on page 79.

IMPORTANT The `MakeLocationAvailable` method should not return until the location is made available.

Syntax

```
HRESULT MakeLocationAvailable(
   [in] BSTR LocationAvailableXML,
   [out,retval] enum ReturnCode *retVal
);
```

Parameters

`LocationAvailableXML`	[in] A LocationAvailable XML block that contains information about the target location.
`retVal`	[out, retval] Returns an error code. Possible values: `0` = The request was completed (`RETURN_SUCCESS`) `1` = Something was wrong with the input, so the request was not completed (`RETURN_BAD_ARGS`) `2` = The request was not completed (`RETURN_FAIL`) For more information, see "ReturnCode enumerated type" on page 384.

MakeLocationAvailable method input

VWorks software passes a LocationAvailable XML block into the `LocationAvailableXML` parameter of the `MakeLocationAvailable` method.

<Go Back

LocationAvailable XML block

The LocationAvailable XML block contains the `LocationAvailable` element and all its children. This XML block contains information about the target location. The contents of the LocationAvailable XML block are different for storage and non-storage devices.

XML structure (storage device)

The value of the `file` attribute for the `Velocity11` element is `MetaData`. See "Velocity11 element" on page 46.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <LocationAvailable>
      <StorageLocation>
         <Coordinates>
            <StorageLocationCoordinate Name='Cassette' ... />
            <StorageLocationCoordinate Name='Slot' ... />
         </Coordinates>
         <Location />
      </StorageLocation>
      <Command />
   </LocationAvailable>
</Velocity11>
```

XML elements and attributes

See "LocationAvailable XML block components" on page 70.

Example of MakeLocationAvailable method input (storage device)

The following sample code is a LocationAvailable XML block that is received by the plugin from VWorks software as a string in the `LocationAvailableXML` parameter of the `MakeLocationAvailable` method. VWorks software tells the plugin to make the target location named `Cassette 1, Slot 1` available for a labware-handling process.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='54ed46ab218a7fbab40a1031a8d05f9d' version='1.0' >
   <LocationAvailable IsRelidding='0' IsSimulating='0' IsSourceLocation='0'
→Location='Cassette 1, Slot 1' Robot='Phantom Robot - 1'
→RobotObjectName='Phantom Robot' >
      <StorageLocation >
         <Coordinates >
            <StorageLocationCoordinate Name='Cassette' Value='1' />
            <StorageLocationCoordinate Name='Slot' Value='1' />
         </Coordinates>
         <Location Group='0' MaxStackHeight='460' Offset='0' Type='1' />
      </StorageLocation>
      <Command Compiler='0' Editor='0' NextTaskToExecute='1' RequiresRefresh='0'
→TaskRequiresLocation='1' VisibleAvailability='1' />
   </LocationAvailable>
</Velocity11>
```

<Go Back

XML structure (non-storage device)

The value of the `file` attribute for the `Velocity11` element is `MetaData`. See "Velocity11 element" on page 416.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <LocationAvailable>
      <StorageLocation>
         <Location />
      </StorageLocation>
      <Command />
   </LocationAvailable>
</Velocity11>
```

XML elements and attributes

See "LocationAvailable XML block components" on page 70.

Example of MakeLocationAvailable method input (non-storage device)

The following sample code is a LocationAvailable XML block that is received by the plugin from VWorks software as a string in the `LocationAvailableXML` parameter of the `MakeLocationAvailable` method. VWorks software tells the plugin to make the target location named `Stage` on the non-storage device named `PlatePad - 1` available for a labware-handling process.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='e28189485b1618286a8456afb3f4f677' version='1.0' >
   <LocationAvailable Device='PlatePad - 1' IsRelidding='0' IsSimulating='0'
→IsSourceLocation='1' Labware='1536 Black Greiner' Location='Stage'
→PlateDevice='PlatePad - 1' PlateLocation='Stage' PlateName='process - 1'
→Robot='Phantom Robot - 1' RobotObjectName='Phantom Robot' >
      <StorageLocation >
         <Location Group='0' MaxStackHeight='460' Offset='0' Type='1' />
      </StorageLocation>
      <Command Compiler='0' Editor='0' NextTaskToExecute='1' RequiresRefresh='0'
→TaskRequiresLocation='1' VisibleAvailability='1' />
   </LocationAvailable>
</Velocity11>
```

Related information

For information about...	See...
IsLocationAvailable method	"IsLocationAvailable method" on page 64
LocationAvailable XML block components	"LocationAvailable XML block components" on page 70

<Go Back

LocationAvailable XML block components

This section defines the elements and attributes that are contained in the LocationAvailable XML block.

LocationAvailable element

The `LocationAvailable` element has two children: `StorageLocation` and `Command`. The `LocationAvailable` element has the following attributes:

Name	Value
`Device`	The target device name.
`IsRelidding`	Indicates whether the labware is relidded at the target location. Possible values: `0` = The labware is not relidded `1` = The labware is relidded
`IsSimulating`	Indicates whether VWorks software is running in simulation mode. Possible values: `0` = VWorks software is not running in simulation mode `1` = VWorks software is running in simulation mode
`IsSourceLocation`	Indicates whether the target location is the same as the source location. Possible values: `0` = The target location is not the same as the source location `1` = The target location is the same as the source location
`Labware`	The labware type.
`Location`	The target location name.
`PlateDevice`	The source device name.
`PlateLocation`	The coordinates (cassette, slot) of the source location. The `Cassette` and `Slot` values are received by the plugin when VWorks software calls the `MakeLocationAvailable` method. See "MakeLocationAvailable method" on page 67.
`PlateName`	The labware name.
`Robot`	The robot name.
`RobotObjectName`	The robot type.

<Go Back

StorageLocation element

The StorageLocation element can have different structures for different device types.

StorageLocation element (Coordinates element child)

The `StorageLocation` element has two children: `Coordinates` and `Location`.

Coordinates element

The `Coordinates` element contains two `StorageLocationCoordinate` elements.

StorageLocationCoordinate element

Each `StorageLocationCoordinate` element has one of the following pairs of `Name` and `Value` attributes.

Name	Value
`Name`	The value `Cassette`.
`Value`	The name of the cassette. *Note:* The cassette name was received with a previous call to the `GetMetaData` method. See "GetMetaData method" on page 45.
`Name`	The value `Slot`.
`Value`	The name of the slot. *Note:* The slot name was received with a previous call to the `GetMetaData` method. See "GetMetaData method" on page 45.

StorageLocation element (Location element child)

The StorageLocation element contains one `Location` element.

Location element

VWorks software ignores the `Location` element in the LocationAvailable XML block. All attributes for this element are set to their default values as follows:

- `Group = 0`
- `MaxStackHeight = 460`
- `Offset = 0`
- `Type = 1`

Command element

When no task is executing or when VWorks software calls the `MakeLocationAvailable` method on a storage device where a task is executing, the `Command` element's `Name` and `PreferredTab` attributes are not specified and these attributes are set to their default values as follows:

- `Compiler = 0`
- `Editor = 0`
- `NextTaskToExecute = 1`
- `RequiresRefresh = 0`
- `TaskRequiresLocation = 1`
- `VisibleAvailability = 1`

<Go Back

Except as noted in the previous paragraph, when a task is executing, the `Command` element provides information about the task. This element contains a `Parameters` element and has the following attributes:

Name	Value
`Compiler`	See "Compiler attribute" on page 399.
`Editor`	See "Editor attribute" on page 400.
`Name`	The name of the task. This attribute is not specified if no VWorks software task is executing.
`NextTaskToExecute`	See "NextTaskToExecute attribute" on page 400.
`PreferredTab`	See "PreferredTab attribute" on page 401. This attribute is not specified if no VWorks software task is executing.
`RequiresRefresh`	See "RequiresRefresh attribute" on page 402.
`TaskRequiresLocation`	See "TaskRequiresLocation attribute" on page 402.
`VisibleAvailability`	See "VisibleAvailability attribute" on page 403.

Parameters element (IsLocationAvailable method, storage device, task is executing)

The `Parameters` element can contain one or more `Parameter` elements, where the `Name` attribute is specific to the task that is named in the `Command` element.

For example, the `Load` task has two `Parameter` elements, `loadIntoByLocation` and `loadIntoByGroup`, while the `Unload` task has only one `Parameter` element, `unloadFrom`.

Parameter element

Each `Parameter` element has one or more pairs of `Name` and `Description` attributes plus the `Scriptable`, `Style`, and `Type` attributes.

For example, the `Unload` task has only one `Parameter` element, while the `Load` task has two `Parameter` elements. The `Name` and `Description` pairs are listed in the following table:

Task	Name attribute	Description attribute
`Load`	`loadIntoByLocation`	`load plate into locations`
	`loadIntoByGroup`	`load plate into groups`
`Unload`	`unloadFrom`	`unload plate from locations`

Parameters element (IsLocationAvailable and MakeLocationAvailable methods, non-storage device, task is executing)

The `Parameters` element contains two `Parameter` elements, where the `Name` attribute has one of the following values:

- `Device to use`

<Go Back

- Location to use

Parameter element (Device to use)

The Device to use parameter specifies the device to be used by the task. This Parameter element contains one Ranges element and has the following attributes:

Name	Value
Description	The description of the parameter.
Name	The value Device to use.
Scriptable	See "Scriptable attribute" on page 410.
Style	See "Style attribute" on page 411.
Type	See "Type attribute" on page 411.
Value	The name of the device to be used.

Ranges element

The Ranges element contains one or more Range elements.

Range element

Each Range element contains the name of a device that can be used by the task. This element has the following attribute:

Name	Value
Value	The name of a device.

Parameter element (Location to use)

The Location to use parameter specifies the location to be used by the task. This Parameter element contains one Ranges element and has the following attributes:

Name	Value
Description	The description of the parameter.
Name	The value Location to use.
Scriptable	See "Scriptable attribute" on page 410.
Style	See "Style attribute" on page 411.
Type	See "Type attribute" on page 411.
Value	The name of the location to be used.

Ranges element

The Ranges element contains one or more Range elements.

<Go Back

Range element

Each `Range` element contains the name of a location that can be used by the task. This element has the following attribute:

Name	Value
`Value`	The name of the location.

<Go Back

PlateDroppedOff method

Description

VWorks software calls the PlateDroppedOff method to notify the plugin that a labware was dropped off at the location specified in the previous call to the MakeLocationAvailable method. (See "MakeLocationAvailable method" on page 67.) A call to this method also provides information about the labware that was dropped off.

IMPORTANT After VWorks software calls the MakeLocationAvailable method, it always calls the PlateDroppedOff, PlatePickedUp, or PlateTransferAborted method before making any other calls to the plugin. See "PlateDroppedOff method" on page 75, "PlatePickedUp method" on page 77, "PlateTransferAborted method" on page 79.

Syntax

```
HRESULT PlateDroppedOff(
   [in] BSTR PlateInfoXML,
   [out,retval] enum ReturnCode *retVal
);
```

Parameters

PlateInfoXML	[in] A Plates XML block that contains information about the labware that was dropped off.
retVal	[out, retval] Returns an error code. Possible values: 0 = The request was completed (RETURN_SUCCESS) 1 = Something was wrong with the input, so the request was not completed (RETURN_BAD_ARGS) 2 = The request was not completed (RETURN_FAIL) For more information, see "ReturnCode enumerated type" on page 384.

PlateDroppedOff method input

VWorks software passes a Plates XML block into the PlateInfoXML parameter of the PlateDroppedOff method.

Plates XML block

The Plates XML block contains the Plates element and all its children. This XML block provides information about labware that are used in methods involving labware-handling processes.

XML structure

The Plates XML block does not contain a Velocity11 element.

<Go Back

Note: The `MountedBy` element is only present if the labware was involved in a Mount task.

```
<?xml version='1.0' encoding='ASCII' ?>
<Plates>
   <Plate />
      <Mountby />
   </Plate>
</Plates>
```

XML elements and attributes

See "Plates XML block components" on page 81.

Example of PlateDroppedOff method input

The following sample code is a Plates XML block that is received by the plugin from VWorks software as a string in the `PlateInfoXML` parameter of the `PlateDroppedOff` method. VWorks software tells the plugin that the labware named `process - 1` was dropped off at the location named `Stage` and that the labware was involved in a Mount task with the labware named `process - 3`.

Note: The `MountedBy` element is only present if the labware was involved in a Mount task.

```
<?xml version='1.0' encoding='ASCII' ?>
<Plates file='PlateInfo' md5sum='b9a7f88767ee8b4b2c70f62171222fea' version='1.0'>
    <Plate Instance_Number='1' Labware='1536 Black Greiner' Location='Stage'
→Name='process - 1' Not_Really_Moving_The_Plate='0' Plate_Has_Lid='0' Simulating='0'
→SourceDevice='PlatePad - 1' SourceLocation='Stage'>
       <MountedBy Instance_Number='1' Labware='1536 Black Greiner' Name='process - 3'
→Plate_Has_Lid='0' />
    </Plate>
</Plates>
```

Related information

For information about...	See...
GetErrorInfo method	"GetErrorInfo method" on page 40
MakeLocationAvailable method	"MakeLocationAvailable method" on page 67
PlatePickedUp method	"PlatePickedUp method" on page 77
PlateTransferAborted method	"PlateTransferAborted method" on page 79
ReturnCode enumerated type	"ReturnCode enumerated type" on page 384

<Go Back

PlatePickedUp method

Description

VWorks software calls the `PlatePickedUp` method to notify the plugin that a labware was picked up at the location specified in the previous call to the `MakeLocationAvailable` method. (See "MakeLocationAvailable method" on page 67.) A call to the `PlatePickedUp` method also provides information about the labware that was picked up. VWorks software calls this method when the device's location is the source of a labware transfer, which includes robot transfers and configured labware.

IMPORTANT After VWorks software calls the `MakeLocationAvailable` method, it always calls the `PlateDroppedOff`, `PlatePickedUp`, or `PlateTransferAborted` method before making any other calls to the plugin. See "PlateDroppedOff method" on page 75, "PlatePickedUp method" on page 77, "PlateTransferAborted method" on page 79.

Syntax

```
HRESULT PlatePickedUp(
   [in] BSTR PlateInfoXML,
   [out,retval] enum ReturnCode *retVal
);
```

Parameters

`PlateInfoXML`	[in] A Plates XML block containing information about the labware that was picked up.
`retVal`	[out, retval] Returns an error code. Possible values: `0` = The request was completed (`RETURN_SUCCESS`) `1` = Something was wrong with the input, so the request was not completed (`RETURN_BAD_ARGS`) `2` = The request was not completed (`RETURN_FAIL`) For more information, see "ReturnCode enumerated type" on page 384.

PlatePickedUp method input

VWorks software passes a Plates XML block into the `PlateInfoXML` parameter of the `PlatePickedUp` method.

Plates XML block

The Plates XML block contains the `Plates` element and its all children. This XML block provides information about labware that are used in methods involving labware-handling processes.

<Go Back

XML structure

The Plates XML block does not contain a `Velocity11` element.

Note: The `MountedBy` element is only present if the labware was involved in a Mount task.

```
<?xml version='1.0' encoding='ASCII' ?>
<Plates>
   <Plate />
      <Mountby />
   </Plate>
</Plates>
```

XML elements and attributes

See "Plates XML block components" on page 81.

Example of PlatePickedUp method input

The following sample code is a Plates XML block that is received by the plugin from VWorks software as a string in the `PlateInfoXML` parameter of the `PlatePickedUp` method. VWorks software tells the plugin that the labware named `process - 1` was picked up at the location named `Location` and that the labware was involved in a Mount task with the labware named `process - 3`.

Note: The `MountedBy` element is only present if the labware was involved in a Mount task.

```
<?xml version='1.0' encoding='ASCII' ?>
<Plates file='PlateInfo' md5sum='1791ea89c01d01a460f8f38626207de9' version='1.0' >
    <Plate Instance_Number='1' Labware='1536 Greiner 782076 blk sqr well flt btm'
→Location='Location' Name='process - 1' Not_Really_Moving_The_Plate='0'
→Plate_Has_Lid='0' Simulating='0'>
       <MountedBy Instance_Number='1' Labware='1536 Black Greiner' Name='process - 3'
→Plate_Has_Lid='0' />
    </Plate>
</Plates>
```

Related information

For information about...	See...
GetErrorInfo method	"GetErrorInfo method" on page 40
MakeLocationAvailable method	"MakeLocationAvailable method" on page 67
PlateDroppedOff method	"PlateDroppedOff method" on page 75
PlateTransferAborted method	"PlateTransferAborted method" on page 79
ReturnCode enumerated type	"ReturnCode enumerated type" on page 384

<Go Back

PlateTransferAborted method

Description

VWorks software calls the `PlateTransferAborted` method to notify the plugin that a labware-transfer process was aborted. This is typically due to a robot malfunction followed by user intervention, that is, when the user clicks the Abort button in the standard error dialog box.

IMPORTANT After VWorks software calls the `MakeLocationAvailable` method, it always calls the `PlateDroppedOff`, `PlatePickedUp`, or `PlateTransferAborted` method before making any other calls to the plugin. See "PlateDroppedOff method" on page 75, "PlatePickedUp method" on page 77, "PlateTransferAborted method" on page 79.

Syntax

```
HRESULT PlateTransferAborted(
   [in] BSTR PlateInfoXML
);
```

Parameter

`PlateInfoXML`	[in] A Plates XML block containing information about the labware that was involved in the aborted labware-transfer process.

PlateTransferAborted method input

VWorks software passes a Plates XML block into the `PlateInfoXML` parameter of the `PlateTransferAborted` method.

Plates XML block

The Plates XML block contains the `Plates` element and all its children. This XML block provides information about labware that are used in methods involving labware-handling processes.

XML structure

The Plates XML block does not contain a `Velocity11` element.

Note: The `MountedBy` element is only present if the labware was involved in a Mount task.

```
<?xml version='1.0' encoding='ASCII' ?>
<Plates>
   <Plate />
      <Mountby />
</Plates>
```

<Go Back

XML elements and attributes

See "Plates XML block components" on page 81.

Example of PlateTransferAborted method input (with Mount task)

The following sample code is a Plates XML block that is received by the plugin from VWorks software as a string in the `PlateInfoXML` parameter of the `PlateTransferAborted` method. VWorks software tells the plugin that the labware named `process - 1` was involved in the aborted labware-transfer process at the location named `Location` and that the labware was involved in a Mount task with the labware named `process - 3`.

Note: The `MountedBy` element is only present if the labware was involved in a Mount task.

```
<?xml version='1.0' encoding='ASCII' ?>
<Plates file='PlateInfo' md5sum='1791ea89c01d01a460f8f38626207de9' version='1.0' >
    <Plate Instance_Number='1' Labware='1536 Greiner 782076 blk sqr well flt btm'
→Location='Location' Name='process - 1' Not_Really_Moving_The_Plate='0'
→Plate_Has_Lid='0' Simulating='0'>
        <MountedBy Instance_Number='1' Labware='1536 Black Greiner' Name='process - 3'
→Plate_Has_Lid='0' />
    </Plate>
</Plates>
```

Related information

For information about...	See...
Abort method	"Abort method" on page 19
MakeLocationAvailable method	"MakeLocationAvailable method" on page 67
PlateDroppedOff method	"PlateDroppedOff method" on page 75
PlatePickedUp method	"PlatePickedUp method" on page 77

<Go Back

Plates XML block components

VWorks software passes a Plates XML block into the `PlateInfoXML` parameter of the `PlateDroppedOff`, `PlatePickedUp`, and `PlateTransferAborted` methods. The following table shows the XML components that are contained in this XML block for each method. The elements and attributes are described in the sections that follow. You can click an element name to jump to the appropriate section.

Method name	PlateDroppedOff		PlatePickedUp		PlateTransferAborted	
Element name Attribute name	No Mount task	Mount task	No Mount task	Mount task	No Mount task	Mount task
Plates	✓	✓	✓	✓	✓	✓
Plate	✓	✓	✓	✓	✓	✓
Instance_Number	✓	✓	✓	✓	✓	✓
Labware	✓	✓	✓	✓	✓	✓
Location	✓	✓	✓	✓	✓	✓
Name	✓	✓	✓	✓	✓	✓
Not_Really_Moving_The_Plate	✓	✓	✓	✓	✓	✓
Plate_Has_Lid	✓	✓	✓	✓	✓	✓
Simulating	✓	✓	✓	✓	✓	✓
SourceDevice	✓	✓				
SourceLocation	✓	✓				
MountedBy		✓		✓		✓
Instance_Number		✓		✓		✓
Labware		✓		✓		✓
Name		✓		✓		✓
Plate_Has_Lid		✓		✓		✓

Plates element

The `Plates` element contains one `Plate` element.

Plate element

The `Plate` element has the following attributes:

Name	Value
`Instance_Number`	The labware instance number.

<Go Back

Name	Value
Labware	The labware type.
Location	The name of the location where the labware was dropped off.
Name	The labware name.
Not_Really_Moving_The_Plate	Indicates whether the robot is moving the labware for delidding/relidding. Possible values: 0 = The robot is not moving the labware for delidding/relidding, but it is performing some other task 1 = The robot is moving the labware for delidding/relidding
Plate_Has_Lid	Indicates whether the labware has a lid. Possible values: 0 = The labware does not have a lid 1 = The labware has a lid
Simulating	Indicates whether VWorks software is running in simulation mode. Possible values: 0 = VWorks software is not running in simulation mode 1 = VWorks software is running in simulation mode
SourceDevice	The name of the device from which the labware was picked up.
SourceLocation	The name of the location on the source device from which the labware was picked up.

MountedBy element

The MountedBy element is only specified when the labware-handling task involves a Mount task. This element contains information about the mounting labware and has the following attributes:

Name	Value
Instance_Number	The mounting labware instance number.
Labware	The mounting labware type.
Name	The mounting labware name.
Plate_Has_Lid	Indicates whether the mounting labware has a lid. Possible values: 0 = The mounting labware does not have a lid 1 = The mounting labware has a lid

PrepareForRun method

Description

VWorks software calls the `PrepareForRun` method to notify the plugin that the user clicked the Start button in the VWorks window. This method, which is called each time a protocol is run, tells the plugin that a run is starting. If the plugin maintains per-run state information, the state should be cleared during this call.

Syntax

```
HRESULT PrepareForRun(
   [in] BSTR LocationInfoXML,
   [out,retval] enum ReturnCode *retVal
);
```

Parameters

`LocationInfoXML`	[in] A Locations XML block that contains the names of the locations on the device and of any labware that is present at each location.
`retVal`	[out, retval] Returns an error code. Possible values: `0` = The request was completed (`RETURN_SUCCESS`) `1` = Something was wrong with the input, so the request was not completed (`RETURN_BAD_ARGS`) `2` = The request was not completed (`RETURN_FAIL`) For more information, see "ReturnCode enumerated type" on page 384.

PrepareForRun method input

VWorks software passes a Locations XML block into the `LocationInfoXML` parameter of the `PrepareForRun` method.

Locations XML block

The Locations XML block contains the `Locations` element and all its children. This XML block provides the names of the locations on the device and of any labware that is present at each location.

VWorks software received the location names from the plugin with a previous call to the `GetMetaData` method. See "GetMetaData method" on page 45.

<Go Back

XML structure

The Locations XML block does not contain a `Velocity11` element.

```
<?xml version='1.0' encoding='ASCII' ?>
<Locations>
   <Location />
   ...
</Locations>
```

Locations element

The `Locations` element contains one or more `Location` elements.

Location element

Each `Location` element contains the name of a location on the device and the type of the labware at that location, if present. This element has the following attributes:

Name	Value
`Labware`	The labware type. If no labware is present at the location, this attribute is not specified.
`Name`	The name of the location.

Example of PrepareForRun method input

The following sample code is a Locations XML block received by the plugin from VWorks software as a string in the `LocationInfoXML` parameter of the `PrepareForRun` method. VWorks software tells the plugin that a protocol run is starting on the labware of type `1536 Greiner 782076 blk sqr well flt btm` at the location named `Stage`.

```
<?xml version='1.0' encoding='ASCII' ?>
<Locations file='LocationInfo' md5sum='e8f85c5eac89c1a0a82d7ff74ae591f8'
→version='1.0' >
   <Location Labware='1536 Greiner 782076 blk sqr well flt btm' Name='Stage' />
</Locations>
```

Related information

For information about...	See...
GetErrorInfo method	"GetErrorInfo method" on page 40
ReturnCode enumerated type	"ReturnCode enumerated type" on page 384

Retry method

Description

VWorks software calls the `Retry` method as follows:

1. The plugin notifies VWorks software that an error occurred during a task.
2. VWorks software does the following:
   a. Calls the `GetErrorInfo` method to get a text string from the plugin that describes the error. See "GetErrorInfo method" on page 40.
   b. Writes the string to the Main Log.
   c. Displays the standard error dialog box, which includes the following components:
      - The error text string.
      - The Abort, Retry, and Ignore and Continue... buttons.

      The figure on page 40 shows a standard error dialog box.
3. If the user clicks the Retry button, VWorks software calls the `Retry` method to tell the plugin to retry the task.

   Because VWorks software assumes the user manually solved the problem that caused the error, the plugin should try to restart the task. The plugin should record the state of the task and retry from the point in the task that makes the most sense given the current state.

   For example, a single-column dispenser that encounters an error after partially filling a labware should not start over, because the dispenser might deliver too much reagent to the already-covered wells. The dispenser should continue as close to the point of interruption as possible to avoid over-dispensing or under-dispensing the wells that were being filled at the time the error occurred.

Syntax

```
HRESULT Retry(
   [out,retval] enum ReturnCode *retVal
);
```

Parameter

`retVal`	[out, retval] Returns an error code. Possible values: `0` = The request was completed (`RETURN_SUCCESS`) `1` = Something was wrong with the input, so the request was not completed (`RETURN_BAD_ARGS`) `2` = The request was not completed (`RETURN_FAIL`) For more information, see "ReturnCode enumerated type" on page 384.

<Go Back

Related information

For information about...	See...
Abort method	“Abort method” on page 19
GetErrorInfo method	“GetErrorInfo method” on page 40
Ignore method	“Ignore method” on page 59
PrepareForRun method	“PrepareForRun method” on page 83
ReturnCode enumerated type	“ReturnCode enumerated type” on page 384

4 IControllerClient interface

VWorks device driver plugins must implement the IControllerClient interface to get a pointer from VWorks software to the IWorksController interface.

This chapter defines the IControllerClient method.

IMPORTANT All VWorks device driver plugins must implement the IWorksDriver, IControllerClient, and IWorksDiags interfaces.

This chapter contains the following topic:

• "SetController method" on page 88

<Go Back

SetController method

Description

When VWorks software loads each plugin, it checks whether the plugin implements the IControllerClient interface. If the plugin does, VWorks software calls the `SetController` method and passes a pointer to itself in the `Controller` parameter. The pointer is valid because VWorks software implements the IWorksController interface. See "IWorksController interface" on page 281.

The plugin stores the pointer and uses it to call the following IWorksController methods, which asks VWorks software to perform actions that the plugin cannot do on its own.

- "NotifyDataChanged method" on page 285
- "NotifyTipOperation method" on page 287
- "OnCloseDiagsDialog method" on page 290
- "PrintToLog method" on page 291
- "Query method" on page 292
- "Update method" on page 347

IMPORTANT To use IWorksController methods, the plugin must first get a pointer to VWorks software, which implements the IWorksController interface.

Syntax

```
HRESULT SetController(
   [in] IWorksController *Controller
);
```

Parameters

`Controller`	[in] A pointer to VWorks software, which implements the IWorksController interface.

Related information

For information about...	See...
IWorksController interface	"IWorksController interface" on page 281
IWorksController NotifyDataChanged method	"NotifyDataChanged method" on page 285
IWorksController NotifyTipOperation method	"NotifyTipOperation method" on page 287

<Go Back

For information about...	See...
IWorksController OnCloseDiagsDialog method	“OnCloseDiagsDialog method” on page 290
IWorksController PrintToLog method	“PrintToLog method” on page 291
IWorksController Query method	“Query method” on page 292
IWorksController Update method	“Update method” on page 347

<Go Back

5 IWorksDiags interface

To open diagnostics dialog boxes in VWorks software, VWorks plugins must implement the IWorksDiags interface.

This chapter defines the IWorksDiags methods.

IMPORTANT All VWorks device driver plugins must implement the IWorksDriver, IControllerClient, and IWorksDiags interfaces.

This chapter contains the following topics:

- "IWorksDiags methods overview" on page 92
- "CloseDiagsDialog method" on page 93
- "ShowDiagsDialog method" on page 94

<Go Back

IWorksDiags methods overview

Use the following table to quickly locate an IWorksDiags method by name, by description, or by page number.

Method	Description	See...
`CloseDiagsDialog`	Tells the plugin to close its diagnostics dialog box.	"CloseDiagsDialog method" on page 93
`IsDiagsDialogOpen`	*Obsolete*. This method should be implemented as `return E_NOTIMPL (0x80004001)`.	
`ShowDiagsDialog`	Tells the plugin to open its diagnostics dialog box.	"ShowDiagsDialog method" on page 94

<Go Back

CloseDiagsDialog method

Description

VWorks software calls the `CloseDiagsDialog` method to tell the plugin to close its diagnostics dialog box.

IMPORTANT To properly inform VWorks software that the diagnostics dialog box is closed, the plugin must call the IWorksController `OnCloseDiagsDialog` method; otherwise, VWorks software assumes that the dialog box is still open and returns an error. See "OnCloseDiagsDialog method" on page 290.

Syntax

```
HRESULT CloseDiagsDialog(
   [out,retval] enum ReturnCode *retVal
);
```

Parameters

`retVal`	[out, retval] Returns an error code. Possible values: `0` = The request was completed (`RETURN_SUCCESS`) `1` = Something was wrong with the input, so the request was not completed (`RETURN_BAD_ARGS`) `2` = The request was not completed (`RETURN_FAIL`) For more information, see "ReturnCode enumerated type" on page 384.

Related information

For information about...	See...
IWorksController OnCloseDiagsDialog method	"OnCloseDiagsDialog method" on page 290
ShowDiagsDialog method	"ShowDiagsDialog method" on page 94

<Go Back

ShowDiagsDialog method

Description

VWorks software calls the `ShowDiagsDialog` method to tell the plugin to open its diagnostics dialog box. This method displays both modal and modeless dialog boxes.

IMPORTANT To properly inform VWorks software that the diagnostics dialog box is closed, the plugin must call the IWorksController `OnCloseDiagsDialog` method; otherwise, VWorks software assumes that the dialog box is still open and returns an error. See "OnCloseDiagsDialog method" on page 290.

Agilent Technologies provides a standard diagnostics dialog box that contains two tabs: one tab for specifying profile settings and the other for controlling the device and viewing device status.

For complex devices, developers of VWorks plugins might want to add controls and components to the standard diagnostics dialog box such as the following:

- Tabs that enable additional functionality
- Embedded tab controls or command buttons that allow the user to access the device vendor's software

Syntax

```
HRESULT ShowDiagsDialog(
   [in] enum SecurityLevel iSecurity,
   [in] VARIANT_BOOL bModal
);
```

<Go Back

Parameters

`iSecurity`	[in] Represents the security level, or access privilege, for the user currently logged in to VWorks software. VWorks software determines the functions that a user can perform in the diagnostics dialog box according to the user's access level privilege. *Note:* Refer to the *VWorks Automation Control Setup Guide* for more information about user accounts and privileges. Possible values: `0` = The access level privilege for the current user is Administrator (`SECURITY_LEVEL_ADMINISTRATOR`) `1` = The access level privilege for the current user is Technician (`SECURITY_LEVEL_TECHNICIAN`) `2` = The access level privilege for the current user is Operator (`SECURITY_LEVEL_OPERATOR`) `3` = The access level privilege for the current user is Guest (`SECURITY_LEVEL_GUEST`) `-1` = No user is currently logged in to VWorks software (`SECURITY_LEVEL_NO_ACCESS`)
`bModal`	[in] Indicates whether the plugin should display a modal or a modeless dialog box. Possible values: `-1` = Display a modal dialog box (`VARIANT_TRUE`) `0` = Display a modeless dialog box (`VARIANT_FALSE`)

Related information

For information about...	See...
CloseDiagsDialog method	"CloseDiagsDialog method" on page 93
IWorksController OnCloseDiagsDialog method	"OnCloseDiagsDialog method" on page 290

<Go Back

6 IBCRDriver interface

VWorks plugins that control barcode readers must implement the IBCRDriver interface.

This chapter defines the IBCRDriver method.

IMPORTANT All VWorks device driver plugins must implement the IWorksDriver, IControllerClient, and IWorksDiags interfaces.

This chapter contains the following topic:

• "Strobe method" on page 98

<Go Back

Strobe method

Description

VWorks software calls the `Strobe` method to tell the plugin to activate the barcode reader and then read the barcode.

Syntax

```
HRESULT Strobe(
   [out] BSTR *bcrxml,
   [out,retval] enum ReturnCode *retVal
);
```

Parameters

`bcrxml`	[out] A Barcode XML element containing the barcode that was read by the barcode reader.
`retVal`	[out, retval] Returns an error code. Possible values: `0` = The request was completed (`RETURN_SUCCESS`) `1` = Something was wrong with the input, so the request was not completed (`RETURN_BAD_ARGS`) `2` = The request was not completed (`RETURN_FAIL`) For more information, see "ReturnCode enumerated type" on page 384.

Strobe method output

The plugin returns a Barcode XML element in the `bcrxml` parameter of the `Strobe` method. This XML element contains the barcode that was read by the barcode reader.

Barcode element

The `Barcode` element has the following attribute:

Name	Value
`Barcode`	The barcode. Required: Yes

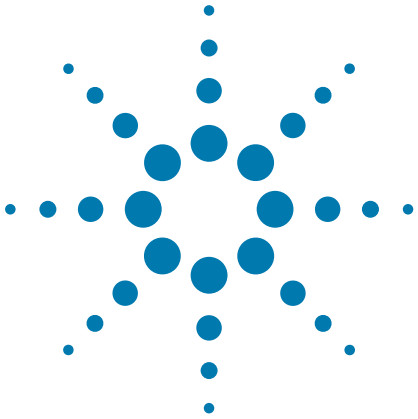
<Go Back

Example of Strobe method output

The following sample code is a Barcode XML element that is returned to VWorks software by the plugin as a string in the `bcrxml` parameter of the `Strobe` method. The code contains the barcode that was read by the barcode reader.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='0090cbcd92e9e12d7dc90765978a7a0d' version='1.0' >
   <Barcode Barcode='barcode1' />
</Velocity11>
```

<Go Back



7 IIODriver interface

VWorks plugins that manage I/O signals must implement the IIODriver interface.

This chapter defines the IIODriver methods.

IMPORTANT All VWorks device driver plugins must implement the IWorksDriver, IControllerClient, and IWorksDiags interfaces.

This chapter contains the following topics:

- "IIODriver methods overview" on page 102
- "EnumPoints method" on page 103
- "Read method" on page 105
- "Set method" on page 108

<Go Back

IIODriver methods overview

Use the following table to quickly locate an IIODriver method by name, by description, or by page number.

Method	Description	See...
`EnumPoints`	Tells the plugin to enumerate all I/O channels for a device.	"EnumPoints method" on page 103
`Read`	Tells the plugin to read the input signal (voltage) on the specified digital or analog input channel.	"Read method" on page 105
`Set`	Tells the plugin to command the specified digital output channel on the device to output the specified voltage.	"Set method" on page 108

Go Back

EnumPoints method

Description

VWorks software calls the `EnumPoints` method to tell the plugin to enumerate all I/O channels for a device.

Syntax

```
HRESULT EnumPoints(
   [out] BSTR *ioxml,
   [out,retval] enum ReturnCode *retVal
);
```

Parameters

`ioxml`	[out] Three XML elements: digitalInputs, analogInputs, and digitalOutputs. Each XML element contains an enumeration array of all digital input, analog input, or digital output channels for a device.
`retVal`	[out, retval] Returns an error code. Possible values: `0` = The request was completed (`RETURN_SUCCESS`) `1` = Something was wrong with the input, so the request was not completed (`RETURN_BAD_ARGS`) `2` = The request was not completed (`RETURN_FAIL`) For more information, see "ReturnCode enumerated type" on page 384.

EnumPoints method output

The plugin returns the following three XML elements in the `ioxml` parameter of the `EnumPoints` method:

- digitalInputs XML element
- analogInputs XML element
- digitalOutputs XML element

The value of the `file` attribute of the `Velocity11` element is `Velocity11`. See "Velocity11 element" on page 416.

digitalInputs, analogInputs, or digitalOutputs elements

Each `digitalInputs`, `analogInputs`, and `digitalOutputs` element contains one or more `name` elements.

Required: No

Default value: An empty string

<Go Back

name element

Each `name` element contains the name of an input or output channel.

Required: No

Default value: An empty string

Example of EnumPoints method output

The following sample code contains the (truncated) digitalInputs, analogInputs, and digitalOutputs XML elements (with enumeration arrays) that are returned to VWorks software by the plugin as a string in the `ioxml` parameter of the `EnumPoints` method.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Velocity11' md5sum='d523224aed7bdf522e1a8901e13d2bf6'
→version='1.1' >
   <digitalInputs >
      <name >Input 1 for Temperature</name>
      <name >Input 2 for Humidity</name>
      <name >Input 3 for Waste bin door</name>
      ...
      <name >Input 19 (00-00-00-00-00-00)</name>
      <name >Input 20 (00-00-00-00-00-00)</name>
   </digitalInputs>
   <analogInputs >
      <name >Fluid level 1</name>
      <name >Fluid level 2</name>
      <name >Humidity</name>
      ...
      <name >Analog 11 (00-00-00-00-00-00)</name>
      <name >Analog 12 (00-00-00-00-00-00)</name>
   </analogInputs>
   <digitalOutputs >
      <name >Ventilation fan</name>
      <name >UV light</name>
      <name >Alarm (audible)</name>
      ...
      <name >Output 15 (00-00-00-00-00-00)</name>
      <name >Output 16 (00-00-00-00-00-00)</name>
   </digitalOutputs>
</Velocity11>
```

Related information

For information about...	See...
Read method	“Read method” on page 105
Set method	“Set method” on page 108

Read method

Description

VWorks software calls the `Read` method to tell the plugin to read the input signal (voltage) on the specified digital or analog input channel.

Syntax

```
HRESULT Read(
   [in,out] BSTR *xml,
   [out,retval] enum ReturnCode *retVal
);
```

Parameters

`xml`	[in, out] The input `xml` parameter contains the following: • digitalInputs XML block with the name of a digital input channel • analogInputs XML block with the name of an analog input channel The output `xml` parameter contains the following: • digitalInput XML element with the input signal (voltage) on the specified digital input channel • analogInput XML element with the input signal (voltage) on the specified analog input channel
`retVal`	[out, retval] Returns an error code. Possible values: `0` = The request was completed (`RETURN_SUCCESS`) `1` = Something was wrong with the input, so the request was not completed (`RETURN_BAD_ARGS`) `2` = The request was not completed (`RETURN_FAIL`) For more information, see "ReturnCode enumerated type" on page 384.

Read method input

VWorks software passes one of the following XML blocks into the `xml` parameter of the `Read` method:

- digitalInputs XML block
- analogInputs XML block

The value of the `file` attribute of the `Velocity11` element is `Velocity11`. See "Velocity11 element" on page 416.

<Go Back

digitalInputs XML block

The digitalInputs XML block contains the `digitalInputs` element and one `name` element. The `name` element contains the name of a digital input channel.

analogInputs XML block

The analogInputs XML block contains the `analogInputs` element and one `name` element. The `name` element contains the name of an analog input channel.

Example of Read method input received by the plugin

The following sample code is an analogInputs XML block that is received by the plugin from VWorks software as a string in the `xml` parameter of the `Read` method. VWorks software asks the plugin for the current input signal (voltage) on the analog input channel named `Humidity`.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Velocity11' md5sum='dabdec8b39169630e4ff61e0388b5142'
→version='1.1' >
   <analogInputs >
      <name >Humidity</name>
   </analogInputs>
</Velocity11>
```

Read method output

The plugin returns a digitalInput XML element or an analogInput XML element in the `xml` parameter of the `Read` method.

digitalInput and analogInput elements

Each `digitalInput` and `analogInput` element has the following attributes:

Name	Value
`name`	The name of the input channel. Required: Yes
`value`	The current input signal (voltage). Required: Yes

Example of Read method input returned by the plugin

The following sample code contains an analogInput XML element that is returned by the plugin to VWorks software as a string in the `xml` parameter of the `Read` method. The code contains the current voltage of `70` volts for the analog input channel named `Humidity`.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Velocity11' md5sum='cc239a8f99203e73b3de0ed8a058f77e'
→version='1.1' >
   <analogInput name='Humidity' value='70' />
</Velocity11>
```

<Go Back

Related information

For information about...	See...
EnumPoints method	“EnumPoints method” on page 103
Set method	“Set method” on page 108

<Go Back

Set method

Description

VWorks software calls the `Set` method to tell the plugin to command the specified digital output channel on the device to output the specified voltage.

Syntax

```
HRESULT Set(
   [in] BSTR xml,
   [out,retval] enum ReturnCode *retVal
);
```

Parameters

`xml`	[in] A digitalOutput XML element that contains the name of a digital output channel on a device and the voltage to set.
`retVal`	[out, retval] Returns an error code. Possible values: `0` = The request was completed (`RETURN_SUCCESS`) `1` = Something was wrong with the input, so the request was not completed (`RETURN_BAD_ARGS`) `2` = The request was not completed (`RETURN_FAIL`) For more information, see "ReturnCode enumerated type" on page 384.

Set method input

VWorks software passes a digitalOutput XML element into the `xml` parameter of the `Set` method.

The value of the `file` attribute of the `Velocity11` element is `Velocity11`. See "Velocity11 element" on page 416.

digitalOutput element

The `digitalOutput` element has the following attributes:

Name	Value
`name`	The name of the output channel.
`value`	The output signal (voltage) to set.

<Go Back

Example of Set method input

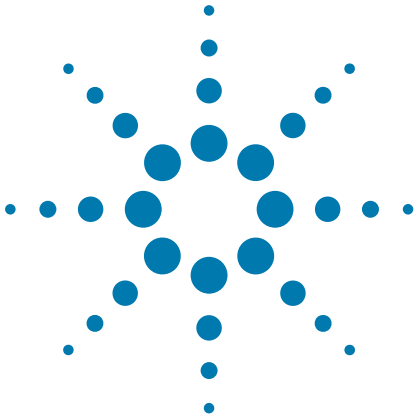
The following sample code contains a digitalOutput XML element that is received by the plugin from VWorks software as a string in the `xml` parameter of the `Set` method. VWorks software tells the plugin to command the digital output channel named `Output channel` to output `100` volts.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Velocity11' md5sum='bfea461a5f6cf84f987ede1376b68cdf'
→version='1.1' >
   <digitalOutput name='Output channel' value='100' />
</Velocity11>
```

Related information

For information about...	See...
EnumPoints method	"EnumPoints method" on page 103
Read method	"Read method" on page 105

<Go Back



8 ILabelerDriver interface

VWorks plugins that perform labware-labeling tasks must implement the ILabelerDriver interface.

This chapter defines the ILabelerDriver methods.

IMPORTANT All VWorks device driver plugins must implement the IWorksDriver, IControllerClient, and IWorksDiags interfaces.

This chapter contains the following topics:

- "ILabelerDriver methods overview" on page 112
- "EnumerateFormats method" on page 113
- "Print method" on page 115
- "PrintAndApply method" on page 117
- "PrinterMetaData XML block components" on page 119

<Go Back

ILabelerDriver methods overview

Use the following table to quickly locate an ILabelerDriver method by name, by description, or by page number.

Method	Description	See...
`EnumerateFormats`	Tells the plugin to enumerate all available label formats.	"EnumerateFormats method" on page 113
`Print`	Tells the plugin to print a labware label using the data provided in the `labelxml` parameter.	"Print method" on page 115
`PrintAndApply`	Tells the plugin to print and apply a labware label using the data provided in the `labelxml` parameter.	"PrintAndApply method" on page 117

EnumerateFormats method

Description

VWorks software calls the `EnumerateFormats` method to tell the plugin to enumerate all available label formats.

Syntax

```
HRESULT EnumerateFormats(
   [out] BSTR *formats,
   [out,retval] enum ReturnCode *retVal
);
```

Parameters

`formats`	[out] A PrinterMetaData XML block that contains the names of all available label formats and their associated field names.
`retVal`	[out, retval] Returns an error code. Possible values: `0` = The request was completed (`RETURN_SUCCESS`) `1` = Something was wrong with the input, so the request was not completed (`RETURN_BAD_ARGS`) `2` = The request was not completed (`RETURN_FAIL`) For more information, see "ReturnCode enumerated type" on page 384.

EnumerateFormats method output

The plugin returns a PrinterMetaData XML block in the `formats` parameter of the `EnumerateFormats` method.

PrinterMetaData XML block

The PrinterMetaData XML block contains the `PrinterMetaData` element and all its children. This XML block provides the names of all available label formats and their associated field names.

<Go Back

XML format

The value of the `file` attribute for the `Velocity11` element is `MetaData`. See "Velocity11 element" on page 416.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <PrinterMetaData>
      <PrinterFormatMetaData>
         <PrinterFormatMetaData>
            <PrinterFieldMetaData>
               <PrinterFieldMetaData />
               ...
            </PrinterFieldMetaData>
         </PrinterFormatMetaData>
         ...
      </PrinterFormatMetaData>
   </PrinterMetaData>
</Velocity11>
```

XML elements and attributes

See "PrinterMetaData XML block components" on page 119.

Example of EnumerateFormats method output

The following sample code is a PrinterFormatMetaData XML block that is returned to VWorks software by the plugin as a string in the `formats` parameter of the `EnumerateFormats` method. The plugin returns the label format named `1` and enumerates its fields.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='883fb23dbd3ba29bded4dbbe6d8b9b5d' version='1.0' >
   <PrinterMetaData >
      <PrinterFormatMetaData >
         <PrinterFormatMetaData Name='1' >
            <PrinterFieldMetaData >
               <PrinterFieldMetaData Name='1' />
               <PrinterFieldMetaData Name='2' />
               <PrinterFieldMetaData Name='3' />
               <PrinterFieldMetaData Name='4' />
               <PrinterFieldMetaData Name='5' />
               <PrinterFieldMetaData Name='6' />
            </PrinterFieldMetaData>
         </PrinterFormatMetaData>
      </PrinterFormatMetaData>
   </PrinterMetaData>
</Velocity11>
```

Related information

For information about...	See...
Print method	"Print method" on page 115
PrintAndApply method	"PrintAndApply method" on page 117

Print method

Description

VWorks software calls the `Print` method to tell the plugin to print a labware label using the data provided in the `labelxml` parameter. The plugin should not *apply* the label.

Syntax

```
HRESULT Print(
   [in] BSTR *labelxml,
   [out,retval] enum ReturnCode *retVal
);
```

Parameters

`labelxml`	[in] A PrinterMetaData XML block that contains the data to be printed on the labware label.
`retVal`	[out, retval] Returns an error code. Possible values: `0` = The request was completed (`RETURN_SUCCESS`) `1` = Something was wrong with the input, so the request was not completed (`RETURN_BAD_ARGS`) `2` = The request was not completed (`RETURN_FAIL`) For more information, see "ReturnCode enumerated type" on page 384.

Print method input

VWorks software passes a PrinterMetaData XML block into the `labelxml` parameter of the `Print` method.

PrinterMetaData XML block

The PrinterFormatMetaData XML block contains the `PrinterMetaData` element and all its children. This XML block provides the name of a label format, its associated field names, and the contents of each field, if any. The PrinterFormatMetaData XML block also tells the plugin not to apply the label.

<Go Back

XML format

The value of the `file` attribute for the `Velocity11` element is `MetaData`. See "Velocity11 element" on page 416.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <PrinterMetaData>
      <PrinterFormatMetaData>
         <PrinterFormatMetaData>
            <PrinterFieldMetaData>
               <PrinterFieldMetaData />
               ...
            </PrinterFieldMetaData>
         </PrinterFormatMetaData>
      </PrinterFormatMetaData>
   </PrinterMetaData>
</Velocity11>
```

XML elements and attributes

See "PrinterMetaData XML block components" on page 119.

Example of Print method input

The following sample code is a PrinterMetaData XML block received by the plugin from VWorks software as a string in the `labelxml` parameter of the `Print` method. VWorks software tells the plugin to print the value `barcode` in the field named `1` using the label format named `Name`. The value of the `Side` attribute tells the plugin not to *apply* the label.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='b1488a186ae7275b123b894d720ea22c' version='1.0' >
   <PrinterMetaData >
      <PrinterFormatMetaData >
         <PrinterFormatMetaData Name='1' Side='No Apply' >
            <PrinterFieldMetaData >
               <PrinterFieldMetaData Name='1' Value='barcode' />
               <PrinterFieldMetaData Name='2' />
               <PrinterFieldMetaData Name='3' />
               <PrinterFieldMetaData Name='4' />
               <PrinterFieldMetaData Name='5' />
               <PrinterFieldMetaData Name='6' />
            </PrinterFieldMetaData>
         </PrinterFormatMetaData>
      </PrinterFormatMetaData>
   </PrinterMetaData>
</Velocity11>
```

Related information

For information about...	See...
EnumerateFormats method	"EnumerateFormats method" on page 113
PrintAndApply method	"PrintAndApply method" on page 117

<Go Back

PrintAndApply method

Description

VWorks software calls the `PrintAndApply` method to tell the plugin to print and apply a labware label using the data provided in the `labelxml` parameter.

Syntax

```
HRESULT PrintAndApply(
   [in] BSTR *labelxml,
   [out,retval] enum ReturnCode *retVal
);
```

Parameters

`labelxml`	[in] A PrinterMetaData XML block that contains the data to be printed on the labware label.
`retVal`	[out, retval] Returns an error code. Possible values: `0` = The request was completed (`RETURN_SUCCESS`) `1` = Something was wrong with the input, so the request was not completed (`RETURN_BAD_ARGS`) `2` = The request was not completed (`RETURN_FAIL`) For more information, see "ReturnCode enumerated type" on page 384.

PrintAndApply method input

VWorks software passes a PrinterMetaData XML block into the `labelxml` parameter of the `PrintAndApply` method.

PrinterMetaData XML block

The PrinterFormatMetaData XML block contains the `PrinterMetaData` element and all its children. This XML block provides the name of a label format, its associated field names, and the contents of each field, if any. The PrinterFormatMetaData XML block also specifies the side of the labware on which to apply the label.

<Go Back

XML format

The value of the `file` attribute for the `Velocity11` element is `MetaData`. See "Velocity11 element" on page 416.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <PrinterMetaData>
      <PrinterFormatMetaData>
         <PrinterFormatMetaData>
            <PrinterFieldMetaData>
               <PrinterFieldMetaData />
               ...
            </PrinterFieldMetaData>
         </PrinterFormatMetaData>
      </PrinterFormatMetaData>
   </PrinterMetaData>
</Velocity11>
```

XML elements and attributes

See "PrinterMetaData XML block components" on page 119.

Example of PrintAndApply method input

The following sample code is a PrinterMetaData XML block received by the plugin from VWorks software as a string in the `labelxml` parameter of the `Print` method. VWorks software tells the plugin to print the value `barcode` in the field named `1` using the label format named `Name`. The value of the `Side` attribute tells the plugin to print *and apply* the label.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='feedb569f1ef7142bd50e4e5ae5a031e' version='1.0'>
   <PrinterMetaData >
      <PrinterFormatMetaData >
         <PrinterFormatMetaData Name='1' Side='South' >
            <PrinterFieldMetaData >
               <PrinterFieldMetaData Name='1' Value='barcode' />
               <PrinterFieldMetaData Name='2' />
               <PrinterFieldMetaData Name='3' />
               <PrinterFieldMetaData Name='4' />
               <PrinterFieldMetaData Name='5' />
               <PrinterFieldMetaData Name='6' />
            </PrinterFieldMetaData>
         </PrinterFormatMetaData>
      </PrinterFormatMetaData>
   </PrinterMetaData>
</Velocity11>
```

Related information

For information about...	See...
EnumerateFormats method	"EnumerateFormats method" on page 113
Print method	"Print method" on page 115

<Go Back

PrinterMetaData XML block components

The plugin returns a PrinterMetaData XML block in the `formats` parameter of the `EnumerateFormats` method. VWorks software passes a PrinterMetaData XML block into the `labelxml` parameter of the `Print` and `PrintAndApply` methods. The following table lists the XML components that are contained in this XML block for each method. The elements and attributes are described in the sections that follow. You can click an element name to jump to the appropriate section.

	Method name		
Element name Attribute name	EnumerateFormats	Print	PrintAndApply
PrinterMetaData	✓	✓	✓
PrinterFormatMetaData (parent)	✓	✓	✓
PrinterFormatMetaData (child)	✓	✓	✓
Name	✓	✓	✓
Side		✓	✓
PrinterFieldMetaData (parent)	✓	✓	✓
PrinterFieldMetaData (child)	✓	✓	✓
Name	✓	✓	✓
Value		✓	✓

PrinterMetaData element

The `PrinterMetaData` element contains one `PrinterFormatMetaData` parent element.

PrinterFormatMetaData element (parent)

For the `EnumerateFormats` method, the `PrinterFormatMetaData` parent element contains one or more `PrinterFormatMetaData` child elements.

For the `Print` and `PrintAndApply` methods, the `PrinterFormatMetaData` parent element contains one `PrinterFormatMetaData` child element.

PrinterFormatMetaData element (child)

The `PrinterFormatMetaData` child element contains one `PrinterFieldMetaData` parent element and has the following attributes:

Name	Value
`Name`	The label format name.

<Go Back

Name	Value
Side	For the Print method, the value No Apply. For the PrintAndApply method, the side of the labware on which to apply the label. Possible values: • North • South • East • West

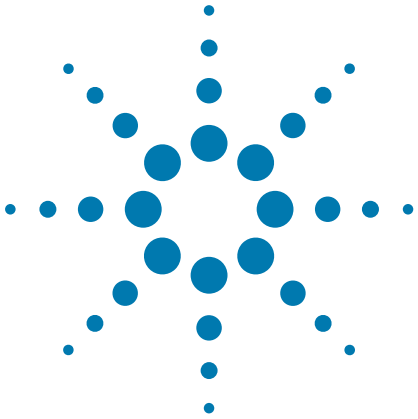
PrinterFieldMetaData element (parent)

The PrinterFieldMetaData parent element contains one or more PrinterFieldMetaData child elements.

PrinterFieldMetaData element (child)

The PrinterFieldMetaData child element has the following attributes:

Name	Value
Name	The field name.
Value	The contents of the field, for example, barcode, date, time, or other static text. If no Value attributes are specified, the plugin will not print anything.



9 ILiddingDriver interface

VWorks plugins that perform delidding and relidding operations must implement the ILiddingDriver interface.

This chapter defines the ILiddingDriver methods.

IMPORTANT All VWorks device driver plugins must implement the IWorksDriver, IControllerClient, and IWorksDiags interfaces.

This chapter contains the following topics:

- "ILiddingDriver methods overview" on page 122
- "LidIsRetained method" on page 123
- "OnDelidMoveComplete method" on page 124
- "OnRelidMoveComplete method" on page 127
- "RobotEndsUpHoldingPlate method" on page 130

<Go Back

ILiddingDriver methods overview

Use the following table to quickly locate an ILiddingDriver method by name, by description, or by page number.

Method	Description	See...
`LidIsRetained`	Asks the plugin whether the device retains the lid after a delid move.	"LidIsRetained method" on page 123
`OnDelidMoveComplete`	Notifies the plugin that a delid operation was executed.	"OnDelidMoveComplete method" on page 124
`OnRelidMoveComplete`	Notifies the plugin that a relid operation was executed.	"OnRelidMoveComplete method" on page 127
`RobotEndsUpHoldingLid`	*Deprecated.* This method should be implemented as `return E_NOTIMPL (0x80004001)`.	
`RobotEndsUpHoldingPlate`	Asks the plugin if the robot is holding the labware at the end of the delidding/relidding process.	"RobotEndsUpHoldingPlate method" on page 130

<Go Back

LidIsRetained method

Description

VWorks software calls the `LidIsRetained` method to ask the plugin whether the device retains the lid after a delid operation.

Note: If the device does not retain the lid, for example, if the device discards the lid as waste, relidding is not possible.

Syntax

```
HRESULT LidIsRetained(
   [out,retval] long *retVal
);
```

Parameters

`retVal`	[out, retval] Indicates whether the device retains the lid after a delid operation. Possible values: `0` = The device does not retain the lid `1` = The device retains the lid

Related information

For information about...	See...
OnDelidMoveComplete method	"OnDelidMoveComplete method" on page 124
OnRelidMoveComplete method	"OnRelidMoveComplete method" on page 127
RobotEndsUpHoldingPlate method	"RobotEndsUpHoldingPlate method" on page 130

<Go Back

OnDelidMoveComplete method

Description

VWorks software calls the OnDelidMoveComplete method to notify the plugin that a delid operation was executed. The plugin determines whether the lid was successfully removed and returns the results to VWorks software.

Syntax

```
HRESULT OnDelidMoveComplete(
   [in] BSTR LiddingXML,
   [out,retval] enum ReturnCode *retVal
);
```

Parameters

LiddingXML	[in] A Command XML block describing the Delid task that caused the delid operation to occur.
retVal	[out, retval] Returns an error code. Possible values: 0 = The delid operation was executed successfully (the request was completed) (RETURN_SUCCESS) 1 = Something was wrong with the input, so the request was not completed (RETURN_BAD_ARGS) 2 = The delid operation failed (the request was not completed) (RETURN_FAIL) For more information, see "ReturnCode enumerated type" on page 384.

OnDelidMoveComplete method input

VWorks software passes a Command XML block into the LiddingXML parameter of the OnDelidMoveComplete method.

Command XML block

The Command XML block for the OnDelidMoveComplete method contains the Command element and all its children. This XML block describes the Delid task that caused the delid operation to occur.

When the plugin receives the Command XML block, it only needs to check the Name attribute of the Location parameter. This attribute is designated by bold text in the following XML structure and input example.

Although VWorks software passes other XML metadata in the Command XML block, this information is of no interest to the plugin.

<Go Back

XML structure

The value of the `file` attribute for the `Velocity11` element is `MetaData`. See "Velocity11 element" on page 416.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Command>
      <Parameters>
         <Parameter ... Name='Plate' ...>
            <Ranges>
               <Range />
               ...
            </Ranges>
         </Parameter>
         <Parameter Name='Labware' ... />
         <Parameter Name='ApproachOffset' ... />
         <Parameter ... Name='Location' ... />
         <Parameter ... Name='Teachpoint' ... />
      </Parameters>
   </Command>
</Velocity11>
```

Parameter element (Location)

The `Location` parameter specifies the name of the location where the labware is to be delidded. If a lid is present at the specified location after the delid operation is executed, the plugin notifies VWorks software that the operation was successful. If a lid is not present, the plugin notifies VWorks software that the delid operation was not successful. This `Parameter` element has the following attributes plus the Scriptable, Style, and Type attributes:

Name	Description
`Description`	The value `Location to use.`
`Name`	The value `Location`.
`Value`	The name of the location.

<Go Back

OnDelidMoveComplete method input

The following code is a Command XML block received by the plugin from VWorks software as a string in the `LiddingXML` parameter of the `OnDelidMoveComplete` method. VWorks software notifies the plugin that the specified delid operation was executed. To determine whether the delid move was successful, the plugin looks for a lid at the location named `Stage`.

```
<?xml version="1.0" encoding="ASCII" ?>
<Velocity11 file="MetaData" md5sum="e23df8c398bc9c9634ce2fcb027c98a1" version="1.0">
   <Command Compiler="128" Editor="14" Name="Delid" NextTaskToExecute="1"
→PreferredTab="Plate Handling" RequiresRefresh="0" TaskRequiresLocation="1"
→VisibleAvailability="1">
      <Parameters>
         <Parameter Description="Delidding method for BenchCel" Hide_if="Constant(1)"
→Name="Delidding method" Scriptable="1" Style="0" Type="2"
→Value="Delid and retract">
            <Ranges>
               <Range Value="Delid and retract" />
               <Range Value="Delid to waste" />
            </Ranges>
         </Parameter>
         <Parameter Name="Plate" Scriptable="1" Style="0" Type="5"
→Value="process - 1 1" />
         <Parameter Name="Labware" Scriptable="1" Style="0" Type="10"
→Value="96 Costar 3894 PS Clr Rnd V Btm" />
         <Parameter Description="Location to use" Name="Location" Scriptable="1"
→Style="0" Type="6" Value="Stage" />
         <Parameter Description="Teachpoint to use" Name="Teachpoint" Scriptable="1"
→Style="0" Type="1" Value="robot teachpoint" />
      </Parameters>
   </Command>
</Velocity11>
```

Related information

For information about...	See...
LidIsRetained method	"LidIsRetained method" on page 123
OnRelidMoveComplete method	"OnRelidMoveComplete method" on page 127
RobotEndsUpHoldingPlate method	"RobotEndsUpHoldingPlate method" on page 130

<Go Back

OnRelidMoveComplete method

Description

VWorks software calls the `OnRelidMoveComplete` method to notify the plugin that a relid operation was executed. The plugin determines whether the lid was successfully replaced and returns the results to VWorks software.

Syntax

```
HRESULT OnRelidMoveComplete(
   [in] BSTR LiddingXML,
   [out,retval] enum ReturnCode *retVal
);
```

Parameters

`LiddingXML`	[in] A Command XML block describing the Relid task that caused the relid operation to occur.
`retVal`	[out, retval] Returns an error code. Possible values: `0` = The relid operation was executed successfully (the request was completed) (`RETURN_SUCCESS`) `1` = Something was wrong with the input, so the request was not completed (`RETURN_BAD_ARGS`) `2` = The relid operation failed (the request was not completed) (`RETURN_FAIL`) For more information, see "ReturnCode enumerated type" on page 384.

OnRelidMoveComplete method input

VWorks software passes a Command XML block into the `LiddingXML` parameter of the `OnRelidMoveComplete` method.

Command XML block (Relid task)

The Command XML block for the `OnRelidMoveComplete` method contains the `Command` element and all its children. This XML block describes the Relid task that caused the relid operation to occur.

When the plugin receives the Command XML block, it only needs to check the `Name` attribute of the `Location` parameter. This attribute is designated by bold text in the following XML structure and input example.

Although VWorks software passes other XML metadata in the Command XML block, this information is of no interest to the plugin.

<Go Back

XML structure

The value of the `file` attribute for the `Velocity11` element is `MetaData`. See "Velocity11 element" on page 416.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Command>
      <Parameters>
         <Parameter ... Name='Plate' ...>
            <Ranges>
               <Range ... />
               ...
            </Ranges>
         </Parameter>
         <Parameter Name='Labware' ... />
         <Parameter Name='ApproachOffset' ... />
         <Parameter ... Name='Location' ... />
         <Parameter ... Name='Teachpoint' ... />
      </Parameters>
   </Command>
</Velocity11>
```

Parameter element (Location)

The `Location` parameter specifies the name of the location where the labware is to be relidded. If a lid is not present at the specified location after the relid operation is executed, the plugin notifies VWorks software that the operation was successful. If a lid is present, the plugin notifies VWorks software that the relid operation was not successful. This `Parameter` element has the following attributes plus the Scriptable, Style, and Type attributes:

Name	Description
`Description`	The value `Location to use`.
`Name`	The value `Location`.
`Value`	The name of the location.

<Go Back

OnRelidMoveComplete method input

The following code is a Command XML block received by the plugin from VWorks software as a string in the `LiddingXML` parameter of the `OnRelidMoveComplete` method. To determine whether the relid operation was successful, the plugin looks for a lid at the location named `Stage`.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='41a32ad097a8b5cfac3642f875d72b97' version='1.0'>
   <Command Compiler='64' Editor='14' Name='Relid' NextTaskToExecute='1'
→PreferredTab='Plate Handling' RequiresRefresh='0' TaskRequiresLocation='1'
→VisibleAvailability='1'>
      <Parameters>
         <Parameter Name='Plate' Scriptable='1' Style='0' Type='5'
→Value='process - 1 1' />
         <Parameter Name='Labware' Scriptable='1' Style='0' Type='10'
→Value='96 Costar 3894 PS Clr Rnd V Btm' />
         <Parameter Description='Location to use' Name='Location' Scriptable='1'
→Style='0' Type='6' Value='Stage' />
         <Parameter Description='Teachpoint to use' Name='Teachpoint' Scriptable='1'
→Style='0' Type='1' Value='robot teachpoint' />
      </Parameters>
   </Command>
</Velocity11>
```

Related information

For information about...	See...
LidIsRetained method	“LidIsRetained method” on page 123
OnDelidMoveComplete method	“OnDelidMoveComplete method” on page 124
RobotEndsUpHoldingPlate method	“RobotEndsUpHoldingPlate method” on page 130

<Go Back

RobotEndsUpHoldingPlate method

Description

VWorks software calls the `RobotEndsUpHoldingPlate` method to ask the plugin if the robot is holding the labware at the end of the delidding/relidding process.

For the Vacuum Delid Station and the Lid Hotel Station, the robot ends up holding the labware.

Syntax

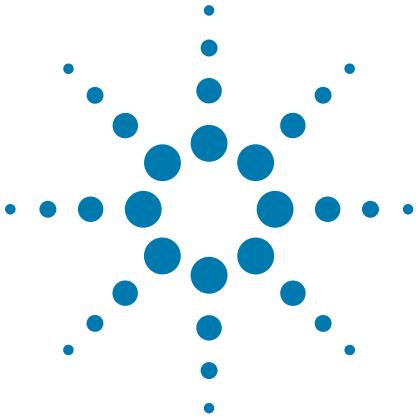
```
HRESULT RobotEndsUpHoldingPlate(
   [out,retval] long *retVal)
);
```

Parameters

`retVal`	[out, retval] Indicates whether the robot is holding the labware at the end of the delidding/relidding process. Possible values: `0` = The robot is not holding the labware `1` = The robot is holding the labware

Related information

For information about...	See...
LidIsRetained method	"LidIsRetained method" on page 123
OnDelidMoveComplete method	"OnDelidMoveComplete method" on page 124
OnRelidMoveComplete method	"OnRelidMoveComplete method" on page 127



10 IMeasurementDriver interface

VWorks plugins that return measurement values and monitor device measurements must implement the IMeasurementDriver interface.

This chapter defines the IMeasurementDriver methods.

IMPORTANT All VWorks device driver plugins must implement the IWorksDriver, IControllerClient, and IWorksDiags interfaces.

This chapter contains the following topics:

- "IMeasurementDriver methods overview" on page 132
- "GetMeasurement method" on page 133
- "GetMeasurementTypes method" on page 134

<Go Back

IMeasurementDriver methods overview

Use the following table to quickly locate an IMeasurementDriver method by name, by description, or by page number.

Method	Description	See...
`GetMeasurement`	Gets the value for the specified measurement type from the plugin.	"GetMeasurement method" on page 133
`GetMeasurementTypes`	Gets all available measurement types from the plugin.	"GetMeasurementTypes method" on page 134

GetMeasurement method

Description

VWorks software calls the `GetMeasurement` method to get the value for the specified measurement type from the plugin.

Syntax

```
HRESULT GetMeasurement(
   [in] BSTR xmlParams,
   [out] DOUBLE *measurement,
   [out,retval] enum ReturnCode *retVal
);
```

Parameters

`xmlParams`	[in] A string that contains the measurement type, such as battery level, power load, and internal temperature. *Note:* Although the name of this parameter implies that it contains XML, the parameter actually contains a string value.
`measurement`	[out] The value of the specified measurement type.
`retVal`	[out, retval] Returns an error code. Possible values: `0` = The request was completed (`RETURN_SUCCESS`) `1` = Something was wrong with the input, so the request was not completed (`RETURN_BAD_ARGS`) `2` = The request was not completed (`RETURN_FAIL`) For more information, see "ReturnCode enumerated type" on page 384.

Related information

For information about...	See...
GetMeasurementTypes method	"GetMeasurementTypes method" on page 134

<Go Back

GetMeasurementTypes method

Description

VWorks software calls the `GetMeasurementTypes` method to get all available measurement types from the plugin, such as battery level, power load, and internal temperature.

Syntax

```
HRESULT GetMeasurementTypes(
   [out] BSTR *xmlTypes,
   [out,retval] enum ReturnCode *retcode
);
```

Parameters

`xmlTypes`	[out] A MeasurementTypes XML block that defines all available measurement types.
`retcode`	[out, retval] Returns an error code. Possible values: `0` = The request was completed (`RETURN_SUCCESS`) `1` = Something was wrong with the input, so the request was not completed (`RETURN_BAD_ARGS`) `2` = The request was not completed (`RETURN_FAIL`) For more information, see "ReturnCode enumerated type" on page 384.

GetMeasurementTypes method output

The plugin returns a MeasurementTypes XML block in the `xmlTypes` parameter of the `GetMeasurementTypes` method.

MeasurementTypes XML block

The MeasurementTypes XML block contains the `MeasurementTypes` element and all its children. This XML block defines the available measurement types, including settings that are specified on the Measurement Manager tab.

<Go Back

XML structure

The value of the `file` attribute for the `Velocity11` element is `Measurement`. See "Velocity11 element" on page 416.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <MeasurementTypes>
      <MeasurementType />
      ...
   </MeasurementTypes>
</Velocity11>
```

MeasurementTypes element

The `MeasurementTypes` element contains one or more `MeasurementType` elements.

MeasurementType element

The `MeasurementType` element has the following attributes:

Name	Value
`CriticalTime`	The time, in seconds, that the measurement is allowed to be above the upper limit or below the lower limit before it is considered to be out of range. Required: No Default value: `0`
`LogAction`	Specifies when to log the measurement value. Possible values: `-1` = Always log the measurement value based on the poll frequency (`LOG_ACTION_ALWAYS`) `0` = Do not log any measurement values (`LOG_ACTION_NONE`) `1` = Log the measurement value only when the value is less than the lower limit and the duration is longer than the critical time (`LOG_ACTION_LOW`) `2` = Log the measurement value only when the value exceeds the upper limit and the duration is longer than the critical time (`LOG_ACTION_HIGH`) `3` = Log the measurement value when the value is less than the lower limit or exceeds the upper limit and when the duration is longer than the critical time (`LOG_ACTION_HIGHLOW`) Required: No Default value: `0`
`LowerLimit`	The lower limit for the measurement type. Required: No Default value: `0.0`
`MeasurementName`	The name of the measurement type. Required: No Default value: No value

<Go Back

Name	Value
PauseAction	Specifies when to pause the scheduler. Possible values: 0 = Do not pause the scheduler for any measurement values (PAUSE_ACTION_NONE) 1 = Pause the scheduler for the measurement value only when the value is less than the lower limit and the duration is longer than the critical time (PAUSE_ACTION_LOW) 2 = Pause the scheduler for the measurement value only when the value exceeds the upper limit and the duration is longer than the critical time (PAUSE_ACTION_HIGH) 3 = Pause the scheduler for the measurement value when the value is less than the lower limit or exceeds the upper limit and when the duration is longer than the critical time (PAUSE_ACTION_HIGHLOW) Required: No Default value: 0
PollFrequency	The frequency at which VWorks software requests measurement values, in seconds. Required: No Default value: 0
Unit	The unit of measurement for the measurement type. Required: No Default value: No value
UpperLimit	The upper limit for the measurement type. Required: No Default value: 0.0

Example of GetMeasurementTypes method output

The following sample code is a MeasurementTypes XML block that is returned to VWorks software by the plugin as a string in the xmlTypes parameter of the GetMeasurementTypes method. The plugin returns the measurement type named Battery level to VWorks software.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Measurement' md5sum='d597fb30274a5effc9c50473338e7522'
→version='1.1' >
   <MeasurementTypes >
      <MeasurementType CriticalTime='5' LogAction='1' LowerLimit='80'
→MeasurementName='Battery level' PauseAction='3' PollFrequency='10' Unit='%'
→UpperLimit='100' />
   </MeasurementTypes>
</Velocity11>
```

<Go Back

Related information

For information about...	See...
GetMeasurement method	“GetMeasurement method” on page 133

<Go Back

11 IPipetteDriver interface

This interface is reserved for internal use. VWorks plugins should not implement the IPipetteDriver interface.

<Go Back

12 IRobotDriver interface

VWorks plugins that use robots to move labware must implement the IRobotDriver interface.

This chapter defines the IRobotDriver methods.

IMPORTANT All VWorks device driver plugins must implement the IWorksDriver, IControllerClient, and IWorksDiags interfaces.

This chapter contains the following topics:

- "CheckPlatePresent method" on page 143
- "DelidRelid method" on page 146
- "GetPlatePresentResult method" on page 150
- "GetSimulationTimes method" on page 153
- "GetTeachPoints method" on page 156
- "Move method" on page 159
- "SetSpeed method" on page 162

<Go Back

IRobotDriver methods overview

Use the following table to quickly locate an IRobotDriver method by name, by description, or by page number.

Method	Description	See...
`CheckBarCode`	*Obsolete*. This method should be implemented as `return E_NOTIMPL (0x80004001)`.	
`CheckPlatePresent`	Asks the plugin if the robot labware sensor can detect a labware using the specified robot gripper offset at the specified location.	"CheckPlatePresent method" on page 143
`DelidRelid`	Tells the plugin to command the robot to delid or relid the specified labware at the specified location.	"DelidRelid method" on page 146
`GetPlatePresentResult`	Gets the results of the last call to the `CheckPlatePresent` method from the plugin.	"GetPlatePresentResult method" on page 150
`GetSimulationTimes`	Gets the average simulation robot movement times for the Slow, Medium, and Fast speed settings from the plugin.	"GetSimulationTimes method" on page 153
`GetSpeed`	*Obsolete*. This method should be implemented as `return E_NOTIMPL (0x80004001)`.	
`GetTeachPoints`	Gets the names of all of a robot's teachpoints from the plugin.	"GetTeachPoints method" on page 156
`Move`	Tells the plugin to command the robot to move to a labware from a pick location to a place location.	"Move method" on page 159
`SetSpeed`	Tells the plugin to set the specified robot-speed settings to their new values.	"SetSpeed method" on page 162

<Go Back

CheckPlatePresent method

Description

VWorks software calls the `CheckPlatePresent` method to ask the plugin if the robot labware sensor can detect a labware using the specified robot gripper offset at the specified location.

Syntax

```
HRESULT CheckPlatePresent(
   [in] BSTR XML,
   [out] BSTR *returnedXML,
   [out,retval] enum ReturnCode *retVal
);
```

Parameters

`XML`	[in] An IRobotDriver_CheckPlatePresent_Input XML element that contains the robot gripper offset and the location to check.
`returnedXML`	[out] An IRobot_CheckPlatePresent_Output XML element that indicates whether a labware is present.
`retVal`	[out, retval] Returns an error code. Possible values: `0` = The request was completed (`RETURN_SUCCESS`) `1` = Something was wrong with the input, so the request was not completed (`RETURN_BAD_ARGS`) `2` = The request was not completed (`RETURN_FAIL`) For more information, see "ReturnCode enumerated type" on page 384.

CheckPlatePresent method input

VWorks software passes an IRobotDriver_CheckPlatePresent_Input XML element into the `XML` parameter of the `CheckPlatePresent` method. This XML element provides the robot gripper offset and the location to check.

IRobotDriver_CheckPlatePresent_Input element

The `IRobotDriver_CheckPlatePresent_Input` element has the following attributes:

Name	Value
`OffsetHeight`	The robot gripper offset, in millimeters.
`Location`	The name of the location to check.

<Go Back

Example of CheckPlatePresent method input

The following sample code is an IRobotDriver_CheckPlatePresent_Input XML element that is received by the plugin from VWorks software as a string in the `XML` parameter of the `CheckPlatePresent` method. VWorks software tells the plugin to determine if the robot labware sensor can detect a labware using the gripper offset height of `0` mm at the location named `Location1`.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='0e237893b059fefa2b1edfdfad089f74' version='1.0' >
   <IRobotDriver_CheckPlatePresent_Input Location='Location1' OffsetHeight='0' />
</Velocity11>
```

CheckPlatePresent method output

The plugin returns an IRobotDriver_CheckPlatePresent_Output XML element in the `returnedXML` parameter of the `CheckPlatePresent` method. This XML element indicates whether a labware is present at the specified location using the specified gripper offset. A labware is present if the robot's labware sensor can detect it.

IRobot_CheckPlatePresent_Output element

The `IRobotDriver_CheckPlatePresent_Output` element has the following attribute:

Name	Value
`Present`	The value of this attribute indicates whether the labware is present at the specified gripper offset. Possible values: `0` = The labware is not present `1` = The labware is present Required: No Default value: `0`

Example of CheckPlatePresent method output

The following sample code is an IRobotDriver_CheckPlatePresent_Output XML element that is returned by the plugin to VWorks software as a string in the `returnedXML` parameter of the `CheckPlatePresent` method. The plugin tells VWorks software that a labware is not present at the specified location, that is, the robot cannot detect a labware.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='c31697b858117fcc668453c48526698b' version='1.0' >
   <IRobotDriver_CheckPlatePresent_Output Present='0' />
</Velocity11>
```

Related information

For information about...	See...
DelidRelid method	"DelidRelid method" on page 146

<Go Back

For information about...	See...
GetPlatePresentResult method	“GetPlatePresentResult method” on page 150
GetSimulationTimes method	“GetSimulationTimes method” on page 153
GetTeachPoints method	“GetTeachPoints method” on page 156
Move method	“Move method” on page 159
ReturnCode enumerated type	“ReturnCode enumerated type” on page 384
SetSpeed method	“SetSpeed method” on page 162

<Go Back

DelidRelid method

Description

VWorks software calls the DelidRelid method to tell the plugin to command the robot to delid or relid the specified labware at the specified location.

Syntax

```
HRESULT DelidRelid(
   [in] BSTR XML,
   [out] BSTR *returnedXML,
   [out,retval] enum ReturnCode *retVal
);
```

Parameters

XML	[in] An IRobotDriver_DelidRelid_Input XML element that contains information required by the robot to perform a Delid or Relid task.
returnedXML	[out] An empty IRobotDriver_DelidRelid_Output XML element.
retVal	[out, retval] Returns an error code. Possible values: 0 = The request was completed (RETURN_SUCCESS) 1 = Something was wrong with the input, so the request was not completed (RETURN_BAD_ARGS) 2 = The request was not completed (RETURN_FAIL) For more information, see "ReturnCode enumerated type" on page 384.

DelidRelid method input

VWorks software passes an IRobotDriver_DelidRelid_Input XML element into the XML parameter of the DelidRelid method. This element provides information required by the robot to perform a Delid or Relid task.

<Go Back

IRobotDriver_DelidRelid_Input element

The `IRobotDriver_DelidRelid_Input` element has the following attributes:

Name	Value
`IntendedDropoffLocationOnNextMove`	The name of the place location. The plugin should use this information to determine the required gripper offset for the place location. The `DelidRelid` method does not actually move the labware to the place location. First the robot should pick up the labware, delid or relid it, and move it to a safe location. Then VWorks software calls the `Move` method to tell the plugin to move the labware from the safe location to the place location. See "Move method" on page 159.
`IsRelid`	Indicates whether the task is Relid or Delid. Possible values: `0` = The task is Delid `1` = The task is Relid
`Labware`	The labware name.
`LiddingLocation`	The name of the teachpoint where the labware is to be relidded or delidded.
`PickupDeviceName`	The name of the pick-location device.
`PickupLocation`	The name of the teachpoint that is set at the pick location.
`PickupLocationName`	The name of the pick location.
`PickupLocationOffset`	The pick-location device gripper offset, in millimeters.
`RiseHeight`	If the labware has a lid, and the lid is retained, the value of this parameter is the lid departure height, in millimeters. For delidding on devices that retain the lid (that is, where the value of `Type` is `LID_HOTEL`), the robot should move the labware downward by the `RiseHeight` before retracting with the delidded labware. For relidding, the robot should lift the labware by the `RiseHeight` before retracting with the lidded labware. If the labware has a lid, but the lid is not retained, the value is the lidded thickness. For delidding on devices that do not retain the lid (that is, where `Type` is `VACUUM_DELIDDER`), the robot should lift the labware by the `RiseHeight` and then move the labware downward by the `RiseHeight` before retracting with the delidded labware. If the labware does not have a lid, the value of the `RiseHeight` parameter is always `0.0`.

<Go Back

Name	Value
Type	Represents the type of lidding device. Possible values: 0 = The lidding device retains the lid (VACUUM_DELIDDER) 1= The lidding device does not retain the lid (LID_HOTEL) See ILiddingDriver "LidIsRetained method" on page 123.
PickupLeanLocation	The name of the teachpoint that is set at the second pick location. A second teachpoint is specified if the labware is to be picked up from a Plate Hotel or other device that holds labware vertically. This second teachpoint is used to determine if any correction is required in the x-axis or y-axis to account for lean in the stack. Otherwise, this attribute has no value.

Example of DelidRelid method input

The following sample code contains an IRobotDriver_DelidRelid_Input XML element that is received by the plugin from VWorks software as a string in the XML parameter of the DelidRelid method. VWorks software tells the plugin to command the robot to delid the labware named lid at the lidding location named Robot Teachpoint - 2.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='7ea07f83fa28f2eb067dd18ae21dd186' version='1.0' >
   <IRobotDriver_DelidRelid_Input IntendedDropoffLocationOnNextMove=
→'Robot Teachpoint - 3' IsRelid='0' Labware='lid' LiddingLocation=
→'Robot Teachpoint - 2' PickupDeviceName='Lid Hotel Station - 1'
→PickupLocation='Robot Teachpoint - 1' PickupLocationName='Upper plate pad'
→PickupLocationOffset='0' RiseHeight='0' Type='1' PickupLeanLocation=
→'Plate Hotel Top' />
</Velocity11>
```

DelidRelid method output

The plugin returns an empty IRobotDriver_DelidRelid_Output XML element in the returnedXML parameter of the DelidRelid method.

Example of DelidRelid method output

The following sample code is an empty IRobotDriver_DelidRelid_Output XML element that is returned by the plugin to VWorks software as a string in the returnedXML parameter of the DelidRelid method.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='cb72e8dc081b48394cc654a62cfe334a' version='1.0' >
   <IRobotDriver_DelidRelid_Output />
</Velocity11>
```

<Go Back

Related information

For information about...	See...
CheckPlatePresent method	“CheckPlatePresent method” on page 143
GetPlatePresentResult method	“GetPlatePresentResult method” on page 150
GetSimulationTimes method	“GetSimulationTimes method” on page 153
GetTeachPoints method	“GetTeachPoints method” on page 156
Move method	“Move method” on page 159
ReturnCode enumerated type	“ReturnCode enumerated type” on page 384
SetSpeed method	“SetSpeed method” on page 162

<Go Back

GetPlatePresentResult method

Description

VWorks software calls the GetPlatePresentResult method to get the result of the last call to the CheckPlatePresent method from the plugin. See "CheckPlatePresent method" on page 143.

Syntax

```
HRESULT GetPlatePresentResult(
   [in] BSTR XML,
   [out] BSTR *returnedXML,
   [out,retval] enum ReturnCode *retVal
);
```

Parameters

XML	[in] An empty IRobotDriver_GetPlatePresentResult_Input XML element.
returnedXML	[out] An IRobotDriver_GetPlatePresentResult_Output XML element that provides the result of the last call to the CheckPlatePresent method.
retVal	[out, retval] Returns an error code. Possible values: 0 = The request was completed (RETURN_SUCCESS) 1 = Something was wrong with the input, so the request was not completed (RETURN_BAD_ARGS) 2 = The request was not completed (RETURN_FAIL) For more information, see "ReturnCode enumerated type" on page 384.

GetPlatePresentResult method input

VWorks software passes an empty IRobotDriver_GetPlatePresentResult_Input XML element into the XML parameter of the GetPlatePresentResult method.

<Go Back

Example of GetPlatePresentResult method input

The following sample code is an empty IRobotDriver_GetPlatePresentResult_Input XML element that is received by the plugin from VWorks software as a string in the `XML` parameter of the `GetPlatePresentResult` method. VWorks software asks the plugin for the result of the last call to the `CheckPlatePresent` method.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='0e237893b059fefa2b1edfdfad089f74' version='1.0' >
   <IRobotDriver_GetPlatePresentResult_Input />
</Velocity11>
```

GetPlatePresentResult method output

The plugin returns an IRobotDriver_GetPlatePresentResult_Output XML element in the `returnedXML` parameter of the `GetPlatePresentResult` method. This XML element provides the result of the last call to the `CheckPlatePresent` method. (See "CheckPlatePresent method" on page 143.)

IRobotDriver_GetPlatePresentResult_Output element

The `IRobotDriver_GetPlatePresentResult_Output` element has the following attribute:

Name	Value
`Present`	Represents the result of the last call to the `CheckPlatePresent` method. Possible values: `0` = The labware is not present `1` = The labware is present Required: No Default value: `0`

Example of GetPlatePresentResult method output

The following sample code is an IRobotDriver_GetPlatePresentResult_Output XML element that is returned by the plugin to VWorks software as a string in the `returnedXML` parameter of the `GetPlatePresentResult` method. The plugin tells VWorks software that the result of the last `CheckPlatePresent` method call was `0` (The labware is not present).

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='c31697b858117fcc668453c48526698b' version='1.0' >
   <IRobotDriver_GetPlatePresentResult_Output Present='0' />
</Velocity11>
```

Related information

For information about...	See...
CheckPlatePresent method	"CheckPlatePresent method" on page 143

<Go Back

For information about...	See...
DelidRelid method	“DelidRelid method” on page 146
GetSimulationTimes method	“GetSimulationTimes method” on page 153
GetTeachPoints method	“GetTeachPoints method” on page 156
Move method	“Move method” on page 159
ReturnCode enumerated type	“ReturnCode enumerated type” on page 384
SetSpeed method	“SetSpeed method” on page 162

<Go Back

GetSimulationTimes method

Description

VWorks software calls the GetSimulationTimes method to get the average simulation robot movement times for the Slow, Medium, and Fast speed settings from the plugin.

Syntax

```
HRESULT GetSimulationTimes(
   [in] BSTR XML,
   [out] BSTR *returnedXML,
   [out,retval] enum ReturnCode *retVal
);
```

Parameters

XML	[in] An empty IRobotDriver_GetSimulationTimes_Input XML element.
returnedXML	[out] An IRobotDriver_GetSimulationTimes_Output XML element that contains the average simulation robot movement times for the Slow, Medium, and Fast speed settings.
retVal	[out, retval] Returns an error code. Possible values: 0 = The request was completed (RETURN_SUCCESS) 1 = Something was wrong with the input, so the request was not completed (RETURN_BAD_ARGS) 2 = The request was not completed (RETURN_FAIL) For more information, see "ReturnCode enumerated type" on page 384.

GetSimulationTimes method input

VWorks software passes an empty IRobotDriver_GetSimulationTimes_Input XML element into the XML parameter of the GetSimulationTimes method.

Example of GetSimulationTimes method input

The following sample code contains an empty IRobotDriver_GetSimulationTimes_Input XML element that is received by the plugin from VWorks software as a string in the XML parameter of the

<Go Back

GetSimulationTimes method. VWorks software asks the plugin for the average simulation robot speeds for the 3-Axis Robot's Slow, Medium, and Fast settings.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='8ea81b7f4630630c501aa89f152db68c' version='1.0' >
   <IRobotDriver_GetSimulationTimes_Input />
</Velocity11>
```

IRobotDriver_GetSimulationTimes_Output XML element

The plugin returns an IRobotDriver_GetSimulationTimes_Output XML element in the returnedXML parameter of the GetSimulationTimes method. This XML element provides the average simulation robot movement times for the Slow, Medium, and Fast speed settings.

IRobotDriver_GetSimulationTimes_Output element

The IRobotDriver_GetSimulationTimes_Output element has the following attributes:

Name	Value
AverageFastMoveTime	The average simulation robot movement time for the Fast speed setting, in seconds. Required: No Default value: 0
AverageMediumMoveTime	The average simulation robot movement time for the Medium speed setting, in seconds. Required: No Default value: 0
AverageSlowMoveTime	The average simulation robot movement time for the Slow speed setting, in seconds. Required: No Default value: 0

Example of GetSimulationTimes method output

The following sample code contains an IRobotDriver_GetSimulationTimes_Output XML element that is returned to VWorks software by the plugin as a string in the returnedXML parameter of the GetSimulationTimes method. The plugin returns the average simulation speeds for the 3-Axis Robot to VWorks software.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='5d148b5a5eec9af40e5b4df77c90988d' version='1.0' >
   <IRobotDriver_GetSimulationTimes_Output AverageFastMoveTime='4'
→AverageMediumMoveTime='6' AverageSlowMoveTime='8' />
</Velocity11>
```

<Go Back

Related information

For information about...	See...
CheckPlatePresent method	“CheckPlatePresent method” on page 143
DelidRelid method	“DelidRelid method” on page 146
GetPlatePresentResult method	“GetPlatePresentResult method” on page 150
GetTeachPoints method	“GetTeachPoints method” on page 156
Move method	“Move method” on page 159
ReturnCode enumerated type	“ReturnCode enumerated type” on page 384
SetSpeed method	“SetSpeed method” on page 162

<Go Back

GetTeachPoints method

Description

VWorks software calls the `GetTeachPoints` method to get the names of all of the robot's teachpoints from the plugin.

Syntax

```
HRESULT GetTeachpoints(
   [in] BSTR XML,
   [out] BSTR *returnedXML,
   [out,retval] enum ReturnCode *retVal
);
```

Parameters

`XML`	[in] A string that contains an empty IRobotDriver_GetTeachpoints_Input XML element.
`returnedXML`	[out] An IRobotDriver_GetTeachpoints_Output XML block that contains a list of teachpoint names.
`retVal`	[out, retval] Returns an error code. Possible values: `0` = The request was completed (`RETURN_SUCCESS`) `1` = Something was wrong with the input, so the request was not completed (`RETURN_BAD_ARGS`) `2` = The request was not completed (`RETURN_FAIL`) For more information, see "ReturnCode enumerated type" on page 384.

GetTeachPoints method input

VWorks software passes an empty IRobotDriver_GetTeachpoints_Input XML element into the `XML` parameter of the `GetTeachPoints` method.

Example of GetTeachPoints method input

The following sample code is an empty IRobotDriver_GetTeachpoints_Input XML element that is received by the plugin from VWorks software as a string in the `XML` parameter of the `GetTeachPoints` method. VWorks software asks the plugin for the names of all the teachpoints for the 3-Axis Robot.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='1741399628c5cfaebce951e6e26821b3' version='1.0' >
   <IRobotDriver_GetTeachpoints_Input />
</Velocity11>
```

<Go Back

GetTeachPoints method output

The plugin returns an IRobotDriver_GetTeachpoints_Output XML block in the `returnedXML` parameter of the `GetTeachPoints` method.

IRobotDriver_GetTeachpoints_Output XML block

The IRobotDriver_GetTeachpoints_Output XML block contains the `IRobotDriver_GetTeachpoints_Output` element and all its children. This XML block provides a list of all of the robot's teachpoints.

XML structure

The value of the `file` attribute for the `Velocity11` element is `MetaData`. See "Velocity11 element" on page 416.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <IRobotDriver_GetTeachpoints_Output>
      <Teachpoints >
         <Value />
         ...
      </Teachpoints>
   </IRobotDriver_GetTeachpoints_Output>
</Velocity11>
```

IRobotDriver_GetTeachpoints_Output XML block

The `IRobotDriver_GetTeachpoints_Output` element contains one `Teachpoints` element.

Teachpoints element

The `Teachpoints` element contains one or more `Value` elements.

Value element

Each `Value` element contains the name of a teachpoint. This element has the following attribute:

Name	Value
`Value`	The name of the teachpoint. Required: Yes

Example of GetTeachPoints method output

The following sample code is an IRobotDriver_GetTeachpoints_Output XML block that is returned by the plugin to VWorks software as a string in the `returnedXML` parameter of the `GetTeachPoints` method. The plugin returns all the names of the 3-Axis Robot's teachpoints to VWorks software.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='c806278dab427bf154e0d2bf56f4aa18' version='1.0' >
   <IRobotDriver_GetTeachpoints_Output >
      <Teachpoints >
         <Value Value='Robot Teachpoint 1' />
         <Value Value='Robot Teachpoint 2' />
         <Value Value='Robot Teachpoint 3' />
         <Value Value='Robot Teachpoint 4' />
      </Teachpoints>
   </IRobotDriver_GetTeachpoints_Output>
</Velocity11>
```

<Go Back

Related information

For information about...	See...
CheckPlatePresent method	“CheckPlatePresent method” on page 143
DelidRelid method	“DelidRelid method” on page 146
GetPlatePresentResult method	“GetPlatePresentResult method” on page 150
GetSimulationTimes method	“GetSimulationTimes method” on page 153
Move method	“Move method” on page 159
ReturnCode enumerated type	“ReturnCode enumerated type” on page 384
SetSpeed method	“SetSpeed method” on page 162

<Go Back

Move method

Description

VWorks software calls the `Move` method to tell the plugin to command the robot to move a labware from a pick location to a place location.

Syntax

```
HRESULT Move(
   [in] BSTR XML,
   [out] BSTR *returnedXML,
   [out,retval] enum ReturnCode *retVal
);
```

Parameters

`XML`	[in] An IRobotDriver_Move_Input XML element that provides information about the pick/place action.
`returnedXML`	[out] An empty IRobotDriver_Move_Output XML element.
`retVal`	[out, retval] Returns an error code. Possible values: `0` = The request was completed (`RETURN_SUCCESS`) `1` = Something was wrong with the input, so the request was not completed (`RETURN_BAD_ARGS`) `2` = The request was not completed (`RETURN_FAIL`) For more information, see "ReturnCode enumerated type" on page 384.

Move method input

VWorks software passes a IRobotDriver_Move_Input XML element into the `XML` parameter of the `Move` method. This XML element provides information about the pick/place action.

IRobotDriver_Move_Input element

The `IRobotDriver_Move_Input` element has the following attributes:

Name	Value
`DropoffLocation`	The name of the teachpoint that is set at the place location.
`DropoffLocationOffset`	The place-location device gripper offset, in millimeters.
`Labware`	The labware name.

<Go Back

Name	Value
PayloadThickness	The thickness, in millimeters, of all the labware on top of the labware that is to be moved. If the mounted labware has lids, the thickness of the lids is also included. If no labware is mounted on the labware to be moved, the value of PayloadThickness is 0.
PickupDeviceName	The pick-location device name.
PickupLocation	The name of the teachpoint that is set at the pick location.
PickupLocationName	The name of the pick location.
PickupLocationOffset	The pick-location device gripper offset, in millimeters.
PlateHasLid	Indicates whether the labware has a lid. Possible values: 0 = The labware does not have a lid 1 = The labware has a lid
PlateSealed	Indicates whether the labware is sealed. Possible values: 0 = The labware is not sealed 1 = The labware is sealed
PickupLeanLocation	The name of the teachpoint that is set at the second pick location. A second teachpoint is specified if the labware is to be picked up from a Plate Hotel or other device that holds labware vertically. This second teachpoint is used to determine if any correction is required in the x-axis or y-axis to account for lean in the stack. Otherwise, this attribute has no value.

Example of Move method input

The following sample code contains an IRobotDriver_Move_Input XML element that is received by the plugin from VWorks software as a string in the XML parameter of the Move method. VWorks software tells the plugin to command the robot to move the labware named 384-well plate from the pick location named Stage to the place location named Dropoff Teachpoint.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='931011d35de982a068bfe3313c3754f6' version='1.0' >
   <IRobotDriver_Move_Input DropoffLocation='Dropoff Teachpoint'
DropoffLocationOffset='0' Labware='384-well plate' PayloadThickness='0'
→PickupDeviceName='PlatePad - 1' PickupLocation='PlatePad - 1 Teachpoint'
→PickupLocationName='Stage' PickupLocationOffset='0' PlateHasLid='0'
→PlateSealed='0' />
</Velocity11>
```

<Go Back

Move method output

The plugin returns an empty IRobotDriver_Move_Output XML element into the `returnedXML` parameter of the `Move` method.

Example of Move method output

The following sample code is an empty IRobotDriver_Move_Output XML element that is returned by the plugin to VWorks software as a string in the `returnedXML` parameter of the `Move` method.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='0090cbcd92e9e12d7dc90765978a7a0d' version='1.0' >
   <IRobotDriver_Move_Output />
</Velocity11>
```

Related information

For information about...	See...
CheckPlatePresent method	"CheckPlatePresent method" on page 143
DelidRelid method	"DelidRelid method" on page 146
GetPlatePresentResult method	"GetPlatePresentResult method" on page 150
GetSimulationTimes method	"GetSimulationTimes method" on page 153
GetTeachPoints method	"GetTeachPoints method" on page 156
ReturnCode enumerated type	"ReturnCode enumerated type" on page 384
SetSpeed method	"SetSpeed method" on page 162

<Go Back

SetSpeed method

Description

VWorks software calls the `SetSpeed` method to tell the plugin to set the specified robot-speed setting to a new value.

Syntax

```
HRESULT SetSpeed(
   [in] BSTR XML,
   [out] BSTR *returnedXML,
   [out,retval] enum ReturnCode *retVal
);
```

Parameters

`XML`	[in] An IRobotDriver_SetSpeed_Input XML element that contains the name of the robot-speed settings and their new values.
`returnedXML`	[out] An empty IRobotDriver_SetSpeed_Output XML element.
`retVal`	[out, retval] Returns an error code. Possible values: `0` = The request was completed (`RETURN_SUCCESS`) `1` = Something was wrong with the input, so the request was not completed (`RETURN_BAD_ARGS`) `2` = The request was not completed (`RETURN_FAIL`) For more information, see "ReturnCode enumerated type" on page 384.

SetSpeed method input

VWorks software passes an IRobotDriver_SetSpeed_Input XML element into the `XML` parameter of the `SetSpeed` method. This XML element provides name of the robot-speed settings and their new values.

<Go Back

IRobotDriver_SetSpeed_Input element

The `IRobotDriver_SetSpeed_Input` element has the following attributes:

Name	Value
`EmptyPayloadSpeedType`	The speed at which the robot can run when the gripper is not carrying any labware. The labware speed settings are ignored. This attribute is used when **Always run at "robot speed" when gripper is empty** is selected in the Options dialog box. Possible values: `0` = Slow `1` = Medium `2` = Fast
`Type`	Represents the robot-speed setting. Possible values: `0` = Slow `1` = Medium `2` = Fast

Example of SetSpeed method input

The following sample code contains an IRobotDriver_SetSpeed_Input XML element that is received by the plugin from VWorks software as a string in the XML parameter of the `SetSpeed` method. VWorks software tells the plugin to set the Fast robot-speed setting to the new value of `2`.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='79363f4956846adf751f115dbe5e52b2' version='1.0' >
   <IRobotDriver_SetSpeed_Input EmptyPayloadSpeedType='2' Type='2' />
</Velocity11>
```

SetSpeed method output

The plugin returns an empty IRobotDriver_SetSpeed_Output XML element in the `returnedXML` parameter of the `SetSpeed` method.

Example of SetSpeed method output

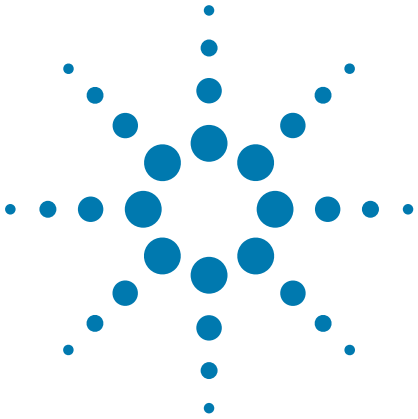
The following sample code is an empty IRobotDriver_SetSpeed_Output XML element that is returned by the plugin to VWorks software as a string in the `returnedXML` parameter of the `SetSpeed` method.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='6c51256d736d242e7f2388a2ff95086f' version='1.0' >
   <IRobotDriver_SetSpeed_Output />
</Velocity11>
```

<Go Back

Related information

For information about...	See...
CheckPlatePresent method	“CheckPlatePresent method” on page 143
DelidRelid method	“DelidRelid method” on page 146
GetPlatePresentResult method	“GetPlatePresentResult method” on page 150
GetSimulationTimes method	“GetSimulationTimes method” on page 153
GetTeachPoints method	“GetTeachPoints method” on page 156
Move method	“Move method” on page 159
ReturnCode enumerated type	“ReturnCode enumerated type” on page 384



13 ISpinDriver interface

VWorks plugins that perform Centrifuge tasks must implement the ISpinDriver interface.

This chapter defines the ISpinDriver method.

IMPORTANT All VWorks device driver plugins must implement the IWorksDriver, IControllerClient, and IWorksDiags interfaces.

This chapter contains the following topic:

• "SpinCycle method" on page 166

<Go Back

SpinCycle method

Description

VWorks software calls the `SpinCycle` method to tell the plugin to spin the loaded labware using the specified motion and time parameters.

Syntax

```
HRESULT SpinCycle(
   [in] DOUBLE vel_percent,
   [in] DOUBLE accel_percent,
   [in] DOUBLE decel_percent,
   [in] enum TIMER_MODES timer_mode,
   [in] LONG time_in_secs,
   [in] BSTR location_name,
   [out,retval] enum ReturnCode *retVal
);
```

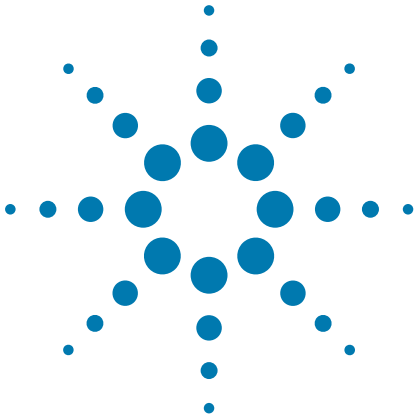
Parameters

`vel_percent`	[in] The rotor velocity, as a percent of the maximum rotor velocity.
`accel_percent`	[in] The acceleration of the centrifuge, as a percent of maximum acceleration.
`decel_percent`	[in] The deceleration of the centrifuge, as a percent of maximum deceleration (braking).
`timer_mode`	[in] Specifies how to implement the spin time. Possible values: `0` = Set the next spin session to last for the specified duration, including the time to accelerate and decelerate (`TIMER_MODE_TOTAL_TIME`) `1` = Set the next spin session to last for the specified duration, excluding the time to accelerate and decelerate (`TIMER_MODE_TIME_AT_SPEED`) `2` = This value is not currently used (`TIMER_MODE_CONTINUOUS_SPIN`)
`time_in_secs`	[in] The length of time to spin the labware in the desired timer mode, in seconds.
`location_name`	[in] The name of the device location to use.

<Go Back

retVal	[out, retval] Returns an error code. Possible values: 0 = The request was completed (RETURN_SUCCESS) 1 = Something was wrong with the input, so the request was not completed (RETURN_BAD_ARGS) 2 = The request was not completed (RETURN_FAIL) For more information, see "ReturnCode enumerated type" on page 384.

<Go Back



14 IStackerDriver interface

VWorks plugins that control stackers must implement the IStackerDriver interface.

This chapter defines the IStackerDriver methods.

IMPORTANT All VWorks device driver plugins must implement the IWorksDriver, IControllerClient, and IWorksDiags interfaces.

This chapter contains the following topics:

- "IStackerDriver methods overview" on page 170
- "IsStackEmpty method" on page 171
- "IsStackFull method" on page 172
- "LoadStack method" on page 173
- "ScanStack method" on page 175
- "SinkPlate method" on page 176
- "SourcePlate method" on page 178
- "UnloadStack method" on page 180

<Go Back

IStackerDriver methods overview

Use the following table to quickly locate an IStackerDriver method by name, by description, or by page number.

Method	Description	See...
`IsStackEmpty`	Asks the plugin whether the stack at the specified location is empty.	"IsStackEmpty method" on page 171
`IsStackFull`	Asks the plugin whether the stack at the specified location is full.	"IsStackFull method" on page 172
`LoadStack`	Tells the plugin to prepare the stack at the specified location for robot access.	"LoadStack method" on page 173
`ScanStack`	Tells the plugin to scan the stack at the specified location.	"ScanStack method" on page 175
`SinkPlate`	Tells the plugin to upstack a labware from the specified location.	"SinkPlate method" on page 176
`SourcePlate`	Tells the plugin to downstack a labware from the specified location.	"SourcePlate method" on page 178
`UnloadStack`	Tells the plugin to release the stack at the specified location so the operator can remove the stack.	"UnloadStack method" on page 180

<Go Back

IsStackEmpty method

Description

VWorks software calls the `IsStackEmpty` method to ask the plugin whether the stack at the specified location is empty.

Note: For devices that do not need to scan the stack before downstacking or upstacking a labware, this method can return `E_NOTIMPL` (`0x80004001`).

Syntax

```
HRESULT IsStackEmpty(
   [in] BSTR Location,
   [out,retval] SHORT *IsEmpty
);
```

Parameters

`Location`	[in] The name of the location of the stack.
`IsEmpty`	[out, retval] Indicates whether the stack is empty. Possible values: `0` = The stack is not empty `1` = The stack is empty

Related information

For information about...	See...
IsStackFull method	"IsStackFull method" on page 172
LoadStack method	"LoadStack method" on page 173
ScanStack method	"ScanStack method" on page 175
SinkPlate method	"SinkPlate method" on page 176
SourcePlate method	"SourcePlate method" on page 178
UnloadStack method	"UnloadStack method" on page 180

<Go Back

IsStackFull method

Description

VWorks software calls the `IsStackFull` method to ask the plugin whether the stack at the specified location is full.

Syntax

```
HRESULT IsStackFull(
   [in] BSTR Location,
   [out,retval] SHORT *IsFull
);
```

Parameters

`Location`	[in] The name of the location of the stack.
`IsFull`	[out, retval] Indicates whether the stack is full. Possible values: `0` = The stack is not full `1` = The stack is full

Related information

For information about...	See...
IsStackEmpty method	"IsStackEmpty method" on page 171
LoadStack method	"LoadStack method" on page 173
ScanStack method	"ScanStack method" on page 175
SinkPlate method	"SinkPlate method" on page 176
SourcePlate method	"SourcePlate method" on page 178
UnloadStack method	"UnloadStack method" on page 180

LoadStack method

Description

VWorks software calls the `LoadStack` method to tell the plugin to prepare the stack at the specified location for robot access. VWorks software calls this method once at the start of a protocol run.

Syntax

```
HRESULT LoadStack(
   [in] BSTR Labware,
   [in] enum PlateFlagsType PlateFlags,
   [in] BSTR Location,
   [out,retval] enum ReturnCode *retVal
);
```

Parameters

`Labware`	[in] The labware type.
`PlateFlags`	[in] Specifies whether the labware has a lid, is sealed, or neither. Possible values: `0` = The labware does not have a lid and is not sealed `1` = The labware has a lid `2` = The labware is sealed
`Location`	[in] The name of the location of the stack.
`retVal`	[out, retval] Returns an error code. Possible values: `0` = The request was completed (`RETURN_SUCCESS`) `1` = Something was wrong with the input, so the request was not completed (`RETURN_BAD_ARGS`) `2` = The request was not completed (`RETURN_FAIL`) For more information, see "ReturnCode enumerated type" on page 384.

Related information

For information about...	See...
IsStackEmpty method	"IsStackEmpty method" on page 171
IsStackFull method	"IsStackFull method" on page 172

<Go Back

For information about...	See...
ReturnCode enumerated type	“ReturnCode enumerated type” on page 384
ScanStack method	“ScanStack method” on page 175
SinkPlate method	“SinkPlate method” on page 176
SourcePlate method	“SourcePlate method” on page 178
UnloadStack method	“UnloadStack method” on page 180

<Go Back

ScanStack method

Description

VWorks software calls the `ScanStack` method to tell the plugin to scan the stack at the specified location.

IMPORTANT VWorks software always calls the `ScanStack` method before the `SinkPlate` or `SourcePlate` method. See "SinkPlate method" on page 176 and "SourcePlate method" on page 178.

Syntax

```
HRESULT ScanStack(
   [in] BSTR Location,
   [out,retval] enum ReturnCode *retVal
);
```

Parameters

`Location`	[in] The name of the location of the stack.
`retVal`	[out, retval] Returns an error code. Possible values: `0` = The request was completed (`RETURN_SUCCESS`) `1` = Something was wrong with the input, so the request was not completed (`RETURN_BAD_ARGS`) `2` = The request was not completed (`RETURN_FAIL`) For more information, see "ReturnCode enumerated type" on page 384.

Related information

For information about...	See...
IsStackEmpty method	"IsStackEmpty method" on page 171
IsStackFull method	"IsStackFull method" on page 172
LoadStack method	"LoadStack method" on page 173
ReturnCode enumerated type	"ReturnCode enumerated type" on page 384
SinkPlate method	"SinkPlate method" on page 176
SourcePlate method	"SourcePlate method" on page 178
UnloadStack method	"UnloadStack method" on page 180

<Go Back

SinkPlate method

Description

VWorks software calls the SinkPlate method to tell the plugin to upstack a labware to the specified location.

If VWorks software calls the SourcePlate method next, VWorks software expects to retrieve the same labware. See "SourcePlate method" on page 178.

IMPORTANT VWorks software always calls the ScanStack method before the SinkPlate or SourcePlate method. See "ScanStack method" on page 175 and "SourcePlate method" on page 178.

Syntax

```
HRESULT SinkPlate(
   [in] BSTR Labware,
   [in] enum PlateFlagsType PlateFlags,
   [in] BSTR SinkToLocation,
   [out,retval] enum ReturnCode *retVal
);
```

Parameters

Labware	[in] The labware type.
PlateFlags	[in] Specifies whether the labware has a lid, is sealed, or neither. Possible values: 0 = The labware does not have a lid and is not sealed 1 = The labware has a lid 2 = The labware is sealed
SinkToLocation	[in] The name of the location of the stack to which the labware is to be upstacked.
retVal	[out, retval] Returns an error code. Possible values: 0 = The request was completed (RETURN_SUCCESS) 1 = Something was wrong with the input, so the request was not completed (RETURN_BAD_ARGS) 2 = The request was not completed (RETURN_FAIL) For more information, see "ReturnCode enumerated type" on page 384.

<Go Back

Related information

For information about...	See...
IsStackEmpty method	“IsStackEmpty method” on page 171
IsStackFull method	“IsStackFull method” on page 172
LoadStack method	“LoadStack method” on page 173
ReturnCode enumerated type	“ReturnCode enumerated type” on page 384
ScanStack method	“ScanStack method” on page 175
SourcePlate method	“SourcePlate method” on page 178
UnloadStack method	“UnloadStack method” on page 180

<Go Back

SourcePlate method

Description

VWorks software calls the `SourcePlate` method to tell the plugin to downstack a labware from the specified location.

IMPORTANT VWorks software always called the `ScanStack` method before the `SinkPlate` or `SourcePlate` method. See "ScanStack method" on page 175 and "SinkPlate method" on page 176.

Syntax

```
HRESULT SourcePlate(
   [in] BSTR Labware,
   [in] enum PlateFlagsType PlateFlags,
   [in] BSTR SourceFromLocation,
   [out,retval] enum ReturnCode *retVal
);
```

Parameters

`Labware`	[in] The labware type.
`PlateFlags`	[in] Specifies whether the labware has a lid, is sealed, or neither. Possible values: `0` = The labware does not have a lid and is not sealed `1` = The labware has a lid `2` = The labware is sealed
`SourceFromLocation`	[in] The name of the location of the stack from which the labware is to be downstacked.
`retVal`	[out, retval] Returns an error code. Possible values: `0` = The request was completed (`RETURN_SUCCESS`) `1` = Something was wrong with the input, so the request was not completed (`RETURN_BAD_ARGS`) `2` = The request was not completed (`RETURN_FAIL`) For more information, see "ReturnCode enumerated type" on page 384.

<Go Back

Related information

For information about...	See...
IsStackEmpty method	“IsStackEmpty method” on page 171
IsStackFull method	“IsStackFull method” on page 172
LoadStack method	“LoadStack method” on page 173
ReturnCode enumerated type	“ReturnCode enumerated type” on page 384
ScanStack method	“ScanStack method” on page 175
SinkPlate method	“SinkPlate method” on page 176
UnloadStack method	“UnloadStack method” on page 180

<Go Back

UnloadStack method

Description

VWorks software calls the `UnloadStack` method to tell the plugin to release the stack at the specified location so the operator can remove the stack.

Syntax

```
HRESULT UnloadStack(
   [in] BSTR Labware,
   [in] enum PlateFlagsType PlateFlags,
   [in] BSTR Location,
   [out,retval] enum ReturnCode *retVal
);
```

Parameters

`Labware`	[in] The labware name.
`PlateFlags`	[in] Specifies whether the labware has a lid, is sealed, or neither. Possible values: `0` = The labware does not have a lid and is not sealed `1` = The labware has a lid `2` = The labware is sealed
`Location`	[in] The name of the location of the stack.
`retVal`	[out, retval] Returns an error code. For more information, see "ReturnCode enumerated type" on page 384. Possible values: `0` = The request was completed (`RETURN_SUCCESS`) `1` = Something was wrong with the input, so the request was not completed (`RETURN_BAD_ARGS`) `2` = The request was not completed (`RETURN_FAIL`)

Related information

For information about...	See...
IsStackEmpty method	"IsStackEmpty method" on page 171
IsStackFull method	"IsStackFull method" on page 172
LoadStack method	"LoadStack method" on page 173

<Go Back

For information about...	See...
ReturnCode enumerated type	“ReturnCode enumerated type” on page 384
ScanStack method	“ScanStack method” on page 175
SinkPlate method	“SinkPlate method” on page 176
SourcePlate method	“SourcePlate method” on page 178

<Go Back

15 IStorageDriver interface

VWorks plugins that control storage devices must implement the IStorageDriver interface.

This chapter defines the IStorageDriver methods.

IMPORTANT All VWorks device driver plugins must implement the IWorksDriver, IControllerClient, and IWorksDiags interfaces.

- "IStorageDriver methods overview" on page 184
- "LoadPlate method" on page 185
- "LookupLocations method" on page 188
- "QueryStorageLocations method" on page 192
- "UnloadPlate method" on page 194

<Go Back

IStorageDriver methods overview

Use the following table to quickly locate an IStorageDriver method by name, by description, or by page number.

Method	Description	See...
`GetStorageLocations`	*Obsolete*. This method should be implemented as `return E_NOTIMPL (0x80004001)`.	
`LoadPlate`	Tells the plugin to accept the specified labware from the robot and either 1) load the labware to the specified location, or 2) place the labware on the storage device's handoff location, where the storage device loads the labware to the specified location.	"LoadPlate method" on page 185
`LookupLocations`	Gets the names from the plugin of all the locations to which or from which the robot can move labware.	"LookupLocations method" on page 188
`QueryStorageLocations`	Provides the plugin with the coordinates of the range of labware to be inventoried.	"QueryStorageLocations method" on page 192
`UnloadPlate`	Tells the plugin to present the specified labware at the specified location to the robot for unloading.	"UnloadPlate method" on page 194

<Go Back

LoadPlate method

Description

VWorks software calls the `LoadPlate` method to tell the plugin to accept the specified labware from the robot. Then robot does one of the following:

- Loads the labware to the specified location
- Places the labware on the storage device's handoff location, and the storage device loads the labware to the specified location

VWorks software calls this method each time a Load task is executed during a protocol run.

IMPORTANT VWorks software always calls the `LookupLocations` method before the `LoadPlate` and `UnloadPlate` methods. See "LookupLocations method" on page 188 and "UnloadPlate method" on page 194.

Syntax

```
HRESULT LoadPlate(
   [in] BSTR Labware,
   [in] enum PlateFlagsType PlateFlags,
   [in] BSTR LoadPlateLocationXML,
   [out,retval] enum ReturnCode *retVal
);
```

Parameters

`Labware`	[in] The labware name.
`PlateFlags`	[in] Specifies whether the labware has a lid, is sealed, or neither. Possible values: `0` = The labware does not have a lid and is not sealed `1` = The labware has a lid `2` = The labware is sealed
`LoadPlateLocationXML`	[in] A StorageLocation XML block that contains information about the location on the storage device to which the specified labware is to be loaded.
`retVal`	[out, retval] Returns an error code. For more information, see "ReturnCode enumerated type" on page 384. Possible values: `0` = The request was completed (`RETURN_SUCCESS`) `1` = Something was wrong with the input, so the request was not completed (`RETURN_BAD_ARGS`) `2` = The request was not completed (`RETURN_FAIL`)

<Go Back

LoadPlate method input

VWorks software passes a StorageLocation XML block into the `LoadPlateLocationXML` parameter of the `LoadPlate` method.

StorageLocation XML block

The StorageLocation XML block contains the `StorageLocation` element and all its children. This XML block provides information about the location on the storage device, including the cassette/slot coordinates, that is used for a Load or Unload task.

XML structure

The value of the `file` attribute for the `Velocity11` element is `MetaData`. See "Velocity11 element" on page 416.

```
<?xml version='1.0' encoding='ASCII'>
<Velocity11>
   <StorageLocation>
      <Coordinates>
         <StorageLocationCoordinate Name='Cassette' ... />
         <StorageLocationCoordinate Name='Slot' ... />
      </Coordinates>
      <Location />
   </StorageLocation>
</Velocity11>
```

StorageLocation element

The `StorageLocation` element has two children: `Coordinates` and `Location`.

Coordinates element

The `Coordinates` element contains two `StorageLocationCoordinate` elements.

StorageLocationCoordinate element

Each `StorageLocationCoordinate` element has one of the following pairs of `Name` and `Value` attributes:

Name	Value
`Name`	The value `Cassette`.
`Value`	The name of the cassette. *Note:* The cassette name was received with a previous call to the `GetMetaData` method. See IWorksDriver "GetMetaData method" on page 45.
`Name`	The value `Slot`.
`Value`	The name of the slot. *Note:* The slot name was received with a previous call to the `GetMetaData` method. See IWorksDriver "GetMetaData method" on page 45.

<Go Back

Location element

The `Location` element contains information about the location on the storage device. The `Location` element is defined in "Location element" on page 407.

Example of LoadPlate method input

The following sample code is a StorageLocation XML block that is received by the plugin from VWorks software as a string in the `LoadPlateLocationXML` parameter of the `LoadPlate` method. VWorks software tells the plugin to accept a labware from the robot. Then the robot is to load the specified labware to the location named `Primary (Load/Unload) Pad`, which is one of the StoreX device's external locations.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='3787f34ae05399b617f2cf072129fb4a' version='1.0' >
   <StorageLocation >
      <Coordinates >
         <StorageLocationCoordinate Name='Cassette' Value='1' />
         <StorageLocationCoordinate Name='Slot' Value='1' />
      </Coordinates>
      <Location Group='0' MaxStackHeight='460' Name='Primary (Load/Unload) Pad'
→Offset='0' Type='4' />
   </StorageLocation>
</Velocity11>
```

Related information

For information about...	See...
LookupLocations method	"LookupLocations method" on page 188
MakeLocationAvailable method	"MakeLocationAvailable method" on page 67
ReturnCode enumerated type	"ReturnCode enumerated type" on page 384
UnloadPlate method	"LoadPlate method" on page 185

<Go Back

LookupLocations method

Description

VWorks software calls the `LookupLocations` method to get the names from the plugin of all the locations to which or from which the robot can move labware.

Note: The *internal location* is the location on the storage device, which is considered inside the system. The *external location* is the location outside the system.

IMPORTANT VWorks software always calls the `LookupLocations` method before the `LoadPlate` and `UnloadPlate` methods. See "LoadPlate method" on page 185 and "UnloadPlate method" on page 194.

Syntax

```
HRESULT LookupLocations(
   [in] VARIANT_BOOL Load,
   [in] BSTR Labware,
   [in] enum PlateFlagsType PlateFlags,
   [in] BSTR StorageLocationXML,
   [out] BSTR *ExternalLocationsXML,
   [out,retval] enum ReturnCode *retVal
);
```

Parameters

`Load`	[in] Indicates whether the labware is about to be loaded or unloaded. Possible values: `-1` = The labware is about to be loaded `0` = The labware is about to be unloaded
`Labware`	[in] The labware name.
`PlateFlags`	[in] Specifies whether the labware has a lid, is sealed, or neither. Possible values: `0` = The labware does not have a lid and is not sealed `1` = The labware has a lid `2` = The labware is sealed
`StorageLocationXML`	[in] A StorageLocation XML block that contains information about the location on the storage device.
`ExternalLocationsXML`	[out] A LocationVector XML block that contains information about the external location.

<Go Back

retVal	[out, retval] Returns an error code. For more information, see "ReturnCode enumerated type" on page 384. Possible values: 0 = The request was completed (RETURN_SUCCESS) 1 = Something was wrong with the input, so the request was not completed (RETURN_BAD_ARGS) 2 = The request was not completed (RETURN_FAIL)

LookupLocations method input

VWorks software passes a StorageLocation XML block into the StorageLocationsXML parameter of the LookupLocations method.

StorageLocation XML block

The StorageLocation XML block contains the StorageLocation element and all its children. This XML block provides information about the location on the storage device, including the cassette/slot coordinates, that is used for a Load or Unload task.

XML structure

The value of the file attribute for the Velocity11 element is MetaData. See "Velocity11 element" on page 416.

```
<?xml version='1.0' encoding='ASCII'>
<Velocity11>
   <StorageLocation>
      <Coordinates>
         <StorageLocationCoordinate Name='Cassette' ... />
         <StorageLocationCoordinate Name='Slot' ... />
      </Coordinates>
      <Location />
   </StorageLocation>
</Velocity11>
```

StorageLocation element

The StorageLocation element has two children: Coordinates and Location.

Coordinates element

The Coordinates element contains two StorageLocationCoordinate elements.

StorageLocationCoordinate element

Each StorageLocationCoordinate element has one of the following pairs of Name and Value attributes.

Name	Value
Name	The value Cassette.
Value	The name of the cassette. *Note:* The cassette name was received with a previous call to the GetMetaData method. See IWorksDriver "GetMetaData method" on page 45.

<Go Back

Name	Value
Name	The value Slot.
Value	The name of the slot. *Note:* The slot name was received with a previous call to the GetMetaData method. See IWorksDriver "GetMetaData method" on page 45.

Location element

The Location element contains information about the location on the storage device. The Location element is defined in "Location element" on page 407.

Example of LookupLocations method input

The following sample code is a StorageLocation XML block that is received by the plugin from VWorks software as a string in the StorageLocationXML parameter of the LookupLocations method. When the value of the Load parameter is -1, VWorks software asks the plugin for all the locations on the storage device at which the robot can load the specified labware.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='dd1caf38ce495aaae3d6957494bd2ef0' version='1.0' >
   <StorageLocation >
      <Coordinates >
         <StorageLocationCoordinate Name='Cassette' Value='1' />
         <StorageLocationCoordinate Name='Slot' Value='1' />
      </Coordinates>
      <Location Group='0' MaxStackHeight='460' Offset='0' Type='1' />
   </StorageLocation>
</Velocity11>
```

LookupLocations method output

The plugin returns a LocationVector XML block in the ExternalLocationsXML parameter of the LookupLocations method.

LocationVector XML block

The LocationVector XML block contains the LocationVector element and all its children. This XML block provides information about the external location that is to be used for a Load or Unload task.

XML structure

The value of the file attribute of the Velocity11 element is MetaData. See "Velocity11 element" on page 416.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <LocationVector>
      <Locations>
         <Location />
         ...
      </Locations>
   </LocationVector>
</Velocity11>
```

<Go Back

LocationVector element

The `LocationVector` element contains one `Locations` element.

Locations element

The `Locations` element contains one or more `Location` elements.

Location element

The `Location` element contains information about the external location. The `Location` element is defined in "Location element" on page 407.

Example of LookupLocations method output

The following sample code is a LocationVector XML block that is returned to VWorks software by the plugin as a string in the `ExternalLocationsXML` parameter of the `LookupLocations` method. The code returns information about `Primary (Load/Unload) Pad`, which is one of the StoreX device's external locations.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='d9def80adb4895a149000be927986a3e' version='1.0' >
   <LocationVector >
      <Locations >
         <Location Group='0' MaxStackHeight='460' Name='Primary (Load/Unload) Pad'
→Offset='0' Type='4' />
      </Locations>
   </LocationVector>
</Velocity11>
```

Related information

For information about...	See...
LoadPlate method	"LoadPlate method" on page 185
MakeLocationAvailable method	"MakeLocationAvailable method" on page 67
ReturnCode enumerated type	"ReturnCode enumerated type" on page 384
UnloadPlate method	"UnloadPlate method" on page 194

<Go Back

QueryStorageLocations method

Description

VWorks software calls the QueryStorageLocations method when the user requests an inventory using the Inventory Editor. The method provides the plugin with a starting cassette/slot and ending cassette/slot. All slots between the start and end are inventoried.

After the plugin performs the inventory, it returns the results of the inventory by creating an InventoryPlateBarcode update with a call to the IWorksController Update method. See "InventoryPlateBarcodes update" on page 358.

Syntax

```
HRESULT QueryStorageLocations(
   [in] BSTR QueryXML,
   [out,retval] enum ReturnCode *retVal
);
```

Parameters

QueryXML	[in] A Velocity11 XML element that specifies the coordinates of the range of labware to be inventoried.
retVal	[out, retval] Returns an error code. For more information, see "ReturnCode enumerated type" on page 384. Possible values: 0 = The request was completed (RETURN_SUCCESS) 1 = Something was wrong with the input, so the request was not completed (RETURN_BAD_ARGS) 2 = The request was not completed (RETURN_FAIL)

QueryStorageLocations method input

VWorks software passes a Velocity11 XML element into the QueryXML parameter of the QueryStorageLocations method. This XML element provides the coordinates of the range of labware to be inventoried.

Velocity11 element

The Velocity11 element has the following attributes:

Name	Value
EndCassette	The cassette at the end of the labware range.
EndSlot	The slot at the end of the labware range.
StartCassette	The cassette at the beginning of the labware range.

<Go Back

Name	Value
StartSlot	The slot at the beginning of the labware range.
file	The value PlateStorageInventory.

Example of QueryStorageLocations method input

The following sample code is a Velocity11 XML element received by the plugin from VWorks software as a string in the QueryXML parameter of the QueryStorageLocations method. VWorks software tells the plugin to inventory the labware at the specified coordinates. The range includes all slots from the start cassette/slot to the end casette/slot.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 EndCassette='2' EndSlot='1' StartCassette='2' StartSlot='1'
→file='PlateStorageInventory' md5sum='30a2242fee50dbc4d5e2b2df229e7574'
→version='1.0' />
```

Related information

For information about...	See...
InventoryPlateBarcodes update (IWorksController Update method)	"InventoryPlateBarcodes update" on page 358
ReturnCode enumerated type	"ReturnCode enumerated type" on page 384

<Go Back

UnloadPlate method

Description

VWorks software calls the `UnloadPlate` method to tell the plugin to present the specified labware to the robot. Then the robot unloads the labware from the specified location or from the device's transfer station. VWorks software calls this method each time an Unload task is executed during a protocol run.

IMPORTANT VWorks software always calls the `LookupLocations` method before the `LoadPlate` and `UnloadPlate` methods. See "LookupLocations method" on page 188 and "LoadPlate method" on page 185.

Syntax

```
HRESULT UnloadPlate(
   [in] BSTR Labware,
   [in] enum PlateFlagsType PlateFlags,
   [in] BSTR UnloadPlateLocationXML,
   [out,retval] enum ReturnCode *retVal
);
```

Parameters

`Labware`	[in] The labware name.
`PlateFlags`	[in] Specifies whether the labware has a lid, is sealed, or neither. Possible values: `0` = The labware does not have a lid and is not sealed `1` = The labware has a lid `2` = The labware is sealed
`UnloadPlateLocationXML`	[in] A StorageLocation XML block that contains information about the location on the storage device from which the robot is to unload the specified labware.
`retVal`	[out, retval] Returns an error code. For more information, see "ReturnCode enumerated type" on page 384. Possible values: `0` = The request was completed (`RETURN_SUCCESS`) `1` = Something was wrong with the input, so the request was not completed (`RETURN_BAD_ARGS`) `2` = The request was not completed (`RETURN_FAIL`)

UnloadPlate method input

VWorks software passes a StorageLocation XML block into the `UnloadPlateLocationXML` parameter of the `UnloadPlate` method.

<Go Back

StorageLocation XML block

The StorageLocation XML block contains the `StorageLocation` element and all its children. This XML block provides information about the location on the storage device that is used for a Load or Unload task.

XML structure

The value of the `file` attribute for the `Velocity11` element is `MetaData`. See "Velocity11 element" on page 416.

```
<?xml version='1.0' encoding='ASCII'>
<Velocity11>
   <StorageLocation>
      <Coordinates>
         <StorageLocationCoordinate Name='Cassette' ... />
         <StorageLocationCoordinate Name='Slot' ... />
      </Coordinates>
      <Location />
   </StorageLocation>
</Velocity11>
```

StorageLocation element

The `StorageLocation` element has two children: `Coordinates` and `Location`.

Coordinates element

The `Coordinates` element contains two `StorageLocationCoordinate` elements.

StorageLocationCoordinate element

Each `StorageLocationCoordinate` element has one of the following pairs of `Name` and `Value` attributes.

Name	Value
`Name`	The value `Cassette`.
`Value`	The name of the cassette. *Note:* The cassette name was received with a previous call to the `GetMetaData` method. See IWorksDriver "GetMetaData method" on page 45.
`Name`	The value `Slot`.
`Value`	The name of the slot. *Note:* The slot name was received with a previous call to the `GetMetaData` method. See IWorksDriver "GetMetaData method" on page 45.

Location element

The `Location` element contains information about the location on the storage device. The `Location` element is defined in "Location element" on page 407.

Example of UnloadPlate method input

The following sample code is a StorageLocation XML block that is received by the plugin from VWorks software as a string in the `UnloadPlateLocationXML` parameter of the `UnloadPlate` method. VWorks

<Go Back

software tells the plugin to accept a labware from the robot. Then the robot is to unload the specified labware to the location named `Primary (Load/Unload) Pad`, which is one of the StoreX device's external locations.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='3787f34ae05399b617f2cf072129fb4a' version='1.0' >
   <StorageLocation >
      <Coordinates >
         <StorageLocationCoordinate Name='Cassette' Value='1' />
         <StorageLocationCoordinate Name='Slot' Value='1' />
      </Coordinates>
      <Location Group='0' MaxStackHeight='460' Name='Primary (Load/Unload) Pad'
→Offset='0' Type='4' />
   </StorageLocation>
</Velocity11>
```

Related information

For information about...	See...
LoadPlate method	"LoadPlate method" on page 185
LookupLocations method	"LookupLocations method" on page 188
MakeLocationAvailable method	"MakeLocationAvailable method" on page 67
ReturnCode enumerated type	"ReturnCode enumerated type" on page 384

16 IVHooks interface

VWorks plugins that want to act on events in VWorks software must implement the IVHooks interface.

This chapter provides the following information about the IVHooks interface:

- Definitions of the XML metadata structures that are returned in the output parameter of IVHooks methods
- Explanations of when and why VWorks software calls IVHooks methods
- Descriptions of how the plugin uses IVHooks methods to respond to events in VWorks software

IMPORTANT All VWorks device driver plugins must implement the IWorksDriver, IControllerClient, and IWorksDiags interfaces.

This chapter contains the following topics:

- "IVHooks methods overview" on page 199
- "IVHooks interface methods output" on page 201
- "Aborted method" on page 203
- "BarCodeMisread method" on page 205
- "BarCodeRead method" on page 210
- "CompileComplete method" on page 214
- "Deadlock method" on page 218
- "Error method" on page 220
- "FileOpened method" on page 222
- "FileSaved method" on page 224
- "GetUserInterface method" on page 226
- "LiquidTransferComplete method" on page 227
- "ProcessFinished method" on page 231
- "ProcessStarting method" on page 234
- "ProtocolFinished method" on page 237
- "ProtocolPaused method" on page 239
- "ProtocolStarted method" on page 241
- "RobotMove method" on page 243
- "RobotPickComplete method" on page 246
- "RobotPlaceComplete method" on page 248
- "ScriptPlateError method" on page 250
- "TaskFinished method" on page 252
- "TaskStarting method" on page 255
- "UserLoggedIn method" on page 258

<Go Back

- “UserLoggedOut method” on page 260

<Go Back

IVHooks methods overview

Use the following table to quickly locate an IVHooks method by name, by event, or by page number.

Method	Event	See...
`Aborted`	A protocol run was aborted.	"Aborted method" on page 203
`AvailablePlateList`	*Obsolete.* This method should be implemented as `return E_NOTIMPL (0x80004001)`.	
`BarCodeMisread`	A barcode misread occurred.	"BarCodeMisread method" on page 205
`BarCodeRead`	A barcode read occurred.	"BarCodeRead method" on page 210
`CompileComplete`	A protocol was compiled.	"CompileComplete method" on page 214
`CustomHook`	A labware was inventoried.	"CustomHook method" on page 216
`CustomMenuClick`	*Reserved for internal use.* This method should be implemented as `return E_NOTIMPL (0x80004001)`.	
`Deadlock`	A deadlock occurred.	"Deadlock method" on page 218
`Error`	An error occurred in VWorks software.	"Error method" on page 220
`FileOpened`	A protocol file, runset file, or device file was opened.	"FileOpened method" on page 222
`FileSaved`	A protocol file, runset file, or device file was saved.	"FileSaved method" on page 224
`GetPlateInfo`	*Obsolete.* This method should be implemented as `return E_NOTIMPL (0x80004001)`.	
`GetUserInterface`	The user chose **Tools** > **Open Hooks Plugin for...** in the VWorks main window and then clicked the plugin's file name.	"GetUserInterface method" on page 226
`LiquidTransferComplete`	A liquid-transfer process was completed.	"LiquidTransferComplete method" on page 227
`PipetProcessFinished`	*Obsolete.* This method should be implemented as `return E_NOTIMPL (0x80004001)`.	

<Go Back

Method	Event	See...
PipetProcessStarting	*Obsolete.* This method should be implemented as return E_NOTIMPL (0x80004001).	
PipetTaskFinished	*Obsolete.* This method should be implemented as return E_NOTIMPL (0x80004001).	
PipetTaskStarting	*Obsolete.* This method should be implemented as return E_NOTIMPL (0x80004001).	
PlateGroupMapping	*Obsolete.* This method should be implemented as return E_NOTIMPL (0x80004001).	
ProcessFinished	A process finished.	"ProcessFinished method" on page 231
ProcessStarting	A process started.	"ProcessStarting method" on page 234
ProtocolFinished	A protocol finished.	"ProtocolFinished method" on page 237
ProtocolPaused	A protocol is paused.	"ProtocolPaused method" on page 239
ProtocolStarted	A protocol started.	"ProtocolStarted method" on page 241
RobotMove	A robot is about to move away from its current location and to a new location.	"RobotMove method" on page 243
RobotPickComplete	A robot picked up a labware.	"RobotPickComplete method" on page 246
RobotPlaceComplete	A robot placed a labware.	"RobotPlaceComplete method" on page 248
ScriptPlateError	An error message was received from a script associated with a protocol.	"ScriptPlateError method" on page 250
TaskFinished	A task finished.	"TaskFinished method" on page 252
TaskStarting	A task started.	"TaskStarting method" on page 255
UserLoggedIn	A user logged in to VWorks software.	"UserLoggedIn method" on page 258
UserLoggedOut	A user logged out of VWorks software.	"UserLoggedOut method" on page 260

IVHooks interface methods output

When VWorks software calls an IVHooks method, the plugin can return the following XML metadata as a string in the output parameters:

- An XML element
- A HookResults XML block
- An XML block other than a HookResults XML block

In some IVHooks methods, the plugin can return either an XML element or a HookResults XML block.

XML elements and other blocks

The XML elements and XML blocks (other than the HookResults XML block) contained in the output parameters of IVHooks methods are defined in the methods that return them.

HookResults XML block

The HookResults XML block contains the `HookResults` element and all its children. This XML block can do one or both of the following:

- Tell VWorks software to write a specified message to the Main Log
- Command the scheduler to pause or abort a runset or a protocol

All HookResults XML blocks returned in the output parameter of IVHooks methods have the same structure, as defined in this section.

XML structure

The HookResults XML block has the following XML structure. The value of the `file` attribute for the `Velocity11` element is `MetaData`.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <HookResults>
      <Results>
         <HookResult />
         ...
      </Results>
   </HookResults>
</Velocity11>
```

Velocity11 element

In addition to the `file`, `md5sum`, and `version` attributes, the `Velocity11` element contains the `Action` attribute for the following methods only:

- "BarCodeMisread method" on page 205
- "BarCodeRead method" on page 210
- "CompileComplete method" on page 214

HookResults element

The `HookResults` element contains one `Results` element.

Results element

The `Results` element contains one or more `HookResult` elements.

<Go Back

HookResult element

The `HookResult` element has the following pairs of `ResultType` and `ResultValue` attributes:

Name	Description/Name-Value
`ResultType`/ `ResultValue`	The output to return to VWorks software. • Writes an Info-type message to the Main Log. `ResultType`: The value `LogMessage`. `ResultValue`: A text string that describes the message. • Writes an Error-type message to the Main Log. `ResultType`: The value `LogError`. `ResultValue`: A text string that describes the message. • Commands the scheduler to pause a runset. `ResultType`: The value `PauseExecution`. `ResultValue`: The value of this attribute must be `True`. • Prepends a message string with `Plugin pause:` followed by a space and then writes the results to the Main Log as an Info-type message. The message is written to the log in the following format: `'Plugin pause: ' + ResultValue` `ResultType`: The value `PauseMessage`. `ResultValue`: A text string that describes the pause message. • Commands the scheduler to abort a protocol. `ResultType`: The value `AbortExecution`. `ResultValue`: The value of this attribute must be `True`. • Prepends a message string with `Plugin abort:` followed by a space and then writes the results to the Main Log as an Info-type message. The message is written to the log in the following format: `'Plugin abort: ' + ResultValue` `ResultType`: The value `AbortMessage`. `ResultValue`: A text string that describes the abort message.

Related information

For information about...	See...
Overview of IVHooks interface methods	"IVHooks methods overview" on page 199

<Go Back

Aborted method

Event

A protocol run was aborted.

Description

VWorks software calls the `Aborted` method after a protocol run is aborted.

Syntax

```
HRESULT Aborted(
   [in] BSTR sXML,
   [in, out] BSTR* sResultXML
);
```

Parameters

`sXML`	[in] An Aborted XML element containing the file path of the protocol that generated the event.
`sResultXML`	[in, out] A HookResults XML block. See "HookResults XML block" on page 201.

Aborted method input

VWorks software passes an Aborted XML element into the `sXML` parameter of the `Aborted` method. This XML element provides the file path of the protocol that generated the event.

Aborted element

The `Aborted` element has the following attribute:

Name	Value
`Path`	The name of the protocol that generated the event. If the protocol has been saved, the value of this attribute is the protocol's file path. If the protocol has not been saved, the value is the default protocol name.

<Go Back

Example of Aborted method input

The following sample code is an Aborted XML element that is received by the plugin from VWorks software as a string in the `sXML` parameter of the `Aborted` method. VWorks software tells the plugin that the protocol named `Protocol File - 1` was aborted.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='dcddf516cdfa849276034a138c8ac016' version='1.0' >
   <Aborted Path='Protocol File - 1' />
</Velocity11>
```

Aborted method output

The plugin returns a HookResults XML block in the `sResultXML` parameter of the `Aborted` method. For information about the structure and contents of this XML block, see "HookResults XML block" on page 201.

Example of Aborted method output

The following sample code is a HookResults XML block that is returned to VWorks software from the plugin as a string in the `sResultXML` parameter of the `Aborted` method. The plugin tells VWorks software to write the following Info-type message to the Main Log:

```
ProtocolFile - 1 was aborted.
```

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='52226bb1b70c756162b551f1f5685a5d' version='1.0'>
   <HookResults>
      <Results>
         <HookResult ResultType='Log Message'
→ResultValue='ProtocolFile - 1 was aborted. />
      </Results>
   </HookResults>
</Velocity11>
```

Related information

For information about...	See...
HookResults XML block	"HookResults XML block" on page 201

<Go Back

BarCodeMisread method

Event

A barcode misread occurred.

Description

VWorks software calls the `BarCodeMisread` method after a barcode misread occurs. The plugin must tell VWorks software what action to take.

Also, if an expected barcode is not found, VWorks software reports a barcode misread and calls the `BarCodeMisread` method.

Syntax

```
HRESULT BarCodeMisread(
   [in] BSTR sXML,
   [in, out] BSTR* sResultXML
);
```

Parameters

`sXML`	[in] A BarCodeMisread XML element containing information about the labware that was involved in the barcode misread.
`sResultXML`	[in, out] A Velocity11 XML element or a HookResults XML block that specifies the action to take after a barcode misread occurs. See "HookResults XML block" on page 201.

BarCodeMisread method input

VWorks software passes a BarCodeMisread XML element into the `sXML` parameter of the `BarCodeMisread` method. This XML element provides information about the labware involved in the barcode misread.

BarCodeMisread element

The `BarCodeMisread` element has the following attributes:

Name	Value
`BarcodeSide`	Represents the side of the labware where the barcode misread occurred. Possible values: • `NORTH_SIDE` • `SOUTH_SIDE` • `EAST_SIDE` • `WEST_SIDE`

<Go Back

Name	Value
BarcodeRead	The barcode as read by the device.
OriginalBarcode	The expected barcode, or else the value No Barcode or no value.
PlateName	The labware name.
InstanceNumber	The labware instance number.
DatabaseID	The database ID of the labware.
Plugin	The name of the plugin.
Labware	The labware type.
Device	The name of the device where the barcode misread occurred.
Location	The name of the location on the device where the barcode misread occurred.
Path	The name of the protocol that generated the event. If the protocol has been saved, the value of this attribute is the protocol's file path. If the protocol has not been saved, the value is the default protocol name.
Action	This attribute is not used by VWorks software.

Example of BarCodeMisread method input

The following sample code is a BarCodeMisread XML element that is received by the plugin from VWorks software as a string in the sXML parameter of the BarCodeMisread method. VWorks software tells the plugin that a barcode misread occurred on the south side of the tip box named NameofPlate.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='849392019ca47102839e845113d11840'
→version='1.0'>
   <BarCodeMisread BarcodeSide='SOUTH_SIDE' BarcodeRead='NAW1001'
→OriginalBarcode='NAW1002' PlateName='NameofPlate' InstanceNumber='1' DatabaseID='1'
→Labware='384 V11 ST10 Tip Box 10734.102' Device='NameofDevice'
→Location='NameofLocation' Path='C:\VWorks Workspace\Protocols\protocol1.pro'/>
</Velocity11>
```

BarCodeMisread output

The plugin returns either a Velocity11 XML element or a HookResults element in the sResultXML parameter of the BarCodeMisread method as follows:

Action VWorks software should take	Output type	Action or ResultType attribute value
Send the labware to the quarantine device.	Velocity11 XML element	Action=BCR_QUARANTINE
Automatically replace the barcode.	Velocity11 XML element	Action=BCR_REPLACE

<Go Back

Action VWorks software should take	Output type	Action or ResultType attribute value
Halt the protocol and prompt the user to resolve the misread.	Velocity11 XML element	`Action=BCR_HALT_REPLACE`
Ignore the error and continue as if the misread has not occurred.	Velocity11 XML element	`Action=BCR_IGNORE`
Write a message to the Main Log.	HookResults XML block	`ResultType=LogMessage` or `ResultType=LogError`
Pause the scheduler.	HookResults XML block	`ResultType=PauseExecution`

Velocity 11 element (Velocity11 XML element)

For the Velocity11 XML element, the `Velocity11` element has the following attributes plus the `md5sum` and `version` attributes:

Name	Value
`file`	The value `BarCodeMisreadResult`.
`Action`	Specifies the action to take after a barcode misread occurs. Possible values: • `BCR_QUARANTINE` Send the labware to the quarantine device. • `BCR_REPLACE` Automatically replace the barcode. • `BCR_HALT_REPLACE` Halt the protocol and prompt the user to resolve the misread. *Note:* This value cannot be used unless the user selected **Halt on bar code misreads** in the Options dialog box. • `BCR_IGNORE` Take no action.
`BarcodeResult`	The barcode to use if the value of the `Action` attribute is `BCR_REPLACE`. If `Action` is not `BCR_REPLACE`, this attribute has no value.

<Go Back

Velocity 11 element (HookResults XML block)

For the HookResults XML block, the `Velocity11` element has the following additional attribute:

Name	Value
`Action`	Specifies the action to take after a barcode misread occurs. Possible values: • `BCR_QUARANTINE` Send the labware to the quarantine device. • `BCR_REPLACE` Automatically replace the barcode. • `BCR_HALT_REPLACE` Halt the protocol and prompt the user to resolve the misread. *Note:* This value cannot be used unless the user selected **Halt on bar code misreads** in the Options dialog box. • `BCR_IGNORE` Take no action.

Example of BarCodeMisread method output (Velocity11 XML element)

The following sample code is a Velocity11 XML element that is returned to VWorks software by the plugin as a string in the `sResultXML` parameter of the `BarCodeMisread` method. The plugin tells VWorks software to replace the barcode that was read with the barcode named `A123456`.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='BarcodeMisreadResult' md5sum='52226bb1b70c756162b551f1f5685a5d'
→version='1.0' Action='BCR_REPLACE' BarcodeResult='A123456' />
```

Example of BarCodeMisread method output (HookResults XML block)

The following sample code is a HookResults XML block that is returned to VWorks software by the plugin as a string in the `sResultXML` parameter of the `BarCodeMisread` method. (See "HookResults XML block" on page 201.) The plugin tells the scheduler to pause the protocol after a barcode misread occurs. The plugin also tells VWorks software to quarantine the labware and to write the following Error-type message to the Main Log:

```
Plugin paused: Barcode is not in the database.
```

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='52226bb1b70c756162b551f1f5685a5d' version='1.0'
→Action='BCR_QUARANTINE'>
   <HookResults>
      <Results>
         <HookResult ResultType='LogError' ResultValue='Plugin paused: Barcode is not
→in the database.' />
         <HookResult ResultType='PauseExecution' ResultValue='True' />
      </Results>
   </HookResults>
</Velocity11>
```

<Go Back

Related information

For information about...	See...
BarCodeRead method	"BarCodeRead method" on page 210
HookResults XML block	"HookResults XML block" on page 201

<Go Back

BarCodeRead method

Event

A barcode read occurred.

Description

VWorks software calls the `BarCodeRead` method after a barcode read occurs. The plugin must tell VWorks software what action to take if a barcode is present on one of the labware's four sides.

Plugins can also use this method to determine if a barcode is expected or valid.

Syntax

```
HRESULT BarCodeRead(
   [in] BSTR sXML,
   [in, out] BSTR* sResultXML
);
```

Parameters

`sXML`	[in] A Velocity11 XML element containing information about the labware that was involved in the barcode read.
`sResultXML`	[in, out] A Velocity11 XML element or a HookResults XML block that specifies the action to take if a barcode is present on one of the labware's four sides. See "HookResults XML block" on page 201.

BarCodeRead method input

VWorks software passes a Velocity11 XML element into the `sXML` parameter of the `BarCodeRead` method. This XML element provides information about the labware involved in the barcode read.

Velocity11 element

The `Velocity11` element has the following attributes:

Name	Value
`NorthBarcode`	The barcode if it is located on the north side of the labware.
`SouthBarcode`	The barcode if it is located on the south side of the labware.

<Go Back

Name	Value
WestBarcode	The barcode if it is located on the west side of the labware.
EastBarcode	The barcode if it is located on the east side of the labware.
PlateName	The labware name.
InstanceNumber	The labware instance number.
DatabaseID	The database ID of the labware.
Plugin	The name of the plugin.
Labware	The labware type.
Device	The device name.
Location	The name of the location on the device where the barcode read occurred.
Path	The name of the protocol that generated the event. If the protocol has been saved, the value of this attribute is the protocol's file path. If the protocol has not been saved, the value is the default protocol name.

Example of BarCodeRead method input

The following sample code is a Velocity11 XML element that is received by the plugin from VWorks software as a string in the sXML parameter of the BarCodeRead method. VWorks software tells the plugin that a barcode read occurred on the south side of the tip box named NameofPlate.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='849392019ca47102839e845113d11840' version='1.0'>
   < BarCodeRead NorthBarcode='BAR0001' SouthBarcode='BAR0002' WestBarcode='BAR0003'
→EastBarcode='BAR0004' PlateName='NameofPlate' InstanceNumber='1' DatabaseID='1'
→Labware='384 V11 ST10 Tip Box 10734.102' Device='NameofDevice'
→Location='NameofLocation' Path='C:\VWorks Workspace\Protocols\protocol1.pro'/>
</Velocity11>
```

BarCodeRead output

The plugin returns either a Velocity11 XML element or a HookResults element in the sResultXML parameter of the BarCodeRead method as follows:

Action VWorks software should take	Output type	Action or ResultType attribute value
Send the labware to the quarantine device.	Velocity11 XML element	Action=BCR_QUARANTINE
Ignore the error and continue as if the misread has not occurred.	Velocity11 XML element	Action=BCR_IGNORE
Write a message to the Main Log.	HookResults XML block	ResultType=LogMessage or ResultType=LogError

<Go Back

Action VWorks software should take	Output type	Action or ResultType attribute value
Pause the scheduler.	HookResults XML block	`ResultType=PauseExecution`

Velocity 11 element (Velocity11 XML element)

For the Velocity11 XML element, the `Velocity11` element has the following attributes plus the `md5sum` and `version` attributes:

Name	Value
`file`	The value `BarCodeReadResult`.
`Action`	Specifies the action to take after a barcode read occurs. Possible values: • `BCR_QUARANTINE` Send the labware to the quarantine device. • `BCR_IGNORE` Take no action.

Velocity 11 element (HookResults XML block)

For the HookResults XML block, the `Velocity11` element has the following additional attribute:

Name	Value
`Action`	Specifies the action to take after a barcode read occurs. Possible values: • `BCR_QUARANTINE` Send the labware to the quarantine device. • `BCR_IGNORE` Take no action.

Example of BarCodeRead method output (Velocity11 XML element)

The following sample code is a Velocity11 element that is returned to VWorks software by the plugin as a string in the `sXML` parameter of the `BarCodeRead` method. The plugin tells VWorks software to take no action after a barcode read occurs.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='BarCodeReadResult' md5sum='52226bb1b70c756162b551f1f5685a5d'
→version='1.0' Action='BCR_IGNORE' />
```

Example of BarCodeRead method output (HookResults XML block)

The following sample code is a HookResults XML block that is returned to VWorks software by the plugin as a string in the `sXML` parameter of the `BarCodeRead` method. (See "HookResults XML block" on page 201.) The

<Go Back

plugin tells the scheduler to pause the protocol. The plugin also tells VWorks software to quarantine the labware and to write the following Error-type message to the Main Log:

```
Plugin pause: Barcode is not in the database.
```

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='52226bb1b70c756162b551f1f5685a5d'
→version='1.0' Action='BCR_QUARANTINE'>
   <HookResults>
      <Results>
         <HookResult ResultType='LogError' ResultValue='Plugin paused: Barcode is not
→in the database.' />
         <HookResult ResultType='PauseExecution' ResultValue='True' />
      </Results>
   </HookResults>
</Velocity11>
```

Related information

For information about...	See...
BarCodeMisread method	“BarCodeMisread method” on page 205
HookResults XML block	“HookResults XML block” on page 201

<Go Back

CompileComplete method

Event

A protocol was compiled.

Description

VWorks software calls the `CompileComplete` method after a protocol is compiled. The plugin tells VWorks software whether the protocol run should proceed if compiler errors occurred.

Syntax

```
HRESULT CompileComplete(
   [in] BSTR sXML,
   [in, out] BSTR* sResultXML
);
```

Parameters

`sXML`	[in] A Velocity11 XML element containing the number of errors and warnings that occurred during the compilation.
`sResultXML`	[in] A Velocity11 XML element that contains the action to take if compiler errors occurred.

CompileComplete method input

VWorks software passes a Velocity11 XML element into the `sXML` parameter of the `CompileComplete` method. This XML element provides the number of errors and warnings that occurred during the compilation.

Velocity11 element

The `Velocity11` element has the following attributes:

Name	Value
`Errors`	The number of errors that occurred when the protocol was compiled.
`Path`	The name of the protocol that generated the event. If the protocol has been saved, the value of this attribute is the protocol's file path. If the protocol has not been saved, the value is the default protocol name.
`Warnings`	The number of warnings that occurred when the protocol was compiled.

<Go Back

Example of CompileComplete method input

The following sample code is a Velocity11 XML element that is received by the plugin from VWorks software in the `sXML` parameter of the `CompileComplete` method. VWorks software tells the plugin that no warnings or errors occurred when the protocol was compiled.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='02ed84e9c28dad9cb815334189ba49ca' version='1.0' >
   <CompileComplete Errors='0' Path='Protocol File - 1' Warnings='0' />
</Velocity11>
```

CompileComplete method output

The plugin returns a Velocity11 XML element in the `sResultXML` parameter of the `CompileComplete` method. This XML element provides the action to take if compiler error occurred.

Velocity11 element

The `Velocity11` element has the following additional attribute:

Name	Value
`Action`	Indicates whether the protocol run should proceed if compiler errors occurred. Possible values: • `AllowErrors` Allow the protocol run to proceed. • Any non-empty string Do not allow the protocol run to proceed.

Example of CompileComplete method output

The following sample code is a Velocity11 XML element that is returned to VWorks software by the plugin as a string in the `sResultXML` parameter of the `CompileComplete` method. The plugin tells VWorks software to allow the protocol to proceed, even if compiler errors occurred.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='849392019ca47102839e845113d11840' version='1.0'
→Action='AllowErrors' />
```

<Go Back

CustomHook method

Event

A labware was inventoried.

Description

VWorks software calls the `CustomHook` method after the storage device inventories a labware. This method is for Liconic incubators that support only west-side barcodes.

Syntax

```
HRESULT CustomHook(
   [in] BSTR sXML,
   [in, out] BSTR* sResultXML
);
```

Parameters

`sXML`	[in] An OnPlateInventoried XML element that contains information about the inventoried labware.
`sResultXML`	[in, out] A HookResults XML block. See "HookResults XML block" on page 201.

CustomHook method input

VWorks software passes an OnPlateInventoried XML element into the `sXML` parameter of the `CustomHook` method. This XML element provides the following information about the labware that was inventoried.

OnPlateInventoried element

The `OnPlateInventoried` element has the following attributes:

Name	Value
`cassette`	The number of the cassette.
`slot`	The number of the slot.
`west_barcode`	The barcode on the west side of the labware.
`device_name`	The device name.

<Go Back

Example of CustomHook method input

The following sample code is an OnPlateInventoried XML element that is received by the plugin from VWorks software as a string in the `sXML` parameter of the `CustomHook` method. The plugin returns information about the labware that was inventoried by the device named `Store X Device Driver`.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='PlateStorageInventory' md5sum='02ed84e9c28dad9cb815334189ba49ca'
→version='1.0' >
   <OnPlateInventoried cassette='1' slot='1' west_barcode='XUB89893-909'
→device_name='StoreX driver - 1' />
</Velocity11>
```

CustomHook method output

The plugin returns a HookResults XML block in the `sResultXML` parameter of the `CustomHook` method. For information about the structure and contents of this XML block, see "HookResults XML block" on page 201.

Example of CustomHook method output

The following sample code is a HookResults XML block that is returned to VWorks software by the plugin as a string in the `sXML` parameter of the `CustomHook` method. The plugin tells VWorks software to write the following Info-type message to the Main Log:

```
A microplate was inventoried.
```

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='52226bb1b70c756162b551f1f5685a5d' version='1.0'>
   <HookResults>
      <Results>
         <HookResult ResultType='LogMessage' ResultValue='A microplate was
→inventoried.' />
      </Results>
   </HookResults>
</Velocity11>
```

Related information

For information about...	See...
HookResults XML block	"HookResults XML block" on page 201

<Go Back

Deadlock method

Event

A deadlock occurred.

Description

VWorks software calls the `Deadlock` method after a deadlock occurs.

Syntax

```
HRESULT Deadlock(
   [in] BSTR sXML,
   [in, out] BSTR* sResultXML
);
```

Parameters

`sXML`	[in] A Deadlock XML element containing the file path to the protocol that generated the event.
`sResultXML`	[in, out] A HookResults XML block. See "HookResults XML block" on page 201.

Deadlock method input

VWorks software passes a Deadlock XML element into the `sXML` parameter of the `Deadlock` method. This XML element provides the file path to the protocol that generated the event.

Deadlock element

The `Deadlock` element has the following attribute:

Name	Value
`Path`	The name of the protocol that generated the event. If the protocol has been saved, the value of this attribute is the protocol's file path. If the protocol has not been saved, the value is the default protocol name.

<Go Back

Example of Deadlock method input

The following sample code is a Deadlock XML element that is received by the plugin from VWorks software as a string in the `sXML` parameter of the `Deadlock` method. VWorks software tells the plugin that the protocol named `Protocol File - 1` encountered a deadlock during execution.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='19e8733a7594b5be747639772d28e706' version='1.0' >
   <Deadlock Path='Protocol File - 1' />
</Velocity11>
```

Deadlock method output

The plugin returns a HookResults XML block in the `sResultXML` parameter of the `Deadlock` method. For information about the structure and contents of this XML block, see "HookResults XML block" on page 201.

Example of Deadlock method output

The following sample code is a HookResults XML block that is returned to VWorks software by the plugin as a string in the `sResultXML` parameter of the `Deadlock` method. The plugin tells VWorks software to write the following Info-type message to the Main Log:

```
A deadlock occurred during execution of ProtocolFile - 1.
```

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='52226bb1b70c756162b551f1f5685a5d' version='1.0'>
   <HookResults>
      <Results>
         <HookResult ResultType='Log Message'
→ResultValue='A deadlock occurred during execution of ProtocolFile - 1.' />
      </Results>
   </HookResults>
</Velocity11>
```

Related information

For information about...	See...
HookResults XML block	"HookResults XML block" on page 201

<Go Back

Error method

Event

An error occurred in VWorks software.

Description

VWorks software calls the `Error` method after an error occurs, that is, the plugin returned a `RETURN_FAIL` error code. See "ReturnCode enumerated type" on page 384.

Syntax

```
HRESULT Error(
   [in] BSTR sXML,
   [in, out] BSTR* sResultXML
);
```

Parameters

`sXML`	[in] An Error XML element containing information about the error that occurred in VWorks software.
`sResultXML`	[in, out] A HookResults XML block. See "HookResults XML block" on page 201.

Error method input

VWorks software passes an Error XML element into the `sXML` parameter of the `Error` method. This XML element provides information about the error that occurred in VWorks software.

Error element

The `Error` element has the following attributes:

Name	Value
`Path`	The name of the protocol that generated the event. If the protocol has been saved, the value of this attribute is the protocol's file path. If the protocol has not been saved, the value is the default protocol name.
`Source`	The source of the error.
`Message`	A text string that describes the error.
`InstanceNumber`	The labware instance number.
`DeviceName`	The device name.

<Go Back

Name	Value
LocationName	The name of the location on the device where the error occurred.
Profile	The profile name.
ProcessName	The labware name.

Example of Error method input

The following sample code is an Error XML element that is received by the plugin from VWorks software as a string in the sXML parameter of the Error method. VWorks software passes information about an error that was generated by the plugin named Plugin - 1.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='02ed84e9c28dad9cb815334189ba49ca' version='1.0' >
   <Error Path='Protocol File - 1' Source='Plugin - 1' Message='Error'
→InstanceNumber='1' DeviceName='Device - 1' LocationName='Location1'
→Profile='Profile1' ProcessName='Process - 1' />
</Velocity11>
```

Error method output

The plugin returns a HookResults XML block in the sResultXML parameter of the Error method. For information about the structure and contents of this XML block, see "HookResults XML block" on page 201.

Example of Error method output

The following sample code is a HookResults XML block that is returned to VWorks software by the plugin as a string in the sResultXML parameter of the Error method. The plugin tells the scheduler to pause the runset. It also tells VWorks software to write the following Error-type message to the Main Log:

```
Unable to move to safe height at end of protocol.
```

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='52226bb1b70c756162b551f1f5685a5d'
→version='1.0' >
   <HookResults>
      <Results>
         <HookResult ResultType='Log Error' ResultValue='Unable to move to safe height
→at end of protocol.' />
         <HookResult ResultType='Pause Execution' ResultValue='True' />
      </Results>
   </HookResults>
</Velocity11>
```

Related information

For information about...	See...
HookResults XML block	"HookResults XML block" on page 201

<Go Back

FileOpened method

Event

A protocol file, runset file, or device file was opened.

Description

VWorks software calls the `FileOpened` method when the user opens a protocol file, a runset file, or a device file.

Syntax

```
HRESULT FileOpened(
   [in] BSTR sXML,
   [in, out] BSTR* sResultXML
);
```

Parameters

`sXML`	[in] A FileLoaded XML element that contains the file path of the opened file.
`sResultXML`	[in, out] A HookResults XML block. See "HookResults XML block" on page 201.

FileOpened method input

VWorks software passes a FileLoaded XML element into the `sXML` parameter of the `FileOpened` method. This XML element provides the file path of the file that was opened.

FileLoaded element

The `FileLoaded` element has the following attribute:

Name	Value
`Path`	The file path of the file that was opened.

<Go Back

Example of FileLoaded method input

The following sample code is FileLoaded XML element that is received by the plugin from VWorks software as a string in the `sXML` parameter of the `FileOpened` method. VWorks software tells the plugin that the protocol file named `protocol1.pro` located in the `C:\V11\V11 Files\Protocols` directory was opened.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='3083237761ca765a7c5389d20beda64e' version='1.0' >
   <FileLoaded Path='C:\V11\V11 Files\Protocols\protocol1.pro' />
</Velocity11>
```

FileOpened method output

The plugin returns a HookResults XML block in the `sResultXML` parameter of the `FileOpened` method. For information about the structure and contents of this XML block, see "HookResults XML block" on page 201.

Example of FileOpened method output

The following sample code is a HookResults XML block that is returned to VWorks software by the plugin as a string in the `sResultXML` parameter of the `FileOpened` method. The plugin tells VWorks software to write the following Info-type message to the Main Log:

```
protocol1.pro was opened.
```

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='52226bb1b70c756162b551f1f5685a5d' version='1.0'>
   <HookResults>
      <Results>
         <HookResult ResultType='Log Message'
→ResultValue='protocol1.pro was opened.' />
      </Results>
   </HookResults>
</Velocity11>
```

Related information

For information about...	See...
HookResults XML block	"HookResults XML block" on page 201

<Go Back

FileSaved method

Event

A protocol file, runset file, or device file was saved.

Description

VWorks software calls the `FileSaved` method when the user saves a protocol file, runset file, or device file.

Syntax

```
HRESULT FileSaved(
   [in] BSTR sXML,
   [in, out] BSTR* sResultXML
);
```

Parameters

`sXML`	[in] A FileSaved XML element that contains the file path of the saved file.
`sResultXML`	[in, out] A HookResults XML block. See "HookResults XML block" on page 201.

FileSaved method input

VWorks software passes a FileSaved XML element into the `sXML` parameter of the `FileSaved` method. This XML element provides the file path of the file that was saved.

FileSaved element

The `FileSaved` element has the following attribute:

Name	Value
`Path`	The file path of the file that was saved.

<Go Back

Example of FileSaved method input

The following sample code is a FileSaved XML element that is received by the plugin from VWorks software as a string in the `sXML` parameter of the `FileSaved` method. VWorks software tells the plugin that the protocol file named `file1.pro` located in the `C:\V11\V11 Files\Protocols` directory was saved.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='912539b6aeeb1373ffccab4d4983d456' version='1.0' >
   <FileSaved Path='C:\V11\V11 Files\Protocols\file1.pro' />
</Velocity11>
```

FileSaved method output

The plugin returns a HookResults XML block in the `sResultXML` parameter of the `FileSaved` method. For information about the structure and contents of this XML block, see "HookResults XML block" on page 201.

Example of FileSaved method output

The following sample code is a HookResults XML block that is returned to VWorks software by the plugin as a string in the `sResultXML` parameter of the `FileSaved` method. The plugin tells VWorks software to write the following Info-type message to the Main Log:

```
file1.pro was saved.
```

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='52226bb1b70c756162b551f1f5685a5d' version='1.0'>
   <HookResults>
      <Results>
         <HookResult ResultType='Log Message' ResultValue='file1.pro was saved.' />
      </Results>
   </HookResults>
</Velocity11>
```

Related information

For information about...	See...
HookResults XML block	"HookResults XML block" on page 201

<Go Back

GetUserInterface method

Event

The user chose **Tools** > **Open Hooks Plugin for...** in the VWorks main window and then clicked the plugin's file name.

Description

VWorks software calls the `GetUserInterface` method when the user chooses **Tools** > **Open Hooks Plugin for...** in the VWorks main window and then clicks the plugin's file name. The plugin responds by opening its user interface in a modal window. A call to this method should not return until the user closes the window. If the plugin does not provide a user interface, it should return `E_NOTIMPL (0x80004001)`.

Syntax

```
HRESULT GetUserInterface(
   [in, out] BSTR* pResultXML
);
```

Parameters

`pResultXML`	[in] An empty string. [out] Any output from the plugin is ignored by VWorks software.

<Go Back

LiquidTransferComplete method

Event

A liquid-transfer process is finished.

Description

VWorks software calls the `LiquidTransferComplete` method after a liquid-transfer process is finished. VWorks software provides information about both the source and destination labware in the input parameters of this method.

The plugin can use the `LiquidTransferComplete` method to keep a liquid management system updated on the locations of compounds in the system, in real time.

Syntax

```
HRESULT LiquidTransferComplete(
   [in] BSTR SourcePlateInfoXML,
   [in] BSTR DestinationPlateInfoXML,
   [in, out] BSTR* pResultXML
);
```

Parameters

`SourcePlateInfoXML`	[in] A LiquidTransferComplete XML element containing information about the source labware that was involved in the liquid-transfer process.
`DestinationPlateInfoXML`	[in] A LiquidTransferComplete XML element containing information about the destination labware that was involved in the liquid-transfer process.
`pResultXML`	[in, out] A HookResults XML block. See "HookResults XML block" on page 201.

SourcePlateInfoXML and DestinationPlateInfoXML parameters

VWorks software passes a LiquidTransferComplete XML element into the following parameters of the `LiquidTransferComplete` method:

- `SourcePlateInfoXML` for the source labware
- `DestinationPlateInfoXML` for the destination labware

This XML element provides information about the labware that was involved in the liquid-transfer process.

<Go Back

LiquidTransferComplete element

The `LiquidTransferComplete` element has the following attributes.

Note: Where an attribute name is prepended by a lowercase letter, `i`=integer, `f`=float, and `s`=string.

Name	Value
`Path`	The name of the protocol that generated the event. If the protocol has been saved, the value of this attribute is the protocol's file path. If the protocol has not been saved, the value is the default protocol name.
`fVolume`	The volume, in microliters.
`iColumn`	The column of the well in the labware in which the liquid-transfer process occurred (1-based).
`sQuadrant`	The set quadrant, which depends on the microplate-to-labware mapping (`1`, `1–4`, or `1–16`). *Note:* Refer to the *VWorks Automation Control User Guide* for more information about quadrant patterns.
`iRow`	The row of the well on the microplate in which the liquid-transfer process occurred (1-based).
`sDevice`	The name of the destination device.
`sPlate`	The labware name.
`sTimestamp`	The date and time of the liquid-transfer process, in the following format: `m/d/yyyy - H:M:S.MS AM(PM)` where `m` = month (`1–12`) `d` = day (`1–31`) `yyyy` = year (`1980–2099`) `H` = hour (`1–12`) `M` = minutes (`0–59`) `S` = seconds (`0–59`) `MS` = milliseconds (`0–59`)
`sWellDescription`	The description of the well.
`sNorthSideBarcode`	The barcode if it is located on the north side of the microplate, or else the value `No bar code`.
`sSouthSideBarcode`	The barcode if it is located on the south side of the microplate, or else the value `No bar code`.
`sEastSideBarcode`	The barcode if it is located on the east side of the microplate, or else the value `No bar code`.

<Go Back

Name	Value
sWestSideBarcode	The barcode if it is located on the west side of the microplate, or else the value No bar code.

Example of LiquidTransferComplete method input for the source microplate

The following sample code is a LiquidTransferComplete XML element that is received by the plugin from VWorks software as a string in the SourcePlateInfoXML parameter of the LiquidTransferComplete method. VWorks software tells the plugin that a liquid-transfer process is finished on the source microplate named process - 1 1.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='87c92b22402e34a7bdd7ab7ff0c39147' version='1.0' >
   <LiquidTransferComplete Path='C:\V11\V11 Files\Protocols\Hooks reference test.pro'
→fVolume='10' iColumn='1' iQuadrant='1' iRow='1' sDevice='Bravo - 1'
→sEastSideBarcode='No bar code' sNorthSideBarcode='No bar code'
→sPlate='process - 1 1' sSouthSideBarcode='No bar code'
→sTimestamp='2010-4-7 21:27:35' sWellDescription='Quadrant 1'
→sWestSideBarcode='No bar code' />
</Velocity11>
```

Example of LiquidTransferComplete method input for the destination microplate

The following sample code is a LiquidTransferComplete XML element that is received from VWorks software by the plugin as a string in the DestinationPlateInfoXML parameter of the LiquidTransferComplete method. VWorks software tells the plugin that a liquid-transfer process is finished on the destination microplate named process - 2 1.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='3f1102a46cf0727053bd6273b867c66c' version='1.0' >
   <LiquidTransferComplete Path='C:\V11\V11 Files\Protocols\Hooks reference test.pro'
→fVolume='10' iColumn='1' iQuadrant='1' iRow='1' sDevice='Bravo - 1'
→sEastSideBarcode='No bar code' sNorthSideBarcode='No bar code'
→sPlate='process - 2 1' sSouthSideBarcode='No bar code'
→sTimestamp='2010-4-7 21:27:37' sWellDescription='Quadrant 1'
→sWestSideBarcode='No bar code' />
</Velocity11>
```

LiquidTransferComplete method output

The plugin returns a HookResults XML block in the pResultXML parameter of the LiquidTransferComplete method. For information about the structure and contents of this XML block, see "HookResults XML block" on page 201.

Example of LiquidTransferComplete method output

The following sample code is a HookResults XML block that is returned to VWorks software by the plugin as a string in the sResultXML parameter of the LiquidTransferComplete method. The plugin tells VWorks software to write the following Info-type message to the Main Log:

```
The liquid-transfer process is finished.
```

<Go Back

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='52226bb1b70c756162b551f1f5685a5d' version='1.0'>
   <HookResults>
      <Results>
         <HookResult ResultType='Log Message'
→ResultValue='The liquid-transfer process is finished.' />
      </Results>
   </HookResults>
</Velocity11>
```

Related information

For information about...	See...
HookResults XML block	"HookResults XML block" on page 201

<Go Back

ProcessFinished method

Event

A process is finished.

Description

VWorks software calls the `ProcessFinished` method after a process is finished.

Syntax

```
HRESULT ProcessFinished(
   [in] BSTR sXML,
   [in, out] BSTR* sResultXML
);
```

Parameters

`sXML`	[in] A ProcessFinishing XML element containing information about the process that is finished.
`sResultXML`	[in, out] A HookResults XML block. See "HookResults XML block" on page 201.

ProcessFinished method input

VWorks software passes a ProcessFinishing XML element into the `sXML` parameter of the `ProcessFinished` method. This XML element provides information about the process that is finished.

ProcessFinishing element

The `ProcessFinishing` element has the following attributes:

Name	Value
`DatabaseID`	The database ID of the labware.
`ProcessName`	The process name.
`InstanceNumber`	The labware instance number.
`PlateName`	The process name. This attribute has the same value as the `ProcessName` attribute.
`Labware`	The labware type.
`TaskCount`	The count of the tasks within the process.

<Go Back

Name	Value
Path	The name of the protocol that generated the event. If the protocol has been saved, the value of this attribute is the protocol's file path. If the protocol has not been saved, the value is the default protocol name.
ProtocolPath	The name of the protocol that generated the event. This attribute has the same value as the Path attribute.
ProtocolType	Represents the protocol type. Possible values: 1 = Startup 2 = Main 3 = Cleanup
NorthSideBarcode	The barcode if it is located on the north side of the labware, or else the value No bar code.
SouthSideBarcode	The barcode if it is located on the south side of the labware, or else the value No bar code.
EastSideBarcode	The barcode if it is located on the east side of the labware, or else the value No bar code.
WestSideBarcode	The barcode, if it is located on the west side of the labware or else the value No bar code.

Example of ProcessFinished method input

The following sample code is a ProcessFinishing XML element that is received by the plugin from VWorks software as a string in the sXML parameter of the ProcessFinished method. VWorks software tells the plugin that the process named process - 1 is finished in the Main Protocol.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='e1461e22aa9869c9b0ed38e3ea1744ed' version='1.0' >
   <ProcessFinishing DatabaseID='-1' EastSideBarcode='No bar code' InstanceNumber='1'
→Labware='1536 Black Greiner' NorthSideBarcode='No bar code' Path='C:\V11\V11
→Files\Protocols\[16784] Quarantine Plate.pro' PlateName='process - 1'
→ProcessName='process - 1' ProtocolPath='C:\V11\V11 Files\Protocols\[16784]
→Quarantine Plate.pro' ProtocolType='2' SouthSideBarcode='No bar code' TaskCount='4'
→WestSideBarcode='No bar code' />
</Velocity11>
```

ProcessFinished method output

The plugin returns a HookResults XML block in the sResultXML parameter of the ProcessFinished method. For information about the structure and contents of this XML block, see "HookResults XML block" on page 201.

<Go Back

Example of ProcessFinished method output

The following sample code is a HookResults XML block that is returned to VWorks software by the plugin as a string in the `sResultXML` parameter of the `ProcessFinished` method. The plugin tells VWorks software to write the following Info-type message to the Main Log:

```
process - 1 is finished.
```

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='52226bb1b70c756162b551f1f5685a5d' version='1.0'>
   <HookResults>
      <Results>
         <HookResult ResultType='Log Message' ResultValue='process - 1 is finished. />
      </Results>
   </HookResults>
</Velocity11>
```

Related information

For information about...	See...
HookResults XML block	"HookResults XML block" on page 201

<Go Back

ProcessStarting method

Event

A process started.

Description

VWorks software calls the `ProcessStarting` method after a process starts.

Syntax

```
HRESULT ProcessStarting(
   [in] BSTR sXML,
   [in, out] BSTR* sResultXML
);
```

Parameters

`sXML`	[in] A ProcessStarting XML element containing information about the process that started.
`sResultXML`	[in, out] A HookResults XML block. See "HookResults XML block" on page 201.

ProcessStarting method input

VWorks software passes a ProcessStarting XML element into the `sXML` parameter of the `ProcessStarting` method. This XML element provides information about the process that started.

ProcessStarting element

The `ProcessStarting` element has the following attributes:

Name	Value
`DatabaseID`	The database ID of the labware.
`ProcessName`	The process name.
`InstanceNumber`	The labware instance number.
`PlateName`	The process name. This attribute has the same value as the `ProcessName` attribute.
`Labware`	The labware type.
`TaskCount`	The count of the tasks within the process.

<Go Back

Name	Value
Path	The name of the protocol that generated the event. If the protocol has been saved, the value of this attribute is the protocol's file path. If the protocol has not been saved, the value is the default protocol name.
ProtocolPath	The name of the protocol that generated the event. This attribute has the same value as the Path attribute.
ProtocolType	Represents the protocol type. Possible values: 1 = Startup 2 = Main 3 = Cleanup
NorthSideBarcode	The barcode if it is located on the north side of the labware, or else the value No bar code.
SouthSideBarcode	The barcode if it is located on the south side of the labware, or else the value No bar code.
EastSideBarcode	The barcode if it is located on the east side of the labware, or else the value No bar code.
WestSideBarcode	The barcode if it is located on the west side of the labware, or else the value No bar code.

Example of ProcessStarting method input

The following sample code is a ProcessStarting XML element that is received by the plugin from VWorks software as a string in the sXML parameter of the ProcessStarting method. VWorks software tells the plugin that the process named process - 1 started in the Main Protocol.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='74d4f5229fbb5f52ddc0e9e2b9a2739f' version='1.0' >
   <ProcessStarting DatabaseID='-1' EastSideBarcode='No bar code' InstanceNumber='1'
→Labware='1536 Black Greiner' NorthSideBarcode='No bar code'
→Path='C:\V11\MyProtocol.pro' PlateName='process - 1' ProcessName='process - 1'
→ProtocolPath='C:\V11\MyProtocol.pro' ProtocolType='2'
→SouthSideBarcode='No bar code' TaskCount='4' WestSideBarcode='No bar code' />
</Velocity11>
```

ProcessStarting method output

The plugin returns a HookResults XML block in the sResultXML parameter of the ProcessStarting method. For information about the structure and contents of this XML block, see "HookResults XML block" on page 201.

<Go Back

Example of ProcessStarting method output

The following sample code is a HookResults XML block that is returned to VWorks software by the plugin as a string in the `sResultXML` parameter of the `ProcessStarting` method. The plugin tells VWorks software to write the following Info-type message to the Main Log:

```
process - 1 started.
```

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='52226bb1b70c756162b551f1f5685a5d' version='1.0'>
   <HookResults>
      <Results>
         <HookResult ResultType='Log Message' ResultValue='process - 1 started.' />
      </Results>
   </HookResults>
</Velocity11>
```

Related information

For information about...	See...
HookResults XML block	"HookResults XML block" on page 201

<Go Back

ProtocolFinished method

Event

A protocol is finished.

Description

VWorks software calls the `ProtocolFinished` method after a protocol is finished.

Syntax

```
HRESULT ProtocolFinished(
   [in] BSTR sXML,
   [in, out] BSTR* sResultXML
);
```

Parameters

`sXML`	[in] A ProtocolFinished XML element containing information about the protocol that is finished.
`sResultXML`	[in, out] A HookResults XML block. See "HookResults XML block" on page 201.

ProtocolFinished method input

VWorks software passes a ProtocolFinished XML element into the `sXML` parameter of the `ProtocolFinished` method. This XML element provides information about the protocol that is finished.

ProtocolFinished element

The `ProtocolFinished` element has the following attributes:

Name	Value
`Path`	The name of the protocol that generated the event. If the protocol has been saved, the value of this attribute is the protocol's file path. If the protocol has not been saved, the value is the default protocol name.

`ProtocolType`	Represents the protocol type.
	Possible values:
	`1 = Startup`
	`2 = Main`
	`3 = Cleanup`

<Go Back

Example of ProtocolFinished method input

The following sample code is a ProtocolFinished XML element that is received by the plugin from VWorks software as a string in the `sXML` parameter of the `ProtocolFinished` method. VWorks software tells the plugin that the protocol named `MyProtocol.pro` is finished in the Cleanup Protocol.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='3b6461ce3a66ad5959be51ec7e40662b' version='1.0' >
   <ProtocolFinished Path='C:\V11\MyProtocol.pro' ProtocolType='3' />
</Velocity11>
```

ProtocolFinished method output

The plugin returns a HookResults XML block in the `sResultXML` parameter of the `ProtocolFinished` method. For information about the structure and contents of this XML block, see "HookResults XML block" on page 201.

Example of ProtocolFinished method output

The following sample code is a HookResults XML block that is returned to VWorks software by the plugin as a string in the `sResultXML` parameter of the `ProtocolFinished` method. The plugin tells VWorks software to write the following Info-type message to the Main Log:

```
MyProtocol.pro is finished.
```

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='52226bb1b70c756162b551f1f5685a5d' version='1.0'>
   <HookResults>
      <Results>
         <HookResult ResultType='Log Message'
→ResultValue='MyProtocol.pro is finished.' />
      </Results>
   </HookResults>
</Velocity11>
```

Related information

For information about...	See...
HookResults XML block	"HookResults XML block" on page 201

<Go Back

ProtocolPaused method

Event

The scheduler is paused.

Description

After the scheduler is paused, VWorks software calls the `ProtocolPaused` method to notify the plugin of the state of the scheduler.

Syntax

```
HRESULT ProtocolPaused(
   [in] BSTR xXML,
   [in,out] BSTR* sResultXML
);
```

Parameters

`xXML`	[in] A ProtocolPaused XML element that contains the state of the scheduler when VWorks software called this method.
`sResultXML`	[in, out] A HookResults XML block. See "HookResults XML block" on page 201.

ProtocolPaused method input

VWorks software passes a ProtocolPaused XML element into the `xXML` parameter of the `ProtocolPaused` method. This XML element provides the state of the scheduler when VWorks software called the `ProtocolPaused` method.

ProtocolPaused element

The `ProtocolPaused` element has the following attribute:

Name	Value
`PauseType`	Represents the state of the scheduler. Possible values: `0` = Paused. The scheduler is paused, so protocols have temporarily stopped running. `1` = Continued. The scheduler is now running the previously paused protocols. `2` = Aborted. The scheduler is about to abort the currently running protocols. See "PauseType enumerated type" on page 382.

<Go Back

Example of ProtocolPaused method input

The following sample code is a ProtocolPaused XML element that is received by the plugin from VWorks software as a string in the `xXML` parameter of the `ProtocolPaused` method. VWorks software tells the plugin that the scheduler is paused, so protocols have temporarily stopped running.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='5ebe83f1832cae73ad07dfe9e6353bf3' version='1.0' >
   <ProtocolPaused PauseType='1' />
</Velocity11>
```

ProtocolPaused method output

The plugin returns a HookResults XML block in the `sResultXML` parameter of the `ProtocolPaused` method. For information about the structure and contents of this XML block, see "HookResults XML block" on page 201.

Example of ProtocolPaused method output

The following sample code is a HookResults XML block that is returned to VWorks software by the plugin as a string in the `sResultXML` parameter of the `ProtocolPaused` method. The plugin tells VWorks software to write the following Info-type message to the Main Log:

```
The scheduler is paused.
```

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='52226bb1b70c756162b551f1f5685a5d' version='1.0'>
   <HookResults>
      <Results>
         <HookResult ResultType='Log Message'
→ResultValue='The scheduler is paused.' />
      </Results>
   </HookResults>
</Velocity11>
```

Related information

For information about...	See...
HookResults XML block	"HookResults XML block" on page 201
PauseType enumerated type	"PauseType enumerated type" on page 382

<Go Back

ProtocolStarted method

Event

A protocol started.

Description

VWorks software calls the `ProtocolStarted` method after a protocol starts.

Syntax

```
HRESULT ProtocolStarted(
   [in] BSTR sXML,
   [in, out] BSTR* sResultXML
);
```

Parameters

`sXML`	[in] A ProtocolStarted XML element containing information about the protocol that started.
`sResultXML`	[in, out] A HookResults XML block. See "HookResults XML block" on page 201.

ProtocolStarted method input

VWorks software passes a ProtocolStarted XML element into the `sXML` parameter of the `ProtocolStarted` method. This XML element provides information about the protocol that started.

ProtocolStarted element

The `ProtocolStarted` element has the following attributes:

Name	Value
`Path`	The name of the protocol that generated the event. If the protocol has been saved, the value of this attribute is the protocol's file path. If the protocol has not been saved, the value is the default protocol name.

`ProtocolType`	Represents the protocol type.
	Possible values:
	`1 = Startup`
	`2 = Main`
	`3 = Cleanup`

<Go Back

Name	Value
Simulating	Indicates whether the protocol is running in simulation mode. Possible values: 0 = The protocol is not running in simulation mode 1 = The protocol is running in simulation mode

Example of ProtocolStarted method input

The following sample code is a ProtocolStarted XML element that is received by the plugin from VWorks software as a string in the sXML parameter of the ProtocolStarted method. VWorks software tells the plugin that the protocol named MyProtocol started in the Startup Protocol and that the protocol is running in simulation mode.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='f60b1a768f9a0cd259bde93752ce8201' version='1.0' >
   <ProtocolStarted Path='C:\V11\MyProtocol.pro' ProtocolType='1' Simulating='1' />
</Velocity11>
```

ProtocolStarted method output

The plugin returns a HookResults XML block in the sResultXML parameter of the ProtocolStarted method. For information about the structure and contents of this XML block, see "HookResults XML block" on page 201.

Example of ProtocolStarted method output

The following sample code is a HookResult XML block that is returned to VWorks software by the plugin as a string in the sResultXML parameter of the ProtocolStarted method. The plugin tells VWorks software to write the following Info-type message to the Main Log:

```
MyProtocol.pro started.
```

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='52226bb1b70c756162b551f1f5685a5d' version='1.0'>
   <HookResults>
      <Results>
         <HookResult ResultType='Log Message' ResultValue='MyProtocol.pro started.' />
      </Results>
   </HookResults>
</Velocity11>
```

Related information

For information about...	See...
HookResults XML block	"HookResults XML block" on page 201

RobotMove method

Event

A robot is about to move away from its current location and to a new location.

Description

VWorks software calls the `RobotMove` method when a robot is about to move away from its current location and to a new location.

Syntax

```
HRESULT RobotMove(
   [in] BSTR sXML,
   [in, out] BSTR* sResultXML
);
```

Parameters

`sXML`	[in] A RobotMove XML element containing information about the robot that is about to move away from its current location and to a new location.
`sResultXML`	[in, out] A HookResults XML block. See "HookResults XML block" on page 201.

RobotMove method input

VWorks software passes a RobotMove XML element into the `sXML` parameter of the `RobotMove` method. This XML element provides information about the robot that is about to move away from its current location and to a new location.

RobotMove element

The `RobotMove` element has the following attributes:

Name	Value
`RobotName`	The robot name.
`InstanceNumber`	The labware instance number.
`ProcessName`	The labware name.
`FromDeviceName`	The name of the device that the robot moved away from.
`FromLocationName`	The name of the location from which the robot moved away.

<Go Back

Name	Value
FromTeachpointName	The name of the teachpoint from which the robot moved away.
ToDeviceName	The name of the device to which the robot moved.
ToLocationName	The name of the location to which the robot moved.
ToTeachpointName	The name of the teachpoint to which the robot moved.

Example of RobotMove method input

The following sample code is a RobotMove XML element that is received by the plugin from VWorks software as a string in the sXML parameter of the RobotMove method. VWorks software tells the plugin that the robot named Phantom Robot - 1 moved from the location named Stage on the device named PlatePad - 1 to the location named Hole on the device named WasteBin - 1.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='6efb2bf07f7229a5bf9a47007cbf76b8' version='1.0' >
   <RobotMove FromDeviceName='PlatePad - 1' FromLocationName='Stage'
→FromTeachpointName='robot teachpoint' InstanceNumber='1' ProcessName='process - 1'
→RobotName='Phantom Robot - 1' ToDeviceName='WasteBin - 1' ToLocationName='Hole'
→ToTeachpointName='robot teachpoint' />
</Velocity11>
```

RobotMove method output

The plugin returns a HookResults XML block in the sResultXML parameter of the RobotMove method. For information about the structure and contents of this XML block, see “HookResults XML block” on page 201.

Example of RobotMove method output

The following sample code is a HookResults XML block that is returned to VWorks software by the plugin as a string in the sResultXML parameter of the RobotMove method. The plugin tells VWorks software to write the following Info-type message to the Main Log:

```
Phantom Robot - 1 is about to move from Stage on
PlatePad - 1 to Hole on WasteBin - 1.
```

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='52226bb1b70c756162b551f1f5685a5d' version='1.0'>
   <HookResults>
      <Results>
         <HookResult ResultType='Log Message'
→ResultValue='Phantom Robot - 1 is about to move from Stage on PlatePad - 1 to Hole
→on WasteBin - 1.' />
      </Results>
   </HookResults>
</Velocity11>
```

<Go Back

Related information

For information about...	See...
HookResults XML block	“HookResults XML block” on page 201

<Go Back

RobotPickComplete method

Event

A robot picked up a labware.

Description

VWorks software calls the `RobotPickComplete` method to notify the plugin that a robot picked up a labware.

Syntax

```
HRESULT RobotPickComplete(
   [in] BSTR sXML,
   [in, out] BSTR* sResultXML
);
```

Parameters

`sXML`	[in] An empty RobotPickComplete XML element.
`sResultXML`	[in, out] A HookResults XML block. See "HookResults XML block" on page 201.

RobotPickComplete method input

VWorks software passes an empty RobotPickComplete XML element into the `sXML` parameter of the `RobotPickComplete` method.

Example of RobotPickComplete method input

The following sample code is an empty RobotPickComplete XML element that is received by the plugin from VWorks software as a string in the `sXML` parameter of the `RobotPickComplete` method. VWorks software notifies the plugin that the robot picked up a labware.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='3c2d69b02b8e7505682cfb59c06a0105' version='1.0' >
   <RobotPickComplete />
</Velocity11>
```

RobotPickComplete method output

The plugin returns a HookResults XML block in the `sResultXML` parameter of the `RobotPickComplete` method. For information about the structure and contents of this XML block, see "HookResults XML block" on page 201.

<Go Back

Example of RobotPickComplete method output

The following sample code is a HookResults XML block that is returned to VWorks software by the plugin as a string in the `sResultXML` parameter of the `RobotPickComplete` method. The plugin tells VWorks software to write the following Info-type message to the Main Log:

```
A plate is picked up.
```

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='52226bb1b70c756162b551f1f5685a5d' version='1.0'>
   <HookResults>
      <Results>
         <HookResult ResultType='Log Message' ResultValue='A plate is picked up.' />
      </Results>
   </HookResults>
</Velocity11>
```

<Go Back

RobotPlaceComplete method

Event

A robot placed a labware.

Description

VWorks software calls the `RobotPlaceComplete` method to notify the plugin that a robot placed a labware.

Syntax

```
HRESULT RobotPlaceComplete(
   [in] BSTR sXML,
   [in, out] BSTR* sResultXML
);
```

Parameters

`sXML`	[in] An empty RobotPlaceComplete XML element.
`sResultXML`	[in, out] A HookResults XML block. See "HookResults XML block" on page 201.

RobotPlaceComplete method input

VWorks software passes an empty RobotPlaceComplete XML element into the `sXML` parameter of the `RobotPlaceComplete` method.

Example of RobotPlaceComplete method input

The following sample code is an empty RobotPlaceComplete XML element that is received by the plugin from VWorks software as a string in the `sXML` parameter of the `RobotPlaceComplete` method. VWorks software notifies the plugin that the robot placed a labware.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='a87004731e03b5c901b140e181e9742a' version='1.0' >
   <RobotPlaceComplete />
</Velocity11>
```

RobotPlaceComplete method output

The plugin returns a HookResults XML block in the `sResultXML` parameter of the `RobotPlaceComplete` method. For information about the structure and contents of this XML block, see "HookResults XML block" on page 201.

<Go Back

Example of RobotPlaceComplete method output

The following sample code is a HookResults XML block that is returned to VWorks software by the plugin as a string in the `sResultXML` parameter of the `RobotPlaceComplete` method. The plugin tells VWorks software to write the following Info-type message to the Main Log:

```
A plate is placed.
```

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='52226bb1b70c756162b551f1f5685a5d' version='1.0'>
   <HookResults>
      <Results>
         <HookResult ResultType='Log Message' ResultValue='A plate is placed.' />
      </Results>
   </HookResults>
</Velocity11>
```

<Go Back

ScriptPlateError method

Event

VWorks software received an error message for the plugin from a script associated with a labware.

Description

VWorks software calls the `ScriptPlateError` method to send an error message to the plugin from a script associated with a labware.

In JavaScript, the following function invokes the `ScriptPlateError` method:

```
plate.reportErrorToPlugin("message to plugin");
```

Syntax

```
HRESULT ScriptPlateError(
   [in] BSTR sXML,
   [in, out] BSTR* sResultXML
);
```

Parameters

`sXML`	[in] A ScriptPlateError XML element that contains information about the labware and the error message from the script.
`sResultXML`	[in, out] A HookResults XML block. See "HookResults XML block" on page 201.

ScriptPlateError method input

VWorks software passes a ScriptPlateError XML element into the `sXML` parameter of the `ScriptPlateError` method. This XML element provides information about the labware and the error message from the script.

ScriptPlateError element

The `ScriptPlateError` element has the following attributes:

Name	Value
`Message`	The error message from the script.
`PlateName`	The labware name.
`NorthSideBarcode`	The barcode if it is located on the north side of the labware, or else the value `No bar code`.

<Go Back

Name	Value
SouthSideBarcode	The barcode if it is located on the south side of the labware, or else the value No bar code.
WestSideBarcode	The barcode if it is located on the west side of the labware, or else the value No bar code.
EastSideBarcode	The barcode if it is located on the east side of the labware, or else the value No bar code.
Labware	The labware type.

Example of ScriptPlateError method input

The following sample code is a ScriptPlateError XML element that is received by the plugin from VWorks software as a string in the sXML parameter of the ScriptPlateError method. VWorks software passes the following error message, which is from a script associated with the microplate named process - 1 1:

```
This is a test error about the microplate.
```

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='d09b4626b67d2290f1197da7a7198ba2' version='1.0' >
   <ScriptPlateError EastSideBarcode='No bar code' Labware='1536 Black Greiner'
→Message='This is a test error about the microplate.' NorthSideBarcode='No bar code'
→PlateName='process - 1 1' SouthSideBarcode='No bar code'
→WestSideBarcode='No bar code' />
</Velocity11>
```

ScriptPlateError method output

The plugin returns a HookResults XML block in the sResultXML parameter of the ScriptPlateError method. For information about the structure and contents of this XML block, see "HookResults XML block" on page 201.

Example of ScriptPlateError method output

The following sample code is a HookResults XML block that is returned to VWorks software by the plugin as a string in the sResultXML parameter of the ScriptPlateError method. The plugin tells VWorks software to pause the protocol.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='52226bb1b70c756162b551f1f5685a5d' version='1.0'>
   <HookResults>
      <Results>
         <HookResult ResultType='Pause Execution' ResultValue='True' />
      </Results>
   </HookResults>
</Velocity11>
```

Related information

For information about...	See...
HookResults XML block	"HookResults XML block" on page 201

<Go Back

TaskFinished method

Event

A task is finished.

Description

VWorks software calls the `TaskFinished` method after a task is finished.

Syntax

```
HRESULT TaskFinished(
   [in] BSTR sXML,
   [in, out] BSTR* sResultXML
);
```

Parameters

`sXML`	[in] A TaskFinishing XML block containing information about the task that is finished and about the protocol that generated the event.
`sResultXML`	[in, out] A HookResults XML block. See "HookResults XML block" on page 201.

TaskFinished method input

VWorks software passes a TaskFinishing XML block into the `sXML` parameter of the `TaskFinished` method.

TaskFinishing XML block

The TaskFinishing XML block contains the `TaskFinishing` element and all its children. This XML block provides information about the task that is finished, including the names of all devices and locations used by the task.

TaskFinishing element

The `TaskFinishing` element has two children: `Devices` and `Locations`. The `TaskFinishing` element has the following attributes:

Name	Value
`Description`	The description of the task.
`InstanceNumber`	The labware instance number.
`ProcessName`	The name of the process.

<Go Back

Name	Value
ProcessIndex	If the task is in a main process, the value is the index of this process in the protocol (zero-based). If the task is in a subprocess, the value is always 0.
TaskIndex	The index of the task in the main process or in a subprocess (zero-based).
Path	The name of the protocol that generated the event. If the protocol has been saved, the value of this attribute is the protocol's file path. If the protocol has not been saved, the value is the default protocol name.
ProtocolPath	The name of the protocol that generated the event. This attribute has the same value as the Path attribute.
ProtocolType	Represents the protocol type. Possible values: 1 = Startup 2 = Main 3 = Cleanup
Spawned	Indicates whether the task is created by a Spawn Process task. Possible values: 0 = The task is not created by a Spawn Process task 1 = The task is created by a Spawn Process task

Devices element

The Devices element contains one or more Value elements.

Value element

Each Value element contains the name of a device used by the task. This element has the following attribute:

Name	Value
Value	The name of a device used by the task.

Locations element

The Locations element contains one or more Value elements.

Value element

Each Value element contains the name of a location used by the task. This element has the following attribute:

Name	Value
Value	The name of a location used by the task.

<Go Back

Example of TaskFinished method input

The following sample code is a TaskFinishing XML element that is received by the plugin from VWorks software as a string in the `sXML` parameter of the `TaskFinished` method. VWorks software tells the plugin that the task named `Place plate at PlatePad - 1 Stage` is finished in the Main Protocol.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='9f02dde81ee0060429833b8b31260c2a' version='1.0' >
   <TaskFinishing Description='Place plate at PlatePad - 1 Stage' InstanceNumber='1'
→Path='C:\V11\MyProtocol.pro' ProcessIndex='0' ProcessName='process - 1'
→ProtocolPath='C:\V11\MyProtocol.pro' ProtocolType='2' Spawned='0' TaskIndex='0' >
      <Devices >
         <Value Value='PlatePad - 1' />
      </Devices>
      <Locations >
         <Value Value='Stage' />
      </Locations>
   </TaskFinishing>
</Velocity11>
```

TaskFinished method output

The plugin returns a HookResults XML block in the `sResultXML` parameter of the `TaskFinished` method. For information about the structure and contents of this XML block, see "HookResults XML block" on page 201.

Example of TaskFinished method output

The following sample code is a HookResults XML block that is returned to VWorks software by the plugin as a string in the `sResultXML` parameter of the `TaskFinished` method. The plugin tells VWorks software to write the following Info-type message to the Main Log:

```
Place plate at PlatePad - 1 Stage task is finished.
```

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='52226bb1b70c756162b551f1f5685a5d' version='1.0'>
   <HookResults>
      <Results>
         <HookResult ResultType='Log Message'
→ResultValue='Place plate at PlatePad - 1 Stage task is finished.' />
      </Results>
   </HookResults>
</Velocity11>
```

Related information

For information about...	See...
HookResults XML block	"HookResults XML block" on page 201

<Go Back

TaskStarting method

Event

A task started.

Description

VWorks software calls the `TaskStarting` method after a task starts.

Syntax

```
HRESULT TaskStarting(
   [in] BSTR sXML,
   [in, out] BSTR* sResultXML
);
```

Parameters

`sXML`	[in] A TaskStarting XML block containing information about the task that started and about the protocol that generated the event.
`sResultXML`	[in, out] A HookResults XML block. See "HookResults XML block" on page 201.

TaskStarting method input

VWorks software passes a TaskStarting XML block into the `sXML` parameter of the `TaskStarting` method.

TaskStarting XML block

The TaskStarting XML block contains the `TaskStarting` element and all its children. This XML block provides information about the task that started, including the names of all devices and locations used by the task.

TaskStarting element

The `TaskStarting` element has two children: `Devices` and `Locations`. The `TaskStarting` element has the following attributes:

Name	Value
`Description`	The description of the task.
`InstanceNumber`	The labware instance number.
`ProcessName`	The name of the process.

<Go Back

Name	Value
ProcessIndex	If the task is in a main process, the value is the index of this process in the protocol (zero-based). If the task is in a subprocess, the value is always 0.
TaskIndex	The index of the task in the main process or in a subprocess (zero-based).
Path	The name of the protocol that generated the event. If the protocol has been saved, the value of this attribute is the protocol's file path. If the protocol has not been saved, the value is the default protocol name.
ProtocolPath	The name of the protocol that generated the event. This attribute has the same value as the Path attribute.
ProtocolType	Represents the protocol type. Possible values: 1 = Startup 2 = Main 3 = Cleanup
Spawned	Indicates whether the task is created by a Spawn Process task. Possible values: 0 = The task is not created by a Spawn Process task 1 = The task is created by a Spawn Process task

Devices element

The Devices element contains one or more Value elements.

Value element

Each Value element contains the name of a device used by the task. This element has the following attribute:

Name	Value
Value	The name of a device used by the task.

Locations element

The Locations element contains one or more Value elements.

Value element

Each Value element contains the name of a location used by the task. This element has the following attribute:

Name	Value
Value	The name of a location used by the task.

<Go Back

Example of TaskStarting method input

The following sample code is a TaskStarting XML element that is received by the plugin from VWorks software as a string in the `sXML` parameter of the `TaskStarting` method. VWorks software tells the plugin that the task named `Place plate at PlatePad - 1 Stage` started in the Main Protocol.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='644ee98f1252ed195132158a86ecbb5c' version='1.0' >
   <TaskStarting Description='Place plate at PlatePad - 1 Stage' InstanceNumber='1'
→Path='C:\V11\MyProtocol.pro' ProcessIndex='0' ProcessName='process - 1'
→ProtocolPath='C:\V11\MyProtocol.pro' ProtocolType='2' Spawned='0' TaskIndex='0' >
      <Devices >
         <Value Value='PlatePad - 1' />
      </Devices>
      <Locations >
         <Value Value='Stage' />
      </Locations>
   </TaskStarting>
</Velocity11>
```

TaskStarting method output

The plugin returns a HookResults XML block in the `sResultXML` parameter of the `TaskStarting` method. For information about the structure and contents of this XML block, see "HookResults XML block" on page 201.

Example of TaskStarting method output

The following sample code is a HookResults XML block that is returned to VWorks software by the plugin as a string in the `sResultXML` parameter of the `TaskStarting` method. The plugin tells VWorks software to write the following Info-type message to the Main Log:

```
Place plate at PlatePad - 1 Stage task started.
```

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='52226bb1b70c756162b551f1f5685a5d' version='1.0'>
   <HookResults>
      <Results>
         <HookResult ResultType='Log Message'
→ResultValue='Place plate at PlatePad - 1 Stage task started.' />
      </Results>
   </HookResults>
</Velocity11>
```

Related information

For information about...	See...
HookResults XML block	"HookResults XML block" on page 201

<Go Back

UserLoggedIn method

Event

A user logged in to VWorks software.

Description

VWorks software calls the `UserLoggedIn` method after a user logs in to VWorks software.

Syntax

```
HRESULT UserLoggedIn(
   [in] BSTR sXML,
   [in, out] BSTR* sResultXML
);
```

Parameters

`sXML`	[in] A LoginComplete XML element that contains the name and security level of the user who logged in to VWorks software.
`sResultXML`	[in, out] A HookResults XML block. See "HookResults XML block" on page 201.

UserLoggedIn method input

VWorks software passes a LoginComplete XML element into the `sXML` parameter of the `UserLoggedIn` method. This XML element provides the name and security level of the user who logged in to VWorks software.

LoginComplete element

The `LoginComplete` element has the following attributes:

Name	Value
`AccessLevel`	Represents the security level (access privilege) for the user currently logged in to VWorks software. Possible values: `0` = Administrator `1` = Technician `2` = Operator `3` = Guest
`UserName`	The name of the user.

<Go Back

Example of LoginComplete method input

The following sample code is a LoginComplete XML element that is received by the plugin from VWorks software as a string in the `sXML` parameter of the `UserLoggedIn` method. VWorks software tells the plugin that the user named `LSinclair` logged in to VWorks software as Administrator.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='d2cbcda7e389f50d52d31e1bfcd19660' version='1.0' >
   <LoginComplete AccessLevel='0' UserName='LSinclair' />
</Velocity11>
```

UserLoggedIn method output

The plugin returns a HookResults XML block in the `sResultXML` parameter of the `UserLoggedIn` method. For information about the structure and contents of this XML block, see "HookResults XML block" on page 201.

Example of UserLoggedIn method output

The following sample code is a HookResults XML block that is returned to VWorks software by the plugin as a string in the `sResultXML` parameter of the `UserLoggedIn` method. The plugin tells VWorks software to write the following Info-type message to the Main Log:

```
LSinclair logged in as Administrator.
```

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='52226bb1b70c756162b551f1f5685a5d' version='1.0'>
   <HookResults>
      <Results>
         <HookResult ResultType='Log Message'
→ResultValue='LSinclair logged in as Administrator.' />
      </Results>
   </HookResults>
</Velocity11>
```

Related information

For information about...	See...
HookResults XML block	"HookResults XML block" on page 201

<Go Back

UserLoggedOut method

Event

A user logged out of VWorks software.

Description

VWorks software calls the `UserLoggedOut` method after a user logs out of VWorks software.

Syntax

```
HRESULT UserLoggedOut(
   [in] BSTR sXML,
   [in, out] BSTR* sResultXML
);
```

Parameters

`sXML`	[in] A LogoutComplete XML element that contains the name of the user who logged out of VWorks software.
`sResultXML`	[in, out] A HookResults XML block. See "HookResults XML block" on page 201.

UserLoggedOut method input

VWorks software passes a LogoutComplete XML element into the `sXML` parameter of the `UserLoggedOut` method. This XML element provides the name of the user who logged out of VWorks software.

LogoutComplete element

The `LogoutComplete` element has the following attribute:

Name	Value
`UserName`	The name of the user.

<Go Back

Example of UserLoggedOut method input

The following sample code is a LogoutComplete XML element that is received from VWorks software by the plugin as a string in the `sXML` parameter of the `UserLoggedOut` method. VWorks software tells the plugin that the user named `LSinclair` logged out of VWorks software.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='07d1a5a4cb08ef463f3213a90a7261e4' version='1.0' >
   <LogoutComplete UserName='LSinclair' />
</Velocity11>
```

UserLoggedOut method output

The plugin returns a HookResults XML block in the `sResultXML` parameter of the `UserLoggedOut` method. For information about the structure and contents of this XML block, see "HookResults XML block" on page 201.

Example of UserLoggedOut method output

The following sample code is a HookResults XML block that is returned to VWorks software by the plugin as a string in the `sResultXML` parameter of the `UserLoggedOut` method. The plugin tells VWorks software to write the following Info-type message to the Main Log:

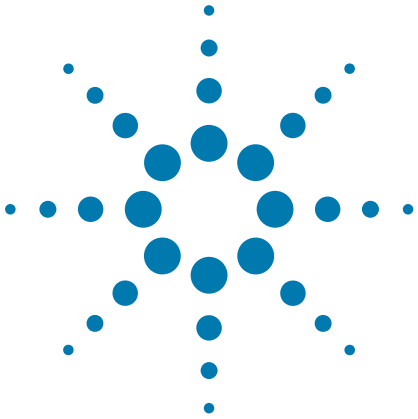
```
LSinclair logged out.
```

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='52226bb1b70c756162b551f1f5685a5d' version='1.0'>
   <HookResults>
      <Results>
         <HookResult ResultType='Log Message' ResultValue='LSinclair logged out.' />
      </Results>
   </HookResults>
</Velocity11>
```

Related information

For information about...	See...
HookResults XML block	"HookResults XML block" on page 201

<Go Back



17 IWorksAsyncDriver interface

VWorks plugins that perform asynchronous tasks must implement the IWorksAsyncDriver interface.

This chapter defines the IWorksAsyncDriver methods.

IMPORTANT All VWorks device driver plugins must implement the IWorksDriver, IControllerClient, and IWorksDiags interfaces.

This chapter contains the following topics:

- "IWorksAsyncDriver methods overview" on page 264
- "Abort method" on page 265
- "GetListOfAsyncTasks method" on page 269
- "Ignore method" on page 271
- "Retry method" on page 274
- "Asynchronous Task Command XML block components" on page 277

<Go Back

IWorksAsyncDriver methods overview

Use the following table to quickly locate an IWorksAsyncDriver method by name, by description, or by page number.

Method	Description	See...
`Abort`	Tells the plugin to terminate an asynchronous task or to terminate all currently executing asynchronous tasks.	"Abort method" on page 265
`GetErrorInfo`	*Deprecated.* This method should be implemented as `return E_NOTIMPL (0x80004001)`. Instead of this method, VWorks software calls the IWorksDriver `GetErrorInfo` method when the plugin reports an error that occurred during a task. See IWorksDriver "GetErrorInfo method" on page 40.	
`GetListOfAsyncTasks`	Gets the list of the currently executing asynchronous tasks from the plugin.	"GetListOfAsyncTasks method" on page 269
`Ignore`	Tells the plugin to ignore an error that occurred during an asynchronous task.	"Ignore method" on page 271
`Retry`	Tells the plugin to retry an asynchronous task.	"Retry method" on page 274

<Go Back

Abort method

Description

VWorks software calls the `Abort` method to terminate a specific asynchronous task or to terminate all currently executing asynchronous tasks.

To terminate a specific asynchronous task

1. The plugin calls the ErrorAbortRetryIgnoreNonBlocking update to notify VWorks software that an error occurred and to provide a literal string that describes the error.
2. VWorks software does the following:
   a. Writes the string to the Main Log.
   b. Displays the standard error dialog box, which includes the following components:
      - The error string
      - The Abort, Ignore and Continue..., Retry, and Diagnostics buttons

      The figure on page 40 shows a standard error dialog box.
3. If the user clicks the Abort button, VWorks software calls the `Abort` method to tell the plugin to terminate the specified asynchronous task.

To terminate all currently executing asynchronous tasks

When the user aborts a run in the Runset Manager, VWorks software calls the `Abort` method to tell the plugin to terminate all currently executing asynchronous tasks. (All protocols in the run contain asynchronous tasks.)

Note: This is not an emergency stop.

Syntax

```
HRESULT Abort(
   [in] BSTR AsyncXML
);
```

Parameters

`AsyncXML`	[in] An Asynchronous Task Command XML block that describes the asynchronous task to be terminated or that contains an `Async_TaskID` parameter whose `Value` attribute is `0`, which means to terminate all currently executing asynchronous tasks.

Abort method input (terminate a specific asynchronous task)

VWorks software passes an Asynchronous Task Command XML block into the `AsyncXML` parameter of the `Abort` method.

<Go Back

Asynchronous Task Command XML block

The Asynchronous Task Command XML block contains the `Command` element and all its children. This XML block describes the asynchronous task to be terminated.

XML structure

The value of the `file` attribute for the `Velocity11` element is `MetaData`. See "Velocity11 element" on page 416.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Command>
      <Parameters>
         <Parameter>
            <Ranges>
               <Range />
               ...
            </Ranges>
         </Parameter>
         ...
      </Parameters>
      <Locations>
         <Value />
         ...
      </Locations>
      <AsyncParameters>
         <AsyncParameter Name='Async_TaskVWorksID' ... />
         <AsyncParameter Name='Async_ErrorDescription' ... />
         <AsyncParameter Name='Async_TaskID' ... />
         <AsyncParameter Name='Async_Location' ... />
      </AsyncParameters>
   </Command>
</Velocity11>
```

XML elements and attributes

See "Asynchronous Task Command XML block components" on page 277.

<Go Back

Example of Abort method input (terminate a specific asynchronous task)

The following code shows a truncated Asynchronous Task Command XML block that is received by the plugin from VWorks software as a string in the `AsyncXML` parameter of the `Abort` method. VWorks software tells the plugin to terminate the asynchronous task named `Shake`, whose plugin-generated asynchronous task ID is `1`.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='2a2e64e9a86ec43f2b2d8058a558f7ab' version='1.0'>
   <Command Compiler='0' Description='Shake plate' Editor='4' Name='Shake'
→NextTaskToExecute='1' ProtocolName='Protocol File - 1.pro' RequiresRefresh='0'
→TaskRequiresLocation='1' VisibleAvailability='1'>
      <Parameters>
         <Parameter Name='Location' Scriptable='1' Style='0' Type='5' Value='9'>
            <Ranges>
               <Range Value='<auto-select>' />
               <Range Value='1' />
               <Range Value='2' />
               ...
               <Range Value='8' />
               <Range Value='9' />
            </Ranges>
         </Parameter>
         <Parameter Description='Mode to operate in' Name='Mode' Scriptable='1'
→Style='0' Type='2' Value='Timed'>
            <Ranges>
               <Range Value='On' />
               <Range Value='Off' />
               <Range Value='Timed' />
            </Ranges>
         </Parameter>
         <Parameter Description='Revolutions per minute' Name='RPM' Scriptable='1'
→Style='0' Type='8' Value='500'>
            <Ranges>
               <Range Value='100' />
               <Range Value='2000' />
            </Ranges>
         </Parameter>
         ...
         <Parameter Description='Amount of time to shake'
→Hide_if='Variable(Mode) != Const('Timed')' Name='Time for operation in Timed mode'
→Scriptable='1' Style='0' Type='8' Units='s' Value='10' />
      </Parameters>
      <Locations>
         <Value Value='9' />
      </Locations>
      <AsyncParameters>
         <AsyncParameter Name='Async_TaskVWorksID' Value='32.13.1' />
         <AsyncParameter Name='Async_ErrorDescription'
→Value='No response received from the pipette controller. It is recommended that you
→click Retry. Ignoring this error may result in an unpredictable move.' />
         <AsyncParameter Name='Async_TaskID' Value='1' />
         <AsyncParameter Name='Async_Location' Value='9' />
      </AsyncParameters>
   </Command>
</Velocity11>
```

Abort method input (terminate all currently executing asynchronous tasks)

VWorks software passes an Asynchronous Task Command XML block into the `AsyncXML` parameter of the `Abort` method.

<Go Back

Asynchronous Task Command XML block

The Asynchronous Task Command XML block contains the `Command` element and one `AsyncParameters` element. The `AsyncParameters` element contains one `AsyncParameter` element, which contains an `Async_TaskID` parameter whose `Value` attribute is `0`.

XML structure

The value of the `file` attribute for the `Velocity11` element is `MetaData`. See "Velocity11 element" on page 416.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Command>
      <AsyncParameters>
         <AsyncParameter Name='Async_TaskID' ... Value='0' />
      </AsyncParameters>
   </Command>
</Velocity11>
```

XML elements and attributes

See "Asynchronous Task Command XML block components" on page 277.

Example of Abort method input (terminate all currently executing asynchronous tasks)

The following code is an Asynchronous Task Command XML block that is received by the plugin from VWorks software as a string in the `AsyncXML` parameter of the `Abort` method. VWorks software tells the plugin to terminate all currently executing asynchronous tasks (the `Async_TaskID` parameter's `Value` element is `0`).

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='db16464f899e42aca9ed0f92b2a5b79e' version='1.0' >
   <Command Compiler='0' Editor='0' NextTaskToExecute='1' RequiresRefresh='0'
TaskRequiresLocation='1' VisibleAvailability='1' >
      <AsyncParameters >
         <AsyncParameter Name='Async_TaskID' Value='0' />
      </AsyncParameters>
   </Command>
</Velocity11>
```

Related information

For information about...	See...
GetListOfAsyncTasks method	"GetListOfAsyncTasks method" on page 269
Ignore method (IWorksAsyncDriver interface)	"Ignore method" on page 271
Retry method (IWorksAsyncDriver interface)	"Retry method" on page 274

<Go Back

GetListOfAsyncTasks method

Description

VWorks software calls the GetListOfAsyncTasks method to get the list of the currently executing asynchronous tasks from the plugin. The list identifies the tasks in the Name (task ID) and Value (location name) attributes of each Parameter element. VWorks software uses this list to keep track of which locations are busy and which are available.

Syntax

```
HRESULT GetListOfAsyncTasks(
   [out,retval] BSTR *AsyncTaskIdLocationXML)
);
```

Parameters

AsyncTaskIdLocationXML	[out, retval] An AsyncTaskList XML block that contains the list of the currently executing asynchronous tasks.

GetListOfAsyncTasks method output

The plugin returns an AsyncTaskList XML block in the AsyncTaskIdLocationXML parameter of the GetListOfAsyncTasks method.

AsyncTaskList XML block

The AsyncTaskList XML block contains the AsyncTaskList element and all its children. This XML block provides the list of asynchronous tasks that are currently executing. The tasks are identified in the Name (task ID) and Value (location name) attributes of each Parameter element.

XML structure

The value of the file attribute for the Velocity11 element is MetaData. See "Velocity11 element" on page 416.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <AsyncTaskList>
      <Parameters>
         <Parameter />
         ...
      </Parameters>
   </AsyncTaskList>
</Velocity11>
```

AsyncTaskList element

The AsyncTaskList element contains one Parameters element.

<Go Back

Parameters element

The `Parameters` element contains one or more `Parameter` elements.

Parameter element

Each `Parameter` element contains the task ID and location name of a currently executing asynchronous task. This element has the following attributes:

:

Name	Value
`Name`	The plugin-generated task ID. Possible values range from `1` to `2147483647` (`INT_MAX`). Required: Yes
`Value`	The name of the location. Required: Yes

Example of GetListOfAsyncTasks method output

The following sample code is an AsyncTaskList XML block that is returned to VWorks software by the plugin as a string in the `AsyncTaskIdLocationXML` parameter of the `GetListOfAsyncTasks` method. The plugin tells VWorks software that the asynchronous tasks named `3`, `11`, and `21` are currently executing.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='e388d6bf4e4b03fefd07ed52c1537a6a' version='1.0' >
   <AsyncTaskList >
      <Parameters >
         <Parameter Name='3' Value='Location1' />
         <Parameter Name='11' Value='Location2' />
         <Parameter Name='21' Value='Location3' />
      </Parameters>
   </AsyncTaskList>
</Velocity11>
```

Related information

For information about...	See...
Abort method (IWorksAsyncDriver interface)	“Abort method” on page 265
Ignore method (IWorksAsyncDriver interface)	“Ignore method” on page 271
Retry method (IWorksAsyncDriver interface)	“Retry method” on page 274

<Go Back

Ignore method

Description

VWorks software calls the `Ignore` method as follows:

1. The plugin calls the ErrorAbortRetryIgnoreNonBlocking update to notify VWorks software that an error occurred and to provide a literal string that describes the error.
2. VWorks software does the following:
   a. Writes the string to the Main Log.
   b. Displays the standard error dialog box, which includes the following components:
      - The error string
      - The Abort, Ignore and Continue..., Retry, and Diagnostics buttons

      The figure on page 40 shows a standard error dialog box.
3. If the user clicks the Ignore and Continue... button, VWorks software calls the `Ignore` method to tell the plugin to ignore the error.

   The plugin should continue the task, if possible, or exit the task if continuing would be dangerous or impossible. A call to the `Ignore` method should not cause an unsafe condition, which could result in a new error or a premature completion of the task.

   If all errors encountered during an asynchronous task are ignored, the system should be able to continue as if the failing step was skipped.

Syntax

```
HRESULT Ignore(
   [in] BSTR AsyncXML,
   [out,retval] enum ReturnCode *retVal
);
```

Parameters

`AsyncXML`	[in] An Asynchronous Task Command XML block that identifies the asynchronous task and describes the error that occurred.
`retVal`	[out, retval] Returns an error code. For more information, see "ReturnCode enumerated type" on page 384. Possible values: `0` = The request was completed (`RETURN_SUCCESS`) `1` = Something was wrong with the input, so the request was not completed (`RETURN_BAD_ARGS`) `2` = The request was not completed (`RETURN_FAIL`)

<Go Back

Ignore method input

VWorks software passes an Asynchronous Task Command XML block into the `AsyncXML` parameter of the `Ignore` method.

Asynchronous Task Command XML block

The Asynchronous Task Command XML block contains the `Command` element and an `AsyncParameters` child element and all its children. This XML block identifies the asynchronous task to be ignored and describes the error that occurred.

XML structure

The value of the `file` attribute for the `Velocity11` element is `MetaData`. See "Velocity11 element" on page 416.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Command>
      <AsyncParameters>
         <AsyncParameter Name='Async_ErrorDescription' ... />
         <AsyncParameter Name='Async_TaskID' ... />
         <AsyncParameter Name='Async_TaskVWorksID' ... />
         <AsyncParameter Name='Async_Location' ... />
      </AsyncParameters>
   </Command>
</Velocity11>
```

XML element and attributes

See "Asynchronous Task Command XML block components" on page 277.

Example of Ignore method input

The following code is an Asynchronous Task Command XML block received by the plugin from VWorks software as a string in the `AsyncXML` parameter of the `Ignore` method. VWorks software tells the plugin to ignore the error that occurred at the location named `Location` during the asynchronous task named `MakeLocationAvailable`, whose plugin-generated task ID is `1`.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='89626582b55d53867fc461da1435da2d' version='1.0' >
   <Command Compiler='0' Editor='0' Name='MakeLocationAvailable'
NextTaskToExecute='1' RequiresRefresh='0' TaskRequiresLocation='1'
→VisibleAvailability='1' Description='MakeLocationAvailable'
→ProtocolName='Protocol File - 1.pro'>
      <AsyncParameters >
         <AsyncParameter Name='Async_ErrorDescription'
→Value='No response received from the pipette controller. It is recommended that you
→click Retry.  Ignoring this error may result in an unpredictable move.' />
         <AsyncParameter Name='Async_TaskID' Value='1' />
         <AsyncParameter Name='Async_TaskVWorksID' />
         <AsyncParameter Name='Async_Location' Value='Location' />
      </AsyncParameters>
   </Command>
</Velocity11>
```

<Go Back

Related information

For information about...	See...
Abort method (IWorksAsyncDriver interface)	“Abort method” on page 265
GetListOfAsyncTasks method	“GetListOfAsyncTasks method” on page 269
Retry method (IWorksAsyncDriver interface)	“Retry method” on page 274

<Go Back

Retry method

Description

VWorks software calls the `Retry` method as follows:

1. The plugin calls the ErrorAbortRetryIgnoreNonBlocking update to notify VWorks software that an error occurred and to provide a literal string that describes the error.
2. VWorks software does the following:
   a. Writes the string to the Main Log.
   b. Displays the standard error dialog box, which includes the following components:
      - The error string
      - The Abort, Ignore and Continue..., Retry, and Diagnostics buttons

      The figure on page 40 shows a standard error dialog box.
3. If the user clicks the Retry button, VWorks software calls the `Retry` method to tell the plugin to retry the asynchronous task.

   Because VWorks software assumes the user manually solved the problem that caused the error, the plugin should try to restart the task. The plugin should record the state of the task and retry from the point in the task that makes the most sense given the current state.

   For example, a single-column dispenser that encounters an error after partially filling a labware should not start over, because the dispenser might deliver too much reagent to the already-covered wells. The dispenser should continue as close to the point of interruption as possible to avoid over-dispensing or under-dispensing the wells that were being filled at the time the error occurred.

Syntax

```
HRESULT Retry(
   [in] BSTR AsyncXML,
   [out,retval] enum ReturnCode *retVal
);
```

Parameters

`AsyncXML`	[in] An Asynchronous Task Command XML block that identifies the asynchronous task and describes the error that occurred.

<Go Back

retVal	[out, retval] Returns an error code. For more information, see "ReturnCode enumerated type" on page 384. Possible values: 0 = The request was completed (RETURN_SUCCESS) 1 = Something was wrong with the input, so the request was not completed (RETURN_BAD_ARGS) 2 = The request was not completed (RETURN_FAIL)

Retry method input

VWorks software passes an Asynchronous Task Command XML block into the AsyncXML parameter of the Retry method.

Asynchronous Task Command XML block

The Asynchronous Task Command XML block contains the Command element and an AsyncParameters child element and all its children. This XML block identifies the asynchronous task to be retried and the error that occurred.

XML structure

The value of the file attribute for the Velocity11 element is MetaData. See "Velocity11 element" on page 416.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Command>
      <AsyncParameters>
         <AsyncParameter Name='Async_ErrorDescription' ... />
         <AsyncParameter Name='Async_TaskID' ... />
         <AsyncParameter Name='Async_TaskVWorksID' ... />
         <AsyncParameter Name='Async_Location' ... />
      </AsyncParameters>
   </Command>
</Velocity11>
```

XML elements and attributes

See "Asynchronous Task Command XML block components" on page 277.

<Go Back

Example of Retry method input

The following code is an Asynchronous Task Command XML block received by the plugin from VWorks software as a string in the `AsyncXML` parameter of the `Ignore` method. VWorks software tells the plugin to retry the asynchronous task named `MakeLocationAvailable`, whose plugin-generated task ID is `1`, after an error occurs at the location named `Location`.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='89626582b55d53867fc461da1435da2d' version='1.0' >
   <Command Compiler='0' Editor='0' Name='MakeLocationAvailable'
→NextTaskToExecute='1' RequiresRefresh='0' TaskRequiresLocation='1'
→VisibleAvailability='1' Description='MakeLocationAvailable'
→ProtocolName='Protocol File - 1.pro'>
      <AsyncParameters >
         <AsyncParameter Name='Async_ErrorDescription'
→Value='No response received from the pipette controller. It is recommended that you
→click Retry.  Ignoring this error may result in an unpredictable move.' />
         <AsyncParameter Name='Async_TaskID' Value='1' />
         <AsyncParameter Name='Async_TaskVWorksID' />
         <AsyncParameter Name='Async_Location' Value='Location' />
      </AsyncParameters>
   </Command>
</Velocity11>
```

Related information

For information about...	See...
Abort method (IWorksAsyncDriver interface)	“Abort method” on page 265
GetListOfAsyncTasks method	“GetListOfAsyncTasks method” on page 269
Ignore method (IWorksAsyncDriver interface)	“Ignore method” on page 271

<Go Back

Asynchronous Task Command XML block components

VWorks software passes an Asynchronous Task Command XML block into the `AsyncXML` parameter of the `Abort`, `Ignore`, and `Retry` methods. The following table lists the XML components that are contained in this XML block for each method. The elements and attributes are described in the sections that follow. You can click an element name to jump to the appropriate section.

The elements and attributes preceded by an asterisk are of interest to an implementer of the IWorksAsyncDriver interface.

	Method name			
Element name Attribute name	Abort—terminate a specific task	Abort—terminate all executing tasks	Ignore	Retry
Command	✓	✓	✓	✓
Compiler	✓	✓	✓	✓
Editor	✓	✓	✓	✓
Description	✓		✓	✓
*Name	✓		✓	✓
NextTaskToExecute	✓	✓	✓	✓
ProtocolName	✓		✓	✓
RequiresRefresh	✓	✓	✓	✓
TaskRequiresLocation	✓	✓	✓	✓
VisibleAvailability	✓	✓	✓	✓
Parameters	✓			
Parameter	✓ (multiple)			
Description	✓			
Hide_if	✓			
*Name	✓			
Scriptable	✓			
Style	✓			
Type	✓			
Units	✓			
*Value	✓			
Ranges	✓			
Range	✓			
Value	✓			

<Go Back

Element name Attribute name	Method name Abort—terminate a specific task	Abort—terminate all executing tasks	Ignore	Retry
Locations	✓			
*Value	✓			
AsyncParameters	✓	✓	✓	✓
AsyncParameter	✓	✓	✓	✓
*Name	✓		✓	✓
Async_ErrorDescription				
Async_TaskID		✓	✓	✓
Async_TaskVWorksID			✓	✓
Async_Location			✓	✓
*Value	✓			

Command element

The `Command` element contains one `Parameters` element, one `Locations` element, and one `AsyncParameters` element. The `Command` element has the following attributes.

Name	Value
`Compiler`	See "Compiler attribute" on page 399.
`Editor`	See "Editor attribute" on page 400.
`*Name`	The name of the asynchronous task.
`NextTaskToExecute`	See "NextTaskToExecute attribute" on page 400.
`RequiresRefresh`	See "RequiresRefresh attribute" on page 402.
`TaskRequiresLocation`	See "TaskRequiresLocation attribute" on page 402.
`VisibleAvailability`	See "VisibleAvailability attribute" on page 403.
`Description`	The description of the asynchronous task.
`ProtocolName`	The name of the protocol that contains the asynchronous task. If the protocol has been saved, the value of this attribute is the protocol's file path. If the protocol has not been saved, the value is the default protocol name.

<Go Back

Parameters element

The `Parameters` element contains one or more `Parameter` elements.

Parameter element

The `Parameter` element can contain one `Ranges` element and has the following attributes:

Name	Value
`Description`	The description of the parameter.
`Hide_if`	See "Hide_if attribute" on page 409.
`*Name`	The name of the parameter.
`Scriptable`	See "Scriptable attribute" on page 410.
`Style`	See "Style attribute" on page 411.
`Type`	See "Type attribute" on page 411.
`Units`	See "Units attribute" on page 412.
`*Value`	See "Value attribute" on page 413.

Ranges element

The `Ranges` element contains one or more `Range` elements.

Range element

The `Range` element has the following attribute:

Name	Value
`Value`	See "Value attribute" on page 415.

Locations element

The `Locations` element contains one of more `Value` elements.

Value element

The `Value` element contains the following attribute:

Name	Value
`Value`	The name of a location used by the asynchronous task.

<Go Back

AsyncParameters element

The `AsyncParameters` element contains five `AsyncParameter` elements.

AsyncParameter element

Each `AsyncParameter` element has one of the following pairs of `Name` and `Value` attributes:

Name	Value
`Name`	The value `Async_TaskVWorksID`. Required: Yes
`Value`	The task ID of the asynchronous task, which is automatically generated by VWorks software and used by VWorks software to identify and manage all asynchronous tasks from all plugins. Required: Yes *Note:* The `AsyncParameters` element contains two `Async_TaskVWorksID` elements.
`Name`	The value `Async_ErrorDescription`. Required: Yes
`Value`	The description of the error that occurred. Required: Yes
`Name`	The value `Async_TaskID`. Required: Yes
`Value`	The task ID of the asynchronous task, which is generated by the plugin and used by the plugin to identify and manage its asynchronous tasks. Required: Yes
`Name`	The value `Async_Location`. Required: Yes
`Value`	The name of the location where the error occurred. Required: Yes

18 IWorksController interface

IMPORTANT To call IWorksController methods, plugins must first implement the IControllerClient interface. See "IControllerClient interface" on page 87.

VWorks software implements the IWorksController interface. Plugins must call IWorksController methods to do the following:

- Notify VWorks software about user activities
- Write messages to the Main Log
- Communicate with another plugin using VWorks software as the intermediary
- Tell VWorks software to perform certain actions
- Request information from VWorks software
- Return information to VWorks software

This chapter defines the IWorksController interface methods.

IMPORTANT All VWorks device driver plugins must implement the IWorksDriver, IControllerClient, and IWorksDiags interfaces.

This chapter contains the following topics:

- "NotifyDataChanged method" on page 285
- "NotifyTipOperation method" on page 287
- "OnCloseDiagsDialog method" on page 290
- "PrintToLog method" on page 291
- "Query method" on page 292
  - "AllDeviceInfo query/response" on page 296
  - "Barcode query/response" on page 300
  - "DeviceLocationTeachpoints query/response" on page 302
  - "GetDeviceName query/response" on page 304
  - "GetIOManagerPointInput query/response" on page 305
  - "GetJavascriptVariable query/response" on page 307
  - "GetProductInfo query/response" on page 312
  - "GetRunSetStatus query/response" on page 313
  - "InterPlugin query/response" on page 319
  - "Labware query/response" on page 322
  - "LocationInformation query/response" on page 327
  - "LocationToTeachpoints query/response" on page 330
  - "PlateVolume query/response" on page 333
  - "ScanBarcode query/response" on page 337
  - "SystemPlateInformation query/response" on page 340

<Go Back

  - "TeachpointInformation query/response" on page 342
- "Update method" on page 347
  - "AsyncTaskFinished update" on page 349
  - "AsyncTaskStarted update" on page 350
  - "Barcode update" on page 350
  - "ErrorAbortRetryIgnoreNonBlocking update" on page 352
  - "InventoryPlateBarcodes update" on page 358
  - "LiquidTransferComplete update" on page 360
  - "RackInfo update" on page 362
  - "RunScript update" on page 363
  - "RunsetAdd update" on page 365
  - "SetIOManagerPointDigitalOutput update" on page 367
  - "Volume update" on page 368

<Go Back

IWorksController methods overview

Use the following table to quickly locate an IWorksController method by name, by description, or by page number.

Method	Description
"NotifyDataChanged method" on page 285	Notifies VWorks software that the user changed a property in the diagnostics dialog box.
"NotifyTipOperation method" on page 287	Notifies VWorks software that the user performed a pipette-tip task using the diagnostics dialog box for a device that uses pipette tips.
"OnCloseDiagsDialog method" on page 290	Notifies VWorks software that the plugin's diagnostics dialog box was closed.
"PrintToLog method" on page 291	Tells VWorks software to print the specified string to the Main Log.
"Query method" on page 292	Requests information from VWorks software or requests information from another plugin, using VWorks software as the intermediary.
Query method categories	
"AllDeviceInfo query/response" on page 296	Gets all device names and device types in the device file.
"Barcode query/response" on page 300	Gets all barcodes on the labware at the specified location.
"DeviceLocationTeachpoints query/response" on page 302	Gets all locations on all devices for which the robot has teachpoints.
"GetDeviceName query/response" on page 304	Gets the device name for the plugin in VWorks software.
"GetIOManagerPointInput query/response" on page 305	Gets the value of the specified digital input signal.
"GetJavascriptVariable query/response" on page 307	Gets the value of the specified JavaScript variable.
"GetProductInfo query/response" on page 312	Gets the name and version of VWorks software that is currently running.
"GetRunSetStatus query/response" on page 313	Gets information about all the protocol runs listed in the Runset Manager.
"InterPlugin query/response" on page 319	Gets information from a destination plugin, using VWorks software as the intermediary. *Note:* This query is only used for interplugin communication.
"Labware query/response" on page 322	Gets the labware entry for the specified labware type.
"LocationInformation query/response" on page 327	Gets the name of the static labware at the specified location. If the location is a stack location, gets the allowable stack height.

<Go Back

Method	Description
"LocationToTeachpoints query/response" on page 330	Gets all teachpoints that have been set at the specified location.
"PlateVolume query/response" on page 333	Gets the current volume of all the wells in the labware at the specified location.
"ScanBarcode query/response" on page 337	Determines whether a barcode scanner should be used to scan the barcode on the specified side of the labware at the specified location.
"SystemPlateInformation query/response" on page 340	Gets the labware type for the specified labware.
"TeachpointInformation query/response" on page 342	Gets the coordinates of the specified teachpoint for the specified robot from the destination plugin. *Note:* The source plugin can also get this information using the InterPlugin query.
"Update method" on page 347	Returns information to VWorks software, tells VWorks software to perform certain actions, or both.
Update method categories	
"AsyncTaskFinished update" on page 349	Tells VWorks software that an asynchronous task is completed.
"AsyncTaskStarted update" on page 350	Tells VWorks software that an asynchronous task is started.
"Barcode update" on page 350	Tells VWorks software to update the barcode on the specified side of the labware at the specified location.
"ErrorAbortRetryIgnoreNonBlocking update" on page 352	When an error occurs during an asynchronous task, tells VWorks software to trigger task error handling.
"InventoryPlateBarcodes update" on page 358	Returns the results of the labware inventory, which was requested by VWorks software with a call to the IStorageDriver `QueryStorageLocations` method. See "QueryStorageLocations method" on page 192.
"LiquidTransferComplete update" on page 360	Tells VWorks software that a liquid transfer is completed.
"RackInfo update" on page 362	Tells VWorks software how many mismatches occurred during a rack check.
"RunScript update" on page 363	Tells VWorks software to execute an arbitrary JavaScript script.
"RunsetAdd update" on page 365	Tells VWorks software to schedule the specified protocol run in the Runset Manager.
"SetIOManagerPointDigitalOutput update" on page 367	Tells VWorks software to output the specified value on the specified digital output channel.
"Volume update" on page 368	Returns the volume change of one or more wells in the labware at the specified location to VWorks software.

<Go Back

NotifyDataChanged method

Description

The plugin calls the `NotifyDataChanged` method to tell VWorks software that the user changed a property in the plugin's diagnostics dialog box. VWorks software can perform operations on this information, such as writing a message to the Main Log.

Syntax

```
HRESULT NotifyDataChanged(
   [in] IControllerClient *Source,
   [in] BSTR ObjectDataChanged
);
```

Parameters

`Source`	[in] The plugin's pointer to itself.
`ObjectDataChanged`	[in] An ObjectDataChanged XML element that contains the new value of the property.

NotifyDataChanged method output

The plugin sends an ObjectDataChanged XML element in the `Source` parameter of the `NotifyDataChanged` method. This XML element provides information about the old property and its new value.

ObjectDataChanged element

The `ObjectDataChanged` element has the following attributes:

Name	Value
`Message`	The message in the following format, where `<ObjectType>` is the name of the property that was changed and `ObjectName` is the new value. The characters following `New object` and the space are in escaped format. `New object <ObjectType>: ObjectName` Required: Yes
`NewValue`	The new value of the property. Required: Yes
`ObjectName`	The property name. Required: Yes

<Go Back

Name	Value
ObjectType	The property type, in escaped format. Required: Yes
OldValue	The old value of the property. Required: Yes
PropertyName	The name of the property in the diagnostics dialog box that was changed. Required: Yes

Example of NotifyDataChanged method output

The following sample code is an ObjectDataChanged XML element that is sent by the plugin to VWorks software as a string in the ObjectDataChanged parameter of the NotifyDataChanged method. The plugin returns the new value of 50 for the property named X/Y Offset Pipetting:East/west offset.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='111f2066886efab8e46ef32de9284e25' version='1.0' >
   <ObjectDataChanged Message='New object &lt;Pipette technique&gt;:
→&quot;technique&quot;' NewValue='50' ObjectName='technique' ObjectType='&lt;Pipette
→technique&gt;' OldValue='0' PropertyName='X/Y Offset Pipetting:East/west offset' />
</Velocity11>
```

Un-escaped attribute values

The following code is the un-escaped portion of the Message attribute value from the previous NotifyDataChanged method output example.

```
<Pipette technique>: "technique"
```

The following code is the un-escaped value of the ObjectType attribute from the previous NotifyDataChanged method output example.

```
<Pipette technique>
```

Related information

For information about...	See...
IControllerClient interface	"IControllerClient interface" on page 87

<Go Back

NotifyTipOperation method

Description

The plugin calls the `NotifyTipOperation` method to tell VWorks software that the user performed a pipette-tip task using the diagnostics dialog box of a device that uses pipette tips. VWorks software can use this information to track pipette-tip usage.

Syntax

```
HRESULT NotifyTipOperation(
   [in] IControllerClient *Source,
   [in] BSTR TipOperationXML
);
```

Parameters

`Source`	[in] The plugin's pointer to itself.
`TipOperationXML`	[in] A DiagnosticsTipOperation XML block that contains information about the specified pipette-tip task involving the specified labware type at the specified location.

NotifyTipOperation method input

VWorks software sends a DiagnosticsTipOperation XML block in the `Source` parameter of the `NotifyTipOperation` method.

DiagnosticsTipOperation XML block

The DiagnosticsTipOperation XML block contains the `DiagnosticsTipOperation` element and all its children. This XML block provides information about the specified pipette-tip task involving the specified labware type at the specified location.

XML structure

The value of the `file` attribute for the `Velocity11` element is `MetaData`. See "Velocity11 element" on page 416.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <DiagnosticsTipOperation>
      <WellSelection>
         <PipetteHeadMode />
         <Wells>
            <Well />
            ...
         </Wells>
      </WellSelection>
   </DiagnosticsTipOperation>
</Velocity11>
```

<Go Back

DiagnosticsTipOperation element

The `DiagnosticsTipOperation` element contains one `WellSelection` element and has the following attributes:

Name	Value
`Labware`	The type of the labware involved in the pipette-tip task. Required: Yes
`Location`	The name of the location where the pipette-tip task occurred. Required: Yes
`Operation`	Represents the type of the pipette-tip task. Possible values: `0` = Tips On `1` = Tips Off Required: No Default value: `1`

WellSelection element

See "WellSelection element" on page 418.

PipetteHeadMode element

See "PipetteHeadMode element" on page 414.

Wells element

See "Wells element" on page 418.

Well element

See "Well element" on page 417.

Example of NotifyTipOperation method output

The following sample code is a DiagnosticsTipOperation XML block that is sent by the plugin to VWorks software as a string in the `TipOperationXML` parameter of the `NotifyTipOperation` method. The plugin returns information about a pipette-tip task involving a labware of type `384 V11 Tip Box ST70 19133.002` at the location named `7`.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='ede80c89e21ca65d4296dfd86c7cd69e' version='1.0' >
   <DiagnosticsTipOperation Labware='384 V11 Tip Box ST70 19133.002' Location='7'
→Operation='0' >
      <WellSelection CanBe16QuadrantPattern='0' CanBeLinked='0'
→CanBeQuadrantPattern='0' IsLinked='0' IsQuadrantPattern='0' OnlyOneSelection='0'
→OverwriteHeadMode='0' QuadrantPattern='0' StartingQuadrant='1'
→LinkedText='From hit pick task' >
         <PipetteHeadMode Channels='1' ColumnCount='24' RowCount='16' SubsetConfig='0'
→SubsetType='0' TipType='0' />
         <Wells>
            <Well Column='0' Row='0' />
         </Wells>
      </WellSelection>
   </DiagnosticsTipOperation>
</Velocity11>
```

<Go Back

Related information

For information about...	See...
IControllerClient interface	“IControllerClient interface” on page 87

<Go Back

OnCloseDiagsDialog method

Description

The plugin calls the `OnCloseDiagsDialog` method to tell VWorks software that the plugin's diagnostics dialog box was closed.

IMPORTANT To properly inform VWorks software that the diagnostics dialog box is closed, the plugin must call the `OnCloseDiagsDialog` method. Otherwise, VWorks software assumes that the dialog box is still open and returns an error when the user starts or resumes a protocol. Before a protocol can run, all diagnostics windows must be closed.

Syntax

```
HRESULT OnCloseDiagsDialog(
   [in] IControllerClient *Source
);
```

Parameters

`Source`	[in] The plugin's pointer to itself.

Related information

For information about...	See...
IWorksDiags CloseDiagsDialog method	"CloseDiagsDialog method" on page 93
IControllerClient interface	"IControllerClient interface" on page 87
IWorksDiags ShowDiagsDialog method	"ShowDiagsDialog method" on page 94

<Go Back

PrintToLog method

Description

The plugin calls the `PrintToLog` method to tell VWorks software to print the specified string (add an entry) to the Main Log.

Syntax

```
HRESULT PrintToLog(
   [in] IControllerClient *Source,
   [in] BSTR StringToPrint
);
```

Parameters

`Source`	[in] The plugin's pointer to itself.
`StringToPrint`	[in] The string to be printed to the Main Log.

Related information

For information about...	See...
IControllerClient interface	"IControllerClient interface" on page 87

<Go Back

Query method

Overview

This section covers the `Query` method. Queries, and their responses, are divided into the following categories. See "Query element" on page 293 for a more complete list that includes descriptions.

- "AllDeviceInfo query/response" on page 296
- "Barcode query/response" on page 300
- "DeviceLocationTeachpoints query/response" on page 302
- "GetDeviceName query/response" on page 304
- "GetIOManagerPointInput query/response" on page 305
- "GetJavascriptVariable query/response" on page 307
- "GetProductInfo query/response" on page 312
- "GetRunSetStatus query/response" on page 313
- "InterPlugin query/response" on page 319
- "Labware query/response" on page 322
- "LocationInformation query/response" on page 327
- "LocationToTeachpoints query/response" on page 330
- "PlateVolume query/response" on page 333
- "ScanBarcode query/response" on page 337
- "SystemPlateInformation query/response" on page 340
- "TeachpointInformation query/response" on page 342

Description

The plugin calls the `Query` method to do one or both of the following:

- Request information from VWorks software
- Request information from another plugin, using VWorks software as the intermediary

  The plugin and VWorks software can use the IWorksController `Query` method in conjunction with the IWorksDriver `ControllerQuery` method to provide a means for two plugins to communicate with each other. See "Interplugin communication" on page 29 for more information.

Syntax

```
HRESULT Query(
   [in] IControllerClient *Source,
   [in] BSTR Query,
   [out,retval] BSTR *QueryResult
);
```

<Go Back

Parameters

`Source`	[in] The plugin's pointer to itself.
`Query`	[in] A Query XML block that contains the query. In this parameter, the plugin passes the query directly to VWorks software, or to the destination plugin through VWorks software.
`QueryResult`	[out, retval] A Response XML block that contains the query response. In this parameter, VWorks software returns its response to the plugin, or VWorks software returns the query response from the destination plugin to the source plugin.

Query XML block

The Query XML block contains the `Query` element and all its children. This XML block is the request from the source plugin for VWorks software, or for the destination plugin sent through VWorks software.

Velocity11 element

For all Query XML blocks, the value of the `file` attribute for the `Velocity11` element is `Query`. See "Velocity11 element" on page 416.

Query element

The `Query` element specifies the type of information that is requested, and for interplugin queries, the names of the source and destination plugins. This element has the following attributes:

Name	Value
`Category`	Represents the type of query. Possible values: • `AllDeviceInfo` Gets all device names and device types in the device file. • `Barcode` Gets all barcodes on the labware at the specified location. • `DeviceLocationTeachpoints` Gets all locations on all devices for which the robot has teachpoints. • `GetDeviceName` Gets the device name for the plugin in VWorks software. • `GetIOManagerPointInput` Gets the value of the specified digital input signal. • `GetJavascriptVariable` Gets the value of the specified JavaScript variable.

<Go Back

Name	Value
	• `GetProductInfo` Gets the name and version of VWorks software that is currently running. • `GetRunSetStatus` Gets information about all the protocol runs listed in the Runset Manager. • `InterPlugin` Gets information from a destination plugin, using VWorks software as the intermediary. *Note:* This query is only used for interplugin communication. • `Labware` Gets the labware entry for the specified labware type. • `LocationInformation` Gets the name of the configured labware at the specified location. If the location is a stack location, gets the allowable stack height. • `LocationToTeachpoints` Gets all teachpoints that have been set at the specified location. • `PlateVolume` Gets the current volume of all the wells in the labware at the specified location. • `ScanBarcode` Determines whether a barcode scanner should be used to scan the barcode on the specified side of the labware at the specified location. • `SystemPlateInformation` Gets the labware type for the specified labware. • `TeachpointInformation` Gets the coordinates of the specified teachpoint for the specified robot from the destination plugin. *Note:* The source plugin can also get this information using the InterPlugin query. Required: Yes
`Source`	The name of the plugin that sends the query. This value is automatically added to the query by VWorks software, but is not used.
`Destination`	For InterPlugin queries, the name of the plugin that should receive the query. The developer of the plugin must specify this attribute. For queries that are directed to VWorks software, this attribute is optional and has no default value.

<Go Back

Child elements

The `Query` element's children are defined in the "query" sections for each value of the `Category` attribute.

Response XML block

The Response XML block contains the `Response` element and all its children. This XML block is the query response that is returned to the source plugin from VWorks software, or from the destination plugin through VWorks software.

Velocity11 element

For all Response XML blocks, the value of the `file` attribute for the `Velocity11` element is `QueryResponse`. See "Velocity11 element" on page 416.

Response element

The `Response` element specifies the type of information that is returned and, for interplugin query responses, the names of the source and destination plugins. This element has the following attributes:

Name	Value
`Category`	Each value of this `Category` attribute matches the `Category` value specified in the query. Possible values: • `AllDeviceInfo` • `Barcode` • `DeviceLocationTeachpoints` • `GetDeviceName` • `GetIOManagerPointInput` • `GetJavascriptVariable` • `GetProductInfo` • `GetRunSetStatus` • `InterPlugin` • `Labware` • `LocationInformation` • `LocationToTeachpoints` • `PlateVolume` • `ScanBarcode` • `SystemPlateInformation` • `TeachpointInformation`
`Source`	The name of the plugin that sends the response.
`Destination`	The name of the plugin that should receive the response, which is the plugin that sent the query.

<Go Back

Child elements

The `Response` element's children are defined in the "query response" sections for each value of the `Category` attribute.

Query/response types

This section describes the query/response types for each value of the `Query` and `Response` elements' `Category` attribute:

- "AllDeviceInfo query/response" on page 296
- "Barcode query/response" on page 300
- "DeviceLocationTeachpoints query/response" on page 302
- "GetDeviceName query/response" on page 304
- "GetIOManagerPointInput query/response" on page 305
- "GetJavascriptVariable query/response" on page 307
- "GetProductInfo query/response" on page 312
- "GetRunSetStatus query/response" on page 313
- "InterPlugin query/response" on page 319
- "Labware query/response" on page 322
- "LocationInformation query/response" on page 327
- "LocationToTeachpoints query/response" on page 330
- "PlateVolume query/response" on page 333
- "ScanBarcode query/response" on page 337
- "SystemPlateInformation query/response" on page 340
- "TeachpointInformation query/response" on page 342

AllDeviceInfo query/response

AllDeviceInfo query

The AllDeviceInfo query requests all the device names and device types in the device file as follows:

- If a protocol is specified, the plugin is requesting the device names and device types in the device file associated with the specified protocol.
- If no protocol is specified, the plugin is requesting the device names and device types in the first device file currently open in VWorks software.

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Query Category='AllDeviceInfo'>
      <Parameters>
         <Parameter Name='ProtocolName' ... />
      </Parameters>
   </Query>
</Velocity11>
```

Parameters element

The `Parameters` element contains one `Parameter` element.

<Go Back

Parameter element

The `Parameter` element has the following attributes plus the Scriptable, Style, and Type attributes:

Name	Value
`Name`	The value `ProtocolName`. Required: Yes
`Value`	If the protocol has been saved, the value of this attribute is the protocol's file path. If the protocol has not been saved, the value is the default protocol name. Required: Yes

Example of an AllDeviceInfo query (ProtocolName is specified)

The following sample code is an AllDeviceInfo query that requests all the device names and device types in the device file that is associated with the protocol named `Protocol File - 1` (the default protocol name).

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Query' md5sum='e720d11b2c02c2b6d183a86e8b7d8b02' version='1.0' >
   <Query Category='AllDeviceInfo' >
      <Parameters >
      <Parameter Name='ProtocolName' Scriptable='1' Style='0' Type='1'
→Value='Protocol File - 1' />
      </Parameters>
   </Query>
</Velocity11>
```

Example of an AllDeviceInfo query (ProtocolName is not specified)

The following sample code is an AllDeviceInfo query that requests the device names and device types in the first device file currently open in VWorks software.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Query' md5sum='3f4f9a4153a60858242372a04167b02e' version='1.0' >
   <Query Category='AllDeviceInfo' />
</Velocity11>
```

AllDeviceInfo query response

The AllDeviceInfo query response returns the device names and device types in the device file as follows:

- If a protocol was specified in the query, VWorks software returns the device names and device types in the device file associated with the specified protocol.
- If no protocol was specified, VWorks software returns the device names and device types in the first device file currently open in VWorks software.

<Go Back

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Response Category='AllDeviceInfo' ... >
      <Parameters>
         <Parameter Name='AllDeviceInfo' ... />
      </Parameters>
   </Response>
</Velocity11>
```

Parameters element

The `Parameters` element contains one `Parameter` element.

Parameter element

The `Parameter` element has the following attributes plus the Scriptable, Style, and Type attributes:

Name	Value
`Name`	The value `AllDeviceInfo`.
`Value`	An escaped DeviceLocationTeachpoints XML block.

DeviceLocationTeachpoints XML block

The DeviceLocationTeachpoints XML block contains the `DeviceLocationTeachpoints` parent element and all its children. This XML block contains all device names and device types in the device file.

For the DeviceLocationTeachpoints XML block that is returned in the AllDeviceInfo query response, only the `DeviceName` and `DeviceType` attributes of the `DeviceLocationTeachpoint` element are specified. The `LocationName`, `RobotName`, and `TeachpointName` attributes are not specified.

XML structure

The value of the `file` attribute for the `Velocity11` element is `MetaData`. See "Velocity11 element" on page 416.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <DeviceLocationTeachpoints>
      <DeviceLocationTeachpoints>
         <DeviceLocationTeachpoint />
         ...
      </DeviceLocationTeachpoints>
   </DeviceLocationTeachpoints>
</Velocity11>
```

DeviceLocationTeachpoints element (parent)

The `DeviceLocationTeachpoints` parent element has one `DeviceLocationTeachpoints` child element.

DeviceLocationTeachpoints element (child)

The `DeviceLocationTeachpoints` child element contains one or more `DeviceLocationTeachpoint` elements.

<Go Back

DeviceLocationTeachpoint element

Each `DeviceLocationTeachpoint` element contains the name and type of a device in the device file. This element has the following attributes:

Name	Value
`DeviceName`	The device name.
`DeviceType`	The device type.

Example of an AllDeviceInfo query response

The following sample code is an AllDeviceInfo query response. If a protocol was specified in the query, the escaped DeviceLocationTeachpoints XML block contains the device names and device types in the device file associated with the protocol. If a protocol was not specified in the query, the escaped DeviceLocationTeachpoints XML block contains the device names and device types in the first device file currently open in VWorks software.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='QueryResponse' md5sum='9cabeded37bbf38afb5737f9a4f520ee'
→version='1.0' >
   <Response Category='AllDeviceInfo' Destination='IWorksController Test - 1' >
      <Parameters >
         <Parameter Name='AllDeviceInfo' Scriptable='1' Style='0' Type='1'
→Value='&lt;?xml version=&apos;1.0&apos; encoding=&apos;ASCII&apos; ?&gt;
→&lt;Velocity11 file=&apos;MetaData&apos;
→md5sum=&apos;281f97ac84c697cf58d44c4d4a723622&apos; version=&apos;1.0&apos; &gt;
→&lt;DeviceLocationTeachpoints &gt; &lt;DeviceLocationTeachpoints &gt;
→&lt;DeviceLocationTeachpoint DeviceName=&apos;Phantom Robot - 1&apos;
→DeviceType=&apos;Phantom Robot&apos; /&gt; &lt;DeviceLocationTeachpoint
→DeviceName=&apos;PlatePad - 1&apos; DeviceType=&apos;PlatePad&apos; /&gt;
→&lt;DeviceLocationTeachpoint DeviceName=&apos;WasteBin - 1&apos;
→DeviceType=&apos;WasteBin&apos; /&gt; &lt;/DeviceLocationTeachpoints&gt;
→&lt;/DeviceLocationTeachpoints&gt; &lt;/Velocity11&gt;' />
      </Parameters>
   </Response>
</Velocity11>
```

Un-escaped DeviceLocationTeachpoints XML block

The following code is the un-escaped DeviceLocationTeachpoints XML block from the preceding AllDeviceInfo query response example.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='281f97ac84c697cf58d44c4d4a723622' version='1.0' >
   <DeviceLocationTeachpoints >
      <DeviceLocationTeachpoints >
         <DeviceLocationTeachpoint DeviceName='Phantom Robot - 1'
→DeviceType='Phantom Robot' />
         <DeviceLocationTeachpoint DeviceName='PlatePad - 1' DeviceType='PlatePad' />
         <DeviceLocationTeachpoint DeviceName='WasteBin - 1' DeviceType='WasteBin' />
      </DeviceLocationTeachpoints>
   </DeviceLocationTeachpoints>
</Velocity11>
```

<Go Back

Barcode query/response

Barcode query

The Barcode query requests all barcodes on the labware at the specified location on the device.

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Query Category='Barcode'>
      <Parameters>
         <Parameter Name='Location' ... />
   </Query>
</Velocity11>
```

Parameters element

The `Parameters` element contains one `Parameter` element.

Parameter element

The `Parameter` element has the following attributes plus the Scriptable, Style, and Type attributes:

Name	Value
`Name`	The value `Location`. Required: Yes
`Value`	The name of the location on the device. Required: Yes

Example of a Barcode query

The following sample code is a Barcode query that requests all barcodes on the labware at the location named `Stage 1`.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Query' md5sum='3a5758c22a5dac2966623e6e08edcf37' version='1.0' >
   <Query Category='Barcode'>
      <Parameters >
         <Parameter Name='Location' Scriptable='1' Style='0' Type='1'
→Value='Stage 1' />
      </Parameters>
   </Query>
```

Barcode query response

The Barcode query response returns the barcodes on the labware at the location specified in the query.

<Go Back

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Response Category='Barcode' ...>
      <Parameters>
         <Parameter Category='Barcode' ... />
         ...
      </Parameters>
   </Response>
</Velocity11>
```

Parameters element

The `Parameters` element contains one or more `Parameter` elements.

Parameter element

The `Parameter` element has the following attributes plus the Scriptable, Style, and Type attributes:

Name	Value
`Category`	The value `Barcode`.
`Name`	Represents the side of the labware. Possible values: `0` = South `1` = West `2` = North `3` = East
`Value`	The barcode.

Example of a Barcode query response

The following sample code is a Barcode query response that returns four barcodes, one for each side of the labware at the location specified in the query.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='QueryResponse' md5sum='7526ebbf819e069f77548aaa4219f96c'
version='1.0' >
   <Response Category='Barcode' Destination='IWorksController Test - 1' />
      <Parameters >
         <Parameter Category='Barcode' Name='0' Scriptable='1' Style='0' Type='1'
→Value='barcode1' />
         <Parameter Category='Barcode' Name='1' Scriptable='1' Style='0' Type='1'
→Value='barcode2' />
         <Parameter Category='Barcode' Name='2' Scriptable='1' Style='0' Type='1'
→Value='barcode3' />
         <Parameter Category='Barcode' Name='3' Scriptable='1' Style='0' Type='1'
→Value='barcode4' />
      </Parameters>
</Velocity11>
```

<Go Back

DeviceLocationTeachpoints query/response

DeviceLocationTeachpoints query

The DeviceLocationTeachpoints query requests all locations on all devices for which a robot has teachpoints. This query contains an empty DeviceLocationTeachpoints Query XML block.

Example of a DeviceLocationTeachpoints query

The following sample code is a DeviceLocationTeachpoints query.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Query' md5sum='aa1c8a1047f1c17f3ae5b8e617266032' version='1.0' >
   <Query Category='DeviceLocationTeachpoints' />
</Velocity11>
```

DeviceLocationTeachpoints query response

The DeviceLocationTeachpoints query response returns all locations on all devices for which the robot has teachpoints.

If the DeviceLocationTeachpoints query is called on a non-robot device, an empty DeviceLocationTeachpoints XML block is returned as the value of the `Parameter` element's `Value` attribute.

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Response Category='DeviceLocationTeachpoints' ...>
      <Parameters>
         <Parameter Name='DeviceLocationTeachpoints' ... />
      </Parameters>
   </Response>
<Velocity11>
```

Parameters element

The `Parameters` element contains one `Parameter` element.

Parameter element

The `Parameter` element has the following attributes plus the Scriptable, Style, and Type attributes:

Name	Value
`Name`	The value `DeviceLocationTeachpoints`.
`Value`	An escaped DeviceLocationTeachpoints XML block.

DeviceLocationTeachpoints XML block

The DeviceLocationTeachpoints XML block contains the `DeviceLocationTeachpoints` parent element and all its children. This XML block returns the locations and devices for the specified robot.

<Go Back

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <DeviceLocationTeachpoints>
      <DeviceLocationTeachpoints>
         <DeviceLocationTeachpoint />
         ...
      </DeviceLocationTeachpoints>
   </DeviceLocationTeachpoints>
</Velocity11>
```

DeviceLocationTeachpoints element (parent)

The `DeviceLocationTeachpoints` parent element has one `DeviceLocationTeachpoints` child element.

DeviceLocationTeachpoints element (child)

The `DeviceLocationTeachpoints` child element contains one or more `DeviceLocationTeachpoint` elements.

DeviceLocationTeachpoint element

The `DeviceLocationTeachpoint` element has the following attributes:

Name	Value
`DeviceName`	The device name.
`DeviceType`	The device type.
`LocationName`	The name of the location on the device.
`RobotName`	The robot name.
`TeachpointName`	The teachpoint name.

Example of a DeviceLocationTeachpoints query response (robot)

The following sample code is a DeviceLocationTeachpoints query response that contains an escaped DeviceLocationTeachpoints XML block. The response contains the location named `Stage` and the device named `PlatePad - 1` for which the robot named `IWorksController Test - 1` has teachpoints.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='QueryResponse' md5sum='7ac5ebff6b67d5d080344d11f508f557'
→version='1.0'>
   <Response Category='DeviceLocationTeachpoints'
→Destination='IWorksController Test - 1'>
      <Parameters>
         <Parameter Name='DeviceLocationTeachpoints' Scriptable='1' Style='0' Type='1'
→Value='&lt;?xml version=&apos;1.0&apos; encoding=&apos;ASCII&apos; ?&gt;
→&lt;Velocity11 file=&apos;MetaData&apos;
→md5sum=&apos;a16eb3001ae9fcd2ff3fd34654eba2d5&apos; version=&apos;1.0&apos; &gt;
→&lt;DeviceLocationTeachpoints &gt; &lt;DeviceLocationTeachpoints &gt;
→&lt;DeviceLocationTeachpoint DeviceName=&apos;PlatePad - 1&apos;
→DeviceType=&apos;PlatePad&apos; LocationName=&apos;Stage&apos;
→RobotName=&apos;IWorksController Test - 1&apos; TeachpointName=&apos;Teachpoint
→1&apos; /&gt; &lt;/DeviceLocationTeachpoints&gt; &lt;/DeviceLocationTeachpoints&gt;
→&lt;/Velocity11&gt;' />
      </Parameters>
   </Response>
</Velocity11>
```

<Go Back

Un-escaped DeviceLocationTeachpoints XML block

The following code is the un-escaped DeviceLocationTeachpoints XML block from the preceding DeviceLocationTeachpoints query response example.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' gmd5sum='a16eb3001ae9fcd2ff3fd34654eba2d5'
→version='1.0' >
   <DeviceLocationTeachpoints >
      <DeviceLocationTeachpoints >
         <DeviceLocationTeachpoint DeviceName='PlatePad - 1' DeviceType='PlatePad'
→LocationName='Stage' RobotName='IWorksController Test - 1'
→TeachpointName='Teachpoint 1' />
      </DeviceLocationTeachpoints>
   </DeviceLocationTeachpoints>
</Velocity11>
```

Example of a DeviceLocationTeachpoints query response (non-robot device)

The following sample code is a DeviceLocationTeachpoints query response for a non-robot device. The response contains an empty DeviceLocationTeachpoints XML block.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='QueryResponse' md5sum='7ac5ebff6b67d5d080344d11f508f557'
→version='1.0'>
   <Response Category='DeviceLocationTeachpoints'
→Destination='IWorksController Test - 1'>
      <Parameters>
         <Parameter Name='DeviceLocationTeachpoints' Scriptable='1' Style='0' Type='1'
→Value='&lt;?xml version=&apos;1.0&apos; encoding=&apos;ASCII&apos; ?&gt;
→&lt;Velocity11 file=&apos;MetaData&apos;
→md5sum=&apos;a16eb3001ae9fcd2ff3fd34654eba2d5&apos;
→version=&apos;1.0&apos; &gt; &lt;DeviceLocationTeachpoints /&gt;
→&lt;/Velocity11&gt;' />
      </Parameters>
   </Response>
</Velocity11>
```

Un-escaped empty DeviceLocationTeachpoints XML block

The following code is the empty un-escaped DeviceLocationTeachpoints XML block from the preceding DeviceLocationTeachpoints query response example.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='a16eb3001ae9fcd2ff3fd34654eba2d5' version='1.0' >
   <DeviceLocationTeachpoints />
</Velocity11>'
```

GetDeviceName query/response

GetDeviceName query

The user can specify a name for each instance of the plugin, but the plugin instance does not know what its device name is. The plugin calls the GetDeviceName query to get its device name in VWorks software. This query contains an empty GetDeviceName Query XML block.

<Go Back

Example of a GetDeviceName query

The following sample code is a GetDeviceName query.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Query' md5sum='caa52e1e5fae3c9dbea6cdc245eb3485' version='1.0' >
   <Query Category='GetDeviceName' />
</Velocity11>
```

GetDeviceName query response

The GetDeviceName query response returns the plugin's device name.

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Response Category='GetDeviceName' ...>
      <Parameters>
         <Parameter Name='DeviceName' ... />
      </Parameters>
   </Response>
<Velocity11>
```

Parameters element

The `Parameters` element contains one `Parameter` element.

Parameter element

The `Parameter` element has the following attributes plus the Scriptable, Style, and Type attributes:

Name	Value
`Name`	The value `DeviceName`.
`Value`	The plugin's device name.

Example of a GetDeviceName query response

The following sample code is a GetDeviceName query response that returns the requested plugin device named `IWorksController Test - 1`.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='QueryResponse' md5sum='c20b42f0195609cb663db3cbbba1d48e'
→version='1.0' >
   <Response Category='GetDeviceName' Destination='IWorksController Test - 1' >
      <Parameters >
         <Parameter Name='DeviceName' Scriptable='1' Style='0' Type='1'
→Value='IWorksController Test - 1' />
      </Parameters>
   </Response>
</Velocity11>
```

GetIOManagerPointInput query/response

GetIOManagerPointInput query

The GetIOManagerPointInput query requests the value of a digital input channel, which is specified in the IO Manager.

<Go Back

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Query Category='GetIOManagerPointInput'>
      <Parameters>
         <Parameter Name='PointName' ... />
      </Parameters>
   </Query>
</Velocity11>
```

Parameters element

The `Parameters` element contains one `Parameter` element.

Parameter element

The `Parameter` element has the following attributes plus the Scriptable, Style, and Type attributes:

Name	Value
`Name`	The value `PointName`. Required: Yes
`Value`	The name of the digital input channel. Required: Yes

Example of a GetIOManagerPointInput query

The following sample code is a GetIOManagerPointInput query that requests the value of the digital input channel named `P1`.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Query' md5sum='625cca67376182141c2e2744fb49bd0d' version='1.0' >
   <Query Category='GetIOManagerPointInput' >
      <Parameters >
         <Parameter Name='PointName' Scriptable='1' Style='0' Type='1' Value='P1' />
      </Parameters>
   </Query>
</Velocity11>
```

GetIOManagerPointInput query response

The GetIOManagerPointInput query response returns the value of the digital input signal that was specified in the query.

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Response Category='GetIOManagerPointInput' ...>
      <Parameters>
         <Parameter Name='PointState' ... />
      </Parameters>
   </Response>
</Velocity11>
```

<Go Back

Parameters element

The `Parameters` element contains one `Parameter` element.

Parameter element

The `Parameter` element has the following attributes plus the Scriptable, Style, and Type attributes:

Name	Value
`Name`	The value `PointState`.
`Value`	The state of the digital input signal.

Example of a GetIOManagerPointInput query response

The following sample code is a GetIOManagerPointInput query response that returns the value `0` for the digital input signal that was specified in the query.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='QueryResponse' md5sum='aa0b07bc1788411376dfc0987d0b9a18'
→version='1.0' >
   <Response Category='GetIOManagerPointInput'
→Destination='IWorksController Test - 1' >
      <Parameters >
         <Parameter Name='PointState' Scriptable='1' Style='0' Type='8' Value='0' />
      </Parameters>
   </Response>
</Velocity11>
```

GetJavascriptVariable query/response

GetJavascriptVariable query

The GetJavascriptVariable query requests the value of the specified JavaScript variable in the specified protocol. The plugin can use this value, for example, to calculate the value of the internal variables for a device.

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Query Category='GetJavascriptVariable'>
      <Parameters>
         <Parameter Name='VariableName' ... />
         <Parameter Name='ProtocolName' ... />
      </Parameters>
   </Query>
</Velocity11>
```

Parameters element

The `Parameters` element contains two `Parameter` elements.

<Go Back

Parameter element

Each `Parameter` element has one of the following pairs of `Name` and `Value` attributes plus the Scriptable, Style, and Type attributes:

Name	Value
`Name`	The value `VariableName`. Required: Yes
`Value`	The name of the JavaScript variable. Required: Yes
`Name`	The value `ProtocolName`. Required: Yes
`Value`	If the protocol has been saved, the value of this attribute is the protocol's file path. If the protocol has not been saved, the value is the default protocol name. Required: Yes

Example of a GetJavascriptVariable query

The following sample code is a GetJavascriptVariable query that requests the value of JavaScript variable `a`, which is contained in the protocol named `Protocol File - 1`.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Query' md5sum='d4966e51cb29ddf7e75634b13288ae6f' version='1.0' >
   <Query Category='GetJavascriptVariable' >
      <Parameters >
         <Parameter Name='VariableName' Scriptable='1' Style='0' Type='1' Value='a' />
         <Parameter Name='ProtocolName' Scriptable='1' Style='0' Type='1'
→Value='Protocol File - 1' />
      </Parameters>
   </Query>
</Velocity11>
```

GetJavascriptVariable query response

The GetJavascriptVariable query response returns the value of the specified JavaScript variable contained in the protocol that is specified in the query.

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Response Category='GetJavascriptVariable' ...>
      <Parameters>
         <Parameter Name='VariableValue' ... />
      </Parameters>
   </Response>
</Velocity11>
```

<Go Back

Parameters element

The `Parameters` element contains on `Parameter` element.

Parameter element

The `Parameter` element has the following attributes plus the Scriptable, Style, and Type attributes:

Name	Value
`Name`	The value `VariableValue`.
`Value`	An escaped JSObject XML block.

JSObject XML block

The JSObject XML block contains the `JSObject` element and all its children. This XML block provides the serialization of the JavaScript variable.

XML structure

The value of the `file` attribute for the `Velocity11` element is `JSSerialize`. See "Velocity11 element" on page 416.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <JSObject />
</Velocity11>
```

JSObject element (parent)

The `JSObject` element can contain one or more `JSObject` child elements or one or more `JSProperty` elements. The element can have one of the following pairs of `Type` and `Value` attributes:

Name	Value
`Type`	The value `Nothing`.
`Value`	Not used for this `Type` attribute. This attribute parameter can be used for initializing the JSObject object.
`Type`	The value `Array`.
`Value`	Not used for this `Type` attribute. The `JSObject` child elements contain the array values as shown in the following XML structure:

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <JSObject >
      <JSObject/>
      ...
   </JSObject >
</Velocity11>
```

<Go Back

Name	Value
Type	The value Hash.
Value	Not used for this Type attribute. The JSObject child elements contain the hash values, as shown in the following XML structure:

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <JSObject>
      <JSProperty >
         <JSObject />
      </JSProperty>
      ...
   </JSObject>
</Velocity11>
```

Type	The value String.
Value	The string value.

Type	The value Double.
Value	The string representation of the double value.

Type	The value Int.
Value	The string representation of the integer value.

For JSObject XML blocks that contain arrays

JSObject element (child)

Each JSObject child element is an array item and has the following attributes:

Name	Value
Type	The value String, Double, or Int. All JSObject children (array items) contained in a JSObject parent element must be of the same type.
Value	The string value or the string representation of the double or integer value.

For JSObject XML blocks that contain a hash table

JSProperty element

The JSProperty element contains one JSObject child element and has the following attribute:

Name	Value
Name	The key name of a key–value pair.

<Go Back

JSObject element (child)

The `JSObject` child element has the following attributes:

Name	Value
`Type`	The value `String`, `Double`, or `Int`.
`Value`	The string value or the string representation of the double or integer value.

Example of a GetJavascriptVariable query response

The following sample code is a GetJavascriptVariable query response that contains an escaped JSSerialize XML block. This XML block contains the value `1` for the JavaScript variable in the protocol that was specified in the query.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='QueryResponse' md5sum='557c26a95ff92a1757130eb78826b86d'
→version='1.0'>
   <Response Category='GetJavascriptVariable'
→Destination='IWorksController Test - 1'>
      <Parameters>
         <Parameter Name='VariableValue' Scriptable='1' Style='0' Type='1'
→Value='&lt;?xml version=&apos;1.0&apos; encoding=&apos;ASCII&apos; ?&gt;
→&lt;Velocity11 file=&apos;JSSerialize&apos;
→md5sum=&apos;620957083f71145b8c66848ae28c28be&apos; version=&apos;1.0&apos; &gt;
→&lt;JSObject Type=&apos;Int&apos; Value=&apos;1&apos; /&gt; &lt;/Velocity11&gt;' />
      </Parameters>
   </Response>
</Velocity11>
```

Un-escaped JSObject XML block (integer value)

The following code is the un-escaped JSObject XML block from the previous GetJavascriptVariable query response example.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='JSSerialize' md5sum='620957083f71145b8c66848ae28c28be'
→version='1.0' >
   <JSObject Type='Int' Value='1' />
</Velocity11>
```

Example of a JSObject XML block for an array

The following sample code is a JavaScript array that holds two values: `94` and `73`.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='JSSerialize' md5sum='620957083f71145b8c66848ae28c28be'
→version='1.0' >
   <JSObject Type='Array'>
      <JSObject Type='Int' Value='94' />
      <JSObject Type='Int' Value='73' />
   </JSObject>
</Velocity11>
```

<Go Back

Example of a JSObject XML block for a hash table

The following sample code is a JavaScript hash table that holds two key–value pairs.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='JSSerialize' md5sum='620957083f71145b8c66848ae28c28be'
→version='1.0' >
   <JSObject Type='Hash'>
      <JSProperty Name='John Smith'>
         <JSObject Type='String' Value='555-1212'>
      </JSProperty>
      <JSProperty Name='Jane Smith'>
         <JSObject Type='String' Value='555-1234'>
      </JSProperty>
   </JSObject>
</Velocity11>
```

GetProductInfo query/response

GetProductInfo query

The GetProductInfo query requests the name and version of VWorks software that is currently running. This query contains an empty GetProductInfo Query XML block.

Example of a GetProductInfo query

The following sample code is a GetProductInfo query.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Query' md5sum='a5b708f8f8701e942678b6d71da82a86' version='1.0' >
   <Query Category='GetProductInfo' />
</Velocity11>
```

GetProductInfo query response

The GetProductInfo query response returns the name and version of VWorks software that is currently running.

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Response Category='GetProductInfo' ...>
      <Parameters>
         <Parameter Name='ApplicationName' ... />
         <Parameter Name='ApplicationVersion' ... />
      </Parameters>
   </Response>
</Velocity11>
```

Parameters element

The `Parameters` element contains two `Parameter` elements.

<Go Back

Parameter element

Each `Parameter` element has one of the following pairs of `Name` and `Value` attributes plus the Scriptable, Style, and Type attributes:

Name	Value
`Name`	The value `ApplicationName`.
`Value`	The application name, which is always VWorks.
`Name`	The value `ApplicationVersion`.
`Value`	The version of VWorks software that is currently running.

Example of a GetProductInfo query response

The following sample code is a GetProductInfo query response that returns the version of VWorks software that is currently running, which is `4.0.0`.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='QueryResponse' md5sum='ac97cfffdf48ca45df3dd516bb88216a'
→version='1.0' >
   <Response Category='GetProductInfo' Destination='IWorksController Test - 1' >
      <Parameters >
         <Parameter Name='ApplicationName' Scriptable='1' Style='0' Type='1'
→Value='VWorks' />
         <Parameter Name='ApplicationVersion' Scriptable='1' Style='0' Type='1'
→Value='4.0.0' />
      </Parameters>
   </Response>
</Velocity11>
```

GetRunSetStatus query/response

GetRunSetStatus query

The GetRunSetStatus query requests information about all protocol runs that are listed in the Runset Manager. This query contains an empty GetRunSetStatus Query XML block. The plugin can use the information, for example, to determine when to add another protocol to the runset.

Example of a GetRunSetStatus query

The following sample code is a GetRunSetStatus query.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Query' md5sum='ab5f678faa81d929a2922f8136704602' version='1.0' >
   <Query Category='GetRunSetStatus' />
</Velocity11>
```

GetRunSetStatus query response

The GetRunSetStatus query response returns information about all protocol runs that are listed in the Runset Manager.

<Go Back

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Response Category='GetRunSetStatus' ...>
      <Parameters >
         <Parameter Name='RunsetXML' ... />
         <Parameter Name='Error' ... />
      </Parameters>
   </Response>
</Velocity11>
```

Parameters element

The `Parameters` element contains two `Parameter` elements.

Parameter element

Each `Parameter` element has one of the following pairs of `Name` and `Value` attributes plus the Scriptable, Style, and Type attributes:

Name	Value
`Name`	The value `RunsetXML`.
`Value`	An escaped Runsets XML block.
`Name`	The value `Error`.
`Value`	A text string describing the error that occurred when the plugin received the requested information. If no error occurred, this attribute is not specified.

Runsets XML block

The Runsets XML block contains the `Runsets` element and all its children. This XML block contains information for all protocol runs that are listed in the Runset Manager.

<Go Back

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Runsets>
      <Runset>
         <Parameters >
            <Parameter Name='Protocol Name' ... />
            <Parameter Name='Runs' ... />
            <Parameter Name='Protocol Notes' ... />
            <Parameter Name='Priority' ... />
            <Parameter Name='ID' ... />
            <Parameter Name='Start_Year' ... />
            <Parameter Name='Start_Month' ... />
            <Parameter Name='Start_Day' ... />
            <Parameter Name='Start_Hour' ... />
            <Parameter Name='Start_Minute' ... />
            <Parameter Name='Start_Second' ... />
            <Parameter Name='State' ... />
            <Parameter Name='Depend ID' ... />
            <Parameter Name='Depend_Day' ... />
            <Parameter Name='Depend_Hour' ... />
            <Parameter Name='Depend_Minute' ... />
            <Parameter Name='Depend_Second' ... />
         </Parameters>
      </Runset>
      ...
   </Runsets>
</Velocity11>
```

Runsets element

The `Runsets` element contains one or more `Runset` elements.

Runset element

The `Runset` element contains one `Parameters` element and has the following attribute:

Name	Value
`Name`	The runset name.

Parameters element

The `Parameters` element contains 17 `Parameter` elements.

Parameter element

Each `Parameter` element has one of the following pairs of `Name` and `Value` attributes:

Name	Value
`Name`	The value `ProtocolName`.
`Value`	If the protocol has been saved, the value of this attribute is the protocol's file path. If the protocol has not been saved, the value is the default protocol name.
`Name`	The value `Runs`.
`Value`	The number of times the protocol is scheduled to run (starts with `1`).

<Go Back

Name	Value
Name	The value ProtocolNotes.
Value	Any notes about the protocol that were entered in the Run Configuration Wizard.
Name	The value Priority.
Value	The priority of the protocol run (starts with 1).
Name	The value ID.
Value	The protocol run ID (starts with 1).
Name	The value Start_Year.
Value	The scheduled start year of the protocol (4 digits).
Name	The value Start_Month.
Value	The scheduled start month of the protocol (1–12).
Name	The value Start_Day.
Value	The scheduled start day of the protocol (1–31).
Name	The value Start_Hour.
Value	The scheduled start hour of the protocol (0–23).
Name	The value Start_Minute.
Value	The scheduled start minutes of the protocol (0–59).
Name	The value Start_Second.
Value	The scheduled start seconds of the protocol (0–59).
Name	The value State.
Value	The state of the run. Possible values: 0 = Run as soon as possible 1 = Run at a fixed time 2 = Depend on another protocol to start 3 = Depend on another protocol to finish 4 = Dependency is broken
Name	The value Depend ID.
Value	The dependent protocol run ID (starting with 1). The value of this attribute can be greater than 0 when the value of the State attribute is 2 or 3; otherwise, the value is 0.
Name	The value Depend_Day.
Value	The days after the dependent protocol starts or finishes (no range). The value of this attribute can be greater than 0 when the value of the State attribute is 2 or 3; otherwise, the value is 0.

<Go Back

Name	Value
Name	The value Depend_Hour.
Value	The hours after the dependent protocol starts or finishes (0–23). The value of this attribute can be greater than 0 when the value of the State attribute is 2 or 3; otherwise, the value is 0.
Name	The value Depend_Minute.
Value	The minutes after the dependent protocol starts or finishes (0–59). The value of this attribute can be greater than 0 when the value of the State attribute is 2 or 3; otherwise, the value is 0.
Name	The value Depend_Second.
Value	The seconds after the dependent protocol starts or finishes (0–59). The value of this attribute can be greater than 0 when the value of the State attribute is 2 or 3; otherwise, the value is 0.

<Go Back

Example of a GetRunSetStatus query response

The following sample code is a GetRunSetStatus query response that contains an escaped Runsets XML block. This XML block returns information that is listed in the Runset Manager for two protocol runs.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='QueryResponse' md5sum='874675d9ec339b5a4810c2fabe932a2f'
version='1.0' >
   <Response Category='GetRunSetStatus' Destination='IWorksController Test - 1' >
      <Parameters >
         <Parameter Name='RunsetXML' Scriptable='1' Style='0' Type='1' Value='&lt;?xml
→version=&apos;1.0&apos; encoding=&apos;ASCII&apos; ?&gt; &lt;Velocity11
→file=&apos;Runset_Data&apos; md5sum=&apos;1d3e31c99c747ac48b15c06c275ec75d&apos;
→version=&apos;1.0&apos; &gt; &lt;Runsets &gt; &lt;Runset Name=&apos;&apos; &gt;
→&lt;Parameters &gt; &lt;Parameter Name=&apos;Protocol Name&apos;
→Value=&apos;C:\VWorks Workspace\Protocol files\Protocol File - 1.pro&apos; /&gt;
→&lt;Parameter Name=&apos;Runs&apos; Value=&apos;2&apos; /&gt; &lt;Parameter
→Name=&apos;Protocol Notes&apos; Value=&apos;&apos; /&gt; &lt;Parameter
→Name=&apos;Priority&apos; Value=&apos;1&apos; /&gt; &lt;Parameter
→Name=&apos;ID&apos; Value=&apos;1&apos; /&gt; &lt;Parameter
→Name=&apos;Start_Year&apos; Value=&apos;2010&apos; /&gt; &lt;Parameter
→Name=&apos;Start_Month&apos; Value=&apos;7&apos; /&gt; &lt;Parameter
→Name=&apos;Start_Day&apos; Value=&apos;1&apos; /&gt; &lt;Parameter
→Name=&apos;Start_Hour&apos; Value=&apos;16&apos; /&gt; &lt;Parameter
→Name=&apos;Start_Minute&apos; Value=&apos;40&apos; /&gt; &lt;Parameter
→Name=&apos;Start_Second&apos; Value=&apos;39&apos; /&gt; &lt;Parameter
→Name=&apos;State&apos; Value=&apos;1&apos; /&gt; &lt;Parameter Name=&apos;Depend
→ID&apos; Value=&apos;0&apos; /&gt; &lt;Parameter Name=&apos;Depend_Day&apos;
→Value=&apos;0&apos; /&gt; &lt;Parameter Name=&apos;Depend_Hour&apos;
→Value=&apos;0&apos; /&gt; &lt;Parameter Name=&apos;Depend_Minute&apos;
→Value=&apos;0&apos; /&gt; &lt;Parameter Name=&apos;Depend_Second&apos;
→Value=&apos;0&apos; /&gt; &lt;/Parameters&gt; &lt;/Runset&gt; &lt;Runset
→Name=&apos;&apos; &gt; &lt;Parameters &gt; &lt;Parameter Name=&apos;Protocol
→Name&apos; Value=&apos;C:\VWorks Workspace\Protocol files\Protocol File -
→2.pro&apos; /&gt; &lt;Parameter Name=&apos;Runs&apos; Value=&apos;10&apos; /&gt;
→&lt;Parameter Name=&apos;Protocol Notes&apos; Value=&apos;&apos; /&gt;
→&lt;Parameter Name=&apos;Priority&apos; Value=&apos;2&apos; /&gt; &lt;Parameter
→Name=&apos;ID&apos; Value=&apos;2&apos; /&gt; &lt;Parameter
→Name=&apos;Start_Year&apos; Value=&apos;2010&apos; /&gt; &lt;Parameter
→Name=&apos;Start_Month&apos; Value=&apos;7&apos; /&gt; &lt;Parameter
→Name=&apos;Start_Day&apos; Value=&apos;2&apos; /&gt; &lt;Parameter
→Name=&apos;Start_Hour&apos; Value=&apos;16&apos; /&gt; &lt;Parameter
→Name=&apos;Start_Minute&apos; Value=&apos;29&apos; /&gt; &lt;Parameter
→Name=&apos;Start_Second&apos; Value=&apos;2&apos; /&gt; &lt;Parameter
→Name=&apos;State&apos; Value=&apos;1&apos; /&gt; &lt;Parameter Name=&apos;Depend
→ID&apos; Value=&apos;0&apos; /&gt; &lt;Parameter Name=&apos;Depend_Day&apos;
→Value=&apos;0&apos; /&gt; &lt;Parameter Name=&apos;Depend_Hour&apos;
→Value=&apos;0&apos; /&gt; &lt;Parameter Name=&apos;Depend_Minute&apos;
→Value=&apos;0&apos; /&gt; &lt;Parameter Name=&apos;Depend_Second&apos;
→Value=&apos;0&apos; /&gt; &lt;/Parameters&gt; &lt;/Runset&gt; &lt;/Runsets&gt;
→&lt;/Velocity11&gt;' />
         <Parameter Name='Error' Scriptable='1' Style='0' Type='1' />
      </Parameters>
   </Response>
</Velocity11>
```

<Go Back

Un-escaped Runsets XML block

The following sample code is the un-escaped Runsets XML block from the previous GetRunSetStatus query response example.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Runset_Data' md5sum='1d3e31c99c747ac48b15c06c275ec75d'
version='1.0' >
   <Runsets >
      <Runset Name='' >
         <Parameters >
            <Parameter Name='Protocol Name' Value='C:\VWorks Workspace\Protocol
files\Protocol File - 1.pro' />
            <Parameter Name='Runs' Value='2' />
            <Parameter Name='Protocol Notes' Value='' />
            <Parameter Name='Priority' Value='1' />
            <Parameter Name='ID' Value='1' />
            <Parameter Name='Start_Year' Value='2010' />
            <Parameter Name='Start_Month' Value='7' />
            <Parameter Name='Start_Day' Value='1' />
            <Parameter Name='Start_Hour' Value='16' />
            <Parameter Name='Start_Minute' Value='40' />
            <Parameter Name='Start_Second' Value='39' />
            <Parameter Name='State' Value='1' />
            <Parameter Name='Depend ID' Value='0' />
            <Parameter Name='Depend_Day' Value='0' />
            <Parameter Name='Depend_Hour' Value='0' />
            <Parameter Name='Depend_Minute' Value='0' />
            <Parameter Name='Depend_Second' Value='0' />
         </Parameters>
      </Runset>
      <Runset Name='' >
         <Parameters >
            <Parameter Name='Protocol Name'
→Value='C:\VWorks Workspace\Protocol files\Protocol File - 2.pro' />
            <Parameter Name='Runs' Value='10' />
            <Parameter Name='Protocol Notes' Value='' />
            <Parameter Name='Priority' Value='2' />
            <Parameter Name='ID' Value='2' />
            <Parameter Name='Start_Year' Value='2010' />
            <Parameter Name='Start_Month' Value='7' />
            <Parameter Name='Start_Day' Value='2' />
            <Parameter Name='Start_Hour' Value='16' />
            <Parameter Name='Start_Minute' Value='29' />
            <Parameter Name='Start_Second' Value='2' />
            <Parameter Name='State' Value='1' />
            <Parameter Name='Depend ID' Value='0' />
            <Parameter Name='Depend_Day' Value='0' />
            <Parameter Name='Depend_Hour' Value='0' />
            <Parameter Name='Depend_Minute' Value='0' />
            <Parameter Name='Depend_Second' Value='0' />
         </Parameters>
      </Runset>
   </Runsets>
</Velocity11>
```

InterPlugin query/response

InterPlugin query

The source plugin sends an InterPlugin query to request information from the destination plugin, using VWorks software as the intermediary. VWorks software returns the response from the destination plugin in the InterPlugin query response.

The developers of the two plugins must define their own mutually agreed-upon values for the following attributes:

<Go Back

- The Name and Value attributes of the query Parameter element
- The Name and Value attributes of the response Parameter elements in the Response XML block

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Query Category='InterPlugin' ...>
      <Parameters>
         <Parameter />
      </Parameters>
   </Query>
</Velocity11>
```

Parameters element

The Parameters element has one Parameter element.

Parameter element

The Parameter element has the following attributes plus the Scriptable, Style, and Type attributes:

Name	Value
Name	The developer-defined name for the Interplugin query parameter. Required: Yes
Value	The developer-defined interplugin query. Required: Yes

Example of an InterPlugin query

The following sample code is an Interplugin query from the source plugin for the destination plugin named IWorksController Test - 2 that requests the value of a.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Query' md5sum='2ec6d3e5530975e9fca67fce701e9bd3' version='1.0' >
   <Query Category='InterPlugin' Destination='IWorksController Test - 2' >
      <Parameters >
         <Parameter Name='InterpluginParameter' Scriptable='1' Style='0' Type='1'
→Value='a' />
      </Parameters>
   </Query>
</Velocity11>
```

InterPlugin query response

The destination plugin returns its response to VWorks software, and then VWorks software returns the response to the source plugin in the InterPlugin query response.

This section presents an example of how Parameter elements might be defined for an Interplugin query response.

<Go Back

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Response Category='InterPlugin' ...>
      <Parameters>
         <Parameter Name='InnerResponse' ... />
      </Parameters>
   </Response>
</Velocity11>
```

Parameters element

The `Parameters` element contains one `Parameter` element.

Parameter element

The `Parameter` element has the following attributes plus the Scriptable, Style, and Type attributes:

Name	Value
`Name`	The value `InnerResponse`. Required: Yes
`Value`	An escaped Interplugin Response XML block. Required: Yes

Interplugin Response XML block

The escaped Interplugin Response XML block contains the destination plugin's response to the query from the source plugin.

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Response Category='InterPlugin' ...>
      <Parameters>
         <Parameter />
         <Parameter />
      </Parameters>
   </Response>
</Velocity11>
```

Parameters element

The `Parameters` element contains two `Parameter` elements: one acknowledges receipt of the query and the other has the query response.

Parameter element

The `Parameter` element has the following attributes plus the Scriptable, Style, and Type attributes:

Name	Value
`Name`	The developer-defined name for the query response parameter. Required: Yes

<Go Back

Name	Value
Value	The query response. Required: Yes

Example of an InterPlugin query response

The following sample code is an Interplugin query response from the plugin named IWorksController Test - 2 for the plugin named IWorksController Test - 1. The response contains an escaped Interplugin Response XML block that returns the value of a, which is 2222222222.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='QueryResponse' md5sum='b53d232c6ee00da01286d91a92d6025c'
→version='1.0'>
   <Response Category='InterPlugin' Destination='IWorksController Test - 1'
→Source='IWorksController Test - 2'>
      <Parameters>
         <Parameter Name='InnerResponse' Scriptable='1' Style='0' Type='1'
→Value='&lt;?xml version=&apos;1.0&apos; encoding=&apos;ASCII&apos; ?&gt;
→&lt;Velocity11 file=&apos;QueryResponse&apos;
→md5sum=&apos;3b449c37c9e8fb774cf67147a540e0dd&apos; version=&apos;1.0&apos; &gt;
→&lt;Response Category=&apos;InterPlugin&apos; Destination=&apos;IWorksController
→Test - 1&apos; Source=&apos;IWorksController Test - 2&apos; &gt; &lt;Parameters &gt;
→&lt;Parameter Name=&apos;InterPlugin Param&apos; Scriptable=&apos;1&apos;
→Style=&apos;0&apos; Type=&apos;1&apos; Value=&apos;Receive a&apos; /&gt;
→&lt;Parameter Name=&apos;Param #2&apos; Scriptable=&apos;1&apos;
→Style=&apos;0&apos; Type=&apos;1&apos; Value=&apos;Value 2222222222&apos; /&gt;
→&lt;/Parameters&gt; &lt;/Response&gt; &lt;/Velocity11&gt;' />
      </Parameters>
   </Response>
</Velocity11>
```

Un-escaped Response XML block

The following sample code is the un-escaped Response XML block from the previous InterPlugin query response example.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='QueryResponse' md5sum='3b449c37c9e8fb774cf67147a540e0dd'
→version='1.0' >
   <Response Category='InterPlugin' Destination='IWorksController Test - 1'
→Source='IWorksController Test - 2' >
      <Parameters >
         <Parameter Name='InterPlugin Param' Scriptable='1' Style='0' Type='1'
→Value='Receive a' />
         <Parameter Name='Param #2' Scriptable='1' Style='0' Type='1'
→Value='Value 2222222222' />
      </Parameters>
   </Response>
</Velocity11>
```

Labware query/response

Labware query

The Labware query requests the labware entry for the specified labware type. The plugin typically makes this request when the user selects a labware in the Task Parameters area or in a diagnostics dialog box.

<Go Back

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Query Category='Labware'>
      <Parameters>
         <Parameter Name='Labware_Entry' ... />
      </Parameters>
   </Query>
</Velocity11>
```

Parameters element

The `Parameters` element contains one `Parameter` element.

Parameter element

The `Parameter` element has the following attributes plus the Scriptable, Style, and Type attributes:

Name	Value
`Name`	The value `Labware_Entry`. Required: Yes
`Value`	The labware type. Required: Yes

Example of a Labware query

The following sample code is a Labware query that requests the labware entry for the `1536 Greiner 782076 blk sqr well flt btm` labware type.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Query' md5sum='611091148e78dae75ad575bf6d17e3a7' version='1.0' >
   <Query Category='Labware' >
      <Parameters >
         <Parameter Name='Labware_Entry' Scriptable='1' Style='0' Type='1'
→Value='1536 Greiner 782076 blk sqr well flt btm' />
      </Parameters>
   </Query>
</Velocity11>
```

Labware query response

The Labware query response returns the labware entry for the labware specified in the query.

<Go Back

XML structure (truncated)

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Response Category='Labware' ...>
      <Parameters>
         <Parameter Name='Labware_Entry' ... />
         <Parameter ... />
         <Parameter Name='3RD_PARTY_TIP_CAPACITY' ... />
         <Parameter Name='A12_NOTCH' Scriptable='1' ... />
         ...
         <Parameter Name='Y_TEACHPOINT_TO_WELL' ... />
         <Parameter Name='Y_WELL_TO_WELL' ... />
         <Parameter Name='Z_TIP_ATTACH_OFFSET' ... />
      </Parameters>
   </Response>
</Velocity11>
```

Parameters element

The `Parameters` element has 62 `Parameter` elements: 60 are labware properties.

Parameter element

Each `Parameter` element has one of the following pairs of `Name` and `Value` attributes plus the Scriptable, Style, and Type attributes:

Name	Value
`Name`	The value `Labware_Entry`.
`Value`	The labware name.
`Name`	The `Name` attribute is not specified for this parameter.
`Value`	The value `0`. This parameter is the `(Default)` value that is automatically generated when a Windows registry key is created.
`Name`	One of the properties from the table in "Labware properties" on page 325. The Labware query response returns a `Parameter` element for each property on the list.
`Value`	The associated Labware Editor property value type. See "Labware Editor property value types" on page 326.

<Go Back

Labware properties

The following table lists the possible values for the `Parameter` element's `Name` attribute, where the specified value is a labware property.

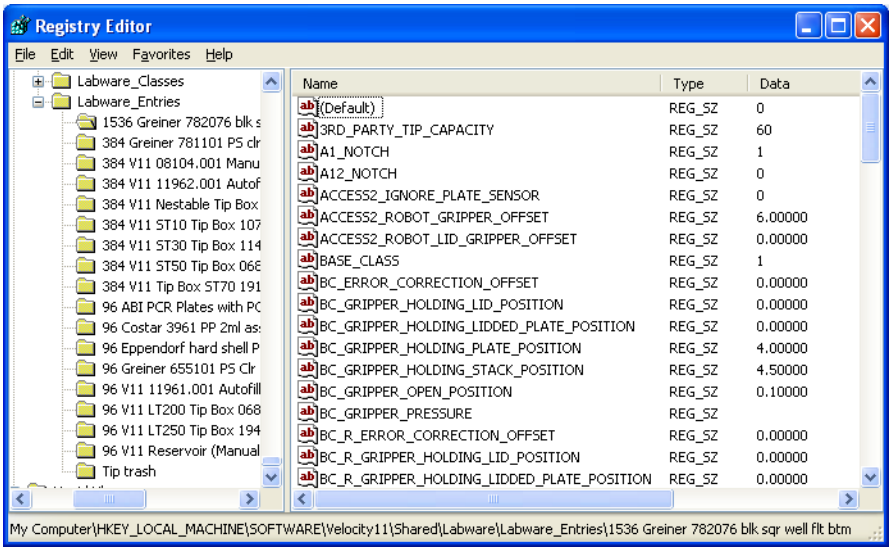
Possible values for the Name attribute

3RD_PARTY_TIP_CAPACITY	MANUFACTURER_PART_NUMBER
A12_NOTCH	MOUNTED_LID_ROBOT_GRIPPER_OFFSET
A1_NOTCH	NAME
BASE_CLASS	NUMBER_OF_WELLS
BC_ERROR_CORRECTION_OFFSET	PRESENTATION_OFFSET
BC_GRIPPER_HOLDING_LIDDED_PLATE_POSITION	ROBOT_GRIPPER_OFFSET
BC_GRIPPER_HOLDING_LID_POSITION	ROBOT_HANDLING_SPEED
BC_GRIPPER_HOLDING_PLATE_POSITION	SEALED_STACKING_THICKNESS
BC_GRIPPER_HOLDING_STACK_POSITION	SEALED_THICKNESS
BC_GRIPPER_OPEN_POSITION	SENSOR_INTENSITY
BC_ROBOT_GRIPPER_OFFSET	SENSOR_OFFSET
BC_SENSOR_OFFSET	SENSOR_THRESHOLD
BC_STACKER_GRIPPER_OFFSET	SENSOR_THRESHOLD_MIN
BRAVO_ROBOT_GRIPPER_OFFSET	SHIM_THICKNESS
CAN_BE_MOUNTED	STACKER_GRIPPER_OFFSET
CAN_BE_SEALED	STACKING_THICKNESS
CAN_HAVE_LID	THICKNESS
CAN_MOUNT	TIPBOX_SOURCE
CHECK_PLATE_ORIENTATION	TIP_CAPACITY
DESCRIPTION	USE_VACUUM_CLAMP
DISPOSABLE_TIP_LENGTH	WELL_BOTTOM_SHAPE
FILTER_TIP_PIN_TOOL_LENGTH	WELL_DEPTH
H12_NOTCH	WELL_DIAMETER
H1_NOTCH	WELL_GEOMETRY
IMAGE_FILENAME	WELL_TIP_VOLUME
LIDDED_STACKING_THICKNESS	X_TEACHPOINT_TO_WELL
LIDDED_THICKNESS	X_WELL_TO_WELL
LID_DEPARTURE_HEIGHT	Y_TEACHPOINT_TO_WELL
LID_RESTING_HEIGHT	Y_WELL_TO_WELL
LOWER_PLATE_AT_VCODE	Z_TIP_ATTACH_OFFSET

<Go Back

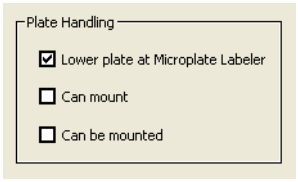
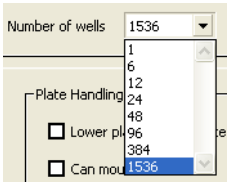
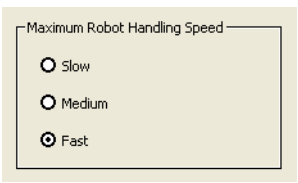
Labware properties in the Windows registry

A labware's properties are listed in the Windows registry. The following figure shows the list of properties for the 1536 Greiner 782076 blk sqr well flt btm labware type.

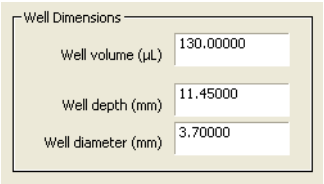
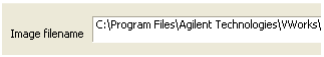


Labware Editor property value types

The value type of the `Parameter` element's `Value` element is determined by its format in the Labware Editor, as defined in the following table:

UI element	Labware Editor component	Value type
Check box		Boolean value, where `0` = unchecked and `1` = checked
Drop-down menu		Text string value of the menu item
Radio button		Integer value of the radio button starting from `1` (top-to-bottom, left-to-right) In the example, the value is `3`.

<Go Back

UI element	Labware Editor component	Value type
Text box (numbers)		Integer or float value
Text box (text)		Text string value

Example of a Labware query response (truncated)

The following sample code shows a truncated Labware query response that returns the labware entry for the `1536 Greiner 782076 blk sqr well flt btm` labware type.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='QueryResponse' md5sum='38ed2376cd4c63610cace3aac066eeaf'
→version='1.0' >
   <Response Category='Labware' Destination='IWorksController Test - 1' >
      <Parameters >
         <Parameter Name='Labware_Entry' Scriptable='1' Style='0' Type='1'
→Value='1536 Greiner 782076 blk sqr well flt btm' />
         <Parameter Scriptable='1' Style='0' Type='1' Value='0' />
         <Parameter Name='3RD_PARTY_TIP_CAPACITY' Scriptable='1' Style='0' Type='1'
→Value='60' />
         <Parameter Name='A12_NOTCH' Scriptable='1' Style='0' Type='1' Value='0' />
         <Parameter Name='A1_NOTCH' Scriptable='1' Style='0' Type='1' Value='1' />
         ...
         <Parameter Name='Y_TEACHPOINT_TO_WELL' Scriptable='1' Style='0' Type='1'
→Value='3.37500' />
         <Parameter Name='Y_WELL_TO_WELL' Scriptable='1' Style='0' Type='1'
→Value='2.25000' />
         <Parameter Name='Z_TIP_ATTACH_OFFSET' Scriptable='1' Style='0' Type='1'
→Value='-1.00000' />
         <Parameter Name='NAME' Scriptable='1' Style='0' Type='1'
→Value='1536 Greiner 782076 blk sqr well flt btm' />
      </Parameters>
   </Response>
</Velocity11>
```

LocationInformation query/response

LocationInformation query

The LocationInformation query requests the name of the configured labware at the specified location on the device. The plugin can use this information, for example, to check for possible errors during protocol compilation.

If the specified location is a stack location, the allowable stack height is also returned in the query response. The plugin can use this information, for example, during protocol execution to determine whether a stack location is full.

<Go Back

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Query Category='LocationInformation'>
      <Parameters>
         <Parameter Name='LocationName' ... />
      </Parameters>
   </Query>
</Velocity11>
```

Parameters element

The `Parameters` element contains one `Parameter` element.

Parameter element

The `Parameter` element has the following attributes plus the Scriptable, Style, and Type attributes:

Name	Value
`Name`	The value `LocationName`. Required: Yes
`Value`	The name of the location on the device. Required: Yes

Example of a LocationInformation query

The following sample code is a LocationInformation query that requests the name of the configured labware at the location named `Stage 1`.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Query' md5sum='78abc5774e120904e7be8581354966ec' version='1.0' >
   <Query Category='LocationInformation' >
      <Parameters >
         <Parameter Name='LocationName' Scriptable='1' Style='0' Type='1'
→Value='Stage 1' />
      </Parameters>
   </Query>
</Velocity11>
```

LocationInformation query response

If configured labware is present at the location, the LocationInformation query response contains the name of the configured labware at the specified location.

If the specified location is a stack location, the query response also contains the allowable stack height.

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Response Category='LocationInformation' ...>
      <Parameters>
         <Parameter Name='PlateStackHeight' ... />
         <Parameter Name='Labware' ... />
      </Parameters>
   </Response>
</Velocity11>
```

<Go Back

Parameters element

The Parameters element contains two Parameter elements.

Parameter element

Each Parameter element has one of the following pairs of Name and Value attributes plus the Scriptable, Style, and Type attributes:

Name	Value
Name	The value PlateStackHeight.
Value	The stack height, in millimeters. This Name and Value pair is only specified for stack locations.
Name	The value Labware.
Value	The labware type. This attribute is not specified if a configured labware is not present at the specified location.

Example of a LocationInformation query response (stack location)

The following LocationInformation query response returns the name of the static labware, 1536 Black Greiner, that is placed at the location specified in the query, along with the stack height of 460 (millimeters).

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='QueryResponse' md5sum='95a109fbd8fb938af71edb3152d9ef03'
→version='1.0' >
   <Response Category='LocationInformation' Destination='IWorksController Test - 1' >
      <Parameters >
         <Parameter Name='PlateStackHeight' Scriptable='1' Style='0' Type='12'
→Value='460' />
         <Parameter Name='Labware' Scriptable='1' Style='0' Type='1'
→Value='1536 Black Greiner' />
      </Parameters>
   </Response>
</Velocity11>
```

Example of a LocationInformation query response (not a stack location, configured labware present)

The following LocationInformation query response returns the type of the configured labware named 1536 Black Greiner that is placed at the location specified in the query.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='QueryResponse' md5sum='95a109fbd8fb938af71edb3152d9ef03'
→version='1.0' >
   <Response Category='LocationInformation' Destination='IWorksController Test - 1' >
      <Parameters >
         <Parameter Name='Labware' Scriptable='1' Style='0' Type='1'
→Value='1536 Black Greiner' />
      </Parameters>
   </Response>
</Velocity11>
```

<Go Back

Example of a LocationInformation query response (not a stack location, no configured labware present)

The following LocationInformation query response returns a LocationInformation Response XML block. Because no configured labware is present, the `Parameter` element's `Value` attribute is not specified.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='QueryResponse' md5sum='95a109fbd8fb938af71edb3152d9ef03'
→version='1.0' >
   <Response Category='LocationInformation' Destination='IWorksController Test - 1' >
      <Parameters >
         <Parameter Name='Labware' Scriptable='1' Style='0' Type='1'/>
      </Parameters>
   </Response>
</Velocity11>
```

LocationToTeachpoints query/response

LocationToTeachpoints query

The LocationToTeachpoints query requests the names of all the teachpoints that have been set at the specified location. The plugin can use this information, for example, to determine how many robots are able to access a certain location.

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Query Category='LocationToTeachpoints'>
      <Parameters>
         <Parameter Name='LocationName' ... />
      </Parameters>
   </Query>
</Velocity11>
```

Parameters element

The `Parameters` element contains one `Parameter` element.

Parameter element

The `Parameter` element has the following attributes plus the Scriptable, Style, and Type attributes:

Name	Value
`Name`	The value `LocationName`. Required: Yes
`Value`	The name of the location on the device. Required: Yes

<Go Back

Example of a LocationToTeachpoints query

The following sample code is a LocationToTeachpoints query that requests the names of all teachpoints that have been set at the location named `Stage 1`.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Query' md5sum='a2977dbfca4f3f4313ce0adf2768b0da' version='1.0' >
   <Query Category='LocationToTeachpoints' >
      <Parameters >
         <Parameter Name='LocationName' Scriptable='1' Style='0' Type='1'
→Value='Stage 1' />
      </Parameters>
   </Query>
</Velocity11>
```

LocationToTeachpoints query response

The LocationToTeachpoints query response contains the names of all teachpoints that have been set at the location specified in the query.

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Response Category='LocationToTeachpoints' ...>
      <Parameters>
         <Parameter />
      </Parameters>
   </Response>
</Velocity11>
```

Parameters element

The `Parameters` element contains one `Parameter` element.

Parameter element

The `Parameter` element has the following attributes plus the Scriptable, Style, and Type attributes:

Name	Value
`Name`	The value `DeviceLocationTeachpoints`.
`Value`	An escaped DeviceLocationTeachpoints XML block.

DeviceLocationTeachpoints XML block

The DeviceLocationTeachpoints XML block contains the `DeviceLocationTeachpoints` element and all its children. This XML block contains the names of all teachpoints that are set at a location.

<Go Back

XML structure

The value of the `file` attribute for the `Velocity11` element is `MetaData`. See "Velocity11 element" on page 416.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <DeviceLocationTeachpoints>
      <DeviceLocationTeachpoints>
         <DeviceLocationTeachpoint />
         ...
      </DeviceLocationTeachpoints>
   </DeviceLocationTeachpoints>
</Velocity11>
```

DeviceLocationTeachpoints element (parent)

The `DeviceLocationTeachpoints` parent element contains one `DeviceLocationTeachpoints` child element.

DeviceLocationTeachpoints element (child)

The `DeviceLocationTeachpoints` child element contains one or more `DeviceLocationTeachpoint` elements.

DeviceLocationTeachpoint element

The `DeviceLocationTeachpoint` element has the following attributes:

Name	Value
`DeviceName`	The device name.
`DeviceType`	The device type.
`LocationName`	The name of the location on the device.
`RobotName`	The robot name.
`TeachpointName`	The teachpoint name.

<Go Back

Example of a LocationToTeachpoints query response

The following sample code is a LocationToTeachpoints query response that returns an escaped DeviceLocationTeachpoints XML block. This XML block contains the names of the following teachpoints at the location named `Stage 1`: `Teachpoint 1`, `Teachpoint 2`, and `Teachpoint 3`.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='QueryResponse' md5sum='d3b9ef7b0797fecbf324f813a8cb0c0a'
→version='1.0' >
   <Response Category='LocationToTeachpoints'
→Destination='IWorksController Test - 1' >
      <Parameters >
         <Parameter Name='DeviceLocationTeachpoints' Scriptable='1' Style='0' Type='1'
→Value='&lt;?xml version=&apos;1.0&apos; encoding=&apos;ASCII&apos; ?&gt;
→&lt;Velocity11 file=&apos;MetaData&apos;
→md5sum=&apos;d2c8c47c4e820342ae53818cb5201de5&apos; version=&apos;1.0&apos; &gt;
→&lt;DeviceLocationTeachpoints &gt; &lt;DeviceLocationTeachpoints &gt;
→&lt;DeviceLocationTeachpoint DeviceName=&apos;IWorksController Test - 1&apos;
→DeviceType=&apos;IWorksController Test&apos; LocationName=&apos;Stage 1&apos;
→RobotName=&apos;3-Axis Robot - 1&apos; RobotType=&apos;IWorksController Test&apos;
→TeachpointName=&apos;Teachpoint 1&apos; /&gt; &lt;DeviceLocationTeachpoint
→DeviceName=&apos;IWorksController Test - 1&apos; DeviceType=&apos;IWorksController
→Test&apos; LocationName=&apos;Stage 1&apos; RobotName=&apos;DDR - 1&apos;
→RobotType=&apos;IWorksController Test&apos; TeachpointName=&apos;Teachpoint 2&apos;
→/&gt; &lt;DeviceLocationTeachpoint DeviceName=&apos;IWorksController Test - 1&apos;
→DeviceType=&apos;IWorksController Test&apos; LocationName=&apos;Stage 1&apos;
→RobotName=&apos;Phantom Robot - 1&apos; RobotType=&apos;Phantom Robot&apos;
→TeachpointName=&apos;Teachpoint 3&apos; /&gt;
→&lt;/DeviceLocationTeachpoints&gt; &lt;/DeviceLocationTeachpoints&gt;
→&lt;/Velocity11&gt;' />
      </Parameters>
   </Response>
</Velocity11>
```

Un-escaped DeviceLocationTeachpoints XML block

The following code is the un-escaped DeviceLocationTeachpoint XML block from the LocationToTeachpoints query response example.

```
<?xml version='1.0' encoding='us-ascii' ?>
<Velocity11 file='MetaData' md5sum='d2c8c47c4e820342ae53818cb5201de5' version='1.0' >
   <DeviceLocationTeachpoints >
      <DeviceLocationTeachpoints >
         <DeviceLocationTeachpoint DeviceName='Plate Sealer' DeviceType='PlateLoc'
→LocationName='Stage 1' RobotName='3-Axis Robot - 1' RobotType='3-Axis Robot'
→TeachpointName='Teachpoint 1' />
         <DeviceLocationTeachpoint DeviceName='Plate Sealer' DeviceType='PlateLoc'
→LocationName='Stage 1' RobotName='DDR - 1' RobotType='Direct Drive Robot'
→TeachpointName='Teachpoint 2' />
         <DeviceLocationTeachpoint DeviceName='Plate Sealer' DeviceType='PlateLoc'
→LocationName='Stage 1' RobotName='Phantom Robot - 1' RobotType='Phantom Robot'
→TeachpointName='Teachpoint 3' />
      </DeviceLocationTeachpoints>
   </DeviceLocationTeachpoints>
</Velocity11>
```

PlateVolume query/response

PlateVolume query

The PlateVolume query requests the current volume of all the wells in the labware at the specified location.

<Go Back

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Query Category='PlateVolume'>
      <Parameters>
         <Parameter Name='LocationInfo' />
      </Parameters>
   </Query>
</Velocity11>
```

Parameters element

The `Parameters` element contains one `Parameter` element.

Parameter element

The `Parameter` element has the following attributes plus the Scriptable, Style, and Type attributes:

Name	Value
`Name`	The value `LocationInfo`. Required: Yes
`Value`	An escaped `VolumeUpdates` element. Required: Yes

VolumeUpdates element

The `VolumeUpdates` element contains the name of the specified location. This element has the following attributes:

Name	Value
`Location`	The name of the location on the device. Required: Yes
`ResetAbsolute`	VWorks software ignores this attribute, so it should be set to `0`.

Example of a PlateVolume query

The following sample code is a PlateVolume query that returns an escaped VolumeUpdates XML element. This XML element contains the specified location.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Query' md5sum='6f4350e79cc7e48b7f7fec7925a857de' version='1.0'>
   <Query Category='PlateVolume'>
      <Parameters>
         <Parameter Name='LocationInfo' Scriptable='1' Style='0' Type='1'
→Value='&lt;?xml version=&apos;1.0&apos; encoding=&apos;ASCII&apos; ?&gt;
→&lt;Velocity11 file=&apos;MetaData&apos;
→md5sum=&apos;79b157293c2417174d4fbc6ca5cb98c3&apos; version=&apos;1.0&apos; &gt;
→&lt;VolumeUpdates Location=&apos;Stage 1&apos; ResetAbsolute=&apos;0&apos; /&gt;
→&lt;/Velocity11&gt;' />
      </Parameters>
   </Query>
</Velocity11>
```

<Go Back

Un-escaped VolumeUpdates XML block

The following code is the un-escaped `VolumeUpdates` element from the previous PlateVolume query example.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='79b157293c2417174d4fbc6ca5cb98c3' version='1.0' >
   <VolumeUpdates Location='Stage 1' ResetAbsolute='0' />
</Velocity11>
```

PlateVolume query response

The PlateVolume query response contains the current volume of all the wells in the labware at the location specified in the query. The response also tells the plugin how to set the volume change.

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Response Category='PlateVolume' ...>
      <Parameters>
         <Parameter Name='PlateVolume' ... />
      </Parameters>
   </Response>
</Velocity11>
```

Parameters element

The `Parameters` element contains one `Parameter` element.

Parameter element

The `Parameter` element has the following attributes plus the Scriptable, Style, and Type attributes:

Name	Value
`Name`	The value `PlateVolume`.
`Value`	An escaped `VolumeUpdates` XML block.

VolumeUpdates XML block

The VolumeUpdates XML block contains the `VolumeUpdates` element and all its children. This XML block provides the volume change for all the wells in the labware and tells the plugin how to set the volume change.

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <VolumeUpdates>
      <VolumeUpdates>
         <VolumeUpdate />
         ...
      </VolumeUpdates>
   </VolumeUpdates>
</Velocity11>
```

<Go Back

VolumeUpdates element (parent)

The `VolumeUpdates` parent element contains one `VolumeUpdates` child element. The parent element has the following attribute:

Name	Value
`ResetAbsolute`	Indicates how to set the volume change specified in the `VolumeUpdate` element. Possible values: `0` = Set the volume change to the current volume plus the volume change specified in the `VolumeUpdate` element `1` = Set the volume change to the volume specified in the `VolumeUpdate` element

VolumeUpdates element (child)

The `VolumeUpdates` child element contains one or more `VolumeUpdate` elements.

VolumeUpdate element

The `VolumeUpdate` element contains the coordinates of the well and the volume change. This element has the following attributes:

Name	Value
`Col`	The column coordinate of the well.
`Row`	The row coordinate of the well.
`VolumeChange`	The volume change, in microliters.

Example of a PlateVolume query response

The following sample code is a PlateVolume query response that returns an escaped VolumeUpdates XML block. This XML block contains the current volume of the six wells in the labware at the location specified in the query.

<Go Back

The query response also tells the plugin to set the volume change to the current volume plus the volume change specified in the `VolumeUpdate` element.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='QueryResponse' md5sum='6e2eb1a9cdc6a4fe367f6fdd6ec9d8df'
version='1.0' >
   <Response Category='PlateVolume' Destination='IWorksController Test - 1' >
      <Parameters >
         <Parameter Name='PlateVolume' Scriptable='1' Style='0' Type='1'
→Value='&lt;?xml version=&apos;1.0&apos; encoding=&apos;ASCII&apos; ?&gt;
→&lt;Velocity11 file=&apos;MetaData&apos;
→md5sum=&apos;485b3ebca3e27731dcb098a2727e6a9a&apos; version=&apos;1.0&apos;  &gt;
→&lt;VolumeUpdates ResetAbsolute=&apos;0&apos; &gt; &lt;VolumeUpdates &gt;
→&lt;VolumeUpdate Col=&apos;0&apos; Row=&apos;0&apos; VolumeChange=&apos;10&apos;
→/&gt; &lt;VolumeUpdate Col=&apos;1&apos; Row=&apos;0&apos;
→VolumeChange=&apos;10&apos; /&gt; &lt;VolumeUpdate Col=&apos;2&apos;
→Row=&apos;0&apos; VolumeChange=&apos;10&apos; /&gt; &lt;VolumeUpdate
→Col=&apos;0&apos; Row=&apos;1&apos; VolumeChange=&apos;10&apos; /&gt;
→&lt;VolumeUpdate Col=&apos;1&apos; Row=&apos;1&apos; VolumeChange=&apos;10&apos;
→/&gt; &lt;VolumeUpdate Col=&apos;2&apos; Row=&apos;1&apos;
→VolumeChange=&apos;10&apos; /&gt; &lt;/VolumeUpdates&gt; &lt;/VolumeUpdates&gt;
→&lt;/Velocity11&gt;' />
      </Parameters>
   </Response>
</Velocity11>
```

Un-escaped VolumeUpdates XML block

The following code is the un-escaped VolumeUpdates XML block from the previous PlateVolume query response example.

```
<?xml version='1.0' encoding='us-ascii' ?>
<Velocity11 file='MetaData' md5sum='485b3ebca3e27731dcb098a2727e6a9a' version='1.0' >
   <VolumeUpdates ResetAbsolute='0' >
      <VolumeUpdates >
         <VolumeUpdate Col='0' Row='0' VolumeChange='10' />
         <VolumeUpdate Col='1' Row='0' VolumeChange='10' />
         <VolumeUpdate Col='2' Row='0' VolumeChange='10' />
         <VolumeUpdate Col='0' Row='1' VolumeChange='10' />
         <VolumeUpdate Col='1' Row='1' VolumeChange='10' />
         <VolumeUpdate Col='2' Row='1' VolumeChange='10' />
      </VolumeUpdates>
   </VolumeUpdates>
</Velocity11>
```

ScanBarcode query/response

ScanBarcode query

The plugin uses the ScanBarcode query to determine whether a barcode scanner should be used to scan the barcode on the specified side of the labware at the specified location.

<Go Back

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Query Category='ScanBarcode'>
      <Parameters>
         <Parameter Name='Location' ... />
         <Parameter Name='Side' ... />
      </Parameters>
   </Query>
</Velocity11>
```

Parameters element

The `Parameters` element contains two `Parameter` elements.

Parameter element

Each `Parameter` element has one of the following pairs of `Name` and `Value` attributes plus the Scriptable, Style, and Type attributes:

Name-Value	Value
`Name`	The value `Location`. Required: Yes
`Value`	The name of the location on the device. Required: Yes
`Name`	The value `Side`. Required: Yes

`Value`	Represents the side of the labware.
	Possible values:
	`0 = South`
	`1 = West`
	`2 = North`
	`3 = East`
	Required: Yes

Example of a ScanBarcode query

The following sample code is a ScanBarcode query that the plugin uses to determine whether a barcode scanner should be used to scan the barcode on the south side of the labware at the location named `Stage 1`.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Query' md5sum='1ec491db4c521e60d87552991b808b2d' version='1.0' >
   <Query Category='ScanBarcode' >
      <Parameters >
         <Parameter Name='Location' Scriptable='1' Style='0' Type='1'
→Value='Stage 1' />
         <Parameter Name='Side' Scriptable='1' Style='0' Type='1' Value='0' />
      </Parameters>
   </Query>
</Velocity11>
```

<Go Back

ScanBarcode query response

The ScanBarcode query response indicates whether a barcode scanner should be used to scan the specified side of the barcode at the location specified in the query.

The response depends on the values of the Barcode information parameters as follows:

- If the value of the `Barcode or header <side>` parameter is `Barcode not in file`, the barcode scanner can scan the barcode on that side of the labware at the specified location.
- If the `Barcode or header <side>` parameter has a value other than `Barcode not in file`, such as `No selection`, the barcode scanner should not scan the barcode on that side of the labware at the specified location.

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Response Category='ScanBarcode' ...>
      <Parameters>
         <Parameter Name='ShouldScan' ... />
      </Parameters>
   </Response>
</Velocity11>
```

Parameters element

The `Parameters` element contains one `Parameter` element.

Parameter element

The `Parameter` element has the following attributes plus the Scriptable, Style, and Type attributes:

Name	Value
`Name`	The value `ShouldScan`.
`Value`	Indicates whether a barcode scanner should be used to scan the barcode. Possible values: `no` = A barcode scanner should not be used `yes` = A barcode scanner should be used

<Go Back

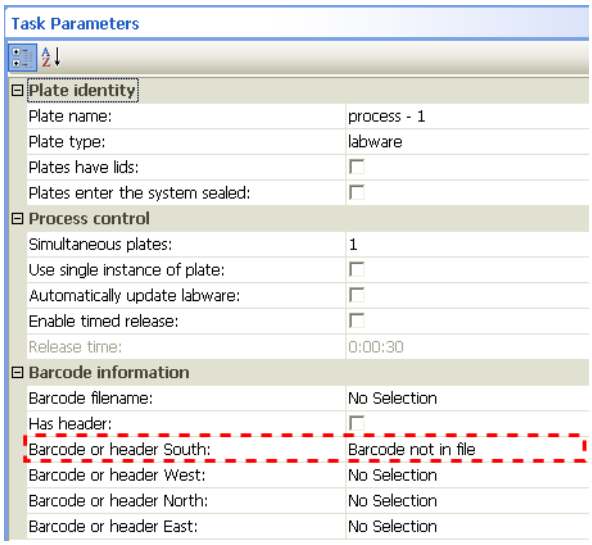
Example of a ScanBarcode query response

The following sample code is a ScanBarcode query response. The response indicates that a barcode scanner should be used to scan the barcode on the specified side of the labware at the location specified in the query.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='QueryResponse' md5sum='34e9f689ab92bb21be7c9075610ef9d2'
→version='1.0' >
   <Response Category='ScanBarcode' Destination='IWorksController Test - 1' >
      <Parameters >
         <Parameter Name='ShouldScan' Scriptable='1' Style='0' Type='1' Value='yes' />
      </Parameters>
   </Response>
</Velocity11>
```

In the "Example of a ScanBarcode query" on page 338, the Barcode information parameters were set to the values in the following figure. Therefore, the value of the `ShouldScan` parameter in the query response is `yes` for the south side of the labware, as in this query response example. The `Response` element for any other side of the labware is empty.

Figure Barcode information parameters



SystemPlateInformation query/response

SystemPlateInformation query

The SystemPlateInformation query requests the labware type for the specified labware.

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Query Category='SystemPlateInformation'>
      <Parameters>
         <Parameter Name='PlateName' ... />
      </Parameters>
   </Query>
</Velocity11>
```

<Go Back

Parameters element

The `Parameters` element contains one `Parameter` element.

Parameter element

The `Parameter` element has the following attributes plus the Scriptable, Style, and Type attributes:

Name	Value
`PlateName`	The value `PlateName`. Required: Yes
`Value`	The labware name. Required: Yes

Example of a SystemPlateInformation query

The following sample code is a SystemPlateInformation query that requests the labware type for the labware named `process - 1`.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Query' md5sum='952b139e0b1a951ec4364faf2cd9555f' version='1.0' >
   <Query Category='SystemPlateInformation' >
      <Parameters >
         <Parameter Name='PlateName' Scriptable='1' Style='0' Type='1'
→Value='process - 1' />
      </Parameters>
   </Query>
</Velocity11>
```

SystemPlateInformation query response

The SystemPlateInformation query response contains the labware type for the labware specified in the query.

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Response Category='SystemPlateInformation' ...>
      <Parameters>
         <Parameter />
      <Parameters>
   </Response>
</Velocity11>
```

Parameters element

The `Parameters` element contains one `Parameter` element.

Parameter element

The `Parameter` element has the following attributes plus the Scriptable, Style, and Type attributes:

Name	Value
`Name`	The value `Labware`.

<Go Back

Name	Value
Value	The labware type.

Example of a SystemPlateInformation query response

The following sample code is a SystemPlateInformation query response that returns the labware of type `1536 Greiner 782076 blk sqr well flt btm` for the labware specified in the query.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='QueryResponse' md5sum='34e9f689ab92bb21be7c9075610ef9d2'
→version='1.0' >
   <Response Category='SystemPlateInformation'
→Destination='IWorksController Test - 1' >
      <Parameters >
         <Parameter Name='Labware' Scriptable='1' Style='0' Type='1'
→Value='1536 Greiner 782076 blk sqr well flt btm' />
      </Parameters>
   </Response>
</Velocity11>
```

TeachpointInformation query/response

When the IWorksController `Query` method is used in conjunction with the IWorksDriver `ControllerQuery` method, the value of the `Category` attribute for the `Query` and `Response` elements is usually the same. (See "ControllerQuery method" on page 29.) However, the TeachpointInformation query/response uses different values for the `Category` attribute, as shown in the following table:

Interface	Method	Category
IWorksController	`Query`	`TeachpointInformation`
IWorksDriver	`ControllerQuery`	`TeachpointValue`

Interplugin communication for the TeachpointInformation query/response is done as follows:

1. To send a query to Plugin B, Plugin A calls the IWorksController `Query` method using the `TeachpointInformation` category, and passes the query to VWorks software in the input parameter. Then the plugin waits for a reply.
2. VWorks software forwards the query to Plugin B by calling the IWorksDriver `ControllerQuery` method using the `TeachpointValue` category, and passing the query in the input parameter.
3. Plugin B returns the query response to VWorks software in the output parameter of the IWorksDriver `ControllerQuery` method using the `TeachpointValue` category.
4. VWorks software forwards the query response to Plugin A in the output parameter of the IWorksController `Query` method using the `TeachpointInformation` category.

<Go Back

TeachpointInformation query

The TeachpointInformation query requests the coordinates of the specified teachpoint for the specified robot. The plugin might use this information, for example, to determine whether teachpoints were set correctly.

The developers of the two plugins must define their own mutually agreed-upon values for the following attributes:

- The `Name` and `Value` attributes of the query `Parameter` element
- The `Name` and `Value` attributes of the response `Parameter` elements in the Response XML block

In addition, the plugin can use the TeachpointInformation query to request the teachpoint coordinates from another plugin, using VWorks software as the intermediary.

Note: The plugin can also get this information using the InterPlugin query. See "InterPlugin query/response" on page 319.

XML structure (IWorksController Query method)

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Query Category='TeachpointInformation'>
      <Parameters>
         <Parameter Name='RobotName' ... />
         <Parameter Name='TeachpointName' ... />
      </Parameters>
   </Query>
</Velocity11>
```

XML structure (IWorksDriver ControllerQuery method)

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Query Category='TeachpointValue'>
      <Parameters>
         <Parameter Name='RobotName' ... />
         <Parameter Name='TeachpointName' ... />
      </Parameters>
   </Query>
</Velocity11>
```

Parameters element

The `Parameters` element contains two `Parameter` elements.

Parameter element

Each `Parameter` element has one of the following pairs of `Name` and `Value` attributes plus the Scriptable, Style, and Type attributes:

Name	Value
`Name`	The value `RobotName`. Required: Yes
`Value`	The robot name. Required: Yes

<Go Back

Name	Value
Name	The value TeachpointName. Required: Yes
Value	The teachpoint name. Required: Yes

Example of a TeachpointInformation query (IWorksController Query method)

The following sample code is a TeachpointInformation query that requests the coordinates of the teachpoint named Teachpoint 1 for the robot named IWorksController Test - 1. Note that the value of the Query element's Category attribute is TeachpointInformation.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Query' md5sum='f2ec1a2d1fe30b9d1def7b0488b31c98' version='1.0' >
   <Query Category='TeachpointInformation' >
      <Parameters >
         <Parameter Name='RobotName' Scriptable='1' Style='0' Type='1'
→Value='IWorksController Test - 1' />
         <Parameter Name='TeachpointName' Scriptable='1' Style='0' Type='1'
→Value='Teachpoint 1' />
      </Parameters>
   </Query>
</Velocity11>
```

Example of a TeachpointInformation query (IWorksDriver ControllerQuery method)

The following sample code is a TeachpointInformation query that requests the coordinates of the teachpoint named Teachpoint 1 for the robot named IWorksController Test - 1. Note that the value of the Query element's Category attribute is TeachpointValue.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Query' md5sum='f2ec1a2d1fe30b9d1def7b0488b31c98' version='1.0' >
   <Query Category='TeachpointValue' >
      <Parameters >
         <Parameter Name='RobotName' Scriptable='1' Style='0' Type='1'
→Value='IWorksController Test - 1' />
         <Parameter Name='TeachpointName' Scriptable='1' Style='0' Type='1'
→Value='Teachpoint 1' />
      </Parameters>
   </Query>
</Velocity11>
```

TeachpointInformation query response

The destination plugin returns its response to VWorks software, and then VWorks software returns the response to the source plugin in the TeachpointInformation query response. This query response returns the coordinates of the specified teachpoint for the robot that is specified in the query.

The developers of the two plugins must define their own mutually agreed-upon values for any Parameter element's Name and Value attributes. The developers must also define the number of Parameter elements that is needed to describe the teachpoint.

<Go Back

This section presents an example of how a TeachpointInformation query response might be returned from the destination plugin.

XML structure (IWorksController Query method)

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Response Category='TeachpointInformation' ...>
      <Parameters>
         <Parameter />
         ...
      <Parameters>
   </Response>
</Velocity11>
```

XML structure (IWorksDriver ControllerQuery method)

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Response Category='TeachpointValue' ...>
      <Parameters>
         <Parameter />
         ...
      <Parameters>
   </Response>
</Velocity11>
```

Parameters element

The `Parameters` element contains one or more `Parameter` elements.

Parameter element

The `Parameter` element has the following attributes plus the Scriptable, Style, and Type attributes:

Name	Value
`Name`	The coordinate name. Required: Yes
`Value`	The coordinate. Required: Yes

<Go Back

Example of a TeachpointInformation query response (IWorksController Query method)

The following sample code is a TeachpointInformation query response that returns the four coordinates of the teachpoint for the robot named `Robot - 1`. Note that the value of the `Response` element's `Category` attribute is `TeachpointInformation`.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='QueryResponse' md5sum='ea5c168a10a90440f3c8811ec910a189'
→version='1.0' >
   <Response Category='TeachpointInformation' Destination='Robot - 1' >
      <Parameters >
         <Parameter Name='Axis1' Scriptable='1' Style='0' Type='12' Value='1' />
         <Parameter Name='Axis2' Scriptable='1' Style='0' Type='12' Value='2' />
         <Parameter Name='Axis3' Scriptable='1' Style='0' Type='12' Value='3' />
         <Parameter Name='Axis4' Scriptable='1' Style='0' Type='12' Value='4' />
      </Parameters>
   </Response>
</Velocity11>
```

Example of a TeachpointInformation query response (IWorksDriver ControllerQuery method)

The following sample code is a TeachpointInformation query response that returns the four coordinates of the teachpoint for the robot named `Robot - 1`. Note that the value of the `Response` element's `Category` attribute is `TeachpointValue`.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='QueryResponse' md5sum='ea5c168a10a90440f3c8811ec910a189'
→version='1.0' >
   <Response Category='TeachpointValue' Destination='Robot - 1' >
      <Parameters >
         <Parameter Name='Axis1' Scriptable='1' Style='0' Type='12' Value='1' />
         <Parameter Name='Axis2' Scriptable='1' Style='0' Type='12' Value='2' />
         <Parameter Name='Axis3' Scriptable='1' Style='0' Type='12' Value='3' />
         <Parameter Name='Axis4' Scriptable='1' Style='0' Type='12' Value='4' />
      </Parameters>
   </Response>
</Velocity11>
```

Related information

For information about...	See...
IWorksDriver ControllerQuery method	"ControllerQuery method" on page 29
IControllerClient interface	"IControllerClient interface" on page 87

<Go Back

Update method

Overview

This section covers the `Update` method. Updates are divided in the following categories. See "Update element" on page 348 for a more complete list that includes descriptions.

- "AsyncTaskFinished update" on page 349
- "AsyncTaskStarted update" on page 350
- "Barcode update" on page 350
- "ErrorAbortRetryIgnoreNonBlocking update" on page 352
- "InventoryPlateBarcodes update" on page 358
- "LiquidTransferComplete update" on page 360
- "RackInfo update" on page 362
- "RunScript update" on page 363
- "RunsetAdd update" on page 365
- "SetIOManagerPointDigitalOutput update" on page 367
- "Volume update" on page 368

Description

The plugin calls the `Update` method to send information to VWorks software, to tell VWorks software to perform certain actions, or both. Using this method enables the plugin to perform tasks that cannot be done by means of existing VWorks software COM interface methods.

Syntax

```
HRESULT Update(
   [in] IControllerClient *Source,
   [in] BSTR Update
);
```

Parameters

`Source`	[in] The plugin's pointer to itself.
`Update`	[in] An Update XML block that contains information, a command for VWorks software, or both.

Update XML block

The Update XML block contains the `Update` element and all its children. This XML block returns information to VWorks software, tells VWorks software to perform certain actions, or both.

<Go Back

Velocity11 element

For all update XML blocks, the value of the `file` attribute for the `Velocity11` element is `Update`. See "Velocity11 element" on page 416.

Update element

The `Update` element specifies the type of update. This element has the following attribute:

Name	Value
`Category`	Represents the type of update. Possible values: • `AsyncTaskFinished` Tells VWorks software that an asynchronous task is completed. • `AsyncTaskStarted` Tells VWorks software that an asynchronous task is started. • `Barcode` Tells VWorks software to update the barcode value on the specified side of the labware at the specified location. • `ErrorAbortRetryIgnoreNonBlocking` Tells VWorks software than an error occurred while executing an asynchronous task. • `InventoryPlateBarcodes` Returns the results of the labware inventory, which was requested by VWorks software with a call to the IStorageDriver `QueryStorageLocations` method. See "QueryStorageLocations method" on page 192. • `LiquidTransferComplete` Tells VWorks software that a liquid transfer is completed. • `RackInfo` Tells VWorks software how many mismatches occurred during a rack check. • `RunScript` Tells VWorks software to execute an arbitrary JavaScript script. • `RunsetAdd` Tells VWorks software to schedule the specified protocol in the Runset Manager. • `SetIOManagerPointDigitalOutput` Tells VWorks software to output the specified value on the specified digital output channel. • `Volume` Returns the volume change of one or more wells in the labware at the specified location to VWorks software. Required: Yes

<Go Back

Update element's children

The `Update` element's children are defined in the "update" sections for each `Category` value.

AsyncTaskFinished update

The AsyncTaskFinished update tells VWorks software that the specified asynchronous task is finished. VWorks software can then remove the task from its asynchronous task management list.

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Update Category='AsyncTaskFinished'>
      <Parameters>
         <Parameter Name='Async_TaskVWorksID' ... />
      </Parameters>
   </Update>
</Velocity11>
```

Parameters element

The `Parameters` element contains one `Parameter` element.

Parameter element

The `Parameter` element has the following attributes plus the Scriptable, Style, and Type attributes:

Name	Value
`Name`	The value `Async_TaskVWorksID`. Required: Yes
`Value`	The task ID of the asynchronous task, which is automatically generated by VWorks software and used to identify and manage all asynchronous tasks from all plugins. Required: Yes

Example of an AsyncTaskFinished update

The following sample code is an AsyncTaskFinished update that tells VWorks software that the asynchronous task with VWorks software task ID `26.18.1` is completed.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Update' md5sum='7550eb75009c4678971e9dba0f07942b' version='1.0' >
   <Update Category='AsyncTaskFinished' >
      <Parameters >
         <Parameter Name='Async_TaskVWorksID' Scriptable='1' Style='0' Type='1'
→Value='26.18.1' />
      </Parameters>
   </Update>
</Velocity11>
```

<Go Back

AsyncTaskStarted update

The AsyncTaskStarted update tells VWorks software that the specified asynchronous task is started. VWorks software can then add the task to its asynchronous task management list and begin to manage the task.

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Update Category='AsyncTaskStarted'>
      <Parameters>
         <Parameter Name='Async_TaskVWorksID' ... />
      </Parameters>
   </Update>
</Velocity11>
```

Parameters element

The `Parameters` element contains one `Parameter` element.

Parameter element

The `Parameter` element has the following attributes plus the Scriptable, Style, and Type attributes:

Name	Value
`Name`	The value `Async_TaskVWorksID`. Required: Yes
`Value`	The task ID of the asynchronous task, which is automatically generated by VWorks software and used to identify and manage all asynchronous tasks from all plugins. Required: Yes

Example of an AsyncTaskStarted update

The following sample code is an AsyncTaskStarted update that tells VWorks software that the asynchronous task with VWorks software task ID `26.18.1` is started.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Update' md5sum='439fd6a65948d19f5bddda47ade97657' version='1.0' >
   <Update Category='AsyncTaskStarted' >
      <Parameters >
         <Parameter Name='Async_TaskVWorksID' Scriptable='1' Style='0' Type='1'
→Value='26.18.1' />
   </Update>
</Velocity11>
```

Barcode update

The Barcode update tells VWorks software to update the barcode on the specified side of the labware at the specified location as follows:

- When a labeler prints and applies a barcode, the plugin returns the new barcode to VWorks software in the Barcode update.

<Go Back

- When a barcode reader reads a barcode, the plugin returns the value to VWorks software in the Barcode update.

IMPORTANT VWorks software cannot call the `BarCodeRead` and `BarCodeMisread` methods unless the plugin implements the IVHooks interface. See "IVHooks interface" on page 197.

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Update Category='Barcode'>
      <Parameters>
         <Parameter Name='Location' ... />
         <Parameter Name='Side' ... />
         <Parameter Name='Barcode' ... />
      </Parameters>
   </Update>
</Velocity11>
```

Parameters element

The `Parameters` element contains three `Parameter` elements.

Parameter element

Each `Parameter` element has one of the following pairs of `Name` and `Value` attributes plus the Scriptable, Style, and Type attributes:

Name	Value
`Name`	The value `Location`. Required: Yes
`Value`	The name of the location on the device. Required: Yes
`Name`	The value `Side`. Required: Yes

`Value`	Represents the side of the labware.
	Possible values:
	`0 = South`
	`1 = West`
	`2 = North`
	`3 = East`
	Required: Yes

`Name`	The value `Barcode`. Required: Yes
`Value`	The barcode. Required: Yes

<Go Back

Example of a Barcode update

The following sample code is a Barcode update that tells VWorks software to update the south-side barcode associated with the labware at the location name `Stage 1`.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Update' md5sum='c0572a1ba800f689c0b787d41497b2a3' version='1.0' >
   <Update Category='Barcode' >
      <Parameters >
         <Parameter Name='Location' Scriptable='1' Style='0' Type='1'
→Value='Stage 1' />
         <Parameter Name='Side' Scriptable='1' Style='0' Type='1' Value='0' />
         <Parameter Name='Barcode' Scriptable='1' Style='0' Type='1'
→Value='barcode' />
      </Parameters>
   </Update>
</Velocity11>
```

ErrorAbortRetryIgnoreNonBlocking update

The ErrorAbortRetryIgnoreNonBlocking update tells VWorks software that an error occurred while executing an asynchronous task. The command metadata included in the update identifies the asynchronous task that returned the error. VWorks software responds to this update by displaying the error message to the user. The complete sequence of events is as follows:

1. The plugin calls the ErrorAbortRetryIgnoreNonBlocking update to notify VWorks software that an error occurred and to provide a literal string that describes the error.
2. VWorks software does the following:
   a. Writes the string to the Main Log.
   b. Displays the standard error dialog box, which includes the following components:
      - The error string
      - The Abort, Ignore and Continue..., Retry, and Diagnostics buttons

      The figure on page 40 shows a standard error dialog box.
3. The user clicks the Abort, Ignore and Continue..., Retry, or Diagnostics button.
4. VWorks software calls the appropriate IWorksAsyncDriver method, `Abort`, `Ignore`, or `Retry`, or the IWorksDiags `ShowDiagsDialog` method. See "Abort method" on page 265, "Ignore method" on page 271, and "Retry method" on page 274, and "ShowDiagsDialog method" on page 94.

The VWorks software error loop can only be terminated with a call to the IWorksAsyncDriver `Abort` method or [when VWorks software receives a return code other than `RETURN_FAIL`.]

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Update Category='ErrorAbortRetryIgnoreNonBlocking' >
      <Parameters >
         <Parameter Name='Async_CommandMetaData' ... />
      </Parameters>
   </Update>
</Velocity11>
```

<Go Back

Parameters element

The `Parameters` element contains one `Parameter` element.

Parameter element

The `Parameter` element has the following attributes plus the Scriptable, Style, and Type attributes:

Name	Value
`Name`	The value `Async_CommandMetaData`. Required: Yes
`Value`	An escaped Command XML block. Required: Yes

Command XML block (Shake task)

The Command XML block contains the `Command` element and all its children. This XML block provides the command metadata for an asynchronous Shake task during which the error that requires handling occurred.

XML structure

The value of the `file` attribute for the `Velocity11` element is `MetaData`. See "Velocity11 element" on page 416.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Command>
      <Parameters>
         <Parameter>
            <Ranges>
               <Range />
               ...
            </Ranges>
         </Parameter>
         ...
      </Parameters>
      <Locations>
         <Value />
         ...
      </Locations>
      <AsyncParameters>
         <AsyncParameter />
         ...
      </AsyncParameters>
   </Command>
</Velocity11>
```

Command element

The `Command` element has three children: `Parameters`, `Locations`, and `AsyncParameters`. The `Command` element has the following attributes:

Name	Value
`Compiler`	See "Compiler attribute" on page 399. Required: No Default value: `0`

<Go Back

Name	Value
Description	The task description. Required: Yes
Editor	See "Editor attribute" on page 400. Required: No Default value: 0
Name	The task name. Required: Yes
NextTaskToExecute	See "NextTaskToExecute attribute" on page 400. Required: No Default value: 1
ProtocolName	The name of the protocol that contains the task. If the protocol has been saved, the value of this attribute is the protocol's file path. If the protocol has not been saved, the value is the default protocol name. Required: Yes
RequiresRefresh	See "RequiresRefresh attribute" on page 402. Required: No Default value: 0
TaskRequiresLocation	See "TaskRequiresLocation attribute" on page 402. Required: No Default value: 1
VisibleAvailability	See "VisibleAvailability attribute" on page 403. Required: No Default value: 1

Parameters element

The Parameters element contains one or more Parameter elements.

Parameter element

The Parameter element contains one Ranges element and has the following attributes plus the Scriptable, Style, and Type attributes:

Name	Value
Description	The parameter description. Required: Yes
Name	The parameter name. Required: Yes

<Go Back

Name	Value
Value	See "Value attribute" on page 413. Required: No Default value: None

Ranges element

The Ranges element contains one or more Range elements.

Range element

The Range element has the following attribute:

Name	Value
Value	See "Value attribute" on page 415. Required: Yes

Locations element

The Locations element contains one of more Value elements.

Value element

Each Value element contains the name of a location that is used by the task. This element has the following attribute:

Name	Value
Value	The name of a location on the device. Required: Yes

AsyncParameters element

The AsyncParameters element contains five AsyncParameter elements.

AsyncParameter element

Each AsyncParameter element has one of the following pairs of Name and Value attributes plus the Style and Type attributes:

Name	Value
Name	The value Async_TaskVWorksID. Required: Yes
Value	The task ID of the asynchronous task, which is automatically generated by VWorks software and used by VWorks software to identify and manage all asynchronous tasks from all plugins. Required: Yes *Note:* The AsyncParameters element contains two Async_TaskWorksID elements.

<Go Back

Name	Value
`Name`	The value `Async_ErrorDescription`. Required: Yes
`Value`	The description of the error that occurred. Required: Yes
`Name`	The value `Async_TaskID`. Required: Yes
`Value`	The task ID of the asynchronous task, which is generated by the plugin and used to identify and manage its asynchronous tasks. Required: Yes
`Name`	The value `Async_Location`. Required: Yes
`Value`	The name of the location where the error occurred. Required: Yes

<Go Back

Example of an ErrorAbortRetryIgnoreNonBlocking update (truncated)

The following sample code shows an ErrorAbortRetryIgnoreNonBlocking update that contains a truncated escaped Update Command XML block. This update tells VWorks software to trigger task error handling because an error occurred during the asynchronous task named `Shake`, which has VWorks software task ID `25.4.1`.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Update' md5sum='cad68c2b476943428bd59f0e1ef1f2b3' version='1.0' >
   <Update Category='ErrorAbortRetryIgnoreNonBlocking' >
      <Parameters >
         <Parameter Name='Async_CommandMetaData' Scriptable='1' Style='0' Type='1'
→Value='&lt;?xml version=&apos;1.0&apos; encoding=&apos;ASCII&apos; ?&gt;
→&lt;Velocity11 file=&apos;MetaData&apos;
→md5sum=&apos;f1805177f41cea7b08356c675b25714d&apos; version=&apos;1.0&apos; &gt;
→&lt;Command Compiler=&apos;0&apos; Description=&apos;Shake plate&apos;
→Editor=&apos;4&apos; Name=&apos;Shake&apos; NextTaskToExecute=&apos;1&apos;
→ProtocolName=&apos;Protocol File - 1&apos; RequiresRefresh=&apos;0&apos;
→TaskRequiresLocation=&apos;1&apos; VisibleAvailability=&apos;1&apos; &gt;
→&lt;Parameters &gt; &lt;Parameter Name=&apos;Location&apos;
→Scriptable=&apos;1&apos; Style=&apos;0&apos; Type=&apos;5&apos; Value=&apos;6&apos;
→&gt; &lt;Ranges &gt; &lt;Range Value=&apos;process - 1&apos; /&gt; &lt;/Ranges&gt;
→&lt;/Parameter&gt; &lt;Parameter Description=&apos;Mode to operate in&apos;
→Name=&apos;Mode&apos; Scriptable=&apos;1&apos; Style=&apos;0&apos;
→Type=&apos;2&apos; Value=&apos;Timed&apos; &gt; &lt;Ranges &gt; &lt;Range
→Value=&apos;On&apos; /&gt; &lt;Range Value=&apos;Off&apos; /&gt; &lt;Range
→Value=&apos;Timed&apos; /&gt; &lt;/Ranges&gt;
...
→&lt;Parameter Name=&apos;Next task command name&apos; Scriptable=&apos;1&apos;
→Style=&apos;0&apos; Type=&apos;1&apos; /&gt; &lt;/Parameters&gt; &lt;Locations &gt;
→&lt;Value Value=&apos;6&apos; /&gt; &lt;/Locations&gt; &lt;AsyncParameters &gt;
→&lt;AsyncParameter Name=&apos;Async_TaskVWorksID&apos; Style=&apos;0&apos;
→Type=&apos;1&apos; Value=&apos;25.4.1&apos; /&gt; &lt;AsyncParameter
→Name=&apos;Async_ErrorDescription&apos; Style=&apos;0&apos; Type=&apos;1&apos;
→Value=&apos;Location 6 Orbital Shaking Station error: Could not set RPM. Make sure
→the device is properly connected and initialized.&apos; /&gt; &lt;AsyncParameter
→Name=&apos;Async_TaskID&apos; Style=&apos;0&apos; Type=&apos;1&apos;
→Value=&apos;1&apos; /&gt; &lt;AsyncParameter Name=&apos;Async_TaskVWorksID&apos;
→Style=&apos;0&apos; Type=&apos;1&apos; Value=&apos;25.4.1&apos; /&gt;
→&lt;AsyncParameter Name=&apos;Async_Location&apos; Style=&apos;0&apos;
→Type=&apos;1&apos; Value=&apos;6&apos; /&gt; &lt;/AsyncParameters&gt;
→&lt;/Command&gt; &lt;/Velocity11&gt;' />
      </Parameters>
   </Update>
</Velocity11>
```

<Go Back

Un-escaped Update Command XML block (truncated)

The following code is the truncated un-escaped Command XML block from the previous ErrorAbortRetryIgnoreNonBlocking update example.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='f1805177f41cea7b08356c675b25714d' version='1.0' >
   <Command Compiler='0' Description='Shake plate' Editor='4' Name='Shake'
→NextTaskToExecute='1' ProtocolName='Protocol File - 1' RequiresRefresh='0'
→TaskRequiresLocation='1' VisibleAvailability='1' >
      <Parameters >
         <Parameter Name='Location' Scriptable='1' Style='0' Type='5' Value='6' >
            <Ranges >
               <Range Value='process - 1' />
            </Ranges>
         </Parameter>
         <Parameter Description='Mode to operate in' Name='Mode' Scriptable='1'
→Style='0' Type='2' Value='Timed' >
            <Ranges >
               <Range Value='On' />
               <Range Value='Off' />
               <Range Value='Timed' />
            </Ranges>
         ...
         <Parameter Name='Next task command name' Scriptable='1' Style='0'
→Type='1' />
      </Parameters>
      <Locations >
         <Value Value='6' />
      </Locations>
      <AsyncParameters >
         <AsyncParameter Name='Async_TaskVWorksID' Style='0' Type='1'
→Value='25.4.1' />
         <AsyncParameter Name='Async_ErrorDescription' Style='0' Type='1'
→Value='Location 6 Orbital Shaking Station error: Could not set RPM.
→Make sure the device is properly connected and initialized.' />
         <AsyncParameter Name='Async_TaskID' Style='0' Type='1' Value='1' />
         <AsyncParameter Name='Async_TaskVWorksID' Style='0' Type='1'
→Value='25.4.1' />
         <AsyncParameter Name='Async_Location' Style='0' Type='1' Value='6' />
      </AsyncParameters>
   </Command>
</Velocity11>
```

InventoryPlateBarcodes update

VWorks software calls the IStorageDriver `QueryStorageLocations` method to tell the plugin to perform a labware inventory. The plugin returns the results of the inventory in the InventoryPlateBarcodes update. See "QueryStorageLocations method" on page 192.

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Update Category='InventoryPlateBarcodes'>
      <Barcode />
      ...
   </Update>
</Velocity11>
```

<Go Back

Barcode element

Each `Barcode` element contains information about one slot in the range of labware that was inventoried. This element has the following attributes:

Name	Value
`BarcodeError`	Indicates whether a barcode read error occurred. Possible values: `0` = No barcode read error occurred `1` = A barcode read error occurred Required: Yes If a read error occurred, VWorks software might report the error. Then the user must decide what to do, for example, check the labware in the slot that reported the error.
`Cassette`	The cassette number. Required: Yes
`PlatePresent`	Indicates whether a labware is present in the specified location. Possible values: `0` = No labware is present `1` = A labware is present Required: Yes
`Slot`	The slot number. Required: Yes
`Value`	The barcode. Required: Yes

Example of an InventoryPlateBarcodes update

The following sample code is an InventoryPlateBarcodes update that returns the results of an IStorageDriver `QueryStorageLocations` method call for three labware.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Update' md5sum='8656d5a7fe6cf8b348c270e4fc101dbb' version='1.0' >
   <Update Category='InventoryPlateBarcodes' >
      <Barcode BarcodeError='0' Cassette='1' PlatePresent='1' Slot='1'
→Value='barcode0001' />
      <Barcode BarcodeError='0' Cassette='1' PlatePresent='1' Slot='2'
→Value='barcode0002' />
      <Barcode BarcodeError='0' Cassette='1' PlatePresent='1' Slot='3'
→Value='barcode0003' />
   </Update>
</Velocity11>
```

<Go Back

If the device is unable to perform a labware inventory, the plugin sends an empty InventoryPlateBarcodes Update XML block to VWorks software, as shown in the following sample code. Then VWorks software generates an error and writes an error message to the Main Log, such as `Failed to update new barcode information: barcode0001`.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Update' md5sum='0b8eb2ad5df221c13e336af98ffec528' version='1.0' >
    <Update Category='InventoryPlateBarcodes' />
</Velocity11>
```

LiquidTransferComplete update

The LiquidTransferComplete update tells VWorks software that the liquid transfer is completed. The plugin might use this update, for example, to provide the new volumes after a Standard Transfer task is completed.

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Update Category='LiquidTransferComplete'>
      <Parameters>
         <Parameter Name='LiquidTransferComplete' ... />
      </Parameters>
   </Update>
</Velocity11>
```

Parameters element

The `Parameters` element contains one `Parameter` element.

Parameter element

The `Parameter` element has the following attributes plus the Scriptable, Style, and Type attributes:

Name	Value
`Name`	The value `LiquidTransferComplete`. Required: Yes
`Value`	An escaped LiquidTransferCompleteUpdate XML block. Required: Yes

LiquidTransferCompleteUpdate XML block

The LiquidTransferCompleteUpdate XML block contains the `LiquidTransferCompleteUpdate` element and all its children. This XML block provides information about the completed liquid transfer.

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <LiquidTransferCompleteUpdate>
      <PipetteHeadMode />
   </LiquidTransferCompleteUpdate>
</Velocity11>
```

<Go Back

LiquidTransferCompleteUpdate element

The `LiquidTransferCompleteUpdate` element contains one `PipetteHeadMode` element and has the following attributes:

Name	Value
`DestLocation`	The location of the labware to which the liquid was transferred. Required: Yes
`SourceLocation`	The location of the labware from which the liquid was transferred. Required: Yes

PipetteHeadMode element

See "PipetteHeadMode element" on page 414.

Example of a LiquidTransferComplete update

The following sample code is a LiquidTransferComplete update that tells VWorks software that the liquid transfer from the source location named `Source Stage` to the destination location named `Destination Stage` is completed.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Update' md5sum='52f0c0fc1135db949ab97541fbb57e6a' version='1.0'>
   <Update Category='LiquidTransferComplete'>
      <Parameters>
         <Parameter Name='LiquidTransferComplete' Scriptable='1' Style='0' Type='1'
→Value='&lt;?xml version=&apos;1.0&apos; encoding=&apos;ASCII&apos; ?&gt;
→&lt;Velocity11 file=&apos;MetaData&apos;
→md5sum=&apos;0fd609ab6ccef7e255a2916357326df9&apos; version=&apos;1.0&apos; &gt;
→&lt;LiquidTransferCompleteUpdate DestLocation=&apos;Destination Stage&apos;
→SourceLocation=&apos;Source Stage&apos; &gt; &lt;PipetteHeadMode
→Channels=&apos;0&apos; ColumnCount=&apos;12&apos; RowCount=&apos;8&apos;
→SubsetConfig=&apos;0&apos; SubsetType=&apos;0&apos; /&gt;
→&lt;/LiquidTransferCompleteUpdate&gt; &lt;/ Velocity11&gt;' />
      </Parameters>
   </Update>
</Velocity11>
```

Un-escaped LiquidTransferCompleteUpdate XML block

The following code is the un-escaped LiquidTransferCompleteUpdate XML block from the previous LiquidTransferComplete update example.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='0fd609ab6ccef7e255a2916357326df9' version='1.0' >
   <LiquidTransferCompleteUpdate DestLocation='Destination Stage'
→SourceLocation='Source Stage' >
      <PipetteHeadMode Channels='0' ColumnCount='12' RowCount='8' SubsetConfig='0'
→SubsetType='0' />
   </LiquidTransferCompleteUpdate>
</Velocity11>
```

<Go Back

RackInfo update

During a rack check, the plugin reads the barcodes and checks the positions of the tubes in a tube rack. Then the plugin compares the results of the rack check to the values in the barcode input file. If a barcode, a position, or both are not the expected value, a mismatch occurred.

The plugin calls the RackInfo update to tell VWorks software how many mismatches occurred during the rack check. VWorks software can store the number of mismatches in the JavaScript variable specified in the RackInfo update and use this variable in subsequent tasks.

Note: Plugins usually provide a task that performs the rack check.

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Update Category='RackInfo'>
      <Parameters>
         <Parameter Name='RackInfoType' ... />
         <Parameter Name='Location' ... />
         <Parameter Name='CheckRackResult' ... />
         <Parameter Name='MismatchesVariable' ... />
         <Parameter Name='Mismatches' ... />
         <Parameter Name='ProtocolName' ... />
      </Parameters>
   </Update>
</Velocity11>
```

Parameters element

The `Parameters` element contains five `Parameter` elements.

Parameter element

Each `Parameter` element has one of the following pairs of `Name` and `Value` attributes plus the Scriptable, Style, and Type attributes:

Name	Value
`Name`	The value `RackInfoType`. Required: Yes
`Value`	The only supported value for this attribute is `CheckRack`. Required: Yes
`Name`	The value `Location`. Required: Yes
`Value`	The name of the location on the device. Required: Yes
`Name`	The value `CheckRackResult`. Required: Yes
`Value`	Indicates whether any mismatches were found during the rack check. Possible values: `0` = One or more mismatches were found `1` = No mismatches were found Required: Yes

<Go Back

Name	Value
Name	The value MismatchesVariable. Required: No
Value	The name of a JavaScript variable set to the number of mismatches that occurred during the rack check. Required: No Default value: None
Name	The value Mismatches. Required: Yes
Value	The number of mismatches that occurred during the rack check. Required: Yes
Name	The value ProtocolName. Required: Yes
Value	If the protocol has been saved, the value of this attribute is the protocol's file path. If the protocol has not been saved, the value is the default protocol name. Required: Yes

Example of a RackInfo update

The following sample code is a RackInfo update that tells VWorks software that 26 mismatches occurred during a rack check at the location named Stage 1.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Update' md5sum='16fa4a78554e3d1f02d7198952c83e71' version='1.0' >
   <Update Category='RackInfo' >
      <Parameters >
         <Parameter Name='RackInfoType' Scriptable='1' Style='0' Type='1'
→Value='CheckRack' />
         <Parameter Name='Location' Scriptable='1' Style='0' Type='1'
→Value='Stage 1' />
         <Parameter Name='CheckRackResult' Scriptable='1' Style='0' Type='0'
→Value='0' />
         <Parameter Name='MismatchesVariable' Scriptable='1' Style='0' Type='1'
→Value='mismatches' />
         <Parameter Name='Mismatches' Scriptable='1' Style='0' Type='8' Value='26' />
         <Parameter Name='ProtocolName' Scriptable='1' Style='0' Type='1'
→Value='Protocol File - 1' />
      </Parameters>
   </Update>
</Velocity11>
```

RunScript update

The RunScript update tells VWorks software to execute an arbitrary JavaScript script, for example, to set the value of variable x to 1 (x=1). The plugin might use this update, for example, to control task execution indirectly by telling VWorks software to run a script that changes a variable value.

<Go Back

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Update Category='RunScript'>
      <Parameters>
         <Parameter Name='Script' ... />
         <Parameter Name='ProtocolName' ... />
      </Parameters>
   </Update>
</Velocity11>
```

Parameters element

The `Parameters` element contains two `Parameter` elements.

Parameter element

Each `Parameter` element has one of the following pairs of `Name` and `Value` attributes plus the Scriptable, Style, and Type attributes:

Name	Value
`Name`	The value `Script`. Required: Yes
`Value`	An escaped JavaScript script. Required: Yes
`Name`	The value `ProtocolName`. Required: Yes
`Value`	If the protocol has been saved, the value of this attribute is the protocol's file path. If the protocol has not been saved, the value is the default protocol name. Required: Yes

Example of a RunScript update

The following code is a RunScript update that tells VWorks software to run the following JavaScript script:

```
print("This is a testing for run script");a=1.
```

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Update' md5sum='711d59e03ab1db8e8d22806ef28b0900' version='1.0' >
   <Update Category='RunScript' >
      <Parameters >
         <Parameter Name='Script' Scriptable='1' Style='0' Type='1'
→Value='print(&quot;This is a testing for run script&quot;);a=1' />
         <Parameter Name='ProtocolName' Scriptable='1' Style='0' Type='1'
→Value='Protocol File - 1' />
      </Parameters>
   </Update>
</Velocity11>
```

<Go Back

Un-escaped JavaScript script

The following code is the un-escaped JavaScript script from the previous RunScript update example.

```
print("This is a test for run script");a=1
```

RunsetAdd update

The RunsetAdd update tells VWorks software to schedule a protocol run in the Runset Manager.

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Update Category='RunsetAdd'>
      <Parameters>
         <Parameter Name='ProtocolName' ... />
         <Parameter Name='StartTime_year' ... />
         <Parameter Name='StartTime_month' ... />
         <Parameter Name='StartTime_day' ... />
         <Parameter Name='StartTime_hour' ... />
         <Parameter Name='StartTime_minute' ... />
         <Parameter Name='StartTime_second' ... />
         <Parameter Name='NumberOfRuns' ... />
         <Parameter Name='PluginName' ... />
         <Parameter Name='PluginFileName' ... />
         <Parameter Name='ProtocolNotes' ... />
      </Parameters>
   </Update>
</Velocity11>
```

Parameters element

The `Parameters` element contains 11 `Parameter` elements.

Parameter element

Each `Parameter` element has one of the following pairs of `Name` and `Value` attributes plus the Scriptable, Style, and Type attributes:

Name	Value
`Name`	The value `ProtocolName`. Required: Yes
`Value`	If the protocol has been saved, the value of this attribute is the protocol's file path. If the protocol has not been saved, the value is the default protocol name. Required: Yes
`Name`	The value `StartTime_year`. Required: Yes
`Value`	The scheduled start year of the protocol (4 digits). Required: Yes
`Name`	The value `StartTime_month`. The scheduled start month of the protocol (`1–12`).
`Value`	Required: Yes

<Go Back

Name	Value
Name	The value StartTime_day. Required: Yes
Value	The scheduled start day of the protocol (1–31). Required: Yes
Name	The value StartTime_hour. Required: Yes
Value	The scheduled start hour of the protocol (1–24). Required: Yes
Name	The value StartTime_minutes. Required: Yes
Value	The scheduled start minutes of the protocol (0–59). Required: Yes
Name	The value StartTime_seconds. Required: Yes
Value	The scheduled start seconds of the protocol (0–59). Required: Yes
Name	The value NumberOfRuns. Required: Yes
Value	The number of times the protocol is scheduled to run. Required: Yes
Name	The value PluginName. Required: Yes
Value	The plugin name. Required: Yes
Name	The value PluginFileName. Required: Yes
Value	The file name of the plugin, for example, MyPlugin.dll. The value can be the same as PluginName. Required: Yes
Name	The value ProtocolNotes. Required: Yes
Value	Any notes about the protocol that were entered in the Run Configuration Wizard. Required: Yes

<Go Back

Example of a RunsetAdd update

The following sample code is a RunsetAdd update that tells VWorks software to add the protocol named `Protocol File - 1` to the Runset Manager and to run the protocol 10 times on April 15, 2010 (2010-04-15), at 8:29:11 p.m. (20:29:11).

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Update' md5sum='965062c18e91f461e97f8736b2cabd75' version='1.0' >
   <Update Category='RunsetAdd' >
      <Parameters >
         <Parameter Name='ProtocolName' Scriptable='1' Style='0' Type='1'
→Value='Protocol File - 1' />
         <Parameter Name='StartTime_year' Scriptable='1' Style='0' Type='8'
→Value='2010' />
         <Parameter Name='StartTime_month' Scriptable='1' Style='0' Type='8'
→Value='4' />
         <Parameter Name='StartTime_day' Scriptable='1' Style='0' Type='8'
→Value='15' />
         <Parameter Name='StartTime_hour' Scriptable='1' Style='0' Type='8'
→Value='20' />
         <Parameter Name='StartTime_minute' Scriptable='1' Style='0' Type='8'
→Value='29' />
         <Parameter Name='StartTime_second' Scriptable='1' Style='0' Type='8'
→Value='11' />
         <Parameter Name='NumberOfRuns' Scriptable='1' Style='0' Type='1'
→Value='10' />
         <Parameter Name='PluginName' Scriptable='1' Style='0' Type='1'
→Value='PluginName' />
         <Parameter Name='PluginFileName' Scriptable='1' Style='0' Type='1'
→Value='PluginFileName' />
         <Parameter Name='ProtocolNotes' Scriptable='1' Style='0' Type='1'
→Value='ProtocolNotes' />
      </Parameters>
   </Update>
</Velocity11>
```

SetIOManagerPointDigitalOutput update

The SetIOManagerPointDigitalOutput update tells VWorks software to output the specified value on the specified digital output channel. The plugin might use this update, for example, to configure a component or subcomponent that is controlled by a digital output signal, such as when the Vacuum Delid Station turns lid suction on or off.

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Update Category='SetIOManagerPointDigitalOutput'>
      <Parameters>
      <Parameters >
         <Parameter Name='PointName' ... />
         <Parameter Name='NewValue' ... />
      </Parameters>
   </Update>
</Velocity11>
```

Parameters element

The `Parameters` element contains two `Parameter` elements.

<Go Back

Parameter element

Each `Parameter` element has one of the following pairs of `Name` and `Value` attributes plus the Scriptable, Style, and Type attributes:

Name	Value
`Name`	The value `PointName`. Required: Yes
`Value`	The name of the digital output signal. Required: Yes
`Name`	The value `NewValue`. Required: Yes
`Value`	The new value for the digital output signal. Possible values: `0` = Off `1` = On Required: Yes

Example of a SetIOManagerPointDigitalOutput update

The following sample code is a SetIOManagerPointDigitalOutput update that tells VWorks software to set the value of the digital output signal named `Vacuum Delid Suction` to the value `1`.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Update' md5sum='85d73cc74a18b2d8e7495c633247e146' version='1.0' >
   <Update Category='SetIOManagerPointDigitalOutput' >
      <Parameters >
         <Parameter Name='PointName' Scriptable='1' Style='0' Type='1'
→Value='Vacuum Delid Suction' />
         <Parameter Name='NewValue' Scriptable='1' Style='0' Type='0' Value='1' />
      </Parameters>
   </Update>
</Velocity11>
```

Volume update

The Volume update provides the volume change for one or more specified wells in the labware at the specified location. The plugin might use this update, for example, to tell VWorks software to update the well volumes in the database after a liquid-handling task is completed.

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <Update Category='Volume'>
      <Parameters>
         <Parameter Name='VolumeChange' ... />
      </Parameters>
   </Update>
</Velocity11>
```

<Go Back

Parameters element

The `Parameters` element contains one `Parameter` element.

Parameter element

The `Parameter` element has the following attributes plus the Scriptable, Style, and Type attributes:

Name	Value
`Name`	The value `VolumeChange`. Required: Yes
`Value`	An escaped VolumeUpdates XML block. Required: Yes

VolumeUpdates XML block

The VolumeUpdates XML block contains the `VolumeUpdates` element and all its children. This XML block provides the volume change for all the wells in the labware at the specified location.

XML structure

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <VolumeUpdates>
      <VolumeUpdates>
         <VolumeUpdate />
         ...
      </VolumeUpdates>
   </VolumeUpdates>
</Velocity11>
```

Volume Updates element (parent)

The parent `VolumeUpdates` element contains one `VolumeUpdates` child element and has the following attributes:

Name	Value
`Location`	The name of the location of the labware. Required: Yes
`ResetAbsolute`	Indicates how to set the volume change specified in the `VolumeUpdate` element. Possible values: `0` = Set the volume change to the current volume plus the volume change specified in the `VolumeUpdate` element `1` = Set the volume change to the volume specified in the `VolumeUpdate` element Required: Yes

<Go Back

VolumeUpdates element (child)

The `VolumeUpdates` child element contains one or more `VolumeUpdate` elements.

VolumeUpdate element

The `VolumeUpdate` element contains the coordinates of the well and the volume change. This element has the following attributes:

Name	Value
`Col`	The column coordinate of the well. Required: Yes
`Row`	The row coordinate of the well. Required: Yes
`VolumeChange`	The volume change, in microliters. Required: Yes

Example of a Volume update

The following sample code is a Volume update that returns an escaped VolumeUpdates XML block. This XML block provides the volume change of 10 wells in the labware at the location named `Stage 1`. The update also tells VWorks software to set the volume change to the current volume plus the volume change specified in the `VolumeUpdate` element.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Update' md5sum='1ba0bdcfe46270ff7751a20d0ff74788' version='1.0'>
   <Update Category='Volume'>
      <Parameters>
         <Parameter Name='VolumeChange' Scriptable='1' Style='0' Type='1'
→Value='&lt;?xml version=&apos;1.0&apos; encoding=&apos;ASCII&apos; ?&gt;
→&lt;Velocity11 file=&apos;MetaData&apos;
→md5sum=&apos;2d50df57d9d719c31dabc56f0a8402ef&apos; version=&apos;1.0&apos; &gt;
→&lt;VolumeUpdates Location=&apos;Stage 1&apos; OptionalDevice=&apos;Optional
→Device&apos; ResetAbsolute=&apos;0&apos; &gt; &lt;VolumeUpdates &gt;
→&lt;VolumeUpdate Col=&apos;0&apos; Row=&apos;0&apos;
→VolumeChange=&apos;0.0001&apos; /&gt; &lt;VolumeUpdate Col=&apos;1&apos;
→Row=&apos;0&apos; VolumeChange=&apos;0.0001&apos; /&gt; &lt;VolumeUpdate
→Col=&apos;2&apos; Row=&apos;0&apos; VolumeChange=&apos;0.0001&apos; /&gt;
→&lt;VolumeUpdate Col=&apos;3&apos; Row=&apos;0&apos;
→VolumeChange=&apos;0.0001&apos; /&gt; &lt;VolumeUpdate Col=&apos;4&apos;
→Row=&apos;0&apos; VolumeChange=&apos;0.0001&apos; /&gt; &lt;VolumeUpdate
→Col=&apos;5&apos; Row=&apos;0&apos; VolumeChange=&apos;0.0001&apos; /&gt;
→&lt;VolumeUpdate Col=&apos;6&apos; Row=&apos;0&apos;
→VolumeChange=&apos;0.0001&apos; /&gt; &lt;VolumeUpdate Col=&apos;7&apos;
→Row=&apos;0&apos; VolumeChange=&apos;0.0001&apos; /&gt; &lt;VolumeUpdate
→Col=&apos;8&apos; Row=&apos;0&apos; VolumeChange=&apos;0.0001&apos; /&gt;
→&lt;VolumeUpdate Col=&apos;9&apos; Row=&apos;0&apos;
→VolumeChange=&apos;0.0001&apos; /&gt; &lt;/VolumeUpdates&gt; &lt;/
→VolumeUpdates&gt; &lt;/Velocity11&gt;' />
      </Parameters>
   </Update>
</Velocity11>
```

<Go Back

Un-escaped VolumeUpdates XML block

The following code is the un-escaped VolumeUpdates XML block from the previous Volume update example.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData' md5sum='2d50df57d9d719c31dabc56f0a8402ef' version='1.0' >
   <VolumeUpdates Location='Stage 1' ResetAbsolute='0' >
      <VolumeUpdates >
         <VolumeUpdate Col='0' Row='0' VolumeChange='0.0001' />
         <VolumeUpdate Col='1' Row='0' VolumeChange='0.0001' />
         <VolumeUpdate Col='2' Row='0' VolumeChange='0.0001' />
         <VolumeUpdate Col='3' Row='0' VolumeChange='0.0001' />
         <VolumeUpdate Col='4' Row='0' VolumeChange='0.0001' />
         <VolumeUpdate Col='5' Row='0' VolumeChange='0.0001' />
         <VolumeUpdate Col='6' Row='0' VolumeChange='0.0001' />
         <VolumeUpdate Col='7' Row='0' VolumeChange='0.0001' />
         <VolumeUpdate Col='8' Row='0' VolumeChange='0.0001' />
         <VolumeUpdate Col='9' Row='0' VolumeChange='0.0001' />
      </VolumeUpdates>
   </VolumeUpdates>
</Velocity11>
```

Related information

For information about...	See...
IVHooks BarCodeMisread method	“BarCodeMisread method” on page 205
IVHooks BarCodeRead method	“BarCodeRead method” on page 210
IControllerClient interface	“IControllerClient interface” on page 87
IStorageDriver QueryStorageLocations method	“QueryStorageLocations method” on page 192

<Go Back

19 IWorksProfiles interface

VWorks software provides a means for exporting protocol files, device files, and profile information to a single compressed file. This file can then be imported into VWorks on a different computer to recreate the configuration from the exporting computer. For VWorks plugins to export nonstandard registry and file data associated with a profile, they must implement the IWorksProfiles interface.

This chapter defines the IWorksProfiles methods.

IMPORTANT All VWorks device driver plugins must implement the IWorksDriver, IControllerClient, and IWorksDiags interfaces.

This chapter contains the following topics:

- "IWorksProfiles methods overview" on page 374
- "ExportXML method" on page 375

<Go Back

IWorksProfiles methods overview

Use the following table to quickly locate an IWorksProfiles method by name, by description, or by page number.

Method	Description	See...
`ExportXML`	Notifies the plugin that the user is exporting a profile.	"ExportXML method" on page 375
`GetConflictedProfileName`	*Reserved for internal use.* This method should be implemented as `return E_NOTIMPL (0x80004001)`.	
`MigrateProfile`	*Reserved for internal use.* This method should be implemented as `return E_NOTIMPL (0x80004001)`.	

<Go Back

ExportXML method

Description

VWorks software calls the `ExportXML` method to notify the plugin that the user is exporting a profile. VWorks software automatically includes the following data in the export file:

All registry keys located in the path HKEY_LOCAL_MACHINE\Velocity11\<RegistryName>\<ProfileName>

where

- <RegistryName> is the value returned by the plugin in response to the `GetMetaData` method call (See IWorksDriver "GetMetaData method" on page 45)
- <ProfileName> is the specified profile

By implementing the `ExportXML` method, the plugin can tell VWorks software to include additional profile data from registry locations, external files, or both in the export file.

IMPORTANT The plugin should not implement the IWorksProfiles interface if it has no additional profile data.

Syntax

```
HRESULT ExportXML(
   [in] BSTR ProfileName,
   [out,retval] BSTR *ExportInfo
);
```

Parameters

`ProfileName`	[in] The name of the profile file that is being exported.
`ExportInfo`	[out, retval] A settings XML block, a files XML block, or both that specify additional registry settings and external files to be included in the export file.

ExportXML method output

The plugin returns a settings XML block, a files XML block, or both in the `ExportInfo` parameter of the `ExportXML` method.

<Go Back

XML structure

The value of the `file` attribute for the `Velocity11` element is the name of the device whose profile is being exported. See "Velocity11 element" on page 416.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11>
   <settings>
      <setting />
      ...
   </settings>
   <files>
      <file />
      ...
   </files>
</Velocity11>
```

settings XML block

The settings XML block contains the `settings` element and all its children. This XML block specifies additional registry settings to be included in the export file for the specified profile.

settings element

The `settings` element contains one or more `setting` elements.

setting element

Each `setting` element specifies the folder path of a registry and the action to take. This element has the following attributes:

Name	Value
`action`	One of the following values: • `CopyRegistryNode` This value is used to copy the specified node and all its children to the export file • `CopyRegistryValue` This value is used to copy a single registry value to the export file
`location`	The relative path to the registry node containing the data to be copied. This path is appended to HKEY_LOCAL_MACHINE\Velocity11\<RegistryName>\ to form the absolute folder path.

files XML block

The files XML block contains the `files` element and all its children. This XML block specifies the external files to be included in the export file for the specified profile.

files element

The `files` element contains one or more `file` elements.

<Go Back

file element

Each `file` element specifies the name and location of an external file and the action to take. This element has the following attributes:

Name	Value
`action`	The value `Copy`.
`entry`	The name of the profile setting that contains the folder path of the external file.
`location`	The file path of the external file.

Examples of ExportXML method output

Registry settings

The following sample code is a settings XML block that is returned to VWorks software by the plugin as a string in the `ExportInfo` parameter of the `ExportXML` method. The code, which is returned by a Stacker, tells VWorks software to include the following data in the export file for the specified profile: the information contained in the Windows registry key located in the `Stacker\settings\` folder.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Stacker' md5sum='646194ae858821b0d32f3cd01631184f'
→version='1.0.0' >
   <settings >
      <setting action='CopyRegistryNode' location='Stacker\settings\' />
   </settings>
</Velocity11>
```

External files

The following sample code is a files XML block that is returned to VWorks software by the plugin as a string in the `ExportInfo` parameter of the `ExportXML` method. The code, which is returned by a KiNEDx Robot, tells VWorks software to include three external files in the export file for the specified profile. The files are located in the `C:\Program Files (x86)\Peak KiNEDx Robot Control DLL\` folder.

- `Parameters3088.ini`
- `C12144 Teach Points 3088.ini`
- `C12144 Waypoints 3088.wpt`

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='KiNEDx Robot' md5sum='e21a1cbdc1c47f49a674722f4697849a'
→version='1.0.0' >
   <files >
      <file action='Copy' entry='Parameters File' location='C:\Program Files
→(x86)\Peak KiNEDx Robot Control DLL\Parameters3088.ini' />
      <file action='Copy' entry='Teachpoints File' location='C:\Program Files
→(x86)\Peak KiNEDx Robot Control DLL\C12144 Teach Points 3088.ini' />
      <file action='Copy' entry='Waypoints File' location='C:\Program Files (x86)\Peak
→KiNEDx Robot Control DLL\C12144 Waypoints 3088.wpt' />
   </files>
</Velocity11>
```

<Go Back

20 Enumerations

This chapter defines the enumerated types and error codes used in VWorks Plugin methods.

This chapter contains the following topics:

- "CompileType enumerated type" on page 380
- "MetaDataType enumerated type" on page 381
- "PauseType enumerated type" on page 382
- "PlateFlagsType enumerated type" on page 383
- "ReturnCode enumerated type" on page 384
- "SecurityLevel enumerated type" on page 386
- "TIMER_MODES enumerated type" on page 387

<Go Back

CompileType enumerated type

The `CompileType` enumerated type represents the current compilation stage.

Constant	Value	Description
`COMPILE_BEGIN`	`0`	Compiling begins.
`COMPILE_TASK_PROCESS`	`1`	Compiling the task in a process in the Main Protocol.
`COMPILE_TASK_SUBPROCESS`	`2`	Compiling the task in a subprocess that uses this device.
`COMPILE_TASK_PREPROCESS`	`3`	Compiling the task in a process in the Startup Protocol.
`COMPILE_TASK_POSTPROCESS`	`4`	Compiling the task in the Cleanup Protocol.
`COMPILE_BEGIN_SUBPROCESS`	`5`	Compiling a subprocess begins.
`COMPILE_END_SUBPROCESS`	`6`	Compiling a subprocess ends.
`COMPILE_END`	`7`	Compiling ends.
`COMPILE_LOOP_BEGIN`	`8`	Compiling a Loop task in a subprocess that uses this device.
`COMPILE_LOOP_END`	`9`	Compiling a Loop End task in a subprocess that uses this device.

MetaDataType enumerated type

The `MetaDataType` represents the type of metadata returned by the plugin when VWorks software calls the IWorksDriver `GetMetaData` method. The first time VWorks software calls the IWorksDriver `GetMetaData` method, the value of the `iDataType` parameter, which is of type `MetaDataType`, is `0`.

Constant	Value	Description
`METADATA_ALL`	`0`	Request for all XML metadata. See "Metadata XML block" on page 46.
`METADATA_DEVICE`	`1`	Request for device metadata only. See "Device XML block" on page 48.
`METADATA_COMMAND`	`2`	Request for command metadata only. See "Command XML block" on page 53.
`METADATA_VERSION`	`3`	Request for version metadata only. See "Versions XML block" on page 52.

<Go Back

PauseType enumerated type

The `PauseType` enumerated type represents the state of the scheduler when VWorks software called the IVHooks `ProtocolPaused` method. See "ProtocolPaused method" on page 239.

Constant	Value	Description
`Paused`	`0`	The scheduler is paused, so protocols have temporarily stopped running.
`Continued`	`1`	The scheduler is now running the previously paused protocols.
`Aborted`	`2`	The scheduler is about to abort the currently running protocol.

<Go Back

PlateFlagsType enumerated type

The `PlateFlagsType` enumerated type specifies whether the labware has a lid, is sealed, or neither.

Constant	Value	Description
`STACK_NORMAL_PLATES`	`0`	The labware does not have a lid and is not sealed.
`STACK_LIDDED_PLATES`	`1`	The labware has a lid.
`STACK_SEALED_PLATES`	`2`	The labware is sealed.

<Go Back

ReturnCode enumerated type

The plugin returns a value of the `ReturnCode` enumerated type in the output parameter of certain methods to indicate whether the action (request) was successful (completed).

Constant	Value	Description
`RETURN_SUCCESS`	`0`	The request was completed.
`RETURN_BAD_ARGS`	`1`	Something was wrong with the input, so the request was not completed. When VWorks software receives a return code of `RETURN_BAD_ARGS`, it calls the IWorksDriver `GetErrorInfo` method to get more information about the error. See "GetErrorInfo method" on page 40.
`RETURN_FAIL`	`2`	The request was not completed. When VWorks software receives a fail error code from the plugin, it assumes that the error is recoverable and enters the *VWorks software error loop*, which is described below. **1** The plugin returns the `RETURN_FAIL` error code to VWorks software. **2** VWorks software does the following: **a** Calls the IWorksDriver `GetErrorInfo` method to get a literal string that describes the error from the plugin. See IWorksDriver "GetErrorInfo method" on page 40. **b** Writes the string to the Main Log. **c** Displays the standard error dialog box, which includes the following components: • The error string • The Abort, Ignore and Continue..., Retry, and Diagnostics buttons The figure on page 40 shows a standard error dialog box. **3** The user clicks the Abort, Ignore and Continue..., Retry, or Diagnostics button. **4** VWorks software calls the appropriate IWorksDriver method, `Abort`, `Ignore`, or `Retry`, or the IWorksDiags `ShowDiagsDialog` method. See "Abort method" on page 19,"Ignore method" on page 59, and "Retry method" on page 85, and "ShowDiagsDialog method" on page 94. The VWorks software error loop can only be terminated with a call to the IWorksDriver `Abort` method or when VWorks software receives a return code other than `RETURN_FAIL`.

Synchronous and asynchronous tasks

For both synchronous and asynchronous tasks, VWorks software tells the plugin to execute a task by calling the IWorksDriver `Command` method.

<Go Back

For synchronous tasks, the plugin finishes executing the task before the `Command` method returns, so the plugin can tell VWorks software whether the task executed successfully. See "Command method" on page 21.

For asynchronous tasks, the `Command` method returns immediately after starting the task, before it knows whether the task completed successfully. Consequently, the plugin always returns `RETURN_SUCCESS` for asynchronous tasks, or `RETURN_BAD_ARGS` if one or more of the command arguments are incorrect. If the asynchronous task ultimately fails, the plugin notifies VWorks software about the failure by calling the IWorksController ErrorAbortRetryIgnoreNonBlocking update, which triggers the error-handling process described in "ErrorAbortRetryIgnoreNonBlocking update" on page 352.

<Go Back

SecurityLevel enumerated type

The `SecurityLevel` enumerated type represents the security level, or access privilege, for the user currently logged in to VWorks software.

Note: Refer to the *VWorks Automation Control Setup Guide* for more information about user accounts and privileges.

Constant	Value	Description
`SECURITY_LEVEL_ADMINISTRATOR`	`0`	The access level privilege for the current user is Administrator.
`SECURITY_LEVEL_TECHNICIAN`	`1`	The access level privilege for the current user is Technician.
`SECURITY_LEVEL_OPERATOR`	`2`	The access level privilege for the current user is Operator.
`SECURITY_LEVEL_GUEST`	`3`	The access level privilege for the current user is Guest.
`SECURITY_LEVEL_NO_ACCESS`	`-1`	No user is currently logged in to VWorks software.

<Go Back

TIMER_MODES enumerated type

The TIMER_MODES enumerated type specifies how the spin time is implemented.

Constant	Value	Description
TIMER_MODE_TOTAL_TIME	0	Sets the next spin session to last for the specified duration, including the time to accelerate and decelerate.
TIMER_MODE_TIME_AT_SPEED	1	Sets the next spin session to last for the specified duration, excluding the time to accelerate and decelerate.
TIMER_MODE_CONTINUOUS_SPIN	2	Not currently used.

<Go Back

21 Testing and debugging

This chapter tells you how to use the IWorksTest utility, which is a diagnostics tool designed to initially debug and test the basic functionality of VWorks plugins.

This chapter contains the following topics:

- "Testing your VWorks plugin" on page 390
- "Testing plugins that implement the IVHooks interface" on page 394

<Go Back

Testing your VWorks plugin

IMPORTANT You should always thoroughly test your plugin using the IWorksTest utility before putting your plugin into production.

Before you begin

Compile your plugin and copy the appropriate *.dll files into the ...\Agilent Technologies\VWorks\Plugins folder as follows:

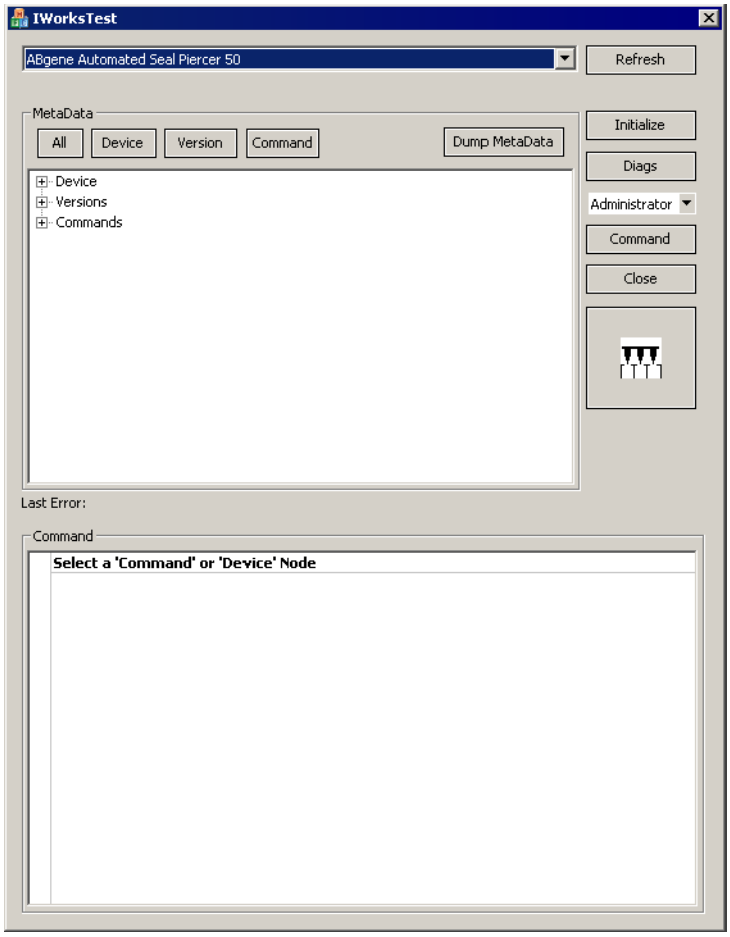
- For Visual Basic, copy the project *.dll and TLB files into the ...\Agilent Technologies\VWorks\Plugins folder
- For C#, copy the project *.dll, TLB, and interop *.dll files into the ...\Agilent Technologies\VWorks\Plugins folder

Starting the IWorksTest utility

To start IWorksTest:

1 Open the ...\Agilent Technologies\VWorks folder.

2 Double-click **IWorksTest.exe**.



<Go Back

3 In the **IWorksTest** dialog box that appears, select your plugin from the list at the top.

The names displayed in this list box are from the `Device` element's `Description` attribute provided in the XML metadata. See "Device element" on page 48.

If your plugin is not on the list, make sure you copied all of the proper *.dll and TLB files into the ...\Agilent Technologies\VWorks\Plugins folder.

Note: You can add breakpoints to your code to determine if the plugin is invoked when IWorksTest is started.

Reviewing and saving your XML metadata

IMPORTANT Your plugin will not load if the XML metadata is not well-formed.

To review your XML metadata:

1 In the **MetaData** area, click one of the following buttons to display all or one of the root nodes:

- **All**
- **Device**
- **Version**
- **Command**

If you provided a 32x32 bitmap icon for the device, the icon should be displayed on the right side of the IWorksTest dialog box.

2 To save a copy of your XML metadata to a file:

a Click the **Dump MetaData** button.

b Enter a **File name** with an .xml extension so you can open the file in an XML editor or in a Web browser.

c Click **Save**.

Testing device initialization

To test the device initialization:

1 Select the **Device** node in the **MetaData** area.

2 Click **Initialize**.

IWorksTest calls the IWorksDriver `Initialize` method. See "Initialize method" on page 61.

If the device is powered on, you can verify that your plugin can communicate with the device and perform any required initialization such as "home the device's motors."

If the device is powered off, you can test the plugin's ability to handle that error condition.

Testing the diagnostics dialog box

To test the diagnostics dialog box for your device:

1 Select a user privilege from the menu under the **Diags** button.

<Go Back

2 Click **Diags**.

IWorksTest calls the IWorksDiags `ShowDiagsDialog` method. See "ShowDiagsDialog method" on page 94.

By selecting different user privileges, you can test the behavior of the diagnostics dialog box under different security levels without having to log in to VWorks software as a different user.

Testing commands

To test individual commands:

1 In the **MetaData** area, expand **Commands** and then select the command you want to run.

2 In the **Command** area, specify the parameter values you want to pass in the method call.

3 Click **Command**.

IWorksTest calls the IWorksDriver `Command` method using the parameter values you specified in step 2. See "Command method" on page 21.

Testing IRobotDriver methods

If your plugin implements the IRobotDriver interface, you can check two of its methods using IWorksTest: `CheckPlatePresent` and `Move`. See "CheckPlatePresent method" on page 143 and "Move method" on page 159.

To test IRobotDriver methods:

1 In the **MetaData** area, expand **Devices** and then expand **Robot**.

2 Select one of the following nodes:

- **Check microplate presence** for the `CheckPlatePresent` method
- **Move microplate** for the `Move` method

3 In the **Command** area, specify the parameter values you want to pass in the method call.

4 Click **Command**.

IWorksTest calls the selected method using the parameter values you specified in step 3.

Related information

For information about...	See...
Device XML block: Device element	"Device element" on page 48
IWorkDiags ShowDiagsDialog method	"ShowDiagsDialog method" on page 94
IWorksDriver Command method	"Command method" on page 21
IWorksDriver Get32x32Bitmap method	"Get32x32Bitmap method" on page 34
IWorksDriver Initialize method	"Initialize method" on page 61

<Go Back

For information about...	See...
IRobotDriver CheckPlatePresent method	“CheckPlatePresent method” on page 143
IRobotDriver Move method	“Move method” on page 159

<Go Back

Testing plugins that implement the IVHooks interface

If your VWorks plugin implements the IVHooks interface, you can perform additional tests.

To test your plugin:

1 In your project code for the `ProtocolStarted` method, add the following code:

Visual Basic

```
Public Sub ProtocolStarted(ByVal sXML As String,
→byRef sResultXML As String)

   MsgBox("Hello from VWorks VHooks Plugin")

End Sub
```

C#

```
Public void ProtocolStarted(String sXML, Ref String sResultXML)
{

   System.Windows.Forms.MessageBox.Show("Hello from VWorks VHooks
→Plugin");

}
```

2 Compile your plugin and copy the appropriate *.dll files into the ...\Agilent Technologies\VWorks\Plugins folder as follows:

- For Visual Basic, copy the project *.dll and TLB files
- For C#, copy the project *.dll, TLB, and interop *.dll files

Note: To save time during testing, temporarily remove all other plugins from the ...\Agilent Technologies\VWorks\Plugins folder to a temporary location. This enables your plugin to load faster in VWorks software and limits the number of messages in the log file to those related to your new plugin.

IMPORTANT When you finish testing, be sure to restore your original plugin files from the temporary location to the ...\Agilent Technologies\VWorks\Plugins folder.

3 Run VWorks software.

4 Check the log to verify that your plugin loaded successfully.

If you do not see your plugin listed in the log, then it failed to load. Typically this is due to either a missing dependency or a missing type library. Any *.dll dependencies must be placed into the ...\Agilent Technologies\VWorks folder and not into the ...\Agilent Technologies\VWorks\Plugins folder.

5 Create a simple protocol and run it.

You should see a message box open when the protocol starts with the message from your plugin.

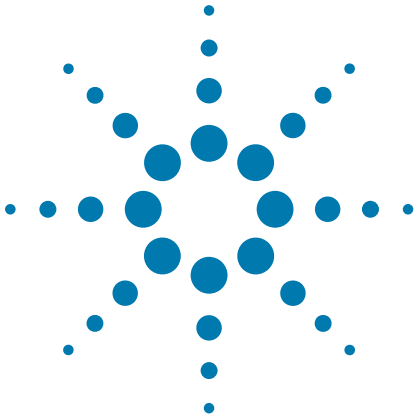
6 Open your plugin's user interface by selecting **Tools > Open Hooks Plugin for**...

<Go Back

Related information

For information about...	See...
IVHooks interface	“IVHooks interface” on page 197

<Go Back



A
Common elements and attributes

This chapter describes some of the XML elements and attributes that are used in multiple methods throughout this guide.

This chapter contains the following topics:

- "Command element" on page 399
  - "Compiler attribute" on page 399
  - "Description attribute" on page 399
  - "DisplayName attribute" on page 400
  - "Editor attribute" on page 400
  - "Name attribute" on page 400
  - "NextTaskToExecute attribute" on page 400
  - "PreferredTab attribute" on page 401
  - "ProtocolName attribute" on page 402
  - "RequiresRefresh attribute" on page 402
  - "TaskRequiresLocation attribute" on page 402
  - "VisibleAvailability attribute" on page 403
- "Device element" on page 403
  - "Description attribute" on page 403
  - "DynamicLocations attribute" on page 403
  - "HardwareManufacturer attribute" on page 404
  - "HasBarcodeReader attribute" on page 404
  - "MiscAttributes attribute" on page 404
  - "Name attribute" on page 404
  - "PreferredTab attribute" on page 405
  - "RegistryName attribute" on page 406
- "Location element" on page 407
  - "Group attribute" on page 407
  - "MaxStackHeight attribute" on page 407
  - "Name attribute" on page 408
  - "Offset attribute" on page 408
  - "Type attribute" on page 408
- "MetaData element" on page 409
- "Parameter element" on page 409
  - "Hide_if attribute" on page 409
  - "Script attribute" on page 410
  - "Scriptable attribute" on page 410

<Go Back

  - "Style attribute" on page 411
  - "Type attribute" on page 411
  - "Units attribute" on page 412
  - "Value attribute" on page 413
  - "ValueToDisplay attribute" on page 413
- "PipetteHeadMode element" on page 414
  - "Channels attribute" on page 414
  - "ColumnCount attribute" on page 414
  - "RowCount attribute" on page 414
  - "SubsetConfig attribute" on page 414
  - "SubsetType attribute" on page 415
  - "TipType attribute" on page 415
- "Ranges element" on page 416
- "Range element" on page 415
  - "Value attribute" on page 415
  - "ValueToDisplay attribute" on page 415
- "Velocity11 element" on page 416
  - "file attribute" on page 416
  - "md5sum attribute" on page 417
  - "version attribute" on page 417
- "Well element" on page 417
  - "Column attribute" on page 417
  - "Row attribute" on page 417
- "Wells element" on page 418
- "WellSelection element" on page 418
  - "CanBe16QuadrantPattern attribute" on page 418
  - "CanBeLinked attribute" on page 418
  - "CanBeQuadrantPattern attribute" on page 418
  - "IsLinked attribute" on page 419
  - "IsQuadrantPattern attribute" on page 419
  - "LinkedText attribute" on page 419
  - "OnlyOneSelection attribute" on page 419
  - "OverwriteHeadMode attribute" on page 419
  - "StartingQuadrant attribute" on page 420

Command element

The `Command` element describes a single task that the device can perform and has the attributes defined in this section.

Compiler attribute

The `Compiler` attribute contains a bitmask that represents the actions that the task performs on the labware.

To determine which value to use, perform a bitwise inclusive OR operation on the actions to be enabled for the task.

Possible values:

`0 = Compiler_No_Action` (Take no action)

`1 = Compiler_Disallow_Sealed_Plates`

`2 = Compiler_Disallow_Unsealed_Plates`

`4 = Compiler_Seals_Plate`

`8 = Compiler_Unseals_Plate`

`16 = Compiler_Disallow_Lidded_Plates`

`32 = Compiler_Disallow_Unlidded_Plates`

`64 = Compiler_Lids_Plate`

`128 = Compiler_Unlids_Plate`

`MAXDWORD = Compiler_All` (VWorks software does not use this value)

Example

Before a Seal Piercer performs a Pierce Plate task, VWorks software needs to know the following:

- If the device allows a sealed labware, but disallows an unsealed labware
- If the labware is unsealed after the Pierce Plate task is completed

To determine the value of the `Compiler` attribute for the Pierce Plate task, perform a bitwise inclusive OR operation on `2` (disallow unsealed labware) and `8` (unseals labware) to get `10`.

If the user selects an invalid condition, VWorks software returns a compiler error.

For a task that seals a labware, performing a bitwise inclusive OR operation on `1` (disallow sealed labware) and `4` (seals labware) to get `5` returns a compiler error.

Required: No

Default value: `0`

Description attribute

The `Description` attribute contains the description of the task.

Required: No

Default value: None

<Go Back

DisplayName attribute

The `DisplayName` attribute contains the task name that is displayed in the user interface. If this attribute is not specified, the value of the `Name` attribute is displayed.

Required: No

Default value: None

See "Name attribute" on page 400.

Editor attribute

The `Editor` attribute contains a bitmask that represents the part of a protocol in which the task is available.

To determine the value to use, perform a bitwise inclusive OR operation on the options to be enabled.

Possible values:

`0` = Disregard all values for this attribute (`Editor_None`)

`1` = Hide the task in the Available Tasks area (`Editor_Hidden`)

`2` = Make the task available in the Main Protocol (`Editor_Primary`)

`4` = Make the task available in a sub-process in the Main Protocol. (`Editor_Secondary`)

`8` = Make the task available in the Startup Protocol and in the Cleanup Protocol (`Editor_PrePost`)

`16` = Make the task available in all editors. This value is the same as the result of performing a bitwise inclusive OR operation on `2`, `4`, and `8` to get `14` (`Editor_Omnipresent`)

`MAXDWORD` = This value is the same as `Editor_Omnipresent` (`Editor_All`)

Example

To determine the value of the `Editor` attribute for a task that is available in the Main Protocol and in the Startup Protocol and Cleanup Protocol, perform a bitwise inclusive OR operation on `2` and `8` to get `10`.

If the value `1` is not specified, the task is always available in the Available Tasks area.

Required: No

Default value: `0`

Name attribute

The `Name` attribute contains the name of the task.

Required: No

Default value: None

NextTaskToExecute attribute

The `NextTaskToExecute` attribute indicates whether this task is the next one to be executed.

Possible values:

`0` = The task is not the next one to be executed

<Go Back

`1` = The task is the next one to be executed

Required: No

Default value: `1`

PreferredTab attribute

The `PreferredTab` attribute contains an option in the Navigation Pane associated with the type of task that the device performs. If the device has more than one task, the `PreferredTab` attribute can be used to specify a different option for each task.

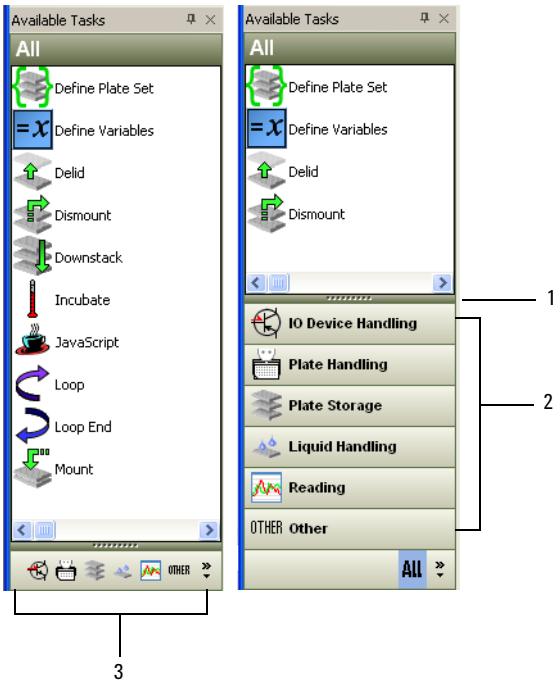
If the `PreferredTab` attribute is specified for the `Command` element, its value overrides the value of the `Device` element's `PreferredTab` attribute.

Possible values:

- `IO Device Handling`
- `Plate Handling`
- `Plate Storage`
- `Liquid Handling`
- `Reading`
- `Other`

To change the quick access buttons from large to small, drag the splitter bar up or down.

In the following figure, the Navigation Pane on the left has small quick access buttons that are displayed horizontally. The Navigation Pane on the right has large quick access buttons that are displayed vertically.



<Go Back

Item	Description	Purpose
1	Horizontal splitter bar	Changes the appearance of the quick access buttons
2	Large quick access button	Displays a list of tasks in the Available Tasks area
3	Small quick access buttons	Displays a list of tasks in the Available Tasks area

Required: No

Default value: None

ProtocolName attribute

The `ProtocolName` attribute contains the name of the protocol that contains the task.

If the protocol has been saved, the value of this attribute is the protocol's file path. If the protocol has not been saved, the value is the default protocol name.

Required: Yes

RequiresRefresh attribute

The `RequiresRefresh` attribute indicates whether VWorks software should always request command metadata from the plugin rather than use cached command metadata.

Possible values:

`0` = VWorks software should not request command metadata from the plugin

`1` = VWorks software should request command metadata from the plugin

For a plugin that provides commands with dynamically changing parameter ranges, the value `1` should be used. Because the plugin does not smartly notify VWorks software of these changes, VWorks software does not know when to request the new command metadata.

Required: No

Default value: `0`

TaskRequiresLocation attribute

The `TaskRequiresLocation` attribute indicates whether the task requires a location.

Possible values:

`0` = The task does not require a location

`1` = The task requires a location

Required: No

Default value: `1`

<Go Back

VisibleAvailability attribute

The `VisibleAvailability` attribute indicates whether the task is displayed in the Available Tasks area.

Possible values:

`0` = The task is not displayed in the Available Tasks area

`1` = The task is displayed in the Available Tasks area

To hide deprecated tasks, set the value of the `VisibleAvailability` attribute to `0`.

Required: No

Default value: `1`

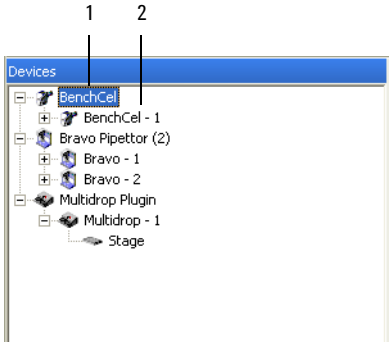
Device element

The `Device` element describes the device to VWorks software and has the attributes defined in this section.

Description attribute

The `Description` attribute contains the description of the device.

The name is displayed in the Device File area, as shown in the following figure.



Item	Description
1	Device description
2	Device name

Required: Yes

DynamicLocations attribute

Currently, VWorks software does not use the `DynamicLocations` attribute.

Indicates whether the device requires dynamic locations.

Possible values:

0 = The device does not require dynamic locations

1 = The device requires dynamic locations

Required: No

Default value: 0

HardwareManufacturer attribute

The HardwareManufacturer attribute contains the name of the device's manufacturer.

Required: No

Default value: None

HasBarcodeReader attribute

The HasBarcodeReader attribute indicates whether the device has a barcode reader.

Possible values:

0 = The device does not have a barcode reader

1 = The device has a barcode reader

Required: No

Default value: 1

MiscAttributes attribute

The MiscAttributes attribute contains a bit field that is reserved for future functionality.

Possible values:

0 = No miscellaneous attributes exist for this device

1 = Allow only combined pick-and-place robot moves; do not allow pick-up-only or place-only moves

2 = Generate a compile-time warning if the location of an upstack or downstack process has not been scanned

4 = The device needs a secondary teachpoint

8 = Allow VWorks software to check if the stack is empty after a pick (used for devices that do not downstack until the move following the Downstack task)

16 = Used for devices that can release stacks

Required: No

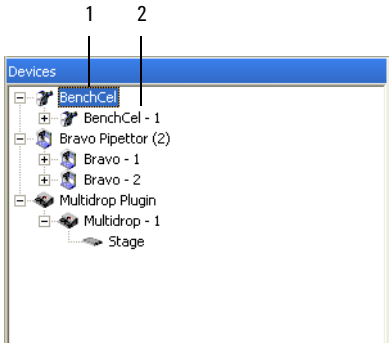
Default value: 0

Name attribute

The Name attribute contains the name of the device.

The name is displayed in the Device File area, as shown in the following figure.

<Go Back



Item	Description
1	Device description
2	Device name

IMPORTANT The device name must be unique, or VWorks software will not load the plugin.

Required: Yes

PreferredTab attribute

The `PreferredTab` attribute contains an option in the Navigation Pane associated with the type of task that the device performs. If the device has more than one task, the `PreferredTab` attribute can be used to specify a different option for each task.

Note: If the device has more than one task, the `Command` element's `PreferredTab` attribute can be used to specify a different option for each task. If the `PreferredTab` attribute is specified for the `Command` element, its value overrides the value of the `Device` element's `PreferredTab` attribute. For a description of the `Command` element's `PreferredTab` attribute, see "PreferredTab attribute" on page 401.

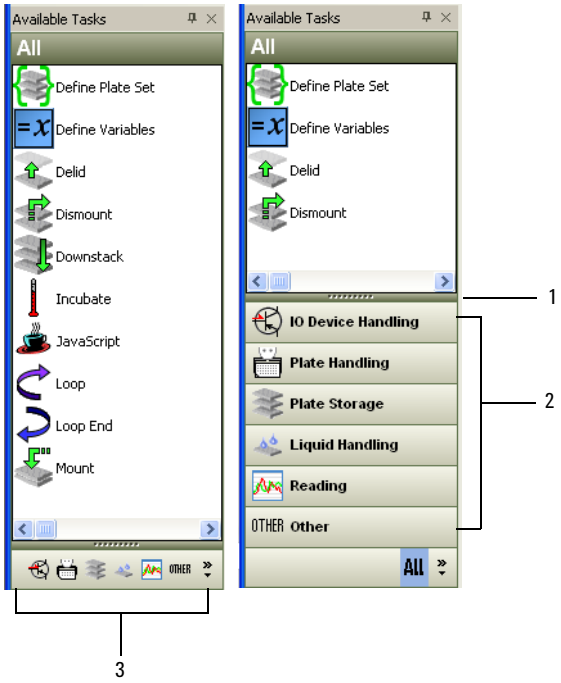
Possible values:

- `IO Device Handling`
- `Plate Handling`
- `Plate Storage`
- `Liquid Handling`
- `Reading`
- `Other`

To change the quick access buttons from large to small, drag the splitter bar up or down.

<Go Back

In the following figure, the Navigation Pane on the left has small quick access buttons that are displayed horizontally. The Navigation Pane on the right has large quick access buttons that are displayed vertically.



Item	Description	Purpose
1	Horizontal splitter bar	Changes the appearance of the quick access buttons
2	Large quick access button	Displays a list of tasks in the Available Tasks area
3	Small quick access buttons	Displays a list of tasks in the Available Tasks area

Required: No

Default value: None

RegistryName attribute

The Windows registry key name. All profile names and attributes are saved in the registry subkeys under this name.

Required: No

Default value: None

<Go Back

Location element

This `Location` element appears in the Device XML block, in the LocationVector XML block, and in the StorageLocation XML block. See "Device XML block" on page 48, "LocationVector XML block" on page 190, "StorageLocation XML block" on page 186, and "StorageLocation XML block" on page 189.

Depending on the XML block that contains it, the `Location` element provides certain information about a location.

Device XML block: Each `Location` element contains the name of a location on the device where labware can be placed.

LocationVector XML block: The `Location` element contains information about the external location.

StorageLocation XML block: The `Location` element contains information about the location on the storage device.

Group attribute

The `Group` attribute is a bitmask that defines a location grouping for this device.

Grouping creates mutually exclusive locations on a device, that is, only one labware can be at a location in the group at a time. To enable this behavior, the `Group` attribute must be set to a value other than `0`.

To determine which value to use, do a bitwise inclusive OR operation on all the groups to which the location should belong. For example, if a location belongs to group 1 and group 2, the value to use is `3`.

Possible values:

`0` = Not exclusive

`1` = group 1

`2` = group 2

`4` = group 3

`8` = group 4

`16` = group 5

`32` = group 6

`64` = group 7

`128` = group 8

`256` = group 9

`512` = group 10

`MAXDWORD` = Exclusivity all

Required: No

Default value: `0`

MaxStackHeight attribute

The `MaxStackHeight` attribute is the maximum height to which a stack of labware is allowed to grow on a device. To determine the number of labware in a stack, divide the stack height by the thickness of the labware.

<Go Back

This attribute is used by stacker devices that return `Location` elements to define the stack locations.

The `MaxStackHeight` attribute is only checked when the value of the `Type` attribute is `2`.

Required: No

Default value: `460.0`

Name attribute

The `Name` attribute contains the name of the location on the device where labware can be placed.

Required: Yes

Offset attribute

The `Offset` attribute is the pick/place height offset, or the delid/relid approach height.

Required: No

Default value: `0.0`

Type attribute

The `Type` attribute is a bitmask that represents the type of access for the location.

To determine the value to use, do a bitwise inclusive OR operation on all the access types for the location.

Possible values:

`0` = No labware are allowed at this location.

`1` = Labware are allowed to be moved to this location.

`2` = Labware are allowed to be stacked at this location.
Adds the Stack Height property to the Location Properties area for a device. This value is used for stack locations only.

`4` = Labware are allowed to be moved into and out of the system at this location.

`8` = Labware are allowed to be incubated at this location.

`16` = Labware are allowed to be delidded/relidded at this location.

`32` = Labware are allowed to be moved into the system at this location.

`64` = Labware are allowed to be moved out of the system at this location.

`128` = A robot is allowed to move a labware into a waste bin at this location.

`256` = Labware are allowed to be mounted at this location.

`512` = Static labware are allowed to be assigned to this location.

`1024` = Only the Centrifuge Loader robot is allowed to access the buckets of the Centrifuge Loader at this location, so private bucket locations are hidden in the device manager.

`MAXDWORD` = All labware are allowed at this location.

Required: No

Default value: `1`

<Go Back

MetaData element

Some VWorks Plugin XML structures contain the `MetaData` element. This element, which is always a child of the `Velocity11` element, has no attributes.

Parameter element

The `Parameter` element contains all information related to a single task parameter, including the following:

- Information needed by VWorks software to properly display the task parameter in the protocol area
- Information needed by the plugin to know the value specified by the user for the parameter when executing the associated task

Category

A name used to group two or more `Parameter` elements into a category.

Required: No

Default value: None

Description

The description of the parameter.

Required: Yes

Hide_if attribute

When the value of the conditional expression in the `Hide_if` attribute is true, the parameter is hidden or made read-only. However, if the `Hide_if` attribute is not specified, the parameter is always visible and read-write.

The following code hides or makes the parameter `ArgName` read-only when the value of `ArgName` is `0` (false).

```
"Variable(ArgName) == Const(0)"
```

<Go Back

Example

In the following sample code, the value of MyParameter will be hidden or made read-only when the value of UpdateCategoryParam is not SetIOManagerPointDigitalOutput. However, UpdateCategoryParam is always visible and read-write because the Hide_if attribute is not specified for this parameter.

```
// Define the "Update Category" parameter.
ObjectInstanceXMLMarshaller::Parameter UpdateCategoryParam;
UpdateCategoryParam._Name = "Update Category";
UpdateCategoryParam._Value = "Some other value";

// Define another parameter.
ObjectInstanceXMLMarshaller::Parameter MyParameter;
MyParameter._Name = "Point Name";
MyParameter._Description = "For IO Manager point input";
MyParameter._Hide_if = "Variable(Update Category)
→!= Const(\"SetIOManagerPointDigitalOutput\")";
```

The following sample code shows the value of the Hide_if attribute for MyParameter. This expression hides or makes MyParameter read-only when the value of UpdateCategoryParam is not set to SetIOManagerPointDigitalOutput.

```
Hide_if = "Variable(Update Category) != Const(\"SetIOManagerPointDigitalOutput\")";
```

Required: No

Default value: None

Script attribute

The Script attribute contains the JavaScript script assigned to the task parameter. The JavaScript variable in the Script attribute can be used to replace the value of the Value attribute.

Required: No

Default value: None

Scriptable attribute

The Scriptable attribute indicates whether a Script Variable dialog box opens when the user selects the parameter in the Task Parameters area and then presses the = (equals) key.

Possible values:

0 = The Script Variable dialog box does not open

1 = The Script Variable dialog box opens

The Scriptable attribute is only used for parameters where the value of the Type attribute is 19 or 30.

Required: No

Default value: 0

<Go Back

Style attribute

The `Style` attribute represents how the parameter is rendered in the Task Parameters area.

Possible values:

`0` = The parameter is always displayed in the Task Parameters area and is read-write

`1` = The parameter is always displayed in the Task Parameters area and is read-only

`2` = The parameter is hidden if the user selects the Hide disabled parameters check box in the Options dialog box; otherwise, the parameter is always displayed in the Task Parameters area and is read-only

Required: No

Default value: `0`

Type attribute

The `Type` attribute represents the type of the field in the Task Parameters area.

Possible values:

`0` = Provides a Boolean check box.

`1` = Allows the user to specify a character string.

`2` = Provides a drop-down list box.

`3` = Provides a drop-down combo box.

`4` = Allows the user to specify a device location.

`5` = Allows the user to specify a labware or a fixed location. If the user specifies a labware, VWorks software selects the location.

`6` = Allows the user to specify both a location and the labware to use. VWorks software then passes the location to the plugin.

`7` = Opens the Well Selection dialog box.

`8` = Allows the user to specify an integer.

`9` = Allows the user to specify a file path.

`10` = Provides a labware drop-down list box.

`11` = Provides a liquid-class drop-down list box.

`12` = Allows the user to specify a decimal fraction.

`13` = Allows the user to specify a file path, where the value can be empty.

`14` = Allows the user to enter a password and displays a series of asterisks to hide the password string.

`15` = Allows the user to specify an IP address.

`16` = Allows the user to select a directory.

`17` = Allows the user to enter a time in the format `hh:mm:ss`.

`18` = Refers to an object in the JavaScript scripting context.

`19` = Allows the user to enter a date. The format depends on the region and language settings.

`20` = Allows the user to enter character strings that can wrap onto multiple lines.

`21` = Opens the Pipette Technique Editor.

<Go Back

22 = Opens the Head Mode Selector dialog box.

23 = Describes the tip positions of a tip box.

24 = Opens the Field Composer dialog box.

25 = Displays the available hit pick format files. For example, when the user clicks the down arrow in the Format file field of the Hit pick replicate task, the list that is displayed is of this type.

26 = *Deprecated.* Used to show the available analog input names in the device file where the plugin resides.

27 = *Deprecated.* Used to show the available digital input names in the device file where the plugin resides.

28 = *Deprecated.* Used to show the available digital output names in the device file where the plugin resides.

29 = Converts a parameter of this type to, and accesses it as, a JavaScript array object.

30 = Allows the user to specify a duration in the format `n Days hh:mm:ss`.

31 = Displays a multi-line text box.

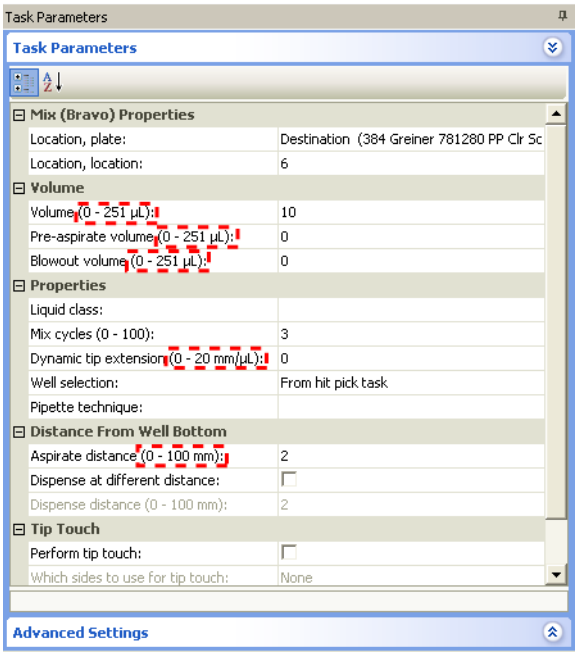
32 = Opens the color palette that enables the user to change the colors of various dialog box components.

Required: Yes

Units attribute

The `Units` attribute specifies the appropriate unit of measure for the parameter.

The unit type is displayed next to the parameter name in the Task Parameters area. The following figure shows three unit types: `µL` for microliters, `mm/µL` for millimeters per microliter, and `mm` for millimeters.



<Go Back

The `Units` attribute is specified for the following values of the `Type` attribute:

- When `Type` is `8` or `12`, two `Range` child elements can contain integers that are used to specify a minimum value and a maximum value.
- When `Type` is `3` or `12`, each `Range` child element can contain an option in a drop-down menu.

Example

In the previous figure, the upper and lower limits for the dynamic tip extension are 0 mm/µL and 20 mm/µL. The corresponding values of the related attributes are as follows:

- The value of the `Parameter` element's `Name` attribute is `Dynamic tip extension`.
- The value of the `Type` attribute is `3`.
- The value of the `Units` attribute is `mm/µL`.
- The values of the two `Range` child elements are `0` and `20`.

Required: No

Default value: None (No units are displayed)

Value attribute

This `Value` attribute is the default value of the `Type` attribute. See "Type attribute" on page 411.

For the following values of the `Type` attribute, the `Value` attribute must contain an escaped XML block, as indicated in the following table:

Value of Type attribute	Escaped XML block required
7	WellSelection XML block
18	JSObject XML block
22	PipetteHeadMode XML block

Required: No

Default value: None

ValueToDisplay attribute

The `ValueToDisplay` attribute specifies the value that is displayed in the Task Parameters area for a `Parameter` element's `Value` attribute.

The values of `ValueToDisplay` and `Value` are not always the same. For example, if `Value` is JavaScript variable `"=variableA"`, then `ValueToDisplay` should be the text string `"variableA"`.

The `ValueToDisplay` attribute, which is not set by the plugin developer, is seldom used. Therefore, the attribute is not shown in any of the sample code in this guide.

Required: No

Default value: None

<Go Back

PipetteHeadMode element

The `PipetteHeadMode` element describes the configuration of the pipette head to be used for liquid transfer operations. This element has the attributes defined in this section.

Channels attribute

The `Channels` attribute specifies the number of channels in the pipette head to use.

Possible values:

`0` = 96-channel head

`1` = 384-channel head

`2` = Column-wise serial dilution with a 96-channel head

`3` = Dispenser with 8 independent tips

`4` = Column-wise serial dilution with a 384-channel head

`5` = Column-wise serial dilution with a 1536-channel head

`6` = Row-wise serial dilution with a 96-channel head

`7` = Row-wise serial dilution with a 384-channel head

`8` = Row-wise serial dilution with a 1536-channel head

`9` = Pin tool

Required: Yes

ColumnCount attribute

The `ColumnCount` attribute contains the number of columns in the pipette head.

Required: Yes

RowCount attribute

The `RowCount` attribute contains the number of rows in the pipette head.

Required: Yes

SubsetConfig attribute

The `SubsetConfig` attribute specifies the pipette channel subset orientation to use.

Possible values:

`0` = Use pipette channels that contain the single channel in the front right corner

`1` = Use pipette channels that contain the single channel in the back right corner

`2` = Use pipette channels that contain the single channel in the back left corner

`3` = Use pipette channels that contain the single channel in the front left corner

<Go Back

Required: No

Default value: 0

SubsetType attribute

The SubsetType attribute specifies which barrels on the pipette head to use.

Possible values:

0 = Use all barrels

1 = Use one or more full columns of barrels

2 = Use one or more full rows of barrels

3 = Use part of one or more columns or rows of barrels, or both

4 = Use a single barrel

Required: No

Default value: 0

TipType attribute

The TipType attribute represents the type of pipette tip.

Possible values:

0 = Small transfer tip

1 = Large transfer tip

2 = Fixed tip

3 = Pin tool

Required: Yes

Range element

The Range element is used to specify a range of values or a list of values for a parameter. This element has the attributes defined in this section.

Value attribute

The value of the Value attribute is determined by the value of the Parameter element's Type attribute as follows:

- When Type is 8 or 12, two Range elements can each have a Value attribute that contains an integer to specify a minimum value and a maximum value.
- When the value of the Type attribute is 3 or 12, each Range element's Value attribute can contain a menu option.

Required: Yes

ValueToDisplay attribute

The ValueToDisplay attribute specifies the value that is displayed in the Task Parameters area for a Parameter element's range of values, as specified in the Range elements' Value attributes.

<Go Back

The values of `ValueToDisplay` and `Value` are not always the same. For example, if `Value` is JavaScript variable `"=variableA"`, then `ValueToDisplay` should be the text string `"variableA"`.

The `ValueToDisplay` attribute, which is not set by the plugin developer, is seldom used. Therefore, the attribute is not shown in any of the sample code in this guide.

Required: No

Default value: None

Ranges element

The `Ranges` element contains one or more `Range` elements and has no attributes.

Velocity11 element

The `Velocity11` element is the root element of most VWorks Plugin XML structures. This element has the attributes defined in this section.

file attribute

The `file` attribute specifies the type of XML structure.

Possible values:

- `BarCodeMisreadResult`

  Used in the Velocity11 XML element for IVHooks `BarCodeMisread` method output.
- `BarCodeReadResult`

  Used in the Velocity11 XML element for IVHooks `BarCodeRead` method output.
- *Device name*. The name of the device whose profile is being exported. Used in the IWorksProfiles ExportXML method.
- `JSSerialize`

  Used in the IWorksController `Query` method's GetJavaScriptVariable query response.
- `MetaData`

  Used in most of the IWorksDriver interface methods.
- `Measurement`

  Used in the IMeasurementDriver `GetMeasurementTypes` method.
- `PlateStorageInventory`

  Used in the IStorageDriver `QueryStorageLocations` method.
- `Query`

  Used in IWorksController `Query` method queries.
- `QueryResponse`

  Used in IWorksController `Query` method query responses.

<Go Back

- `Runset_Data`

  Used in the IWorksController `Query` method's GetRunSetStatus query response.
- `Update`

  Used in the IWorksController `Update` method.
- `Velocity11`

  Used in the IIODriver `EnumPoints`, `Read`, and `Set` methods.

Required: No

Default value: `MetaData`

md5sum attribute

The `md5sum` attribute contains a 128-bit hash value that can be used to verify the integrity of an XML block.

VWorks software provides an `md5sum` calculation by first preparing XML metadata strings with an `md5sum` attribute value of 32 zeros. Then an `md5sum` is calculated, and the `md5sum` attribute value of 32 zeros is replaced with a 32-digit hex code.

Required: No

Default value: None

version attribute

Currently, the value of the `version` attribute is always `1.0`.

Required: Yes

Well element

The `Well` element specifies the location of a well within a microplate, where the upper left corner of the microplate is considered well (1,1). This element has the attributes defined in this section.

Column attribute

The `Column` attribute specifies the leftmost column of the quadrant.

Required: Yes

Row attribute

The `Row` attribute specifies the top row of the quadrant.

Required: Yes

<Go Back

Wells element

The `Wells` element contains one or more `Well` elements and has no attributes.

WellSelection element

IMPORTANT RESERVED FOR INTERNAL USE.

The WellSelection XML block's `WellSelection` element is used by VWorks software to support the quadrant pattern looping feature. The element's attributes should *never* be changed from their default values or from the values assigned to them by VWorks software. The `WellSelection` element contains the `PipetteHeadMode` and `Wells` elements and has the attributes defined in this section. See "PipetteHeadMode element" on page 414 and "Wells element" on page 418.

CanBe16QuadrantPattern attribute

The `CanBe16QuadrantPattern` attribute indicates whether a quadrant pattern can be used for a group of 16 wells.

Possible values:

`0` = A quadrant pattern cannot be used for a group of 16 wells

`1` = A quadrant pattern can be used for a group of 16 wells

Default value: `0`

CanBeLinked attribute

The `CanBeLinked` attribute indicates whether a Well selection field can be linked to a Hit Pick Replication task.

Possible values:

`0` = A Well selection field cannot be linked to a Hit Pick Replication task

`1` = A Well selection field can be linked to a Hit Pick Replication task

Default value: `0`

CanBeQuadrantPattern attribute

The `CanBeQuadrantPattern` attribute indicates whether conditions exist so that a quadrant pattern can be used during well selection.

Possible values:

`0` = A quadrant pattern cannot be used during well selection

`1` = A quadrant pattern can be used during well selection

Default value: `0`

<Go Back

IsLinked attribute

The `IsLinked` attribute indicates whether a Well selection field is linked to a Hit Pick Replication task.

Possible values:

`0` = A Well selection field is not linked to a Hit Pick Replication task

`1` = A Well selection field is linked to a Hit Pick Replication task

Default value: `0`

IsQuadrantPattern attribute

The `IsQuadrantPattern` attribute indicates whether a quadrant pattern is specified for the well selection.

Possible values:

`0` = A quadrant pattern is not specified for the well selection

`1` = A quadrant pattern is specified for the well selection

If the value of `CanBeQuadrantPattern` is `0`, the `IsQuadrantPattern` attribute is ignored by VWorks software.

Default value: `0`

LinkedText attribute

The `LinkedText` attribute contains a string of text that is associated with the link.

If the value of `CanBeLinked` or `IsLinked` is `0`, the `LinkedText` attribute is ignored by VWorks software.

Default value: None

OnlyOneSelection attribute

The `OnlyOneSelection` attribute indicates whether all other wells are deselected when a user selects a well in the Well Selection dialog box.

Possible values:

`0` = All other wells are not deselected when the user selects a well in the Well Selection dialog box

`1` = All other wells are deselected when the user selects a well in the Well Selection dialog box

Default value: `0`

OverwriteHeadMode attribute

The `OverwriteHeadMode` attribute is not used by VWorks software and should be set to `0`.

`QuadrantPattern`

Represents the quadrant pattern.

Possible values:

`0` = Left-to-right, top-to-bottom

`1` = Right-to-left, top-to-bottom

`2` = Top-to-bottom, left-to-right

`3` = Bottom-to-top, left-to-right

`4` = Clockwise (not available for 16-quadrant patterns)

`5` = Counterclockwise (not available for 16-quadrant patterns)

Default value: `0`

StartingQuadrant attribute

The `StartingQuadrant` attribute specifies the number of the starting quadrant in the pattern.

Possible values are `1` through `16`, depending on the number of wells per pipette-head channels or per pin tool pins.

Default value: `1`

B
Methods Terminology

This chapter defines some of the terms used to describe VWorks Plugin methods.

destination device

The device to which a labware is to be moved, or dropped off.

destination location

The location on the destination device to which a labware is to be moved, or dropped off.

external location

The location outside the system.

file path

The fully qualified file name (absolute path), for example, C:\VWorks Workspace\Protocols\myProtocol.pro.

folder path

The container folder where the file resides, for example, ...\Agilent Technologies\VWorks\Plugins.

internal location

The location on the storage device, which is considered inside the system.

sink

verb. To sink a labware is the same as to upstack a labware.

source

verb. To source a labware is the same as to downstack a labware.

source device

The device from which a labware is moved, or picked up.

source location

The location on the source device from which a labware is moved, or picked up.

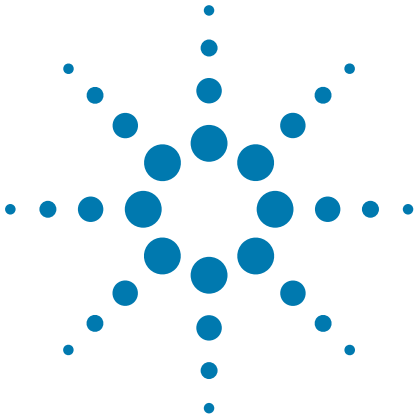
target device

The device that is the object of the IWorksDriver `IsLocationAvailable` and `MakeLocationAvailable` methods. See "IsLocationAvailable method" on page 64 and "MakeLocationAvailable method" on page 67.

<Go Back

target location

The location on the target device that is the object of the IWorksDriver `IsLocationAvailable` and `MakeLocationAvailable` methods. The target location is usually the destination location; however, the source location can also be the target location. See "IsLocationAvailable method" on page 64 and "MakeLocationAvailable method" on page 67.



C
Obsolete, deprecated, and reserved for internal use methods

This chapter lists the methods that are obsolete, deprecated, or reserved for internal use in this version of VWorks software.

Obsolete methods

Obsolete methods should be implemented as `return E_NOTIMPL (0x80004001)`.

IRobotDriver

The following IRobotDriver methods are obsolete in this version of VWorks software:

- `CheckBarcode`
- `GetSpeed`

IStorageDriver interface

The following IStorageDriver method is obsolete in this version of VWorks software:

- `GetStorageLocations`

  This information previously returned by this method is now returned by the IWorksDriver `GetMetaData` method. See "GetMetaData method" on page 45.

IVHooks interface

The following IVHooks methods are obsolete in this version of VWorks software:

- `AvailablePlateList`
- `GetPlateInfo`
- `PipetProcessFinished`
- `PipetProcessStarting`
- `PipetTaskFinished`
- `PipetTaskStarting`
- `PlateGroupMapping`

<Go Back

IWorksDiags interface

The following IWorksDiags method is obsolete in this version of VWorks software:

- `IsDiagsDialogOpen`

IWorksDriver interface

The following IWorksDriver method is obsolete in this version of VWorks software:

- `ShowDiagsDialog`

  Instead of this method, VWorks software calls the IWorksDiags `ShowDiagsDialog` method to display a diagnostics dialog box. See "ShowDiagsDialog method" on page 94.

Deprecated methods

IWorksAsyncDriver interface

The following IWorksAsyncDriver method is deprecated in this version of VWorks software:

- `GetErrorInfo`

  Instead of this method, VWorks software calls the IWorksDriver `GetErrorInfo` method when the plugin reports an error that occurred during a task. See "GetErrorInfo method" on page 40.

ILiddingDriver interface

The following ILiddingDriver method is deprecated in this version of VWorks software:

- `RobotEndsUpHoldingLid`

<Go Back

Reserved for internal use methods

IVHooks interface

The following methods are reserved for internal Agilent Technologies use in this version of VWorks software:

- `CustomMenuClick`

IWorksProfiles interface

The following methods are reserved for internal Agilent Technologies use in this version of VWorks software:

- `GetConflictedProfileName`
- `MigrateProfile`

<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.

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

A

Abort method
    IWorksAsyncDriver interface *265*
    IWorksDriver interface *19*
Aborted method *203*
AllDeviceInfo
    query *296*
    query response *297*
Asynchronous Task Command XML block components *277*
AsyncTaskFinished update *349*
AsyncTaskStarted update *350*
attributes, common
    See elements and attributes, common
AvailablePlateList method, *obsolete* 199

B

Barcode
    query *300*
    query response *300*
Barcode update *350*
BarCodeMisread method *205*
BarCodeRead method *210*

C

C#, writing a plugin in *11*
CanBe16QuadrantPattern attribute, defined *418*
CanBeLinked attribute, defined *418*
CanBeQuadrantPattern attribute, defined *418*
Category element (Parameter element), defined *409*
Channels attribute, defined *414*
CheckBarcode method, *obsolete* 142
CheckPlatePresent method *143*
Close method *20*
CloseDiagsDialog method *93*
Column attribute, defined *417*
ColumnCount attribute, defined *414*
command
    See task *21*
Command element
    defined *399*
    Compiler attribute *399*
    Description attribute *399*
    DisplayName attribute *400*
    Editor attribute *400*
    Name attribute *400*
    NextTaskToExecute attribute *400*
    PreferredTab attribute *401*
    ProtocolName attribute *402*
    RequiresRefresh attribute *402*
    TaskRequiresLocation attribute *402*
    VisibleAvailability attribute *403*
Command method *21*
commands, testing in IWorksTest utility *392*
common attributes
    See elements and attributes, common
common elements
    See elements and attributes, common
common elements and attributes *397–420*
    See elements and attributes, common
Compile method *25*
CompileComplete method *214*
Compiler attribute, defined *399*
CompileType enumerated type *380*
context-sensitive help *viii*
ControllerQuery method *29*
CustomHook method *216*
CustomMenuClick method, *reserved for internal use* 199

D

Deadlock method *218*
debugging plugins
    See IWorksTest utility *389*
DelidRelid method *146*
deprecated methods *424*
    ILiddingDriver RobotEndsUpHoldingLid *424*
    IWorksAsyncDriver GetErrorInfo *424*
Description attribute (Command element), defined *399*
Description attribute (Device element), defined *403*
Description attribute (Parameter element), defined *409*
destination device, defined *421*
destination location, defined *421*
Device element
    defined *403*
    Description attribute *403*
    DynamicLocations attribute *403*
    HardwareManufacturer attribute *404*
    HasBarcodeReader attribute *404*
    MiscAttributes attribute *404*
    Name attribute *404*
    PreferredTab attribute *405*
    RegistryName attribute *406*
device initialization, testing in IWorksTest utility *391*
Device XML block *48*

DeviceLocationTeachpoints
query *302*
query response *302*
diagnostics dialog box, testing in IWorksTest utility *391*
DisplayName attribute, defined *400*
DynamicLocations attribute, defined *403*

E

Editor attribute, defined *400*
elements and attributes, common *397–420*
elements, common
See elements and attributes, common
empty Query and Update XML blocks, defined *6*
empty XML blocks and elements, defined *6*
enumerated types *379–387*
CompileType *380*
MetadataType *381*
PauseType *382*
PlateFlagsType *383*
ReturnCode *384*
SecurityLevel *386*
TIMER_MODES *387*
EnumerateFormats method *113*
enumerations
See enumerated types
EnumPoints method *103*
error codes *384*
Error method *220*
ErrorAbortRetryIgnoreNonBlocking update *352*
escaped XML block, defined *5*
ExportXML method *375*
external location, defined *421*

F

file attribute, defined *416*
file path, defined *421*
FileOpened method *222*
FileSaved method *224*
folder path, defined *421*

G

Get32x32Bitmap method *34*
GetConflictedProfileName method, *reserved for internal use* 374
GetDescription method *36*
GetDeviceName
query *304*
query response *305*
GetErrorInfo method
IWorksAsyncDriver interface, *deprecated* 264
IWorksDriver interface *40*
GetIOManagerPointInput
query *305*
query response *306*
GetJavascriptVariable
query *307*
query response *308*
GetLayoutBitmap method *42*
GetListOfAsyncTasks method *269*
GetMeasurement method *133*
GetMeasurementTypes method *134*
GetMetaData method *45–58*
Command XML block *53*
Device XML block *48*
Metadata XML block *46*
Versions XML block *52*
GetPlateInfo method, *obsolete* 199
GetPlatePresentResult method *150*
GetProductInfo
query *312*
query response *312*
GetRunSetStatus
query *313*
query response *313*
GetSimulationTimes method *153*
GetSpeed method, *obsolete* 142
GetStorageLocations method, *obsolete* 184
GetTeachPoints method *156*
GetUserInterface method *226*
Group attribute, defined *407*

H

HardwareManufacturer attribute, defined *404*
HasBarcodeReader attribute, defined *404*
Hide_if attribute, defined *409*
HookResults XML block *201*

I

IBCRDriver interface *97*
Strobe method *98*
IControllerClient interface *87*
SetController method *88*
Ignore method
IWorksAsyncDriver interface *271*
IWorksDriver interface *59*
IIODriver interface *101–109*
methods overview *102*
EnumPoints method *103*
Read method *105*
Set method *108*
ILabelerDriver interface *111–120*
methods overview *112*
EnumerateFormats method *113*

<Go Back

Print method *115*
PrintAndApply method *117*
ILiddingDriver interface *121—130*
methods overview *122*
LidIsRetained method *123*
OnDelidMoveComplete *124*
OnRelidMoveComplete method *127*
RobotEndsUpHoldingLid method, *deprecated* 122
RobotEndsUpHoldingPlate method *130*
IMeasurementDriver interface *131—137*
methods overview *132*
GetMeasurement method *133*
GetMeasurementTypes *134*
information exchange using XML strings *2*
Initialize method *61*
interfaces
interfaces overview *2*
IBCRDriver *97*
IControllerClient *87*
IIODriver *101*
ILabelerDriver *111*
ILiddingDriver *121—130*
IMeasurementDriver *131*
IPipetteDriver, *reserved for internal use* *139*
IRobotDriver *141*
ISpinDriver *165*
IStackerDriver *169*
IStorageDriver *183*
IVHooks *197*
IWorksAsyncDriver *263*
IWorksController *281*
IWorksDiags *91*
IWorksDriver *15*
IWorksProfiles *373*
internal location, defined *421*
InterPlugin
query *319*
query response *320*
InventoryPlateBarcodes update *358*
IPipetteDriver interface, *reserved for internal use* 139
IRobotDriver CheckPlatePresent method, testing in IWorksTest utility *392*
IRobotDriver interface *141—164*
methods overview *142*
CheckBarcode method, *obsolete* 142
CheckPlatePresent method *143*
DelidRelid method *146*
GetPlatePresentResult method *150*
GetSimulationTimes method *153*
GetSpeed method, *obsolete* 142
GetTeachPoints method *156*
Move method *159*
SetSpeed method *162*
IRobotDriver methods, testing in IWorksTest utility *392*
IRobotDriver Move method, testing in IWorksTest utility *392*
IsDiagsDialogOpen method, *obsolete* 92
IsLinked attribute, defined *419*
IsLocationAvailable method *64*
ISpinDriver interface *165*
SpinCycle method *166*
IsQuadrantPattern attribute, defined *419*
IsStackEmpty method *171*
IsStackFull method *172*
IStackerDriver interface *169—181*
methods overview *170*
IsStackEmpty method *171*
IsStackFull method *172*
LoadStack method *173*
ScanStack method *175*
SinkPlate method *176*
SourcePlate method *178*
UnloadStack method *180*
IStorageDriver interface *183—196*
methods overview *184*
GetStorageLocations method, *obsolete* 184
LoadPlate method *185*
LookupLocations method *188*
QueryStorageLocations method *192*
UnloadPlate method *194*
IVHooks interface *197—261*
methods overview *199*
Aborted method *203*
AvailablePlateList method, *obsolete* 199
BarCodeMisread method *205*
BarCodeRead method *210*
CompileComplete method *214*
CustomHook method *216*
CustomMenuClick method, *reserved for internal use* 199
Deadlock method *218*
Error method *220*
FileOpened method *222*
FileSaved method *224*
GetPlateInfo method, *obsolete* 199
GetUserInterface *226*
HookResults XML block *201*
LiquidTransferComplete method *227*
PipetProcessFinished method, *obsolete* 199
PipetProcessStarting method, *obsolete* 200
PipetTaskFinished method, *obsolete* 200

PipetTaskStarting method, *obsolete* 200
PlateGroupMapping method, *obsolete* 200
ProcessFinished method *231*
ProcessStarting method *234*
ProtocolFinished method *237*
ProtocolPaused method *239*
ProtocolStarted method *241*
RobotMove method *243*
RobotPickComplete method *246*
RobotPlaceComplete method *248*
ScriptPlateError method *250*
TaskFinished method *252*, *255*
UserLoggedIn method *258*
UserLoggedOut method *260*
IWorksAsyncDriver interface *263–280*
methods overview *264*
Abort method *265*
GetErrorInfo method, *deprecated* 264
GetListOfAsyncTasks method *269*
Ignore method *271*
Retry method *274*
IWorksController interface *281–371*
methods overview *283*
NotifyDataChanged method *285*
NotifyTipOperation method *287*
OnCloseDiagsDialog method *290*
PrintToLog method *291*
Query method *292*
Update method *347*
IWorksDiags interface *91–95*
methods overview *92*
CloseDiagsDialog method *93*
IsDiagsDialogOpen method, *obsolete* 92
ShowDiagsDialog method *94*
IWorksDriver interface *15–86*
methods overview *17*
Abort method *19*
Close method *20*
Command method *21*
Compile method *25*
ControllerQuery method *29*
Get32x32Bitmap method *34*
GetDescription method *36*
GetErrorInfo method *40*
GetLayoutBitmap method *42*
GetMetaData method *45*
Ignore method *59*
Initialize method *61*
IsLocationAvailable method *64*
MakeLocationAvailable method *67*
PlateDroppedOff method *75*
PlatePickedUp method *77*
PlateTransferAborted method *79*
PrepareForRun method *83*
Retry method *85*
ShowDiagsDialog method, *obsolete* 18
IWorksProfiles GetConflictedProfileName method, *reserved for internal use* 374
IWorksProfiles interface *373–378*
methods overview *374*
ExportXML method *375*
IWorksProfiles MigrateProfile method, *reserved for internal use* 374
IWorksTest utility
preparing for testing *390*
reviewing and saving XML metadata *391*
starting *390*
testing commands *392*
testing device initialization *391*
testing diagnostics dialog box *391*
testing IRobotDriver methods *392*
testing plugins that implement IVHooks interface *394*

K

knowledge base *vii*

L

Labware
query *322*
query response *323*
LidIsRetained method *123*
LinkedText attribute, defined *419*
LiquidTransferComplete method *227*
LiquidTransferComplete update *360*
loading plugin into VWorks software *12*
LoadPlate method *185*
LoadStack method *173*
Location element (Device XML block, LocationVector XML block, StorageLocation XML block)
defined *407*
Group attribute *407*
MaxStackHeight attribute *407*
Name attribute *408*
Offset attribute *408*
Type attribute *408*
LocationAvailable XML block components *70*
LocationInformation
query *327*
query response *328*
LocationToTeachpoints

query *330*
query response *331*
LookupLocations method *188*

M

MakeLocationAvailable method *67*
MaxStackHeight attribute, defined *407*
md5sum attribute, defined *417*
MetaData element, defined *409*
metadata terminology *5*
MetadataType enumerated type *381*
method terminology *421–422*
methods
deprecated *424*
IBCRDriver interface *98*
IControllerClient interface *88*
IIODriver interface *103–109*
ILabelerDriver interface *113–120*
IMeasurementDriver interface *133–137*
IRobotDriver interface *143–164*
IStackerDriver interface *171–181*
IStorageDriver interface *185–196*
IVHooks interface *203–261*
IWorksAsyncDriver interface *265–280*
IWorksController interface *285–371*
IWorksDiagsDialog interface *93–95*
IWorksDriver interface *19–86*
IWorksProfiles interface *375*
obsolete *423*
reserved internal use *425*
See also individual methods by name
MigrateProfile method, *reserved for internal use* 374
MiscAttributes attribute, defined *404*
Move method *159*

N

Name attribute (Command element), defined *400*
Name attribute (Device element), defined *404*
Name attribute (Location element), defined *408*
NextTaskToExecute attribute, defined *400*
no value (attribute value), defined *7*
not specified (attribute value), defined *6*
NotifyDataChanged method *285*
NotifyTipOperation method *287*

O

obsolete methods *423*
IRobotDriver
CheckBarcode *423*
GetSpeed *423*
IStorageDriver GetStorageLocations *423*
IVHooks
AvailablePlateList *423*
GetPlateInfo *423*
PipetProcessFinished *423*
PipetProcessStarting *423*
PipetTaskFinished *423*
PipetTaskStarting *423*
PlateGroupMapping *423*
IWorksDiags IsDiagsDialogOpen *424*
IWorksDriver ShowDiagsDialog *424*
Offset attribute, defined *408*
OnCloseDiagsDialog method *290*
OnDelidMoveComplete method *124*
online help *vii*
OnlyOneSelection attribute, defined *419*
OnRelidMoveComplete method *127*
overview of interfaces *2*
OverwriteHeadMode attribute, defined *419*

P

Parameter element
defined *409*
Category element *409*
Description attribute *409*
Hide_if attribute *409*
Script attribute *410*
Scriptable attribute *410*
Style attribute *411*
Type attribute *411*
Units attribute *412*
Value attribute *413*
ValueToDisplay attribute *413*
PauseType enumerated type *382*
PDF guide *vii*
PipetProcessFinished method, *obsolete* 199
PipetProcessStarting method, *obsolete* 200
PipetTaskFinished method, *obsolete* 200
PipetTaskStarting method, *obsolete* 200
PipetteHeadMode element
defined *414*
Channels attribute *414*
ColumnCount attribute *414*
RowCount attribute *414*
SubsetConfig attribute *414*
SubsetType attribute *415*
TipType attribute *415*
PlateDroppedOff method *75*
PlateFlagsType enumerated type *383*
PlateGroupMapping method, *obsolete* 200
PlatePickedUp method *77*
Plates XML block components *81*
PlateTransferAborted method *79*
PlateVolume
query *333*

<Go Back

query response *335*
PreferredTab attribute (Command element), defined *401*
PreferredTab attribute (Device element), defined *405*
PrepareForRun method *83*
Print method *115*
PrintAndApply method *117*
PrinterMetaData XML block components *119*
PrintToLog method *291*
ProcessFinished method *231*
ProcessStarting method *234*
ProtocolFinished method *237*
ProtocolName attribute (Command element), defined *402*
ProtocolPaused method *239*
ProtocolStarted method *241*

Q

query
AllDeviceInfo *296*
Barcode *300*
DeviceLocationTeachpoints *302*
GetDeviceName *304*
GetIOManagerPointInput *305*
GetJavascriptVariable *307*
GetProductInfo *312*
GetRunSetStatus *313*
InterPlugin *319*
Labware *322*
LocationInformation *327*
LocationToTeachpoints *330*
PlateVolume *333*
ScanBarcode *337*
SystemPlateInformation *340*
TeachpointInformation *342*
Query method *292*
AllDeviceInfo query/response *296*
Barcode query/response *300*
DeviceLocationTeachpoints query/response *302*
GetDeviceName query/response *304*
GetIOManagerPointInput query/response *305*
GetJavascriptVariable query/response *307*
GetProductInfo query/response *312*
GetRunSetStatus query/response *313*
InterPlugin query/response *319*
Labware query/response *322*
LocationInformation query/response *327*
LocationToTeachpoints query/response *330*
PlateVolume query/response *333*
ScanBarcode query/response *337*
SystemPlateInformation query/response *340*
TeachpointInformation query/response *342*
query response
AllDeviceInfo *296*
Barcode *300*
DeviceLocationTeachpoints *302*
GetDeviceName *304*
GetIOManagerPointInput *305*
GetJavascriptVariable *307*
GetProductInfo *312*
GetRunSetStatus *313*
InterPlugin *319*
Labware *322*
LocationInformation *327*
LocationToTeachpoints *330*
PlateVolume *333*
ScanBarcode *337*
SystemPlateInformation *340*
TeachpointInformation *342*
Query XML block *293*
empty, defined *6*
QueryStorageLocations method *192*

R

RackInfo update *362*
Range element
defined *415*
Value attribute *415*
ValueToDisplay attribute *415*
Ranges element, defined *416*
Read method *105*
RegistryName attribute, defined *406*
RequiresRefresh attribute, defined *402*
reserved for internal use methods *425*
IVHooks CustomMenuClick *425*
IWorksProfiles
GetConflictedProfileName *425*
MigrateProfile *425*
Response XML block *295*
Retry method
IWorksAsyncDriver *274*
IWorksDriver interface *85*
ReturnCode enumerated type *384*
RobotEndsUpHoldingLid, *deprecated* 122
RobotEndsUpHoldingPlate method *130*
RobotMove method *243*
RobotPickComplete method *246*
RobotPlaceComplete method *248*
Row attribute, defined *417*

RowCount attribute, defined *414*
RunScript update *363*
RunsetAdd update *365*

S

ScanBarcode
query *337*
query response *339*
ScanStack method *175*
Script attribute, defined *410*
Scriptable attribute, defined *410*
ScriptPlateError method *250*
SecurityLevel enumerated type *386*
Set method *108*
SetController method *88*
SetIOManagerPointDigitalOutput update *367*
SetSpeed method *162*
ShowDiagsDialog method
IWorksDiags interface *94*
IWorksDriver interface, *obsolete* 18
sink (labware), defined *421*
SinkPlate method *176*
source (labware), defined *421*
source device, defined *421*
source location, defined *421*
SourcePlate method *178*
SpinCycle method *166*
StartingQuadrant attribute, defined *420*
Strobe method *98*
Style attribute, defined *411*
SubsetConfig attribute, defined *414*
SubsetType attribute, defined *415*
SystemPlateInformation
query *340*
query response *341*

T

target device, defined *421*
target location, defined *422*
TaskFinished method *252*, *255*
TaskRequiresLocation attribute, defined *402*
TeachpointInformation
query *343*
query response *344*
terminology
method *421–422*
XML metadata *5*
testing plugins
See IWorksTest utility *389*
TIMER_MODES enumerated type *387*
TipType attribute, defined *415*
Type attribute (Location element), defined *408*
Type attribute (Parameter element), defined *411*

U

Units attribute, defined *412*
UnloadPlate method *194*
UnloadStack method *180*
update
See Update method
Update method *347*
AsyncTaskFinished update *349*
AsyncTaskStarted update *350*
Barcode update *350*
ErrorAbortRetryIgnoreNonBlocking update *352*
InventoryPlateBarcodes update *358*
LiquidTransferComplete update *360*
RackInfo update *362*
RunScript update *363*
RunsetAdd update *365*
SetIOManagerPointDigitalOutput update *367*
Volume update *368*
Update XML block, empty, defined *6*
UserLoggedIn method *258*
UserLoggedOut method *260*

V

Value attribute (Parameter element), defined *413*
Value attribute (Range element), defined *415*
ValueToDisplay attribute (Parameter element), defined *413*
ValueToDisplay attribute (Range element), defined *415*
Velocity11 element
defined *416*
file attribute *416*
md5sum attribute *417*
version attribute *417*
version attribute, defined *417*
Versions XML block *52*
VisibleAvailability attribute, defined *403*
Visual Basic .NET, writing a plugin in *10*
Volume update *368*

W

Well element
defined *417*
Column attribute *417*
Row attribute *417*
Wells element, defined *418*
WellSelection element
CanBe16QuadrantPattern attribute *418*
CanBeLinked attribute *418*
CanBeQuadrantPattern attribute *418*
IsLinked attribute *419*

IsQuadrantPattern attribute *419*
LinkedText attribute *419*
OnlyOneSelection attribute *419*
OverwriteHeadMode attribute *419*
StartingQuadrant attribute *420*
WellSelection element, defined *418*
writing a plugin *9*
in C# *11*
in Visual Basic .NET *10*

X

XML attribute values, identifying *6*
XML block
defined *5*
empty, defined *6*
escaped defined *5*
XML element
defined *5*
empty, defined *6*
XML metadata terminology *5*
XML strings, information exchange using *2*
XML structures, types of *5*

Agilent Technologies

Developer Guide

G5415-90065