IWorks Device Driver Programming Interface and VWorks Hooks Interface

Developer Guide

Agilent Technologies

Notices

User Guide Part Number

G5415-90007

July/2007

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 12081443513
euroservice.automation@agilent.com

Documentation feedback:
documentation.automation@agilent.com

Web: http://www.agilent.com

Acknowledgements

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

Warranty

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

Technology Licenses

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

Restricted Rights Legend

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

Safety Noticies

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.

Agilent Technologies

Letter to our Customers

Dear Customer,

The Agilent Technologies acquisition of Velocity11 resulted in the following changes:

- Creation of Agilent Technologies Automation Solutions, formerly Velocity11
- Renaming of some Velocity11 products
- New Customer Service and Technical Support contact information
- New website address for product information

Please make a note of the following changes as they impact this user guide.

Velocity11 product name changes

Velocity11 product name	Changes to ...
Access2 Automated Microplate Loader	Automated Centrifuge Loader
Element Automation System	BioCel 900 System
IWorks Device Driver Programming Interface	VWorks DCL Interface
PlatePierce Seal Piercing Station	Microplate Seal Piercer
VCode Barcode Print and Apply Station	Microplate Barcode Labeler
Velocity11 Robot	3-Axis Robot
VHooks Integration Interface	VWorks Hooks Interface
VPrep Pipetting System	Vertical Pipetting Station
VSpin Microplate Centrifuge	Microplate Centrifuge
VStack Labware Stacker	Labware Stacker

New contact information

Documentation feedback: documentation.automation@agilent.com

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 12081443513
euroservice.automation@agilent.com

Web: http://www.agilent.com

Contents

Chapter 1. Introduction 1
About this guide 2
VWorks software basics 4
About Velocity11 User Guides 9

Chapter 2. Development environment 11
Component Object Model (COM) interface 12
The COM interfaces 13
Writing a plug-in 15

Chapter 3. Writing XML metadata for IWorks software . 19
Overview 20
IWorksDriver XML metadata 21
<Velocity11> element 25
<MetaData> element 26
<Device> element 27
<Locations> element 28
<Parameters> element 31
<Ranges> element 35
<StorageDimensions> element 36
<Versions> element 38
<Commands> element 39
XML blocks 42

Chapter 4. IWorksDriver interface reference 45
Method list 46
Common IWorks software enumerations 47
Abort method 49
Command method 49
Compile method 50
ControllerQuery method 52
Get32x32Bitmap method 53
GetDescription method 54
GetErrorInfo method 55
GetMetaData method 56
GetLayoutBitmap method 57
Ignore method 59
Initialize method 60

IsLocationAvailable method 61
MakeLocationAvailable method 62
PlateDroppedOff method 65
PlatePickedUp method 66
PlateTransferAborted method 67
PrepareForRun method 68
Retry method 69
ShowDiagsDialog method 70

Chapter 5. IWorksController interface reference 71
Overview 72
SetController method 73
PrintToLog method 73
Query method 74
Update method 101

Chapter 6. IStackerDriver interface reference 115
Method list 116
IsStackEmpty method 116
IsStackFull method 117
LoadStack method 118
SinkPlate method 119
SourcePlate method 120
UnloadStack method 121

Chapter 7. IPipetteDriver interface reference 123
Method list 124
GetHeadChange method 124
GetProfileHeadType method 126
GetTaskType method 127
GetTipChange method 129
GetVolumeChange method 131
ImplementsPipetteWizards method 133

Chapter 8. IRobotDriver interface reference 135
Method list 136
GetTeachPoints method 137
Move method 137

Chapter 9. IStorageDriver interface reference 139
Method list 140
GetStorageLocations method 140
LoadPlate method 143
LookupLocations method 145

QueryStorageLocations method 147
UnloadPlate method 148

Chapter 10. VHooks software interface reference 151
VHooks XML parameters 153
Method list 154
Aborted method 156
AvailablePlateList method 157
BarCodeRead method 157
BarCodeMisread method 159
CompileComplete method 161
CustomMenuClick method 162
Deadlock method 164
Error method 164
FileOpened method 166
FileSaved method 167
GetUserInterface method 168
LiquidTransferComplete method 168
PipetProcessFinished method 171
PipetProcessStarting method 173
PipetTaskFinished method 174
PipetTaskStarting method 176
PlateGroupMapping method 177
ProcessFinished method 178
ProcessStarting method 180
ProtocolFinished method 181
ProtocolStarted method 182
RobotMove method 184
RobotPickComplete method 185
RobotPlaceComplete method 186
ScriptPlateError method 187
TaskFinished method 188
TaskStarting method 190
UserLoggedIn method 191
UserLoggedOut method 192

Chapter 11. Testing and debugging 193
Testing your device driver plug-in 194
Testing your VHooks plug-in 197

Introduction 1

This chapter introduces the *IWorks Device Driver Programming Interface and VHooks Integration Interface Developer Guide*, and contains the following topics:

- ❑ "About this guide" on page 2
- ❑ "VWorks software basics" on page 4
- ❑ "About Velocity11 User Guides" on page 9

About this guide

About this topic

This topic presents an overview of the information covered in this developer guide.

This guide describes:

- ❑ IWorks™ Device Driver Programming Interface
- ❑ VHooks™ Integration Interface

IWorks software is used to create device driver plug-ins controlled by VWorks software.

VHooks software is used to monitor events and to communicate with a Laboratory Information Managenment System (LIMS) using VWorks software.

Who should read this guide

You should read this guide if you are a software developer who wants to:

- ❑ Develop a device driver plug-in to configure hardware that is controlled by VWorks software.
- ❑ Develop or update an application plug-in that needs to respond to VWorks software events.

Developers should have programming knowledge of the Microsoft Windows Component Object Model (COM). Some familiarity with XML is also helpful though several examples are provided in this guide.

What this guide covers

This guide includes the following:

- ❑ Description of the IWorks software interfaces
- ❑ Procedures on setting up a device driver plug-in development project using Microsoft Visual Studio
- ❑ Description of the VHooks software interface
- ❑ Procedures on how to test and debug your plug-in

Device driver defined

A Velocity11 device driver enables VWorks software to control and communicate with a specific type of device. Each type of device that you operate with VWorks software requires a device driver.

For example, VWorks software uses the:

- ❑ VPrep Pipettor device driver to communicate with the Velocity11 VPrep Pipettor device
- ❑ Softmax Reader device driver to communicate with Molecular Devices readers

Plug-in defined

A plug-in is a software program that when added to another program extends it. Velocity11 device drivers are built as plug-ins.

What is IWorks?

IWorks is the VWorks software Device Driver API built as a COM interface. The common interface allows the creation of a device driver plug-in without the necessity of changing the VWorks software. Using IWorks, you can develop plug-ins to add devices to your VWorks software installation.

What is VHooks?

VHooks is the VWorks software Event Monitoring API built as a COM interface. A VHooks plug-in is a software program that uses the VHooks COM interface to respond to events in VWorks software.

Using IWorks and VHooks software together

IWorks and VHooks software are independent plug-ins each with a specific purpose. They may or may not be used together, depending on your objectives.

Other guides to read

Information about the operation of VWorks software is available in the *VWorks Version 3 Automation Control User Guide*.

Software version

This guide documents the implementation of IWorks and VHooks to develop driver plug-ins for version 3 of VWorks software.

Related topics

For information about...	See...
VWorks software	“VWorks software basics” on page 4
Writing a plug-in	“Writing a plug-in” on page 15
COM interface	“Component Object Model (COM) interface” on page 12

VWorks software basics

About this topic

The setup and operation of VWorks software is covered comprehensively in the *VWorks Version 3 Automation Control User Guide*. This topic covers the main concepts that you need to become familiar with before writing a driver plug-in.

Getting started

When you start VWorks software, all driver plug-ins that are located in the C:*VWorks_install_dir*\plugins folder are loaded into VWorks software where *VWorks_install_dir* is the directory where VWorks software was installed. If you are testing your plug-in and make changes to it, you must restart VWorks software to reload the plug-in.

Create an administrator account that you can use when testing your plug-in.

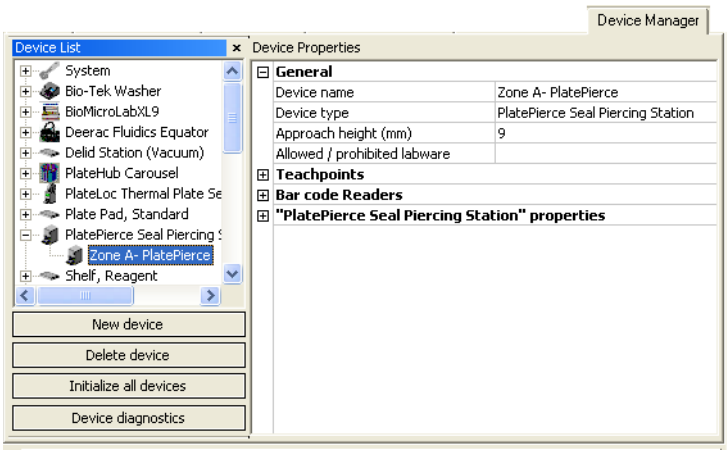
All loaded plug-ins and their version numbers (provided by the plug-in using metadata) are shown in the About VWorks dialog box.

Device defined

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

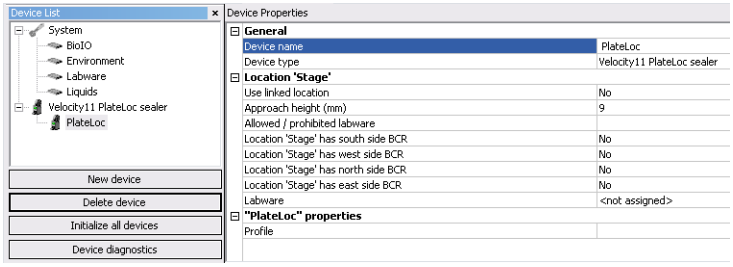
Device manager

The device manager, is used to set up and configure the devices within the system. All devices, including robots, controlled by VWorks software must be set up in the device manager. In addition, labware and liquid classes used by some instruments are configured in the device manager.



Device properties

Configuring devices in the Device Manager requires setting the Device Properties for the device.



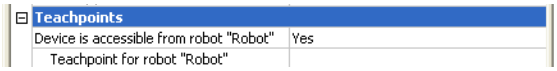
The following list describes the categories of device properties that might have be set when configuring a device:

- ❑ General

  General information about the device such as name, type, allowed labware and robot approach height.

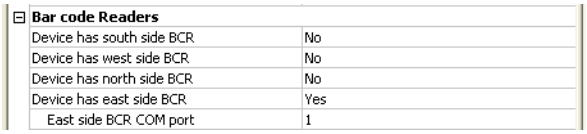
- ❑ Teachpoints

  Teachpoints are coordinates in space that a robot travels to in order to interact with a device. Only the devices that are accessible by robots have teachpoints.



- ❑ Barcode Readers

  These properties include the location and communication port.



- ❑ Locations

  For hardware devices that have more than one robot-accessible labware position, the approach height, allowable/prohibited labware, teachpoint, and barcode properties are located under Location groups. A location is any place on or in the device where VWorks software will access user labware.

Location 'Frame 1'	
Use linked location	No
Location is accessible from robot 'Robot'	No
Approach height (mm)	9
Allowed / prohibited labware	
Location 'Frame 1' has south side BCR	No
Location 'Frame 1' has west side BCR	No
Location 'Frame 1' has north side BCR	No
Location 'Frame 1' has east side BCR	No
Labware	<not assigned>

- *Device* Properties

  The device properties are initialization parameters that are passed to the driver plug-in to initialize the device. The Velocity11 convention is to package the device parameters into a device profile. The profile also contains the communication settings needed for communication between a device and VWorks software.

"PlateLoc Thermal Plate Sealer" properties	
Profile name	ProfileForPlateLoc1

The device profile is accessed the device profile from the device diagnostics.

Labware

Labware (plates, lids, tip boxes, and so on) that is used by devices is defined in the VWorks software labware editor. The labware definitions are stored in the Windows registry.

Device file

All of the information regarding a device is stored or referenced in the device file. This includes the profile settings, device properties, and labware definitions.

Device diagnostics

Device diagnostics are used for:

- Troubleshooting
- Setting teachpoints
- Performing manual operations outside a protocol
- Creating and editing profiles

The driver plug-in developer can determine whether to allow access to diagnostics based on the user account's privileges.

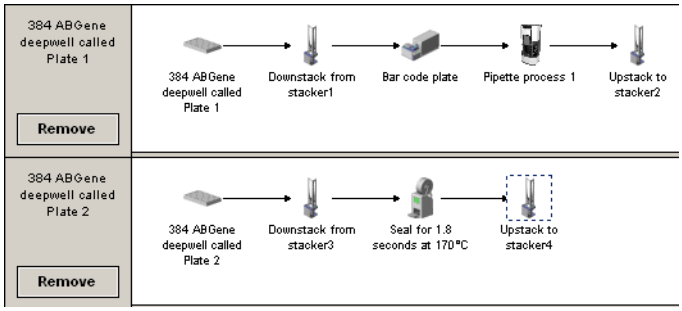
Tasks, processes, and protocols

A task is an operation usually performed by a device, on one or more plates, and is represented by an icon in the protocol editor. It has associated parameters that are defined in the Task Parameters toolbar.

A process is a sequence of tasks that are performed on a plate icon.

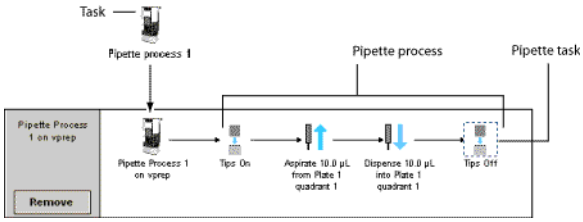
A protocol is a collection of processes that run at the same time.

The following diagram shows one protocol with two processes.



A pipette process task is the parent of a sequence of pipette tasks that perform liquid handling procedures using a VPrep Pipettor. A pipette process is sufficiently complex that it has to be defined by a group of separate sub-tasks, each with its own parameters.

The following diagram describes the relationship between a task, pipette process task, and a pipette process.



Saving and opening protocol files

In the protocol editor, you add plates (each plate represents a parallel process), update plate settings, add tasks (update task settings in the task parameters toolbar).

When you save a protocol file, the information is written to a file in XML. The name of the current device file is also saved with the protocol file.

When you open a protocol file, VWorks software reads and parses the XML. If the device file specified in the protocol file is not currently open, it is opened and all the devices are initialized.

Note: The devices are not initialized if simulation is on.

The device file associated with a protocol file can be changed in Protocol Options.

Compiling, saving, and loading protocols

When you compile a protocol, all defined devices are initialized once unless the Simulator is on or the device properties were modified. The compile will also check for any range errors in the device properties.

When you save a device file, the device configuration is written to a file in XML. The file is saved with a .dev extension. You can open the device file in an XML editor or any browser that contains an XML parser, such as Internet Explorer, version 4 or later.

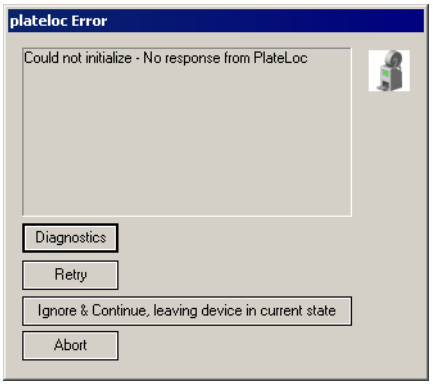
When you load a device file, VWorks software reads and parses the XML. If the file name is different from the previous device file, the device file is also compiled.

Running a protocol

To run a protocol, the user clicks **Start** and specifies the number of times to run the protocol. When a protocol is run, VWorks software automatically compiles it and checks for errors.

Error handling

VWorks software has a standard error dialog as shown in the example below:



The user is given four possible actions when an error occurs:

- *Diagnostics*. Opens the device diagnostics
- *Retry*. Try the operation again
- *Ignore*. Continue with the next operation
- *Abort*. End the current task and optionally stop the protocol

Debugging

You can use the human robot as a debugging tool to test a new driver plug-in in VWorks software. The human robot allows you to operate as the robot. When a protocol runs, dialog boxes open and direct you to move plates.

Related topics

For information about...	See...
What is covered in this guide	“About this guide” on page 2
Writing a plug-in	“Writing a plug-in” on page 15
COM interface	“Component Object Model (COM) interface” on page 12

About Velocity11 User Guides

About this topic

This topic describes the different formats of Velocity11 user information and explains how to access the user information.

Formats available

Velocity11 user information is provided to you as:

- ❑ Online help
- ❑ A PDF file
- ❑ A printed book

The information in each format is the same but each format has different benefits.

Where to find user information

Online help

The online help is added to your computer with the Velocity11 lab automation system software installation.

PDF file

The PDF file of the user guide is on the software CD that is supplied with the product.

Velocity11 website

You can search the online help or download the latest version of any PDF file from the Velocity11 website at www.velocity11.com.

Note: All Velocity11 user information can be searched from the website at www.velocity11.com.

Online help

The online help is the best format to use when you are working at the computer and when you want to perform fast or advanced searches for information.

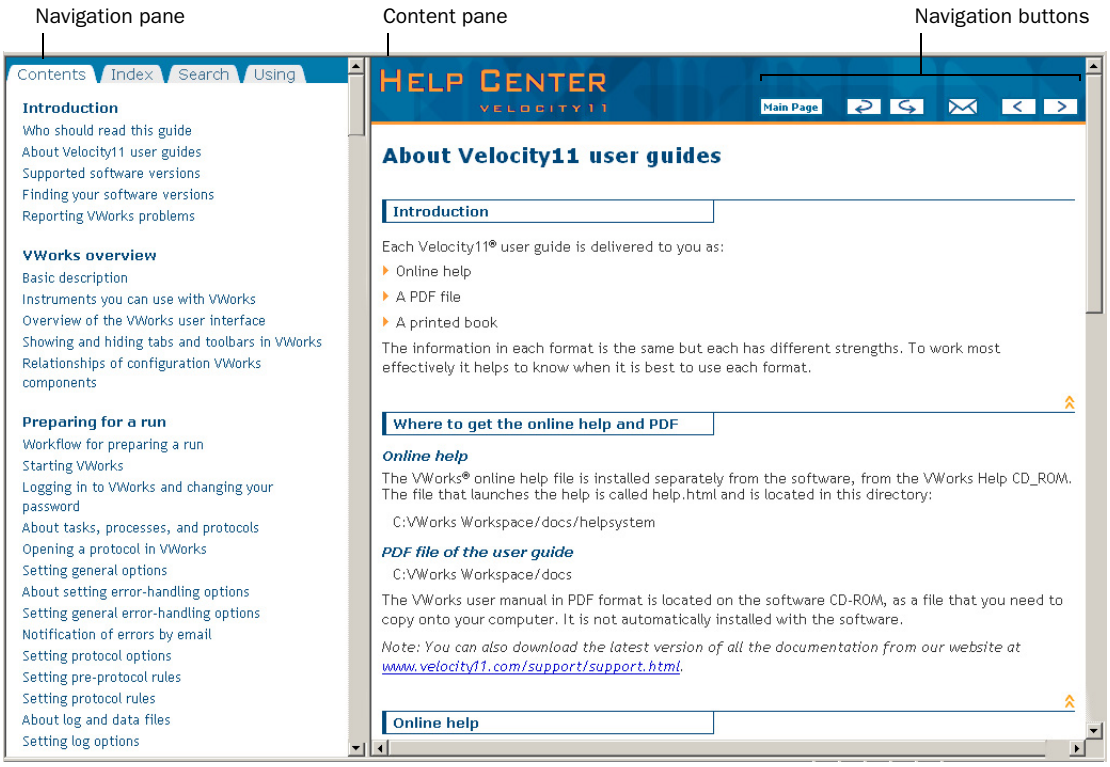
To open the online help:

1. In the Velocity11 lab automation software, press F1. The online help window opens.

Main features

The online help window contains the following:

- ❑ *Navigation pane*. Consists of four tabs. The Contents, Index, and Search tabs provide different ways to locate information. The Using tab contains information about using the help system.
- ❑ *Content pane*. Displays the online help topics.
- ❑ *Navigation buttons*. Enables you to navigate through the pages.The online help includes a navigation pane, content pane, and navigation buttons.



PDF user guides

Computer requirements

To open a user guide in PDF format, you need a PDF viewer. You can download a free PDF viewer from the internet.

Printing and searching

The user guides in PDF format are mainly for printing additional copies. You can perform simple searches in the PDF file, although these searches are much slower than online help searches.

More information

For more information about using PDF documents, see the user documentation for the PDF viewer.

Related topics

For information about...	See...
VWorks software	“VWorks software basics” on page 4
Writing a plug-in	“Writing a plug-in” on page 15
COM interface	“Component Object Model (COM) interface” on page 12

Development environment 2

This chapter describes the development environment and how to start developing your plug-in. It contains the following topics:

❑ "Component Object Model (COM) interface" on page 12
❑ "The COM interfaces" on page 13
❑ "Writing a plug-in" on page 15

Component Object Model (COM) interface

Introduction

Component Object Model (COM) technology was developed by Microsoft Corporation as a language-independent object-oriented system to enable software components to communicate. Each COM component makes its functionality available through one or more interfaces. All access to components is made through the interface methods.

IWorks and VHooks software are component-based APIs (using the Component Object Model, or COM, standard) that provide an interface for developers to write plug-ins for VWorks.

COM was chosen for its language-neutral development environment. Using COM objects, you can write a driver plug-in in the following languages:

- Visual Basic .NET
- C#
- Visual Basic 6
- C++

Conformance to interface

A VWorks driver plug-in is a COM object that conforms to the COM interfaces that define the VWorks Device Driver API (IWorks).

The IWorks device driver plug-in interface only uses methods. By not using events, problems with asynchronous communication between the driver and VWorks are prevented.

Threading model

IWorks is designed to make the developer's job easier by handling the application threading. All calls into the IWorksDriver interface do not need to return until they are completed. All calls are synchronous.

Accommodating version changes

In using COM technology, Velocity11 can make changes to an individual interface without having it affect any others.

If a method is added or removed, it may affect your plug-in. You should code the plug-in to allow for updates to methods.

About third-party device software

Third-party device software can come with DLL drivers or possibly an application, or device software may be non-existent and the third-party just publishes how to interface with their device.

How much work you do depends on the quality of the code that the vendor provides. You may only need to write a simple wrapper for an existing DLL or may have to write the entire driver.

Related topics

For information about...	See...
VWorks software	"VWorks software basics" on page 4
Writing a plug-in	"Writing a plug-in" on page 15
COM interfaces	"The COM interfaces" on page 13

The COM interfaces

About this topic

This topic describes the COM interfaces for the IWorks Device Driver API and the VHooks Event Monitoring API.

IWorks COM interfaces

The IWorks Device Driver API provides a method for developing a device driver plug-in to allow VWorks to control a hardware device in the system.

There are five interfaces that define the COM interface for IWorks.

Interface	Description
IWorksDriver	All drivers must implement this interface. With this interface, the driver plug-in provides information about the device to VWorks, handles device initialization, provides error handling required for the device and responds to requests from VWorks.
IRobotDriver	Drivers that move plates between devices implement this interface.
IPipetteDriver	Drivers that provide liquid handling operations implement this interface.
IStackerDriver	Drivers that introduce plates to the system one plate at a time implement this interface.
IStorageDriver	Drivers that interface with storage devices implement this interface.
IWorksController	This interface provides a way for the driver plug-in to request information from VWorks. Drivers that need to write messages to the log file, request information from VWorks or update information in VWorks implement this interface.

All data passed between the plug-in and VWorks is in the form of generated XML strings.

VWorks uses commands to tell a device to do something. If the device returns before the command is complete, VWorks expects to receive an

error code. When the command is finished, VWorks expects to receive a success confirmation.

Consider breaking commands up into smaller pieces. If, for example, a downstack operation is represented by a single command, the entire command must be repeated when the user clicks **Retry** in response to an error. If the downstack operation is divided into three parts, and the error is in the last part, only that part needs to be repeated.

VHooks COM interface

The VHooks Event Monitoring API provides a method for acting on events in VWorks.

The one interface that defines the COM interface for VHooks is called the VHooksInterface.

Related topics

For information about...	See...
IWorksDriver interface	“IWorksDriver interface reference” on page 45
IRobotDriver interface	“IRobotDriver interface reference” on page 135
IPipetteDriver interface	“IPipetteDriver interface reference” on page 123
IStackerDriver interface	“IStackerDriver interface reference” on page 115
IStorageDriver interface	“IStorageDriver interface reference” on page 139
IWorksController interface	“IWorksController interface reference” on page 71
VHooksInterface interface	“VHooks software interface reference” on page 151
VWorks software	“VWorks software basics” on page 4
Writing a plug-in	“Writing a plug-in” on page 15
COM interface	“Component Object Model (COM) interface” on page 12

Writing a plug-in

About this topic

This topic provides detailed instructions on how to start your plug-in project in Visual Basic .NET and in C# using Microsoft Visual Studio.

These instructions are valid for Microsoft Visual Studio 2005. If you are using a different version, please refer to your Microsoft Visual Studio documentation.

Getting started

Start with a code skeleton in the language of your choice.

Note: Contact Velocity11 to obtain a code skeleton for your project.

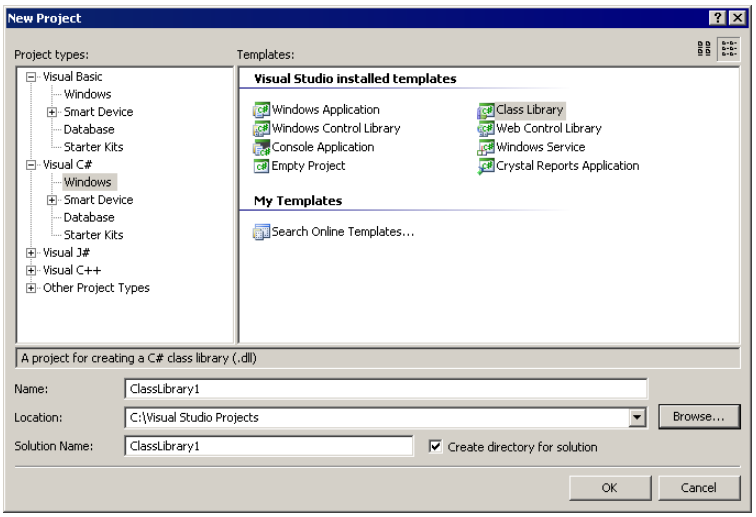
We recommend that you develop your plug-in with Microsoft Visual Studio.

If you are writing a device driver plug-in, you need to design the XML metadata for your driver plug-in.

Starting a project in Visual Basic .NET

To start your project in Visual Basic .NET:

1. In Microsoft Visual Basic Studio, create a new project by selecting **File** > **New** > **Project**.



2. For a Visual Basic project type, select **Class Library** from the templates, enter a **Name** for your project, and click **OK**.
3. To successfully interact with IWorks or VHooks software, you must tell the compiler to generate a type library (TLB) and register it with COM:
   a. Edit the properties for your project by selecting **Project** > *projectname* **Properties**.
   b. Click the **Compile** tab and select the **Register for COM interop** check box.
   c. Save your changes.
4. Add a reference to the COM interface in your project.

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

b. Click the **COM** tab.

c. Choose the type library that corresponds to the interface for which you are developing:

Choose this type library	If you are developing...
IWorksDriver 1.0 Type Library	A device driver plug-in using the IWorks Device Driver API.
VHookInterface 1.0 Type Library	A plug-in that responds to VWorks events and are using the VHooks Event Monitoring API

Note: If you do not see the type library listed, click the **Browse** tab and navigate to the VWorks bin directory to find the type library.

d. Click **OK**.

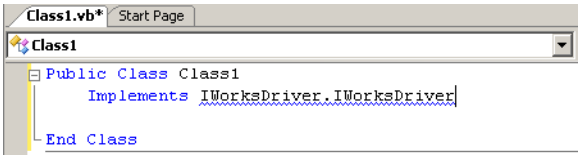
5. In the code window, add an `Implements` statement on the line following the `Class` statement.

   The format for the `Implements` statement is:

   ```
   Implements Interfacename.Interfacemember.
   ```

   Refer to this table to find the Interfacename and Interfacemember you need for the type of plug-in you are writing.

Plug-in for...	Interfacename	Interfacemember
IWorks	IWorksDriver	IControllerClient
	IWorksDriver	IPipetteDriver
	IWorksDriver	IRobotDriver
	IWorksDriver	IStackerDriver
	IWorksDriver	IWorksController
VHooks	VHookInterfaceLib	IVHooks



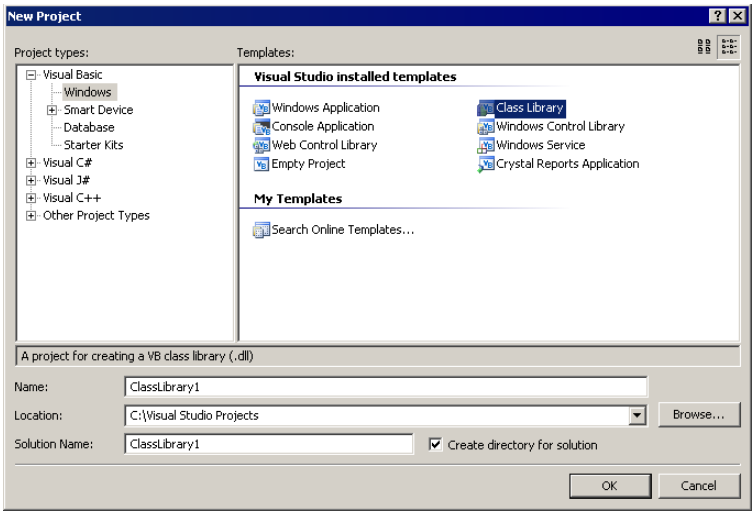
6. Press **Enter** and Visual Studio automatically provides empty method bodies for each method defined in the interface.

7. Add your code to implement the methods of the interface.

Starting a project in C#

To start your project in C#:

1. Create a new project for a C# class library by selecting **File** > **New** > **Project**.



2. For a C# project type, select **Class Library** from the templates, enter a **Name** for your project, and click **OK**.
3. To successfully interact with IWorks or VHooks software, you must tell the compiler to additionally generate a type library (TLB) and register it for COM interop:
   a. Edit the properties for your project by selecting **Project** > *projectname* **Properties**.
   b. Click the **Build** tab and select the **Register for COM interop** check box.
   c. Click the **Application** tab and click **Assembly Information**. Select the **Make assembly COM-Visible** check box.
   d. Save your changes.
4. Add a reference to the COM interface in your project.
   a. Select **Project** > **Add Reference.**
   b. Click the **COM** tab.
   c. Choose the type library that corresponds to the interface for which you are developing:

Choose this type library	If you are developing...
IWorksDriver 1.0 Type Library	A device driver plug-in using the IWorks Device Driver API.
VHookInterface 1.0 Type Library	A plug-in that responds to VWorks events and are using the VHooks Event Monitoring API

Note: If you do not see the type library listed, click the **Browse** tab and navigate to the VWorks bin directory to find the type library.

d. Click **OK**.

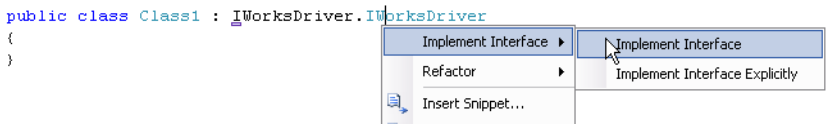
5. In the code window, add the interface declaration to the class. Add the following to the end of the class declaration:

   `: Interfacename.Interfacemember`

   Refer to this table to find the Interfacename and Interfacemember you need for the type of plug-in you are writing.

Plug-in for...	Interfacename	Interfacemember
IWorks	IWorksDriver	IControllerClient
	IWorksDriver	IPipetteDriver
	IWorksDriver	IRobotDriver
	IWorksDriver	IStackerDriver
	IWorksDriver	IWorksController
VHooks	VHookInterfaceLib	IVHooks

6. Right-click on the Interfacemember and select **Implement Interface** > **Implement Interface** to have empty method bodies automatically created for each method in the interface.



7. Add your code to implement the methods of the interface.

Loading your plug-in into VWorks

To get your plug-in loaded for testing:

1. Compile your device driver plug-in.
2. Copy the project DLL, TLB, and the interop DLL files to the C:*VWorks_install_dir*\plugins folder.

   For Visual Basic, you do not need to copy the interop DLL file to the VWorks plugins folder. However for C#, the interop DLL file must be copied to the plugins folder.
3. You can now test your driver plug-in using IWorksTest and VWorks.

Related topics

For information about...	See...
Writing XML metadata for your driver plug-in	“Writing XML metadata for IWorks software” on page 19
Testing your device driver or your VHooks plug-in	“Testing and debugging” on page 193

Writing XML metadata for IWorks software

3

This chapter describes the XML metadata used in the IWorks COM interfaces.

Note: The VHooks oftware interface uses a different XML syntax that is covered in "VHooks software interface reference" on page 151.

This chapter covers the following topics:

- "Overview" on page 20
- "IWorksDriver XML metadata" on page 21
- "<Velocity11> element" on page 25
- "<MetaData> element" on page 26
- "<Device> element" on page 27
- "<Locations> element" on page 28
- "<Parameters> element" on page 31
- "<Ranges> element" on page 35
- "<StorageDimensions> element" on page 36
- "<Versions> element" on page 38
- "<Commands> element" on page 39
- "XML blocks" on page 42

Overview

About this topic

This topic provides defines XML metadata used in the VWorks Device Driver API and describes XML metadata terminology used in this guide.

Using XML metadata in your plug-in

XML metadata is the primary method of how information is passed between VWorks and the driver plug-in. IWorks uses XML strings as parameters in the COM interface methods. Your driver plug-in needs to:

- ❑ Parse XML strings from VWorks provided as input parameters in the COM interface methods
- ❑ Construct well-formed XML strings as output parameters that are sent back to VWorks in the COM interface methods

You can write your own utilities to build the XML strings or consider using the Microsoft COM implementation of the XML Document Object Model (DOM) classes that enable you to construct an XML document in memory. Refer to the MSDN DOM documentation online at http://windowssdk.msdn.microsoft.com/en-us/library/ms766487.aspx for more information.

!! IMPORTANT !! All XML strings must be well-formed or the plug-in will fail to load when VWorks is started.

XML syntax

There are two types of XML strings used in the IWorks interface methods:

- ❑ IWorksDriver XML metadata

  The IWorksDriver XML metadata provides a description of the device, version information for the plug-in and all the commands (tasks) that the device is capable of performing.

  This XML string is only used in the GetMetaData method. All driver plug-ins must implement the GetMetaData method and provide this XML to VWorks.

- ❑ XML blocks

  An XML block is a portion of an XML byte string that includes an element and all its children. The XML block also always includes the XML declaration, the <Velocity11> document element, and may include the <MetaData> element.

  For example, a Device XML block can contain one <Parameters> element, one <Locations> element, and one <StorageDimensions> element. The following is an example of a Device XML block for the BenchCel.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData'
 md5sum='39c81028377422320090ca89839eff31' version='1.0'>
   <MetaData>
      <Device Description='BenchCel' MiscAttributes='1'
       Name='BenchCel' PreferredTab='Plate Handling'>
         <Parameters>
```

```
            <Parameter Name='Profile' Style='0' Type='2' />
         </Parameters>
         <Locations>
            <Location Group='0' Name='Stacker 1' Offset='0' Type='2'
             />
            <Location Group='0' Name='Stacker 2' Offset='0' Type='2'
             />
         </Locations>
         <StorageDimensions DirectStorageAccess='0' />
      </Device>
   </MetaData>
</Velocity11>
```

Related topics

For information about...	See...
XML metadata	"IWorksDriver XML metadata" on page 21
Writing a plug-in	"Writing a plug-in" on page 15
COM interface	"Component Object Model (COM) interface" on page 12

IWorksDriver XML metadata

About this topic

This topic provides an overview of the IWorksDriver XML metadata structure.

High-level structure

The first line of the XML string is always the XML declaration:

```
<?XML version='1.0' encoding='ASCII' ?>
```

The second line is the always the document element <Velocity11>. This line is always followed by the <MetaData> element. The entire metadata structure contains three sections: Device, Versions, and Commands. The following code shows a high-level view of the structure. The details of the three sections follows.

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

Device XML block

Within the main device description XML, there is one device XML block used to describe the device to VWorks. The high-level structure for the device XML section is:

```
<Device>
   <Parameters>
      <Parameter></Parameter>
      ...
   </Parameters>
   <Locations>
      <Location></Location>
      ...
   </Locations>
   <StorageDimensions />
</Device>
```

You can have multiple <Parameter> and multiple <Location> elements.

Note: In the GetMetaData method, the device XML block can be an input parameter and when it is, it will always include the XML declaration, the <Velocity11> element and the <MetaData> element as well as the device specific XML shown here.

Versions XML block

Within the main device description XML, there is one version XML block used to describe the driver plug-in version information to VWorks. The high-level structure for the version XML section is:

```
<Versions>
   <Version />
      ...
</Versions>
```

You can have multiple <Version> elements. Only the information provided in the first <Version> element is displayed in the **About VWorks** dialog box.

Commands XML block

Within the IWorksDriver metadata XML, there is one Commands XML block that contains descriptions of all the commands (tasks) the device can perform. The high-level structure for the Commands XML section is:

```
<Commands>
   <Command>
      <Parameters>
         <Parameter>
            <Ranges>
               <Range></Range>
               ...
            </Ranges>
         </Parameter>
         ...
      </Parameters>
   </Command>
   ...
</Commands>
```

You can have multiple <Command> elements within the <Commands> element. You can have multiple <Parameter> elements for each <Command> and multiple <Range> elements for each <Ranges> element.

Note: In the GetMetaData method, a possible input parameter is a command XML block. This parameter will always include the XML declaration, the <Velocity11> root element and the <MetaData> element as well as a single <Command> XML block.

IWorksDriver metadata XML example

The following is an example of the IWorksDriver metadata XML for a Velocity11 PlateLoc device.

```
<?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>
```

Related topics

For information about...	See...
Providing the device information to VWorks	“GetMetaData method” on page 56

<Velocity11> element

Description

The <Velocity11> element is the document element for all XML blocks.

Attributes

Name	Description	Optional	Default
file	Describes the type of XML block. The valid values are: ❑ MetaData ❑ LocationInfo Used for LocationInfo XML blocks ❑ PlateInfo Used for PlateInfo XML blocks ❑ Query Used for queries from the plug-in ❑ QueryResponse Used for responses to a query from IWorks controller ❑ Update Used for update requests sent to VWorks using the IWorks controller	No	None
md5sum	A 128-bit hash value that can be used to verify the integrity of the XML block. VWorks provides an md5sum calculation by first preparing the XML block with the md5sum attribute value filled with 32 zeros. Then an md5sum of the XML block is calculated and the md5sum attribute value of 32 zeros is replaced with the 32 digit hex code.	Yes	None
version	Currently this is always `1.0`	No	None

Parent elements

The <Velocity11> element is the document of the XML metadata and has no parent element.

Child elements

Element	See...
<MetaData>	"<MetaData> element" on page 26

Related topics

For information about...	See...
XML metadata	"IWorksDriver XML metadata" on page 21
Writing a plug-in	"Writing a plug-in" on page 15
XML blocks	"XML blocks" on page 42

<MetaData> element

Description

The <MetaData> element follows the <Velocity11> document element.

Attributes

There are no attributes for the <MetaData> element.

Parent elements

Element	See...
<Velocity11>	"<Velocity11> element" on page 25

Child elements

Element	See...
<Device>	"<Device> element" on page 27
<Versions>	"<Versions> element" on page 38
<Commands>	"<Commands> element" on page 39

Related topics

For information about...	See...
XML metadata	"IWorksDriver XML metadata" on page 21
Writing a plug-in	"Writing a plug-in" on page 15
XML blocks	"XML blocks" on page 42

<Device> element

Description

The <Device> element describes the general device properties and the tab in the Protocol Tasks toolbar where the device's tasks are displayed.

Attributes

Name	Description	Optional	Default
Name	This device name is displayed as the name of device in the device manager. *Note:* The device name must be unique. VWorks will not load the plug-in if the name is not unique.	No	None
Description	The device description is displayed in the Device Type dropdown menu for the Device Properties.	No	None
MiscAttributes	This is a bit field reserved for future added functionality, The current possible values for MiscAttribute are: 0 = No miscellaneous attributes for this device. 1 = Only allow combined pick and place robot moves. Do not allow pick up only or place only moves.	Yes	0
PreferredTab	Name of the tab in the Protocol Tasks toolbar where the task is displayed based on the type of task. The possible values are: ❑ Liquid Handling ❑ Plate Handling ❑ Plate Storage ❑ Reading ❑ Other	Yes	None

Parent elements

Element	See...
<MetaData>	"<MetaData> element" on page 26

Child elements

Element	See...
<Parameters>	"<Parameters> element" on page 31
<Locations>	"<Locations> element" on page 28

Related topics

For information about...	See...
XML metadata	"IWorksDriver XML metadata" on page 21
Writing a plug-in	"Writing a plug-in" on page 15
XML blocks	"XML blocks" on page 42

<Locations> element

Description

The <Locations> element is used to group one or more <Location> elements.

Attributes

The <Locations> element has no attributes.

Parent elements

Element	See...
<Device>	"<Device> element" on page 27

Child elements

Element	See...
<Location>	"<Location> element" on page 29

<Location> element

The <Location> element describes the possible locations for the device.

Attributes

Name	Description	Optional	Default
Group	A bitmask that defines a location grouping for this device in cases where a plate can be in multiple locations but an operation can only be performed when the plate is in one of the locations within that group making the grouping exclusive respect to commands. To determine the value to use, calculate the sum of the individual values for each feature that you want to enable. The possible bitmask values for Group are: 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	Yes	0
Name	The location name displayed in VWorks Device Properties	No	None
MaxStackHeight	Used for stack locations only. The maximum height that a stack of plates is allowed to grow to on that device. (The total number of plates is the stack height /plate thickness.)	Yes	500
Offset	Unused	Yes	None

Name	Description	Optional	Default
Type	Defines the type of access for this location. The possible values for Type are: 0 = No access 1 = Normal access 2 = Stack access Adds a Stack height setting to the Device Properties. 3 = Storage access	Yes	1

Parent elements

Element	See...
<Locations>	"<Locations> element" on page 28

Child elements

The <Location> element has no child elements.

Related topics

For information about...	See...
XML metadata	"IWorksDriver XML metadata" on page 21
Writing a plug-in	"Writing a plug-in" on page 15
XML blocks	"XML blocks" on page 42

<Parameters> element

Description

The <Parameters> element is used to group one or more <Parameter> elements.

Attributes

The <Parameters> element has no attributes.

Parent elements

Element	See...
<Device>	"<Device> element" on page 27
<Commands>	"<Commands> element" on page 39

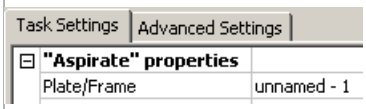
Child elements

Element	See...
<Parameter>	"<Parameter> element" on page 31

<Parameter> element

The <Parameter> elements describe the device parameters and the command (task) parameters.

Attributes

Name	Description	Optional	Default
Category	Adds a category label to the parameters displayed in the Task Parameters toolbar. Assign each parameter the appropriate category and the parameters are displayed grouped by category in the Device Properties.	Yes	None
Description	Description displayed under the Task icon in the Protocol Editor.	No	None
Name	Name displayed in the Task Parameters toolbar. 	No	0
Style	A bitmask that determines how the parameters are rendered the Task Parameters toolbar. The valid values for Style are: 0 = Allows the user to make changes to the parameter value 1 = The value for the parameter is rendered as read only and the user cannot change the value	Yes	0

Name	Description	Optional	Default
Type	The type of field in the Task Parameters toolbar. The possible values for Type are: 0 = Boolean. Displayed as a check box (pt_bool) 1 = Allows the user to specify a character string (pt_string) 2 = Provides a drop-down list box (pt_string_droplist) 3 = Provides a drop-down combo box (pt_string_dropdown) 4 = Allows user to specify a location but VWorks will not place a plate there (pt_fixed_location) 5 = Allows user to specify a plate or a fixed location. If a plate is specified, VWorks selects the location. (pt_auto_location) 6 = User must specify both the location and which plate to use. VWorks passes the chosen location to the plug-in. (pt_manual_location) 7 = Provides a well selection dialog (pt_well_selection) See description below. 8 = Allows the user to specify an integer (pt_integer) 9 = Allows the user to specify the a file path (pt_path) 10 = Provides a labware drop-down list box (pt_labware) 11 = Provides a liquid class drop-down list box (pt_liquid) 12 = Allows the user to specify a decimal fraction, such as 5.34 (pt_float) 13 = Allows user to specify a file path where the value can be empty (pt_optional_path)	Yes	1
Units	Specify the appropriate units for the parameter. The units will display next to the field name in the Task Parameters toolbar. Some examples include mm for millimeters, s for seconds, and oC for degrees Celsius.	Yes	No units displayed
Value	The default value for the parameter. If the parameter type indicates a well selection field (Type='7'), then this field must contain a Well Selection XML block to describe how the plate will be accessed. Values in this XML block provide the default data in VWorks. If the user modifies the settings, the changes are sent to the plug-in using the GetMetaData method. Since this XML block is included as data for an attribute, the XML must have the special characters escaped. *Note:* The use of an XML block here instead of additional attributes in the <Parameter> element simplifies the user interface and allows for future expansion.	No	None

Wells Selection XML block

The Wells selection XML block indicates the number of wells in the plate and how the wells are to be accessed by the pipettor.

XML structure

```
<Velocity11>
   <Channels />
   <Quadrant />
</Velocity11>
```

<Channels> element

The <Channels> element specifies the dilution method. The attribute for this element is:

Name	Value	Description
value	0	96 channel head
	1	384 channel head
	2	Column-wise serial dilution with a 96 channel head (ws_8d)
	3	Dispenser has 8 independent tips (ws_8i)
	4	Column-wise serial dilution with a 384 channel head (ws16d)
	5	ws_32d
	6	Row-wise serial dilutions with a 96 channel head.
	7	Row-wise serial dilutions with a 384 channel head

<Quadrant> element

A quadrant is an evenly spaced array of locations that is addressable by the tips on a pipette head.

The <Quadrant> element specifies upper-left well of the quadrant. The attributes for this element are:

Name	Value	Description
column	integer	The left most column of the quadrant.
row	integer	The top row of the quadrant.

Example

```
<Velocity11 file='MetaData'
md5sum='83892830e93bd93l9384cd939df3d03f'
version='1.0'>
   <Channels value='0' />
   <Quadrant column='0' row='0' />
   <Quadrant column='1' row='0' />
   <Quadrant column='0' row='1' />
   <Quadrant column='1' row='1' />
</Velocity11>
```

Note: Since this XML block is included with the XML as a value of an attribute, you must replace the <, >, &, ‘, and “ characters with the appropriate character entity (< , > , &, ', and ") as shown in the following example:

```
&lt;Velocity11 file=&apos;MetaData&apos;
md5sum=&apos;83892830e93bd93l9384cd939df3d03f&
apos; version=&apos;1.0&apos;&gt;
   &lt;Channels value=&apos;0&apos; /&gt;
   &lt;Quadrant column=&apos;0&apos;
   row=&apos;0&apos; /&gt;
   &lt;Quadrant column=&apos;1&apos;
   row=&apos;0&apos; /&gt;
   &lt;Quadrant column=&apos;0&apos;
   row=&apos;1&apos; /&gt;
   &lt;Quadrant column=&apos;1&apos;
   row=&apos;1&apos; /&gt;
&lt;/Velocity11&gt;
```

Parent elements

Element	See...
<Parameters>	“<Parameters> element” on page 31

Child elements

Element	See...
<Ranges>	“<Ranges> element” on page 35

Related topics

For information about...	See...
XML metadata	“IWorksDriver XML metadata” on page 21
Writing a plug-in	“Writing a plug-in” on page 15

For information about...	See...
XML blocks	"XML blocks" on page 42

<Ranges> element

Description

The <Ranges> element is used to group one or more <Range> elements when specifying a range of values or a list of values for a parameter.

Attributes

The <Ranges> element has no attributes.

Parent elements

Element	See...
<Parameter>	"<Parameter> element" on page 31

Child elements

Element	See...
<Range>	"<Range> element" on page 35

<Range> element

The <Range> element provides range values for a parameter. You can have one or more <Range> elements within a <Ranges> element.

Attributes

Name	Description
Value	How the range is specified depends on the Type specified for the parent <Parameter> element. For example, for an integer you can specify the minimum value and the maximum value using two <Range> elements. For a dropdown menu type, you would specify the possible values for the dropdown menu in each <Range> element.

Parent elements

Element	See...
<Ranges>	"<Ranges> element" on page 35

Child elements

The <Range> element has no child elements.

Related topics

For information about...	See...
XML metadata	"IWorksDriver XML metadata" on page 21
Writing a plug-in	"Writing a plug-in" on page 15
XML blocks	"XML blocks" on page 42

<StorageDimensions> element

Description

The <StorageDimensions> element is used only for storage devices. Use this element to describe the size of a storage location (a rack) in a storage device.

Attributes

Name	Description	Optional	Default
DirectStorageAccess	Specifies whether the robot reaches into the device's storage area (for example a multiple-sided storage carousel) or if the device has an external staging area (for example, STX incubator or Cytomat storage devices). 1 = Robot accesses plates directly in the device 0 = Robot accesses plates on an external staging area	Yes	1
Name0	Name used to describe a column (cassette) of plates.	Yes	None
Name1	Name used to describe what holds each plate.	Yes	None

Parent elements

Element	See...
<Device>	"<Device> element" on page 27

Child elements

Element	See...
<Dimension>	"<Dimensions> element" on page 37
<StorageDimension>	"<Dimensions> element" on page 37

<Dimensions> element

The <Dimensions> element groups one or more <StorageDimension> elements.

Attributes

The <Dimensions> element has no attributes.

Parent elements

Element	See...
<StorageDimensions>	"<StorageDimensions> element" on page 36

Child elements

Element	See...
<StorageDimension>	"<Dimensions> element" on page 37

<StorageDimension> element

The <StorageDimension> element describes the capacity of the storage device. You will have a <StorageDimension> element for each column of plates.

Attributes

The <StorageDimension> element has the following attributes:

Attribute Name	Description
Size	Specifies the number of plates per column.

Parent elements

Element	See...
<Dimension>	"<Dimensions> element" on page 37

Child elements

The <StorageDimension> element has no child elements.

Related topics

For information about...	See...
XML metadata	"IWorksDriver XML metadata" on page 21
Writing a plug-in	"Writing a plug-in" on page 15
XML blocks	"XML blocks" on page 42

<Versions> element

Description

The <Versions> element is used to group one or more <Version> elements that provide version information for the plug-in.

Attributes

The <Versions> element has no attributes.

Parent elements

Element	See...
<MetaData>	"<MetaData> element" on page 26

Child elements

Element	See...
<Version>	"<Version> element" on page 38

<Version> element

Describe the version information for the plug-in. You can have one or more <Version> elements to keep a revision history for your plug-in. Only the first <Version> element information is shown in the About VWorks dialog box.

Attributes

Attribute	Description	Optional	Default
Author	Name of person writing the plug-in.	Yes	None
Company	Your company's name.	Yes	None
Date	Date the plug-in is finished.	Yes	None
Name	Name of the plug-in.	No	None
Version	Version number for the plug-in. This version number is shown in the About VWorks dialog box.	No	None

Parent elements

Element	See...
<Versions>	"<Versions> element" on page 38

Child elements

The <Version> element has no child elements.

Related topics

For information about...	See...
XML metadata	“IWorksDriver XML metadata” on page 21
Writing a plug-in	“Writing a plug-in” on page 15
XML blocks	“XML blocks” on page 42

<Commands> element

Description

The <Commands> element is used to group one or more <Command> elements that describe each task the plug-in can perform.

Attributes

The <Commands> element has no attributes.

Parent elements

Element	See...
<MetaData>	“<MetaData> element” on page 26

Child elements

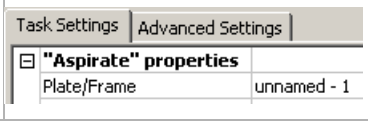
Element	See...
<Command>	“<Command> element” on page 40

<Command> element

Include a <Command> element for each task the device can perform.

Attributes

Name	**Description**	**Optional**	**Default**
Compiler	A bitmask to define the valid plate configurations. To determine the value to use, calculate the sum of the individual values for each feature that you want to enable. The valid bitmask values are: 0 = No action 1 = Disallow_Sealed_Plates 2 = Disallow_Unsealed_Plates 4 = Seals_Plate 8 = Unseals_Plate 16 = Disallow_Lidded_Plates 32 = Disallow_Unlidded_Plates 64 = Lids_Plate 128 = Unlids_Plate MAXDWORD = Everything allowed *Note:* If the user selects an invalid condition, a compiler error is generated.	Yes	0
Description	The description is displayed under the task icon in the Protocol Editor.	No	None
Editor	Specify the editors in which the task is available using this bitmask. To determine the value to use, calculate the sum of the individual values for each feature that you want to enable. The bitmask values are: 0 = Task is not available in any editor 1 = Hide a task from the list of available tasks 2 = Task is available in the Protocol editor 4 = Task is available in the Pipette Process Editor. 8 = Task is available in the Pre-Protocol Editor and the Post-Protocol Editor 16 = Task is available in the Pipette Process editor of other devices	Yes	0

Name	**Description**	**Optional**	**Default**
Name	The name of the task is displayed in the Settings tab of the Task Parameters toolbar. 	No	None

Parent elements

Element	**See...**
<Commands>	"<Commands> element" on page 39

Child elements

Element	**See...**
<Parameters>	"<Parameters> element" on page 31

Related topics

For information about...	**See...**
XML metadata	"IWorksDriver XML metadata" on page 21
Writing a plug-in	"Writing a plug-in" on page 15
XML blocks	"XML blocks" on page 42

XML blocks

About this topic

This topic describes the XML blocks that are used in several IWorks software methods. XML blocks that are only used for a particular method are described with the method.

Command XML block

The Command XML block is a subset of the IWorksDriver metadata XML describing only a single command.

Command XML block example

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='PlateInfo'
 md5sum='39c81028377422320090ca89839eff31' version='1.0' >
   <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>
</Velocity11>
```

Device XML block

The Device XML block is a subset of the IWorksDriver metadata XML describing the device.

Device XML block example

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData'
 md5sum='39c81028377422320090ca89839eff31' version='1.0'>
   <MetaData>
      <Device Description='BenchCel' MiscAttributes='1'
       Name='BenchCel' PreferredTab='Plate Handling'>
         <Parameters>
            <Parameter Name='Profile' Style='0' Type='2' />
         </Parameters>
         <Locations>
```

```
            <Location Group='0' Name='Stacker 1' Offset='0' Type='2'
             />
            <Location Group='0' Name='Stacker 2' Offset='0' Type='2'
             />
         </Locations>
         <StorageDimensions DirectStorageAccess='0' />
      </Device>
   </MetaData>
</Velocity11>
```

PlateInfo XML block

The PlateInfoXML block is used in methods that involve plate handling.

XML structure

```
<?xml version='1.0' encoding='ASCII'>
<Velocity11>
   <Plates>
      <Plate/>
      ...
   </Plates>
</Velocity11>
```

<Plates> element

The <Plates> element is used to group one or more <Plates> elements. This element has no attributes.

<Plate> element

The <Plate> element describes the plate by name, labware, location, and instance. You can have one or more <Plate> elements within a <Plates> element. The <Plate> element has the following attributes:

Attribute	Comments
Name	The user's name for the plate
Labware	The type of plate More information about the labware is available using the IWorksController interface's Query method.
Location	The name of location
Instance_Number	The plate's instance number

PlateInfoXML block example

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='PlateInfo'
 md5sum='39c810283774223200 90ca89839eff31' version='1.0' >
    <Plates>
      <Plate Name='PlateA' Labware='384 V11 ST10 Tip Box 10734'
       Location='Position1' Instance_Number='1'/>
```

```
        <Plate Name='PlateB' Labware='384 V11 ST10 Tip Box 10734'
         Location='Position2' Instance_Number='2'/>
        <Plate Name='PlateC' Labware='384 V11 ST10 Tip Box 10734'
         Location='Position3' Instance_Number='3'/>
    </Plates>
</Velocity11>
```

Related topics

For information about...	See...
IWorksDriver interface methods that use the PlateInfo XML block	❑ "PlateDroppedOff method" on page 65 ❑ "PlatePickedUp method" on page 66 ❑ "PlateTransferAborted method" on page 67
Obtaining labware information using the Query method	"Query method" on page 74

IWorksDriver interface reference

4

This chapter describes the IWorksDriver COM interface methods. All driver plug-ins must implement the 20 IWorksDriver interface methods described in this chapter.

This chapter covers the following topics:

- ❑ “Method list” on page 46
- ❑ “Common IWorks software enumerations” on page 47
- ❑ “Abort method” on page 49
- ❑ “Command method” on page 49
- ❑ “Compile method” on page 50
- ❑ “ControllerQuery method” on page 52
- ❑ “Get32x32Bitmap method” on page 53
- ❑ “GetDescription method” on page 54
- ❑ “GetErrorInfo method” on page 55
- ❑ “GetMetaData method” on page 56
- ❑ “GetLayoutBitmap method” on page 57
- ❑ “Ignore method” on page 59
- ❑ “Initialize method” on page 60
- ❑ “IsLocationAvailable method” on page 61
- ❑ “MakeLocationAvailable method” on page 62
- ❑ “PlateDroppedOff method” on page 65
- ❑ “PlatePickedUp method” on page 66
- ❑ “PlateTransferAborted method” on page 67
- ❑ “PrepareForRun method” on page 68
- ❑ “Retry method” on page 69
- ❑ “ShowDiagsDialog method” on page 70

Method list

About this topic This topic lists the IWorksDriver interface methods.

IWorksDriver methods The following table provides an alphabetical listing of the IWorksDriver COM interface methods.

Method	Description	See...
Abort	User request to Abort the process upon an error	"Abort method" on page 49
Close	Terminate the connection to the device	Planned for a future release.
Command	Perform the requested task	"Command method" on page 49
Compile	Allows driver to log errors and warnings upon a compile	"Compile method" on page 50
ControllerQuery	Provides a method for plug-in to plug-in communication	"ControllerQuery method" on page 52
Get32x32Bitmap	Provide a 32x32 bitmap icon for a given command	"Get32x32Bitmap method" on page 53
GetDescription	Provide a description of the given command	"GetDescription method" on page 54
GetErrorInfo	Provide the last error	"GetErrorInfo method" on page 55
GetLayoutBitmap	Provide a layout bitmap given a layout XML	"GetLayoutBitmap method" on page 57
GetMetaData	Return the specific driver metadata to VWorks software	"GetMetaData method" on page 56
Ignore	User request to ignore the last error	"Ignore method" on page 59
Initialize	Initialize the device	"Initialize method" on page 60
IsLocationAvailable	Respond to query of whether a location is accessible	"IsLocationAvailable method" on page 61
MakeLocationAvailable	Respond to request to make a specific location accessible	"MakeLocationAvailable method" on page 62
PlateDroppedOff	Notification that a plate has been dropped off	"PlateDroppedOff method" on page 65
PlatePickedUp	Notification that a plate has been picked up	"The PlateInfoXML string is an XML block containing information about the plate that was dropped off." on page 65

Method	Description	See...
PlateTransferAborted	Notification that a pending transfer has been aborted	"PlateTransferAborted method" on page 67
PrepareForRun	Notification to prepare for a run	"PrepareForRun method" on page 68
Retry	User request to retry the last command	"Retry method" on page 69
ShowDiagsDialog	Show a diagnostics panel	"ShowDiagsDialog method" on page 70

Common IWorks software enumerations

About this topic

This topic describes the common constants used in several of the COM interfaces defined in IWorks software.

PlateFlagsType values

The PlateFlagsType indicates the state of the plate.

Note: Currently `STACK_NORMAL_PLATES` is the only value used. This parameter will be removed in a future release from the associated methods.

Constant	Value	Description
STACK_NORMAL_PLATES	0	Normal plates
STACK_LIDDED_PLATES	1	Unused
STACK_SEALED_PLATES	2	Unused

Return Code values

The methods use the ReturnCode as an output parameter to indicate to VWorks software whether the action was successful or not. The valid values for the ReturnCode are:

Constant	Value	Description
RETURN_SUCCESS	0	The procedure completed successfully.
RETURN_BAD_ARGS	1	A non-recoverable error occurred. *Note:* This return code triggers a call to GetErrorInfo method to retrieve further error information.

Constant	Value	Description
RETURN_FAIL	2	A procedure failed and the error is recoverable. *Note:* This return code triggers the "VWorks software error loop" described below.

RETURN_FAIL error loop

When VWorks software receives a return code of `RETURN_FAIL`, this implies a recoverable error. The user is provided a choice of Abort, Retry, Ignore, or Diagnose. The user's selection results in a call to the appropriate IWorksDriver method (Abort, Retry, Ignore, or ShowDiagsDialog). This is known as the "VWorks software error loop". The only way out of the error loop is through an Abort command, or if VWorks software receives a return code other than `RETURN_FAIL`.

Related topics

For information about...	See...
XML metadata	"IWorksDriver XML metadata" on page 21
Writing a plug-in	"Writing a plug-in" on page 15
IWorks software interface methods	"Method list" on page 46

Abort method

Description

The call to the Abort method can occur in two possible ways:

- ❑ The user has clicked **Abort** in the standard error dialog box.
- ❑ A protocol run is aborted in the middle of a run.

Note: This is not an emergency stop.

The plug-in should leave the operational state cleanly and safely, and expect to return control of the plate to VWorks software.

Syntax

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

Parameters

The Abort method has no parameters.

Related topics

For information about...	See...
Error handling methods	❑ "Ignore method" on page 59 ❑ "Retry method" on page 69
Providing an error message to VWorks software	"GetErrorInfo method" on page 55

Command method

Description

The Command method is required if the device has any tasks. The Command method is called as a request from VWorks software to begin executing the specified command, and the plug-in should not return until the command has completed.

Syntax

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

Parameters

The Command method has the following parameters:

- ❑ [in] BSTR CommandXML

  In a previous call to the GetMetaData method (with the MetaDataType METADATA_ALL or METADATA_COMMAND), VWorks software received the command metadata that describes the device's commands and the associated parameters for each command.

  The CommandXML parameter is a command XML block that describes the command to be executed.

- [out,retval] ReturnCode* retVal

  Indicates whether or not the request was completed.

Related topics

For information about...	See...
Example of a command XML block	“Command XML block” on page 42
Description of possible return codes	“Return Code values” on page 47
Testing the Command method	“Testing commands” on page 196

Compile method

Description

The Compile method is called for each stage in the protocol compile sequence to provide a way for the plug-in to log errors and warnings for a protocol compile. The plug-in must accumulate state changes during the compile sequence and report any errors encountered during the compile.

Syntax

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

Parameters

The Compile method has the following parameters:

- [in] CompileType iCompileType

  VWorks software calls the Compile method for the different stages of the protocol compilation and each time a different `CompileType` is provided to indicate the current stage of the compile. The valid values for CompileType are:

Value	Description
0	Compile begins
1	Compiling task process
2	Compiling task sub-process
3	Compiling task pre-process
4	Compiling task post-process
5	Compile sub-process begins
6	Compile sub-process ends
7	Compile ends
8	Compile loop begins

Value	Description
9	Compile loop ends

- [in] BSTR MetaDataXML

  The MetaDataXML parameter contains a Command XML block. The content depends on the compilation stage.

- [out,retval] BSTR *CompileResultXML

  The CompileResultsXML parameter is a CompilerErrors XML block provided by the plug-in.

  Returns the empty string if there were no errors or warnings to report to VWorks software. Otherwise, return a string containing a CompilerErrors XML block that contains one or more compile errors.

CompilerErrors XML block

The CompilerErrors XML block describes any compile errors the plug-in needs to report to VWorks software.

XML structure

```
<Velocity11>
   <CompilerErrors>
      <CompilerError />
      ...
   </CompilerErrors>
</Velocity11>
```

<CompilerErrors> element

The <CompilerErrors> element includes one or more <CompileError> elements that describe each error.

<CompileError> element

You can have one or more <CompileError> elements. This element has the following attributes:

Attribute Name	Description
Value	A string representing the error or warning
ErrorType	The number that indicates whether this was an error or warning: ❑ Compiler_Error = 0 ❑ Compiler_Warning = 1

Example

```
<?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>
```

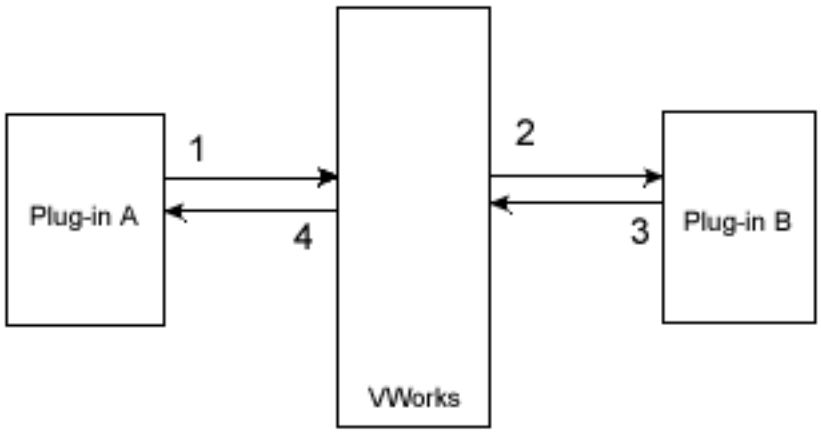
Related topics

For information about...	See...
<Velocity11> element attributes	"<Velocity11> element" on page 25

ControllerQuery method

Description

The ControllerQuery method is used in conjunction with IWorksController Query method to provide a way for two plug-ins to communicate with each other. The communication is initiated by one plug-in with a call to IWorksController Query. This diagram shows how the communication is accomplished with VWorks software as the intermediary.



1. Plug-in A uses IWorksController Query method to initiate a request to Plug-in B and waits for a reply.
2. VWorks software forwards the request to plug-in B using the IWorksDriver ControllerQuery method.
3. Plug-in B provides a result in the output parameter for the ControllerQuery method.
4. VWorks software forwards the result to plug-in A in the output parameter for the IWorksController Query method.

Syntax

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

Parameters

The ControllerQuery has the following parameters:

- [in] BSTR Query
- [out,retval] BSTR* QueryResult

The Query and QueryResult parameters are passed directly between plug-ins, with VWorks software being the intermediary. The contents of these strings is determined by the plug-in developer, who either wrote both plug-ins, or documented the queries to which this plug-in understands and responds.

Related topics

For information about...	See...
IWorksController Query method	"Query method" on page 74

Get32x32Bitmap method

Description

The Get32x32Bitmap method is called when VWorks software requests a graphic for the device or for a specific command (task).

Syntax

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

Parameters

The Get32x32Bitmap method has the following parameters:

- [in] BSTR CommandName

  If CommandName is empty, VWorks software is requesting a graphic that represents the device.

  If CommandName is a string, VWorks software is requesting a graphic to represent a specific task (command) in the protocol editor. In this case, the CommandName parameter contains a Command XML block that was obtained from a previous call to the GetMetaData method for a command.

- [out, retval] IPictureDisp** ppPicture

  The ppPicture parameter is an IPictureDisp object containing the windows bitmap handle 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/.

An example task graphic with description text underneath is shown below.

Related topics

For information about...	See...
Get bitmap showing how plates are arranged on the device	"GetLayoutBitmap method" on page 57

GetDescription method

Description

The GetDescription method retrieves the dynamic description of the command (task) described by CommandXML. The description is only used when logging messages and for displaying a dynamic description under the task icon associated with the command.

Syntax

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

Parameters

The GetDescription method has the following parameters:

- [in] BSTR CommandXML

  The CommandXML parameter is a single Command from the Commands XML block returned from a previous call to the GetMetaData method. Any updates from the user are also included.

- [in] VARIANT_BOOL Verbose

  If the Verbose parameter is set to VARIANT_TRUE, this is a request for a log entry describing the task in detail. Otherwise, this is a request for a short description of the task to place under the task icon in the protocol editor.

- [out,retval] BSTR *Description

  The Description parameter should contain either a detailed description or a short description of the task for the command provided in the CommandXML parameter. You can use the CommandXML parameter to extract details of the command for a dynamic text description of the command.

Related topics

For information about...	See...
Example of a command XML block	"Command XML block" on page 42

GetErrorInfo method

Description

The GetErrorInfo method is called when a plug-in method that has a ReturnCode output parameter returns a value other than RETURN_SUCCESS.

The plug-in should return a string that describes the error. Use this to give the user more information about the error. VWorks software writes the error string to the error log and displays the error string in the standard error dialog box.

Syntax

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

Parameters

The GetErrorInfo method has the following parameter:

- [out,retval] BSTR* ErrorInfo

  This is the literal string representation of the device error. Velocity11 does not currently support error codes. You can embed error codes in the string if necessary.

Related topics

For information about...	See...
Error handling methods	❑ "Abort method" on page 49 ❑ "Ignore method" on page 59 ❑ "Retry method" on page 69

GetMetaData method

Description

The GetMetaData method is called to request the specific driver information. This method is initially called when VWorks software is started and the plug-in is first loaded. The first time this method is called, the request is for all metadata including device, versions, and commands. All devices must provide this method.

The GetMetaData method is also called to communicate with the plug-in when the user changes device parameters or task parameters.

Syntax

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

Parameters

The GetMetaData method has the following parameters:

- [in] MetaDataType iDataType

  Indicates the type of metadata being requested. The first time the GetMetaData method is called, the value for this parameter is 0 requesting all metadata XML for the device. The valid values for MetaDataType are:

Value	Description
0	Request for all metadata (IWorksDriver metadata XML)
1	Request for device metadata only (Device XML block)
2	Request for command metadata only (Command XML block)
3	Request for version metadata only (Version XML block)

- [in] BSTR current_metadata

  This parameter is an XML string that can contain a device or command XML block that reflects the current state of user specified parameters. This is useful for devices that need to modify the metadata returned depending on user settings, before the device has been initialized.

  The first time the GetMetaData method is called, when the plug-in is first loaded, this parameter is empty.

- [out,retval] BSTR *MetaData

  The plug-in should provide the XML metadata that VWorks software has requested based on the iDataType parameter. The plug-in can modify the data based on the user settings specified in the current_metadata parameter.

Related topics

For information about...	See...
Metadata XML structure	"Writing XML metadata for IWorks software" on page 19
All metadata XML example	"IWorksDriver metadata XML example" on page 23
XML blocks examples	"XML blocks" on page 42

GetLayoutBitmap method

Description

The GetLayoutBitmap method is called whenever a bitmap showing the plates arrayed on the device is required. Currently this is a function of the Configure Labware task, but may be incorporated into other areas in the future. The plug-in should parse the layout info, and render a dynamic bitmap that shows the specified labware and where it belongs on the device.

This visual representation can aid a user in the placement of the labware. If the plug-in does not return a bitmap, VWorks software uses a text representation instead.

Syntax

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

Parameters

The GetLayoutBitmap method has the following parameters:

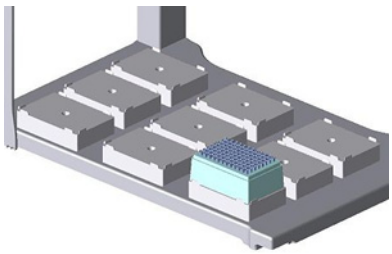
- [in] BSTR LayoutInfoXML

  The LayoutInfoXML parameter contains a LayoutInfo XML block that indicates the labware VWorks software wants to represent on this device.

- [out,retval] IPictureDisp** ppPicture

  The ppPicture parameter is an IPictureDisp object containing the windows bitmap handle for the image. A suggested size for this bitmap is 480 x 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/.

  An example layout bitmap graphic is shown below.



LayoutInfo XML block

The LayoutInfo XML block describes the locations and labware for which VWorks software is requesting a visual representation.

XML structure

The file attribute for the <Velocity11> element is MetaData.

```
<Velocity11>
   <MetaData>
      <Layout>
      ...
   </MetaData>
</Velocity11>
```

<Layout> element

You can have one or more <Layout> elements. This element has the following attributes:

Attribute Name	Description
Name	The name of the location
Labware	The user's name for the labware to represent. More information about the labware is available by using the IWorksController interface's Query method.

Example

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData'
 md5sum='f7038cb092493fad9312e4164873c8a2' version='1.0'>
   <MetaData>
      <Layout Name='location1' Labware='384 V11 ST10 Tip Box
       10734.102' />
      <Layout Name='location2' Labware='384 V11 ST10 Tip Box
       10734.102' />
      <Layout Name='location3' Labware='96 V11 11961.001 Autofilling
       MicroWash' />
      <Layout Name='location4' Labware='96 V11 11961.001 Autofilling
       MicroWash' />
   </MetaData>
</Velocity11>
```

Related topics

For information about...	See...
Getting labware information from VWorks software	"Query method" on page 74
<Velocity11> element attributes	"<Velocity11> element" on page 25

Ignore method

Description

The Ignore method is called when a user has clicked **Ignore** in the error dialog box. The plug-in should continue the operation if possible, or exit the operation if continuing would be dangerous or impossible. This should not result in an unsafe condition, but could result in a new error or a premature completion of the operation.

If all errors encountered during a device operation are ignored, the end result should be a state where the system can continue as if the failing step was skipped.

Syntax

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

Parameters

The Ignore method has the following parameter:

- [out,retval] ReturnCode* retVal

  Indicates whether or not the request was completed.

Related topics

For information about...	See...
Error handling methods	❑ "Abort method" on page 49 ❑ "Retry method" on page 69
Providing an error message to VWorks software	"GetErrorInfo method" on page 55
Description of possible return codes	"Return Code values" on page 47

Initialize method

Description

The Initialize method is called to initialize the device. The plug-in is expected to do everything that is necessary to bring the device into a state that allows it to accept commands (open serial port, set baud rate, home motors, and so on). The method should execute synchronously, meaning it should not return until the device initialization is complete.

Syntax

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

Parameters

The Initialize method has the following parameters:

- [in] BSTR CommandXML

  CommandXML is passed the parameters for the initialization, which are defined in the Device metadata block.

  Using the metadata received from the plug-in with a previous call to GetMetaData with a request for device metadata, VWorks software reformats that data as a command XML block. This command block will also include any changes the user might have made to the parameters.

- [out,retval] ReturnCode* retVal

  Returns the appropriate ReturnCode value:

  - RETURN_SUCCESS means the device has successfully initialized.
  - RETURN_BAD_ARGS indicates something was wrong with the input that prevented the device from initializing.
  - RETURN_FAIL indicates that the device attempted to initialize, but failed.

Related topics

For information about...	See...
Example Command XML block	“Command XML block” on page 42
Description of possible return codes	“Return Code values” on page 47

IsLocationAvailable method

Description

The IsLocationAvailable method is called (repeatedly) during task scheduling to determine if the specified location is available for use.

The IsLocationAvailable method is intended for complicated multi-robot systems to prevent one robot reaching into the envelope of another robot, potentially causing a robot crash.

This method prevents devices with complicated geometry or very long operations from scheduling tasks on the device until pre-conditions are met. It is not intended as a means to pre-empt the scheduler, or to wait for a pre-condition to occur.

It is critical that this function take as little time as possible, as it will be called repeatedly during scheduling.

Syntax

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

Parameters

The IsLocationAvailable method has the following parameters:

- [in] BSTR LocationAvailableXML
- [out,retval] VARIANT_BOOL* Available

  The return value is whether or not a MakeLocationAvailable method call is possible at this time.

  Generally plug-ins should return TRUE from this method. In this case, the scheduler will either call the MakeLocationAvailable method and begin a command cycle on the device, or call the IsLocationAvailable method again at a later time.

  Note: A return value of TRUE does not necessarily mean that VWorks software will use the specified location.

  Return FALSE if there is a pending condition that cannot be resolved by a call to the MakeLocationAvailable method.

Related topics

For information about...	See...
LocationAvailable XML block	“LocationAvailable XML block” on page 62
MakeLocationAvailable method	“PlateDroppedOff method” on page 65

MakeLocationAvailable method

Description

The MakeLocationAvailable method is called when a plate has been scheduled to be delivered to the specified location. The plug-in should perform all actions necessary to ensure that the requested location is available for access by the named robot.

For example, a device might need to open a door and extend a plate stage at this point. This method should not return until the operation has completed. This method is always followed by either PlateDroppedOff, PlatePickedUp, or PlateTransferAborted methods.

To reduce protocol duration, you might return a success value, and then do the work to make sure that the location is available.

Syntax

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

Parameters

The MakeLocationAvailable method has the following parameters:

- [in] BSTR LocationAvailableXML
- [out,retval] ReturnCode* retVal

  Indicates whether or not the request was completed.

LocationAvailable XML block

The LocationAvailable XML block lists the device locations (defined by the device XML block provided to VWorks software with a previous call to the GetMetaData method) and any labware that has been placed there by the device manager.

XML structure

The file attribute for the <Velocity11> element is MetaData.

```
<Velocity11>
   <LocationAvailable>
      <StorageLocation >
         <Coordinates >
            <StorageLocationCoordinate/>
            ...
         </Coordinates>
         <Location />
      </StorageLocation>
      <Command>
         <Parameters>
            <Parameter />
            ...
         </Parameters>
      </Command>
   </LocationAvailable>
</Velocity11>
```

<LocationAvailable> element

The <LocationsAvailable> element has the following attributes:

Attribute Name	Description
Device	The device name.
Location	The plug-in's name for the requested location on the device. This should match one of the locations returned previously with the GetMetaData method in the device metadata.
Robot	The user's name for the robot. *Note:* If the Device and Robot values are the same, then the plug-in can treat this as an internal motion.

The <LocationAvailable> element has two child elements:

- <StorageLocation>
- <Command>.

<StorageLocation> element

The <StorageLocation> element is provided for storage devices only. Other device driver plug-ins can ignore this element. The <StorageLocation> element has no attributes. This provides the coordinates for the location. See “StorageLocation XML block” on page 143 for a detailed description of this element.

<Command> element

The <Command> element provides the command that the location for which the location is being made available. The <Command> element has the standard <Command> element attributes.

Example

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData'
md5sum='200901a13dc2736cc812d80b6cfd0213'
version='1.0' >
   <LocationAvailable Device='Bravo-1'
   Location='8' Robot='Bravo-1' >
      <StorageLocation >
         <Coordinates >
            <StorageLocationCoordinate
            Name='Cassette' Value='1' />
            <StorageLocationCoordinate
            Name='Slot' Value='1' />
         </Coordinates>
         <Location Group='0'
         MaxStackHeight='500' Offset='0'
         Type='1' />
      </StorageLocation>
      <Command Compiler='0' Description='Read
      the plate' Editor='2' Name='Read'>
         <Parameters>
            <Parameter Name='Protocol' Style='0'
            Type='2' Value='Simulation protocol
            1' />
            ...
         </Parameters>
      </Command>
   </LocationAvailable>
</Velocity11>
```

Related topics

For information about...	See...
Description of the possible return codes	"Return Code values" on page 47
Plate delivery cycle methods	❑ "PlatePickedUp method" on page 66 ❑ "PlateDroppedOff method" on page 65 ❑ "PlateTransferAborted method" on page 67
StorageLocation XML block for location coordinates	"StorageLocation XML block" on page 143
<Location> element attributes	"<Location> element" on page 29
<Command> element attributes	"<Command> element" on page 40
<Velocity11> element attributes	"<Velocity11> element" on page 25

PlateDroppedOff method

Description

The PlateDroppedOff method is called when the device's location is the target for a plate transfer. This includes robot transfers (which are preceded by a call to the MakeLocationAvailable method), and configured labware.

A call to either PlateDroppedOff, PlatePickedUp, or PlateTransferAborted will always occur before any other calls to this plug-in once the call to the MakeLocationAvailable method has been performed.

Syntax

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

Parameters

The PlateDroppedOff method has the following parameter:

❑ [in] BSTR PlateInfoXML

The PlateInfoXML string is an XML block containing information about the plate that was dropped off.

Related topics

For information about...	See...
The PlateInfo XML block	"PlateInfo XML block" on page 43

For information about...	See...
Plate delivery cycle methods	❑ "PlatePickedUp method" on page 66 ❑ "PlateTransferAborted method" on page 67

PlatePickedUp method

Description

The PlatePickedUp method is called when the device's location is the source of a plate transfer. This includes robot transfers (which are preceded by a call to the MakeLocationAvailable method), and configured labware.

A call to either PlateDroppedOff, PlatePickedUp, or PlateTransferAborted will always occur before any other calls to this plug-in once the call to the MakeLocationAvailable method has been performed.

Syntax

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

Parameters

The PlatePickedUp method has the following parameter:

❑ [in] BSTR PlateInfoXML

The PlateInfoXML string is an XML block containing information about the plate that was picked up.

Related topics

For information about...	See...
The PlateInfo XML block	"PlateInfo XML block" on page 43
Plate delivery cycle methods	❑ "PlateDroppedOff method" on page 65 ❑ "PlateTransferAborted method" on page 67

PlateTransferAborted method

Description

The PlateTransferAborted method is called when a plate transfer operation was aborted (typically due to a robot malfunction followed by user intervention).

A call to either PlateDroppedOff, PlatePickedUp, or PlateTransferAborted will always occur before any other calls to this plug-in once the call to the MakeLocationAvailable method has been performed.

Syntax

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

Parameters

The PlateTransferAborted method has the following parameter:

- [in] BSTR PlateInfoXML

  The PlateInfoXML string is an XML block containing information about the plate transfer that was aborted.

Related topics

For information about...	See...
The PlateInfo XML block	"PlateInfo XML block" on page 43
Plate delivery cycle methods	❑ "PlateDroppedOff method" on page 65 ❑ "PlatePickedUp method" on page 66

PrepareForRun method

Description

The PrepareForRun method is called when user clicks the **Start** button in VWorks software. This method notifies the plug-in that a run is starting and is called each time a protocol is run. If the plug-in maintains per-run state information, this state should be cleared during this call.

Syntax

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

Parameters

The PrepareForRun method has the following parameters:

- [in] BSTR LocationInfoXML

  The LocationInfoXML parameter is an LocationInfo XML block that lists the device locations.

- [out,retval] ReturnCode* retVal

  Indicates whether or not the request was completed.

LocationInfo XML block

The LocationInfo XML block lists the device locations (defined by the device XML block provided to VWorks software with a previous call to the GetMetaData method) and any labware that has been placed there by the device manager.

XML structure

The file attribute for the <Velocity11> element is LocationInfo.

```
<Velocity11>
   <Locations>
      <Location />
      ...
   </Locations>
</Velocity11>
```

<Locations> element

The <Locations> element is used to group one or more <Location> elements. This element has no attributes and the only child element is the <Location> element.

<Location> element

The <Location> element has the following attributes:

Attribute Name	Description
Name	Name of the location
Labware	Name of the labware

Example

```
<?xml version='1.0' encoding='ASCII' ?>
   <Velocity11 file='LocationInfo'
   md5sum='849392019ca47102839e845113d11840'
   version='1.0' >
      <Locations>
         <Location Name='Stage' Labware='96 V11
         LT200 Tip Box 06880.002' />
         ...
      </Locations>
</Velocity11>
```

Related topics

For information about...	See...
<Velocity11> element attributes	"<Velocity11> element" on page 25
Description of possible return codes	"Return Code values" on page 47

Retry method

Description

The Retry method is called when a user has clicked **Retry** in the standard error dialog box.

The plug-in should retry the operation which caused the error because it is assumed that the user has manually solved the problem that caused the error. The plug-in should record the state of the operation and retry from the point in the operation that makes the most sense given the current state.

For example, a single-column dispenser that encounters an error after partially filling a plate should not start over (and thus deliver too much reagent to the already covered wells), but rather continue as close to the point of interruption as possible (possibly over or under-dispensing the wells that were being filled at time of error).

Syntax

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

Parameters

The Retry method has the following parameter:

- [out,retval] ReturnCode* retVal

  Indicates whether or not the request was completed. A return code of RETURN_FAIL will return to the error recovery loop.

Related topics

For information about...	See...
Error handling methods	❑ "Abort method" on page 49 ❑ "Ignore method" on page 59
Providing an error message to VWorks software	"GetErrorInfo method" on page 55
Description of possible return codes	"Return Code values" on page 47

ShowDiagsDialog method

Description

The ShowDiagsDialog method is called when the user wants to open diagnostics, the dialog box used to control the device directly and in real time.

You can either open a dialog box provided by the device's vendor or you can create your own.

Syntax

```
HRESULT ShowDiagsDialog([in] SecurityLevel
iSecurity);
```

Parameters

The ShowDiagsDialog method has the following parameter:

❑ [in] SecurityLevel iSecurity

Indicates the security level for the user currently logged into VWorks software. The valid values for SecurityLevel are:

Value	Description
0	Access level privilege for the current user is Administrator
1	Access level privilege for the current user is Technician
2	Access level privilege for the current user is Operator
3	Access level privilege for the current user is Guest
4	No user is currently logged into VWorks software

Related topics

For information about...	See...
Testing the ShowDiagsDialog method	"Testing the diagnostics dialog" on page 195

IWorksController interface reference

5

This chapter describes the IControllerClient, the IWorksController interfaces, and the methods for both.

Note: You must also implement the IWorksDriver interface methods if you implement the IControllerClient and IWorksController methods.

This chapter covers the following topics:

- ❑ "Overview" on page 72
- ❑ "SetController method" on page 73
- ❑ "PrintToLog method" on page 73
- ❑ "Query method" on page 74
- ❑ "Update method" on page 101

Overview

About this topic

This topic describes the IControllerClient and IWorksController interfaces and lists the methods.

Introduction

The IControllerClient and IWorksController interfaces provide a way for the driver plug-in to request that VWorks software perform any of following functions.

- ❑ Print to the VWorks software log file
- ❑ Query information from VWorks software
- ❑ Update information in VWorks software

To use the IWorksController interface methods, you must first implement the IControllerClient interface method SetController to get a pointer to the IWorksController.

Method list

The following table lists the IControllerClient and IWorksController interface methods.

Method	Description	See...
IControllerClient methods		
SetController	Tells the driver how to talk to the controller. Required before using the IWorksController class methods.	"SetController method" on page 73
IWorksController methods		
PrintToLog	Tells VWorks software to print the given string to the main log file.	"PrintToLog method" on page 73
Query	Sends a query to VWorks software requesting information.	"Query method" on page 74
Update	Sends an update request to VWorks software.	"Update method" on page 101

SetController method

Description

Use the SetController method to obtain a pointer to the VWorks software controller so you can use the IWorksController class methods. This method is called from VWorks software when the plug-in is initially loaded.

Syntax

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

Parameters

The SetController method has the following parameter:

- [in] IWorksController *Controller

Related topics

For information about...	See...
Calling VWorks software with a request to print to the log	"PrintToLog method" on page 73
Calling VWorks software with a query	"Query method" on page 74
Calling VWorks software with an update	"Update method" on page 101

PrintToLog method

Description

Use the PrintToLog method to request that VWorks software print the provided string to the Log toolbar and the protocol log.

Syntax

```
HRESULT PrintToLog([in] IControllerClient
*Source, [in] BSTR StringToPrint);
```

Parameters

The PrintToLog method has the following parameters:

- [in] IControllerClient *Source

  Pointer to the plug-in so that VWorks software can identify the source of the request.

- [in] BSTR StringToPrint

  String to print to the log file.

Related topics

For information about...	See...
Getting the IControllerClient pointer	"SetController method" on page 73

Query method

Description

Use the Query method to send a request for information from VWorks software.

Syntax

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

Parameters

The Query method has the following parameters:

- [in] IControllerClient *Source

  Pointer to the plug-in so that VWorks software can identify the source of the request.

- [in] BSTR Query

  A Query XML block to describe the query to VWorks software.

- [out, retval] BSTR* QueryResult

  A QueryResponse XML block containing the results of the query from VWorks software.

Query XML block

The Query XML block contains information about the request to VWorks software.

XML structure

```
<Velocity11>
   <Query>
      <Parameters>
         <Parameter />
         ...
      </Parameters>
   </Query>
</Velocity11>
```

<Query> element

Specifies the type of information you are requesting. The attributes for this element are:

Attribute Name	Description
Category	The category describes the type of query. The valid values are: ❑ `Barcode` Get the bar code from the plate at the specified location. ❑ `DeviceLocationTeachpoints` Get all locations on all devices for which the robot has teachpoints. ❑ `GetJavascriptVariable` Get the value of a specified JavaScript variable. ❑ `GetRunSetStatus` Get information about the current or next scheduled protocol from the run-set manager. ❑ `InterPlugin` Provides a method for plug-in to plug-in communication. ❑ `IsDeviceManagerDirty` Determine if the device manager has been modified since the last time it was saved. ❑ `Labware` Get more information about a specified labware entry. ❑ `LocationToTeachpoints` Get the teachpoints for a specific location for the device. ❑ `ScanBarcode` Determine whether a bar code scanner should be used. ❑ `SystemPlateInformation` Get the type of labware for the specified plate. ❑ `TeachpointInformation` Get information on the specified robot's teachpoint.
Destination	For InterPlugin queries, specify the name of the plug-in that should receive the query. For queries that are directed to VWorks software, this attribute is optional.
Source	Specifies the name of the plug-in making the request. This attribute is required only for InterPlugin queries.

<Parameters> element

The <Parameters> element groups zero or more <Parameter> elements and has no attributes.

<Parameter> element

There can be zero or more <Parameter> elements. Whether a <Parameter> element is required depends on the type of query (specified in the <Query> element Category attribute).

Query response XML block

The query Response XML block contains the requested information.

XML structure

```
<Velocity11>
   <Response>
      <Parameters>
         <Parameter />
         ...
      </Parameters>
   </Response>
</Velocity11>
```

<Response> element

The <Response> element indicates the type of information that was requested. The attributes for this element are:

Attribute Name	Description
Category	The category value in the response matches the category provided in the query. The valid values are: ❑ Barcode ❑ DeviceLocationTeachpoints ❑ GetJavascriptVariable ❑ GetRunSetStatus ❑ InterPlugin ❑ IsDeviceManagerDirty ❑ Labware ❑ LocationToTeachpoints ❑ ScanBarcode ❑ SystemPlateInformation ❑ TeachpointInformation
Destination	This attribute is only used in InterPlugin responses to specify the name of the plug-in that should receive the response.
Source	This attribute is only used in InterPlugin responses to specify the name of the plug-in making the request.

<Parameters> element

The <Parameters> element groups zero or more <Parameter> elements and has no attributes.

<Parameter> element

There can be zero or more <Parameter> elements. Whether a <Parameter> element is required depends on the type of query response (specified in the <Response> element Category attribute).

Barcode query XML block

Use the Barcode query to request a barcode for a plate at a specified location.

Note: The Barcode Query XML does not include the <Parameters> element.

XML structure

```
<Velocity11>
   <Query>
      <Parameter />
   </Query>
</Velocity11>
```

<Parameter> element

Provide the location of the plate with the following attribute pair:

Attribute Name	Description
Name	`Location`
Value	The name of the location.

Example

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Query'
md5sum='f7038cb092493fad9312e4164873c8a2'
version='1.0' >
   <Query Category='Barcode'>
      <Parameter Name='Location' Value='stage1'
      />
   </Query>
</Velocity11>
```

Barcode query response XML block

VWorks software provides the following response for a Barcode query.

Note: The Barcode query Response XML does not include the <Parameters> element.

XML structure

```
<Velocity11>
```

```
    <Response>
      <Parameter>
        <values>
          <value />
          ...
        </values>
      </Parameter>
    </Response>
  </Velocity11>
```

<Parameter> element

The <Parameter> element has the following attribute:

Attribute Name	Description
Name	`Barcodes`

<values> element

The <values> element will contain one or more <values> elements. This element has no attributes.

<value> element

The <value> elements contain the bar code information requested from VWorks software. The <value> attribute has the following attributes:

Attribute Name	Description
side	A number to indicate the side of the plate that has the bar code (0–3).
value	The bar code.

Example

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='QueryResponse'
md5sum='f7038cb092493fad9312e4164873c8a2'
version='1.0' >
```

```
   <Response Category='Barcode'>
      <Parameters>
         <Parameter Name='Barcodes'>
            <values>
               <value side='0' value='2B99A1' />
               <value side='1' value='2B9988' />
               <value side='2' value='' />
               <value side='3' value='' />
            </values>
         </Parameter>
      </Parameters>
   </Response>
</Velocity11>
```

Device Location Teachpoints query XML block

Use the DeviceLocationTeachpoints query to request that VWorks software provide all locations on all devices that the robot is able to reach.

Note: If a non-robot device uses this query, an empty DeviceLocationTeachpoints XML block is returned.

The <DeviceLocationTeachpoints> query XML block has no <Parameter> elements.

Example

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Query'
md5sum='f7038cb092493fad9312e4164873c8a2'
version='1.0' >
   <Query Category='DeviceLocationTeachpoints'>
      <Parameters />
   </Query>
</Velocity11>
```

Device Location Teachpoints query response XML block

VWorks software provides all locations on all devices for which the robot has teachpoints.

XML structure

```
<Velocity11>
   <Response>
      <Parameters>
         <Parameter />
      </Parameters>
   </Response>
<Velocity11>
```

<Parameter> element

The <Parameter> element has the following attributes:

Attribute Name	Description
Name	`DeviceLocationTeachpoints`
Value	An escaped DeviceLocationTeachpoints XML block.

DeviceLocationTeachpoints XML block

The Value attribute of the DeviceLocationTeachpoints Response <Parameter> contains an escaped DeviceLocationTeachpoints XML block.

XML structure

```
<DeviceLocationTeachpoints>
      <DeviceLocationTeachpoint />
      ...
</DeviceLocationTeachpoints>
```

<DeviceLocationTeachpoints> element

The <DeviceLocationTeachpoints> element groups one or more <DeviceLocationTeachpoint> elements and has no attributes.

<DeviceLocationTeachpoint> element

Each <DeviceLocationTeachpoint> element contains information for each location. This element has the following attributes:

Attribute Name	Description
DeviceName	The name of the device that called the query.
DeviceType	The type of device that called the query.
LocationName	The name of the location on the device.
RobotName	The name of the robot on which the location is taught.
RobotType	The type of robot.
TeachpointName	The name of the teachpoint that the robot has associated with the specified location.

DeviceLocationTeachpoints query response XML example

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='QueryResponse'
md5sum='f7038cb092493fad9312e4164873c8a2'
version='1.0' >
   <Response
   Category='DeviceLocationTeachpoints'>
      <Parameters>
         <Parameter
         Name='DeviceLocationTeachpoints'
         Value='&lt;DeviceLocationTeachpoints&g
         t; &lt;DeviceLocationTeachpoint
         DeviceName=&quot;DeviceA&quot;
         DeviceType=&quot;plugin:Static
         Stack&quot; …./&gt;
         &lt;DeviceLocationTeachpoint
         DeviceName=&quot;DeviceB&quot;
         DeviceType=&quot;plugin:Static
         Stack&quot; …./&gt; &lt;/
         DeviceLocationTeachpoints&gt;' />
      </Parameters>
   </Response>
</Velocity11>
```

Get Javascript Variable query XML block

Use the GetJavascriptVariable query to obtain the value of a specified JavaScript variable.

XML structure

```
<Velocity11>
   <Query>
      <Parameters>
         <Parameter />
      </Parameters>
   </Query>
</Velocity11>
```

<Parameter> element

The <Parameter> element has the following attribute:

Attribute Name	Description
VariableName	The name of the JavaScript variable for which the value is needed.

Example

```
<?xml version='1.0' encoding='ASCII' ?>
```

```
<Velocity11 file='Query'
md5sum='06fae07c4f1f38aa7e8d970b97b94fde'versi
on='1.0' >
   <Query Category='GetJavascriptVariable'>
      <Parameters>
         <Parameter VariableName='MyVariable' /
         >
      </Parameters>
   </Query>
</Velocity11>
```

Get Javascript Variable query response XML block

The GetJavascriptVariable query Response XML block contains the following information:

XML structure

```
<Velocity11>
   <Response>
      <Parameters>
         <Parameter />
      </Parameters>
   </Response>
</Velocity11>
```

<Parameter> element

The <Parameter> element contains the information requested and has the following attribute pair:

Attribute Name	Description
Name	`VariableValue`
Value	An escaped JSObject XML block.

JSObject XML block

The JSObject XML block provides the serialization of the JavaScript variable.

XML structure

```
<JSObject>
   <JSProperty />
</JSObject>
```

<JSObject> element

The <JSObject> element has the following possible attribute pairs:

Attribute Name	Description
❑ Type ❑ Value	❑ `Nothing` ❑ Not used
❑ Type ❑ Value	❑ `Array` ❑ Not used—The array values are provided in the child <JSObject> elements
❑ Type ❑ Value	❑ `Hash` ❑ Not used—The values are provided in the child <JSObject> and <JSProperty> elements
❑ Type ❑ Value	❑ `String` ❑ The string value
❑ Type ❑ Value	❑ `Double` ❑ The string representation of the double value
❑ Type ❑ Value	❑ `Int` ❑ The string representation of the integer value

JSObject XML block array example

This example shows an array of integers.

```
<JSObject Type='Array'>
      <JSObject Type='Int' Value='94' />
      <JSObject Type='Int' Value='73' />
</JSObject>
```

JSObject XML block double example

```
<JSObject Type='Double' Value='590.91' />
```

JSObject XML block hash example

```
<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>
```

GetJavascriptVariable query response XML example

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='QueryResponse'
md5sum='f7038cb092493fad9312e4164873c8a2'
version='1.0' >
   <Response Category='JavascriptVariable'>
      <Parameters>
         <Parameter Name='VariableValue'
         Value='&lt;JSObject&gt;
         Type=&apos;Int&apos;
         Value=&apos;89&apos; /&gt;' />
      </Parameters>
   </Response>
</Velocity11>
```

Get Run Set Status query XML block

Use the GetRunSetStatus query to obtain information about the current or next scheduled protocol in the run-set manager.

The GetRunSetStatus query has no <Parameter> elements.

Example

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Query'
md5sum='06fae07c4f1f38aa7e8d970b97b94fde'versi
on='1.0' >
   <Query Category='GetRunSetStatus'>
      <Parameters />
   </Query>
</Velocity11>
```

Get Run Set Status query response XML block

VWorks software provides information about the current protocol or the next scheduled protocol in the run-set manager for the GetRunSetStatus query.

XML structure

```
<Velocity11>
   <Response>
      <Parameters>
         <Parameter />
      </Parameters>
   </Response>
</Velocity11>
```

<Parameter> element

The <Parameter> element contains the requested information and has the following attribute pairs:

Attribute Name	Description
❑ Name ❑ Value	❑ `ProtocolName` ❑ The name and full path of the protocol file.
❑ Name ❑ Value	❑ `StartTime_year` ❑ The start year (4 digits).
❑ Name ❑ Value	❑ `StartTime_month` ❑ The start month 1–12.
❑ Name ❑ Value	❑ `StartTime_day` ❑ The start day 1–31.
❑ Name ❑ Value	❑ `StartTime_hour` ❑ The start hour 1–24.
❑ Name ❑ Value	❑ `StartTime_minutes` ❑ The start minute 0–59.
❑ Name ❑ Value	❑ `StartTime_seconds` ❑ The start seconds 0–59.
❑ Name ❑ Value	❑ `NumberOfRuns` ❑ The number of runs to perform.
❑ Name ❑ Value	❑ `PluginFileName` ❑ Filename of the plug-in.
❑ Name ❑ Value	❑ `ProtocolNotes` ❑ The protocol notes associated with the run.

Example

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11
file='QueryResponse'md5sum='486f8ddc4a1bf6c8f0
8ae5f1f079f7fe' version='1.0' >
```

```
<Response Category='GetRunSetStatus'>
   <Parameters>
      <Parameter Name='ProtocolName'
      Value='C:\Protocols\sample-
      protocol.pro' />
      <Parameter Name='StartTime_year'
      Value='2007' />
      <Parameter Name='StartTime_month'
      Value='2' />
      <Parameter Name='StartTime_day'
      Value='5' />
      <Parameter Name='StartTime_hour'
      Value='10' />
      <Parameter Name='StartTime_minutes'
      Value='0' />
      <Parameter Name='StartTime_seconds'
      Value='0' />
      <Parameter Name='NumberOfRuns'
      Value='2' />
      <Parameter Name='PluginFileName'
      Value='my_PluginName_filename' />
      <Parameter Name='ProtocolNotes'
      Value='Notes about this protocol' />
   </Parameters>
  </Response>
</Velocity11>
```

InterPlugin query XML block

Use the InterPlugin query for plug-in to plug-in communication.

XML structure

```
<Velocity11>
   <Query>
      <Parameters>
         <Parameter />
      </Parameters>
   </Query>
</Velocity11>
```

<Parameter> element

The <Parameter> element contains the requested information. This element has the following attributes:

Attribute Name	Description
Name	`InnerQuery`
Value	An escaped XML block for the destination plug-in.

Example

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Query'
md5sum='06fae07c4f1f38aa7e8d970b97b94fde'versi
on='1.0' >
   <Query Category='InterPlugin'
   Destination='kinedx' Source='stack 1' >
      <Parameters>
         <Parameter Name='InnerQuery'
         Value='&lt;?xml
         version=&apos;1.0&apos;
         encoding=&apos;ASCII&apos; ?&gt;
         &lt;Velocity11
         file=&apos;MetaData&apos;
         md5sum=&apos;af35bf19a9febc11ab2d51f22
         d6b61b3&apos; version=&apos;1.0&apos;
         &gt; &lt;Query
         AccessLocation=&apos;stack3
         access&apos; BaseLocation=&apos;stack3
         base&apos; Category=&apos;scan&apos;
         SecurityLevel=&apos;0&apos; &gt;
         &lt;Parameters &gt; &lt;Parameter
         Name=&apos;Stack-height scripting
         variable&apos; Style=&apos;0&apos;
         Type=&apos;1&apos;
         Value=&apos;static_stack_height&apos;
         &gt; &lt;/Parameters&gt; &lt;/
         Query&gt; &lt; Velocity11&gt;' />
      </Parameters>
   </Query>
</Velocity11>
```

InterPlugin query response XML block

The InterPlugin query Response XML block contains the information returned from the other plug-in.

XML structure

```
<Velocity11>
```

```
   <Response>
      <Parameters>
         <Parameter />
      </Parameters>
   </Response>
</Velocity11>
```

<Parameter> element

The <Parameter> element contains the information requested and has the following attributes:

Attribute Name	Description
Name	`InnerResponse`
Value	An escaped XML block from the plug-in originating the request.

Example

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11
file='QueryResponse'md5sum='486f8ddc4a1bf6c8f0
8ae5f1f079f7fe' version='1.0' >
   <Response Category='InterPlugin'
   Destination='stack 1' Source='kinedx' >
      <Parameters>
         <Parameter Name='InnerResponse'
         Value='&lt;?xml
         version=&apos;1.0&apos;
         encoding=&apos;ASCII&apos;
         ?&gt;&lt;Velocity11
         file=&apos;MetaData&apos;
         md5sum=&apos;cbd936a09d0b2e2243666851b
         2d0425f&apos;version=&apos;1.0&apos;
         &gt;&lt;Response
         Result=&apos;0&apos;StackPosition=&apo
         s;-178.961&apos; Success=&apos;1&apos;
         ValidStackPosition=&apos;1&apos; &gt;
         &lt;Parameters &gt;&lt;Parameter
         Name=&apos;Object Found&apos;
         Style=&apos;0&apos; Type=&apos;0&apos;
         Value=&apos;1&apos; /&gt;&lt;/
         Parameters&gt; &lt;/Response&gt; &lt;/
         Velocity11&gt;' />
      </Parameters>
   </Response>
</Velocity11>
```

Is Device Manager Dirty query XML block

Use the IsDeviceManagerDirty query to determine if the device manager has been modified since the last time it was saved.

The IsDeviceManagerDirty query has no <Parameter> elements.

Example

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Query'
md5sum='06fae07c4f1f38aa7e8d970b97b94fde'versi
on='1.0' >
   <Query Category='IsDeviceManagerDirty'>
      <Parameters />
   </Query>
</Velocity11>
```

Is Device Manager Dirty query response XML block

VWorks software provides whether the device manager has been modified since the last save in a response to the IsDeviceManagerDirty query.

XML structure

```
<Velocity11>
   <Response>
      <Parameters>
         <Parameter />
      </Parameters>
   </Response>
</Velocity11>
```

<Parameter> element

The <Parameter> element contains the information requested and has the following attribute pair:

Attribute Name	Description
Name	`Dirty`
Value	Indicates whether the device manager has been modified. The possible values are: 0 = The device manager has not been modified 1 = The device manager has been modified since the last time it was saved.

Example

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11
file='QueryResponse'md5sum='486f8ddc4a1bf6c8f0
8ae5f1f079f7fe' version='1.0' >
```

```
   <Response Category='IsDeviceManagerDirty'>
      <Parameters>
         <Parameter Name='Dirty' Value='0' />
      </Parameters>
   </Response>
</Velocity11>
```

Labware query XML block

Use the Labware query XML block to obtain details about a specific type of labware from VWorks software.

Note: The Labware Query XML does not include the <Parameters> element.

XML structure

```
<Velocity11>
   <Query>
      <Parameter />
   </Query>
</Velocity11>
```

<Parameter> element

The <Parameter> element has the following attributes:

Attribute Name	Description
Name	`Labware_Entry`
Value	The name of labware type.

Example

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Query'
md5sum='f7038cb092493fad9312e4164873c8a2'
version='1.0' >
   <Query Category='Labware'>
      <Parameter Name='Labware_Entry'
      Value='384 V11 ST10 Tip Box 10734.102' />
   </Query>
</Velocity11>
```

Labware query response XML block

VWorks software provides the following information for a Labware query.

Note: The Labware query Response XML does not include the <Parameters> element.

XML structure

```
<Velocity11>
   <Response>
      <Parameter />
         <values>
            <value />
            ...
         </values>
   </Response>
</Velocity11>
```

<Parameter> element

The <Parameter> element has the following attributes:

Note: This is the same information provided in the Labware Query XML block.

Attribute Name	Description
Name	`Labware_Entry`
Value	The name of labware type.

<values> element

The <values> element will contain one or more <value> elements and has no attributes.

<value> element

The <value> elements contain the labware information from VWorks software. You will receive a <value> element for each labware property (including general properties, plate properties, stacker properties, and VPrep/well properties) regardless of the labware type. Properties that do not apply to the specific labware will have the value attribute set to 0. The <value> element has the following attributes:

Attribute Name	Description
name	The name of the labware property. The valid properties in the response include: A12_NOTCH, A1_NOTCH, BASE_CLASS BC_ERROR_CORRECTION_OFFSET, BC_GRIPPER_HOLDING_LIDDED_PLATE_POSITION, BC_GRIPPER_HOLDING_LID_POSITION, BC_GRIPPER_HOLDING_PLATE_POSITION, BC_GRIPPER_HOLDING_STACK_POSITION, BC_GRIPPER_OPEN_POSITION, BC_ROBOT_GRIPPER_OFFSET, CAN_BE_MOUNTED, CAN_BE_SEALED, CAN_HAVE_LID, CAN_MOUNT, CHECK_PLATE_ORIENTATION, DESCRIPTION, DISPOSABLE_TIP_LENGTH, FILTER_TIP_PIN_TOOL_LENGTH, H12_NOTCH, H1_NOTCH, LIDDED_STACKING_THICKNESS, LIDDED_THICKNESS, LID_DEPARTURE_HEIGHT, LID_RESTING_HEIGHT, LOWER_PLATE_AT_VCODE, MANUFACTURER_PART_NUMBER, MOUNTED_LID_ROBOT_GRIPPER_OFFSET, NUMBER_OF_WELLS, PRESENDTAITON_OFFSET, ROBOT_GRIPPER_OFFSET, ROBOT_HANDLING_SPEED, SEALED_STACKING_THICKNESS, SEALED_THICKNESS, SENSOR_INTENSITY, SENSOR_OFFSET, SENSOR_THRESHOLD, SENSOR_THRESHOLD_MIN, STACKER_GRIPPER_OFFSET, STACKING_THICKNESS, THICKNESS, USE_VACUUM_CLAMP, WELL_BOTTOM_SHAPE, WELL_DEPTH, WELL_DIAMETER, WELL_GEOMETRY, WELL_TIP_VOLUME, X_TEACHPOINT_TO_WELL, X_WELL_TO_WELL, Y_TEACHPOINT_TO_WELL, Y_WELL_TO_WELL, Z_TIP_ATTACH_OFFSET, NAME
type	A type code that indicates the type of data. This is always a 1 to indicate a string.
value	The value of the labware property.

Example

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='QueryResponse'
md5sum='f7038cb092493fad9312e4164873c8a2'
version='1.0' >
```

```
<Response Category='Labware'>
   <Parameter Name='Labware_Entry' Value='96
   Costar 3693 PS LVol White rnd well' />
      <values>
         <value name='NAME' type='1'
         value='96 Costar 3693 PS LVol White
         rnd well' />
         <value
         name='MANUFACTURER_PART_NUMBER'
         type='1' value='3693' />
         <value name='NUMBER_OF_WELLS'
         type='1' value='96' />
         <value name='BASE_CLASS' type='1'
         value='1' />
         ...
      </values>
   </Response>
</Velocity11>
```

Location To Teachpoints query XML block

Use the LocationToTeachpoints query to request that VWorks software find the location on the device for the specified location name and provide all the teachpoints taught for that location.

XML structure

```
<Velocity11>
   <Query>
      <Parameters />
         <Parameter />
      </Parameters>
   </Query>
</Velocity11>
```

<Parameter> element

The <Parameter> element has the following attributes:

Attribute Name	Description
Name	`LocationName`
Value	The name of the location on the device.

Example

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Query'
md5sum='f7038cb092493fad9312e4164873c8a2'
version='1.0' >
   <Query Category='LocationToTeachpoints'>
      <Parameter Name='LocationName'
      Value='Stage1' />
   </Query>
</Velocity11>
```

Location To Teachpoints query response XML block

VWorks software provides all the teachpoints taught for the location specified in the LocationToTeachpoints query.

XML structure

```
<Velocity11>
   <Response>
      <Parameters>
         <Parameter />
         ...
      </Parameters>
   </Response>
<Velocity11>
```

<Parameter> element

The <Parameter> element has the following attributes:

Attribute Name	Description
Name	`DeviceLocationTeachpoints`
Value	An escaped DeviceLocationTeachpoints XML block.

DeviceLocationTeachpoints XML block

The Value attribute of the LocationToTeachpoints Response <Parameter> contains an escaped DeviceLocationTeachpoints XML block.

XML structure

```
<DeviceLocationTeachpoints>
      <DeviceLocationTeachpoint />
      ...
</DeviceLocationTeachpoints>
```

<DeviceLocationTeachpoints> element

The <DeviceLocationTeachpoints> element groups one or more <DeviceLocationTeachpoint> elements and has no attributes.

<DeviceLocationTeachpoint> element

Each <DeviceLocationTeachpoint> element contains information for one of the teachpoints. The <DeviceLocationTeachpoint> element has the following attributes:

Attribute Name	Description
DeviceName	The name of the device that called the query.
DeviceType	The type of device that called the query.
LocationName	The name of the location on the device.
RobotName	The name of the robot on which the location is taught.
RobotType	The type of robot.
TeachpointName	The name of the teachpoint that the robot has associated with the specified location.

LocationToTeachpoints query response XML example

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='QueryResponse'
md5sum='f7038cb092493fad9312e4164873c8a2'
version='1.0' >
   <Response Category='LocationToTeachpoints'>
      <Parameters>
         <Parameter
         Name='DeviceLocationTeachpoints'
         Value='&lt;DeviceLocationTeachpoints&g
         t; &lt;DeviceLocationTeachpoint
         DeviceName=&apos;DeviceA&apos;
         DeviceType=&apos;plugin:Static
         Stack&apos; …./&gt;
         &lt;DeviceLocationTeachpoint
         DeviceName=&apos;DeviceA&apos;
         DeviceType=&apos;plugin:Static
         Stack&apos; …./&gt; &lt;/
         DeviceLocationTeachpoints&gt;' />
      </Parameters>
   </Response>
</Velocity11>
```

Scan Barcode query XML block

Use the ScanBarcode to query whether a bar code scanner should be used. The response is determined by the process parameter barcode control settings in VWorks software.

Note: The ScanBarcode Query XML does not include the <Parameters> element.

XML structure

```
<Velocity11>
   <Query>
      <Parameter />
      ...
   </Query>
</Velocity11>
```

<Parameter> element

Provide two parameter elements with the following attribute pairs:

Attribute Name	Description
❑ Name ❑ Value	❑ `Location` ❑ The name of the location.
❑ Name ❑ Value	❑ `Side` ❑ The number to indicate the side of the plate as 0–3.

Example

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Query'
md5sum='f7038cb092493fad9312e4164873c8a2'
version='1.0' >
   <Query Category='ScanBarcode'>
      <Parameter Name='Location' Value='stage1'
      />
      <Parameter Name='Side' Value='0' />
   </Query>
</Velocity11>
```

Scan Barcode query response XML block

VWorks software provides the following Response XML block for a ScanBarcode query.

Note: The ScanBarcode query Response XML does not include the <Parameters> element.

XML structure

```
<Velocity11>
   <Response>
      <Parameter />
   </Response>
</Velocity11>
```

<Parameter> element

The <Parameter> element has the following attributes:

Attribute Name	Description
Name	`ShouldScan`
Value	Indicates whether or not a bar code scanner should be used. The valid values are: `yes` = A bar code scanner should be used. `no =` A barcode scanner should not be used.

Example

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='QueryResponse'
md5sum='f7038cb092493fad9312e4164873c8a2'
version='1.0' >
   <Response Category='ScanBarcode'>
      <Parameter Name='ShouldScan' value='yes'
      />
   </Response>
</Velocity11>
```

System Plate Information query XML block

Use the SystemPlateInformation query to get the type of labware for the specified plate.

XML structure

```
<Velocity11>
   <Query Category='SystemPlateInformation'>
      <Parameters>
         <Parameter />
      </Parameters>
   </Query>
</Velocity11>
```

<Parameter> element

The <Parameter> element has the following attribute:

Attribute Name	Description
PlateName	The name of the plate for which you want the type of labware.

Example

```
<?xml version='1.0' encoding='ASCII' ?>
```

```
<Velocity11 file='Query'
md5sum='f7038cb092493fad9312e4164873c8a2'
version='1.0' >
   <Query Category='SystemPlateInformation'>
      <Parameters>
         <Parameter PlateName='PlateAB' />
      </Parameters>
   </Query>
</Velocity11>
```

System Plate Information query response XML block

VWorks software provides the following information for a SystemPlateInformation query.

XML structure

```
<Velocity11>
   <Response>
      <Parameters>
         <Parameter />
      <Parameters>
   </Response>
</Velocity11>
```

<Parameter> element

The <Parameter> element has the following attribute pair:

Attribute Name	Description
Name	`Labware`
Value	The name of the labware for the plate.

Example

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='QueryResponse'
md5sum='f7038cb092493fad9312e4164873c8a2'
version='1.0' >
   <Response Category='SystemPlateInformation'>
      <Parameters>
         <Parameter Name='Labware' Value='96 V11
         LT200 Tip Box 06880.002' />
      </Parameters>
   </Response>
</Velocity11>
```

Teachpoint Information query XML block

Use the TeachpointInformation query to request more information about a specified robot's teachpoint.

XML structure

```
<Velocity11>
   <Query>
      <Parameters>
         <Parameter />
      </Parameters>
   </Query>
</Velocity11>
```

<Parameter> element

The <Parameter> element has the following attributes:

Attribute Name	Description
RobotName	The name of the robot.
TeachpointName	The name of the teachpoint for the specified robot.

Example

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Query'
md5sum='f7038cb092493fad9312e4164873c8a2'
version='1.0' >
   <Query Category='TeachpointInformation'>
      <Parameter RobotName='RobotA'
      TeachpointName='Teachpoint1' />
   </Query>
</Velocity11>
```

Teachpoint Information query response XML block

VWorks software provides the following information for a TeachpointInformation query.

XML structure

```
<Velocity11>
   <Response>
      <Parameters>
         <Parameter />
      <Parameters>
   </Response>
</Velocity11>
```

<Parameter> element

The <Parameter> element has the following attribute pairs:

Attribute Name	Description
Name	The name of the axis.
Value	The coordinate for the axis.

Example

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='QueryResponse'
md5sum='f7038cb092493fad9312e4164873c8a2'
version='1.0' >
   <Response Category='TeachpointInformation'>
      <Parameters>
         <Parameter Name='X-axis' Value='50' />
         <Parameter Name='Y-axis' Value='100' />
      </Parameters>
   </Response>
</Velocity11>
```

Related topics

For information about...	See...
Getting the IControllerClient pointer	“SetController method” on page 73
Plug-in to plug-in communication	“ControllerQuery method” on page 52

Update method

Description

Use the Update method to send an update request to VWorks software.

Syntax

```
HRESULT Update([in] IControllerClient *Source,
[in] BSTR Update);
```

Parameters

The Update method has the following parameters:

- [in] IControllerClient *Source

  Pointer to the plug-in so that VWorks software can identify the source of the request.

- [in] BSTR Update

  An Update XML block used to specify the update to VWorks software.

Update XML block

The Update XML block contains information about the request to VWorks software.

XML structure

```
<Velocity11>
   <Update>
      ...
   </Update>
</Velocity11>
```

<Update> element

Specify the type of update request you are making. The attribute for this element is:

Attribute Name	Description
Category	The category describes the type of update. The valid values are: ❑ `Barcode` Update the barcode value for a specific side of a plate instance at a specific location. ❑ `InventoryPlateBarcodes` Provide the barcode information from a requested plate inventory using the IStorageDriver interface QueryStorageLocations method. ❑ `Menu` Add a custom menu to the context menu in the inventory editor's Inventory Management tab. ❑ `RackInfo` Provide information to VWorks software about a tube rack. ❑ `RunScript` Execute an arbitrary script, for example set a variable "x=1". ❑ `RunsetAdd` Schedule a protocol in the run-set manager. ❑ `StackHeight` Inform VWorks software of the initial height of a stack. ❑ `TermSuccess` Successfully terminate the currently running protocol. ❑ `Volume` Update the volume in a plate well.

<Parameters> element

The <Parameters> element groups one or more <Parameter> elements and has no attributes.

<Parameter> element

The <Parameter> element provides the information for the update.

Barcode update XML block

Use the Barcode update XML block to update the barcode value for a specific side of a plate instance at a specific location. If there is a bar code mismatch, barcode error handling is triggered.

When the barcode update is used, VWorks software implements it as if the barcode was read from a barcode scanner and therefore it triggers the BarcodeRead and BarcodeMisread VHooks event methods and a plate can be quarantined if appropriate.

XML structure

```
<Velocity11>
```

```
    <Update>
       <Parameters>
          <Parameter />
       </Parameters>
    </Update>
</Velocity11>
```

<Parameter> element

The <Parameter> element has the following attribute pairs:

Attribute Name	Description
❑ Name ❑ Value	❑ `Location` ❑ Location name
❑ Name ❑ Value	❑ `Side` ❑ Number to indicate the side of the plate as 0–3
❑ Name ❑ Value	❑ `Barcode` ❑ The barcode

Example

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Update'
md5sum='f7038cb092493fad9312e4164873c8a2'
version='1.0' >
   <Update Category='Barcode'>
      <Parameters>
         <Parameter Name='Location'
         Value='stage1' />
         <Parameter Name='Side' Value='0' />
         <Parameter Name='Barcode'
         Value='982AB02' />
      </Parameters>
   </Update>
</Velocity11>
```

Inventory Plate Barcodes update XML block

Use the InventoryPlateBarcodes update XML block to provide the bar code information from a plate inventory request using the IStorageDriver interface QueryStorageLocations method.

XML structure

```
<Velocity11>
   <Update>
      <Barcode />
      ...
   </Update>
</Velocity11>
```

<Barcode> element

Include a <Barcode> element for each slot in the plate range. The <Barcode> element has the following attributes:

Attribute Name	Description
BarcodeError	Indicate whether a bar code error occurred using one of the following values: 0 = No bar code read error 1 = An error occurred in reading the bar code
Cassette	The cassette number.
PlatePresent	Indicate whether a plate is present in the location using one of the following values: 1 = Plate is present 0 = No plate present
Slot	The slot number.
Value	The bar code

Example

If the device is unable to perform a plate inventory, provide the following InventoryPlateBarcodes update XML block:

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

The following is an example of an InventoryPlateBarcodes update XML block after performing a successful inventory on two plates:

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

```
   <Update Category='InventoryPlateBarcodes' >
      <Barcode BarcodeError='0' Cassette='1'
      PlatePresent='1' Slot='1'
      Value='ABCDEFGH' />
      <Barcode BarcodeError='0' Cassette='1'
      PlatePresent='1' Slot='2'
      Value='ABCDRRXT' />
   </Update>
</Velocity11>
```

Menu update XML block

Use the Menu update XML block to add a custom menu to the context menu in the inventory editor's Inventory Management tab.

XML Structure

```
<Velocity11>
   <Update>
      <Parameters>
         <Parameter />
      </Parameters>
   </Update>
</Velocity11>
```

<Parameter> element

The <Parameter> element has the following attribute pairs:

Attribute Name	Description
❑ Name ❑ Value	❑ `MenuName` ❑ `Inventory` or `Separator`
❑ Name ❑ Value	❑ `MenuItem` ❑ Menu item name to appear in the context menu

Example

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Update'
md5sum='f7038cb092493fad9312e4164873c8a2'
version='1.0' >
```

```
<Update Category='Menu'>
   <Parameter Name='MenuName'
   Value='Inventory' />
   <Parameter Name='MenuItem' Value='new
   menu item'/>
</Update>
</Velocity11>
```

RackInfo update XML block

Use the RackInfo update XML block to provide information to VWorks software about a tube rack.

XML structure

```
<Velocity11>
   <Update>
      <Parameters>
         <Parameter />
      </Parameters>
   </Update>
</Velocity11>
```

<Parameter> element

The <Parameter> element has the following attribute pairs:

Attribute Name	Description
❑ Name ❑ Value	❑ `RackInfoType` ❑ `CheckRack` is the only currently supported value for `RackInfoType`. This occurs when the plug-in checks the rack by reading the barcodes of the individual tubes in a rack and comparing them to a file that contains the expected tube barcodes. When checking the rack position is just as important as the barcode when considering a match.
❑ Name ❑ Value	❑ `MismatchesVariable` ❑ The name of a JavaScript variable that is set to the number of mismatches encountered in the rack check. This attribute is optional.
❑ Name ❑ Value	❑ `Mismatches` ❑ The number of mismatches that occurred in the rack check.

Attribute Name	Description
❑ Name ❑ Value	❑ `CheckRackResult` ❑ Indicates whether any mismatches were encountered in the rack check. 1 = No mismatches found 0 = One or more mismatches found
❑ Name ❑ Value	❑ `Location` ❑ The value for `Location` is the name of the location within the device.

Example

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Update'
md5sum='f7038cb092493fad9312e4164873c8a2'
version='1.0' >
   <Update Category='RackInfo'>
      <Parameters>
         <Parameter Name='RackInfoType'
         Value='CheckRack' />
         <Parameter Name='MismatchesVariable'
         Value='mismatch-variable' />
         <Parameter Name='Mismatches' Value='0'
         />
         <Parameter Name='CheckRackResult'
         Value='1' />
         <Parameter Name='Location'
         Value='Location1' />
      </Parameters>
   </Update>
</Velocity11>
```

RunScript update XML block

Use the RunScript update XML block to execute an arbitrary script, for example set a variable “x=1”.

XML structure

```
<Velocity11>
   <Update>
      <Parameter />
   </Update>
</Velocity11>
```

<Parameter> element

The <Parameter> element has the following attribute pair:

Attribute Name	Description
Name	The name of the script.
Value	The value that is passed directly to the JavaScript engine.

Example

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Update'
md5sum='f7038cb092493fad9312e4164873c8a2'
version='1.0' >
   <Update Category='RunScript'>
      <Parameter Name='sample.js' Value='1' />
   </Update>
</Velocity11>
```

RunsetAdd update XML block

Use the RunsetAdd update XML block to schedule a protocol in the runset manager.

XML structure

```
<Velocity11>
   <Update>
      <Parameters>
         <Parameter />
      </Parameters>
   </Update>
</Velocity11>
```

<Parameter> element

The <Parameter> element has the following attribute pairs:

Attribute Name	Description
❑ Name ❑ Value	❑ `ProtocolName` ❑ The name and full path of the protocol file.
❑ Name ❑ Value	❑ `StartTime_year` ❑ Start year (4 digits).
❑ Name ❑ Value	❑ `StartTime_month` ❑ Start month 1–12.

Attribute Name	Description
❑ Name ❑ Value	❑ `StartTime_day` ❑ Start day1–31.
❑ Name ❑ Value	❑ `StartTime_hour` ❑ Start hour 1–24.
❑ Name ❑ Value	❑ `StartTime_minutes` ❑ Start minute 0–59.
❑ Name ❑ Value	❑ `StartTime_seconds` ❑ Start seconds 0–59.
❑ Name ❑ Value	❑ `NumberOfRuns` ❑ Number of runs to perform.
❑ Name ❑ Value	❑ `PluginName_filename` ❑ Filename of the plug-in. This name is displayed in the Plugin field of the run-set manager so you can see which plug-in has scheduled the protocol.
❑ Name ❑ Value	❑ `ProtocolNotes` ❑ Protocol notes are associated with the run and are included in the run-set manager's Protocol Notes.

Example

The following example shows a request to add a protocol called sample-protocol.pro to the run-set manager that will run two times on February 5, 2007 at 10pm.

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

```
<Update Category= 'RunsetAdd'>
   <Parameters>
      <Parameter Name='ProtocolName' Value=
      'C:\Protocols\sample-protocol.pro' />
      <Parameter Name='StartTime_year'
      Value= '2007' />
      <Parameter Name='StartTime_month'
      Value= '2' />
      <Parameter Name='StartTime_day' Value=
      '5' />
      <Parameter Name='StartTime_hour'
      Value= '10' />
      <Parameter Name='StartTime_minutes'
      Value= '0' />
      <Parameter Name='StartTime_seconds'
      Value= '0' />
      <Parameter Name='NumberOfRuns' Value=
      '2' />
      <Parameter Name='PluginName_filename'
      Value='my_PluginName_filename' />
      <Parameter Name='ProtocolNotes' Value=
      'Notes about this protocol' />
     </Parameters>
   </Update>
</Velocity11>
```

StackHeight update XML block

Use the StackHeight update XML block to provide the initial height of a plate stack.

XML structure

```
<Velocity11>
   <Update>
      <Parameters>
         <Parameter />
      </Parameters>
   </Update>
</Velocity11>
```

<Parameter> element

The <Parameter> element has the following attribute pair:

Attribute Name	Description
Name	`InitialStackHeight`

Attribute Name	Description
Value	The stack height in millimeters.

Example

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Update'
md5sum='0b8eb2ad5df221c13e336af98ffec528'
version='1.0' >
   <Update Category= 'StackHeight'>
      <Parameters>
         <Parameter Name='InitialStackHeight'
         Value='80' />
      </Parameters>
   </Update>
</Velocity11>
```

TermSucess update XML block

Use the TermSuccess update XML block to successfully terminate a currently running protocol by setting the number of plates remaining to zero. There are no elements except the <Update> element for this XML block.

Example

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='Update'
md5sum='f7038cb092493fad9312e4164873c8a2'
version='1.0'>
   <Update Category='TermSuccess'>
   </Update>
</Velocity11>
```

Volume update XML block

Use the Volume update XML block to update the volume in plate wells.

XML structure

```
<Velocity11>
   <Update>
      <Parameters>
         <Parameter />
      </Parameters>
   </Update>
</Velocity11>
```

<Parameter> element

The <Parameter> element has the following attribute pair:

Attribute Name	Description
Name	`VolumeChange`
Value	An escaped VolumeUpdates XML block.

VolumeUpdates XML block

The Value attribute of the Volume update XML <Parameter> contains an escaped VolumeUpdates XML block.

XML structure

```
<Velocity11>
   <VolumeUpdates>
      <VolumeUpdates>
         <VolumeUpdate />
         ...
      </VolumeUpdates>
   </VolumeUpdates>
</Velocity11>
```

<VolumeUpdates> element

The first occurrence of this the <VolumeUpdates> element has the following attributes:

Attribute Name	Description
Location	The location name.
ResetAbsolute	Indicates whether the volume change is an offset from the current volume or a new volume. 0 = Set the volume to the current volume plus the volume change specified in the <VolumeUpdate> element. 1 = Set to the volume to the volume specified in the <VolumeUpdate> element.

<VolumeUpdate> element

You can have one or more <VolumeUpdate> elements. Each one specifies a change in volume for a well on the plate. The <VolumeUpdate> element has the following attributes:

Attribute Name	Description
Col	The column coordinate of the well.
Row	The row coordinate of the well.
VolumeChange	The volume change in microliters.

VolumeUpdates XML block example

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData'
md5sum='15f7ddd15cbcebf43d47755f08462d03'versio
n='1.0' >
   <VolumeUpdates
   Location='MyDeviceExposedLocationName'
   ResetAbsolute='1' >
      <VolumeUpdates >
         <VolumeUpdate Col='3' Row='2'
         VolumeChange='0.05' />
         ...
         <VolumeUpdate Col='3' Row='3'
         VolumeChange='0.06' />
      </VolumeUpdates>
   </VolumeUpdates>
</Velocity11>
```

Volume update XML block example

The complete example for the Volume XML block for the update method follows:

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData'
md5sum='15f7ddd15cbcebf43d47755f08462d03'versio
n='1.0' >
   <Update Category= "Volume">
      <Parameters>
         <Parameter Name="VolumeChange"
         Value='&lt;?xml
         version=&apos;1.0&apos;
         encoding=&apos;ASCII&apos; ?&gt;
         &lt;Velocity11
         file=&apos;MetaData&apos;
         md5sum=&apos;e2c82c0cd85b14a80364f073a
         aae45b6&apos; version=&apos;1.0&apos;
         &gt; &lt;VolumeUpdates
         Location=&apos;stage1&apos;
         ResetAbsolute=&apos;l&apos; &gt;
         &lt;VolumeUpdates &gt;
         &lt;VolumeUpdate Col=&apos;3&apos;
         Row=&apos;2&apos;
         VolumeChange=&apos0.05&apos; /&gt;
         &lt;/VolumeUpdates&gt; &lt;/
```

```
            VolumeUpdates&gt; &gt;/Velocity11&lt;'
            />
        </Parameters>
    </Update>
</Velocity11>
```

Related topics

For information about...	See...
Plate inventory request from VWorks software	“QueryStorageLocations method” on page 147
Getting the IControllerClient pointer	“SetController method” on page 73

IStackerDriver interface reference

6

This chapter describes the IStackerDriver interface and its methods. These methods are used to add and remove plates from the system.

Note: You must also implement the IWorksDriver interface methods if you implement the IStackerDriver methods.

This chapter covers the following topics:

- ❑ "Method list" on page 116
- ❑ "IsStackEmpty method" on page 116
- ❑ "IsStackFull method" on page 117
- ❑ "LoadStack method" on page 118
- ❑ "SinkPlate method" on page 119
- ❑ "SourcePlate method" on page 120
- ❑ "UnloadStack method" on page 121

Method list

About this topic This topic lists the IStackerDriver interface methods.

IStackerDriver methods The following table provides an alphabetical listing of the IStackerDriver interface methods.

Method	Description	See...
IsStackEmpty	Indicate if the stack at the specified location is empty	"IsStackEmpty method" on page 116
IsStackFull	Indicate if the stack at the specified location is full	"IsStackFull method" on page 117
LoadStack	Prepare the stack for robot access	"LoadStack method" on page 118
SinkPlate	Accept a plate from the robot and place it on a stack	"SinkPlate method" on page 119
SourcePlate	Present a plate to the robot	"SourcePlate method" on page 120
UnloadStack	Release a stack of plates so the operator can remove the stack	"UnloadStack method" on page 121

IsStackEmpty method

Description The IsStackEmpty method is called to determine if the stack at the given location is empty or not.

Syntax

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

Parameters The IsStackEmpty method has the following parameters:

- [in] BSTR Location
- [out,retval] SHORT* isEmpty

  The valid return values are:

Value	Description
0	Indicates the stack is not empty
1	Indicates the stack is empty

Related topics

For information about...	See...
Is stack full	"IsStackFull method" on page 117

IsStackFull method

Description0

The IsStackFull method is called to determine if the stack at the given location is full or not.

Syntax

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

Parameters

The IsStackFull method has the following parameters:

- [in] BSTR Location

  The location name.

- [out,retval] SHORT* isFull

  The valid return values are:

Value	Description
0	Indicates the stack is not full
1	Indicates the stack is full

Related topics

For information about...	See...
Is stack empty	"IsStackEmpty method" on page 116

LoadStack method

Description

The LoadStack method is called when VWorks software makes a request for the driver plug-in to prepare the stack for robot access. This occurs once at the start of a protocol run.

Syntax

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

Parameters

The LoadStack method has the following parameters:

- [in] BSTR Labware

  The labware name.

- [in] PlateFlagsType PlateFlags

  Indicates whether plate type is normal, lidded, or sealed.

- [in] BSTR Location

  The location name.

- [out,retval] ReturnCode *retVal

  Indicate whether or not the operation completed successfully.

Related topics

For information about...	See...
PlateFlagsType values	“PlateFlagsType values” on page 47
Return codes	“Return Code values” on page 47
Unload stack	“UnloadStack method” on page 121

SinkPlate method

Description

The SinkPlate method is called as a request from VWorks software to accept a plate from the robot, and place it on a stack. If the SourcePlate method is called next, VWorks software expects to retrieve the same plate.

Syntax

```
HRESULT SinkPlate([in] BSTR Labware, [in]
PlateFlagsType PlateFlags, [in] BSTR
SinkToLocation, [out,retval] ReturnCode*
retVal);
```

Parameters

The SinkPlate method has the following parameters:

- [in] BSTR Labware

  The labware name.

- [in] PlateFlagsType PlateFlags

  Indicates whether plate type is normal, lidded, or sealed.

- [in] BSTR SinkToLocation

  The destination location name.

- [out,retval] ReturnCode *retVal

  Indicate whether or not the operation completed successfully.

Related topics

For information about...	See...
PlateFlagsType values	“PlateFlagsType values” on page 47
Return codes	“Return Code values” on page 47
A request to present a plate to the robot	“SourcePlate method” on page 120

SourcePlate method

Description

The SourcePlate method is called when VWorks software requests the driver plug-in to present a plate to the robot.

Syntax

```
HRESULT SourcePlate([in] BSTR Labware, [in]
PlateFlagsType PlateFlags, [in] BSTR
SourceFromLocation, [out,retval] ReturnCode*
retVal);
```

Parameters

The SourcePlate method has the following parameters:

- [in] BSTR Labware

  The labware name.

- [in] PlateFlagsType PlateFlags

  Indicates whether plate type is normal, lidded, or sealed.

- [in] BSTR SourceFromLocation

  The source location name.

- [out,retval] ReturnCode *retVal

  Indicates whether or not the request was completed successfully.

Related topics

For information about...	See...
PlateFlagsType values	“PlateFlagsType values” on page 47
Return codes	“Return Code values” on page 47
A request to accept a plate from the robot	“SinkPlate method” on page 119

UnloadStack method

Description

The UnloadStack method is called when VWorks software requests the driver plug-in to release a stack of plates so that the operator can remove the stack.

Syntax

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

Parameters

The UnloadStack method has the following parameters:

- [in] BSTR Labware

  The labware name.

- [in] PlateFlagsType PlateFlags

  Indicates whether plate type is normal, lidded, or sealed.

- [in] BSTR Location

  The location name.

- [out,retval] ReturnCode *retVal

  Indicates whether or not the request was completed.

Related topics

For information about...	See...
PlateFlagsType values	“PlateFlagsType values” on page 47
Return codes	“Return Code values” on page 47

IPipetteDriver interface reference

7

This chapter describes the IPipetteDriver interface and its methods.

Note: You must also implement the IWorksDriver interface methods if you implement the IPipetteDriver methods.

This chapter covers the following topics:

- ❑ "Method list" on page 124
- ❑ "GetHeadChange method" on page 124
- ❑ "GetProfileHeadType method" on page 126
- ❑ "GetTaskType method" on page 127
- ❑ "GetTipChange method" on page 129
- ❑ "GetVolumeChange method" on page 131
- ❑ "ImplementsPipetteWizards method" on page 133

Method list

About this topic

This topic lists the IPipetteDriver interface methods.

IPipetteDriver methods

The following table provides an alphabetical listing of the IPipetteDriver interface methods.

Method	Description	See...
GetHeadChange	Upon a head change command, provide information about the new head	"GetHeadChange method" on page 124
GetProfileHeadType	For a given profile, provide the head type	"GetProfileHeadType method" on page 126
GetTaskType	Provide type of task	"GetTaskType method" on page 127
GetTipChange	For a tip changing command, provide information about the tip change	"GetTipChange method" on page 129
GetVolumeChange	Provide volume change in tips for a given fluid handling command	"GetVolumeChange method" on page 131
ImplementsPipetteWizards	Respond to query of whether the device implements serial dilution and other wizards	"ImplementsPipetteWizards method" on page 133

GetHeadChange method

Description

The GetHeadChange method is called when VWorks software requests information about a new head upon a head change command.

The driver plug-in provides VWorks software with details about the attached head for a particular task. Information you need to provide includes:

- ❑ Number of channels in the head
- ❑ Whether or not the tips are disposable
- ❑ Minimum and maximum volume ranges
- ❑ Name of the head profile

Syntax

```
HRESULT GetHeadChange([in] BSTR CommandXML,
[out,retval] BSTR* HeadChangeXML);
```

Parameters

The GetHeadChange method has the following parameters:

- [in] BSTR CommandXML

  The CommandXML parameter describes the particular task for which VWorks software is requesting information about the attached head.

  In a previous call to the GetMetaData method (with the MetaDataType METADATA_ALL or METADATA_COMMAND), VWorks software received the command metadata that describes the device's commands and the associated parameters for each command.

- [out,retval] BSTR *HeadChangeXML

  A PipetteHead XML block that describes the information about the attached head.

PipetteHead XML block

Provide information about the attached head in the PipetteHead XML block.

XML structure

```
<Velocity11>
   <PipetteHead />
</Velocity11>
```

<PipetteHead> element

The <PipetteHead> element has the following attributes:

Attribute Name	Description
Channels	The number of channels in the head.
Disposable	Whether or not the tips are disposable. The valid values are: 0 = Tips are fixed (not disposable) 1 = Tips are disposable
MaxRange	The maximum volume allowed.
MinRange	The minimum volume allowed.
Name	The name of the head profile.

Example

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData'
md5sum='07885f2690dba0d44ca74c70133981d7'
version='1.0' >
   <PipetteHead Channels='1' Disposable='0'
   MaxRange='50' MinRange='0' Name='384F, 50
   µL' />
</Velocity11>
```

Related topics

For information about...	See...
Example of a command XML block	"Command XML block" on page 42
<Velocity11> element attributes	"<Velocity11> element" on page 25
GetMetaData method	"GetMetaData method" on page 56

GetProfileHeadType method

Description

The GetProfileHeadType method is called when VWorks software requests information about a head type selected in the profile that is currently initialized.

Note: This is not necessarily the same as the head information returned from a call to the GetHeadChange method. However, the XML data the plug-in provides as output is the same as the HeadChange XML block in the GetHeadChange method.

Syntax

```
HRESULT ProfileHeadType([out,retval] BSTR*
HeadTypeXML);
```

Parameters

The GetProfileHeadType method has the following parameter:

- [out,retval] BSTR *HeadTypeXML

  A PipetteHead XML block that describes the information about the head type selected in a profile.

Related topics

For information about...	See...
Example of a PipetteHead XML block	"GetHeadChange method" on page 124

GetTaskType method

Description

The GetTaskType method is called when VWorks software requests information about a particular pipette task. This is called when the Command metadata is first received from the plug-in.

Syntax

```
HRESULT GetTaskType([in] BSTR commandXML,
[out,retval] BSTR* TaskTypeXML);
```

Parameters

The GetTaskType method has the following parameters:

- [in] BSTR CommandXML

  The CommandXML parameter describes the particular pipette task for which VWorks software is requesting the task type.

  In a previous call to the GetMetaData method (with the MetaDataType METADATA_ALL or METADATA_COMMAND), VWorks software received the command metadata that describes the device's commands and the associated parameters for each command.

- [out,retval] BSTR *TaskTypeXML

  A PipetteTaskQuery XML block that describes the type of pipette task such as aspirate, dispense, mix, wash, tips on, tips off, and so on.

PipetteTaskQuery XML block

The PipetteTaskQuery XML block describes the type of pipette task.

XML structure

```
<Velocity11>
   <PipetteTaskQuery />
</Velocity11>
```

<PipetteTaskQuery> element

The <PipetteTaskQuery> element has the following attributes:

Attribute Name	Description
TaskType	A bitmask that defines the type of pipette task. The valid values are: 0x0001 = Primary aspirate 0x0002 = Primary dispense 0x0004 = Primary mix 0x0008 = Primary wash 0x0010 = Primary tips on 0x0020 = Primary tips off 0x0040 = Secondary can change volume level 0x0080 = Secondary can change tip state 0x0100 = Secondary washes – requires a wash station 0x0200 = Can change how head behaves 0x0800 = Pumps reagent

Example

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData'
md5sum='2df99a030b9dc4d37c535d0846f2beeb'
version='1.0' >
   <PipetteTaskQuery TaskType='1' />
</Velocity11>
```

Related topics

For information about...	See...
Example of a command XML block	“Command XML block” on page 42
<Velocity11> element attributes	“<Velocity11> element” on page 25
GetMetaData method	“GetMetaData method” on page 56

GetTipChange method

Description

The GetTipChange method is called when VWorks software requests information about a tip change upon a tip change command. The driver plug-in also provides VWorks software with the capacity of the tips pressed onto the head.

Syntax

```
HRESULT GetTipChange([in] BSTR CommandXML,
[out,retval] BSTR* TipChangeXML);
```

Parameters

The GetTipChange method has the following parameters:

- [in] BSTR CommandXML

  The CommandXML parameter describes the particular pipette task for which VWorks software is requesting tip capacity.

  In a previous call to the GetMetaData method (with the MetaDataType METADATA_ALL or METADATA_COMMAND), VWorks software received the command metadata that describes the device's commands and the associated parameters for each command.

- [out,retval] BSTR *TipChangeXML

  A TipChange XML block that provides whether the tip task was for tips on or tips off and the tip capacity.

TipChange XML block

The TipChange XML block whether the tip task was for tips on or tips off and the tip capacity.

XML structure

```
<Velocity11>
   <TipChanges>
   <TipChanges >
      <TipChange />
   </TipChanges>
   </TipChanges>
</Velocity11>
```

<TipChanges> element

The <TipChanges> element is used to group one or more <TipChange> elements.

<TipChange> element

The <TipChange> element has the following attributes:

Attribute Name	Description
Operation	The tip task. The valid values are: 0 = all tips on 1 = all tips off
TipCapacity	The capacity of the tip in microliters.

Example

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData'
md5sum='b2cfee36b4d8f1e74b7166b2dcc66141'
version='1.0' >
   <TipChanges>
   <TipChanges>
      <TipChange Operation='0' TipCapacity='0'
      />
   </TipChanges>
   </TipChanges>
</Velocity11>
```

Related topics

For information about...	See...
Example of a command XML block	“Command XML block” on page 42
<Velocity11> element attributes	“<Velocity11> element” on page 25
GetMetaData method	“GetMetaData method” on page 56

GetVolumeChange method

Description

The GetVolumeChange method is called when VWorks software requests information about a volume change in tips for a given fluid handling command.

The driver plug-in provides to VWorks software the amount of volume that is aspirated or dispensed for the specified task.

Syntax

```
HRESULT GetVolumeChange([in] BSTR CommandXML,
[out,retval] BSTR* VolumeChangeXML);
```

Parameters

The GetVolumeChange method has the following parameters:

- [in] BSTR CommandXML

  The CommandXML parameter describes the particular pipette task for which VWorks software is requesting information about the volume change.

  In a previous call to the GetMetaData method (with the MetaDataType METADATA_ALL or METADATA_COMMAND), VWorks software received the command metadata that describes the device's commands and the associated parameters for each command.

- [out,retval] BSTR *VolumeChangeXML

  A VolumeChange XML block that provides the task type and the change in volume.

VolumeChange XML block

The VolumeChange XML block describes the task type and the change in volume for the given command.

XML structure

```
<Velocity11>
   <VolumeChanges >
   <VolumeChanges >
      <VolumeChange />
   </VolumeChanges>
   </VolumeChanges>
</Velocity11>
```

<VolumeChanges> element

The <VolumeChanges> element is used to group one or more <VolumeChange> elements.

<VolumeChange> element

The <VolumeChange> element has the following attributes:

Attribute Name	Description
Operation	The task operation. The valid values are: 0 = Aspirate liquid 1 = Aspirate air 2 = Dispense 3 = Empty tips 4 = Survey
VolumeChange	The absolute value of the volume change in microliters.
WellSelection	An escaped Wells Selection XML block that describes the number of wells in the plate and how the wells are to be accessed by the pipettor. See “Wells Selection XML block” on page 33 for a complete description of this XML block.

Example

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData'
md5sum='6797fba63962ed3800be4a138361bf50'
version='1.0' >
   <VolumeChanges>
   <VolumeChanges>
      <VolumeChange Operation='0'
      VolumeChange='10' WellSelection='&lt;?xml
      version=&apos;1.0&apos;
      encoding=&apos;ASCII&apos; ?&gt;
      &lt;Velocity11 file=&apos;MetaData&apos;
      md5sum=&apos;e2c82c0cd85b14a80364f073aaae
      45b6&apos; version=&apos;1.0&apos; &gt;
      &lt;Channels Value=&apos;1&apos; /&gt;
      &lt;Quadrant Column=&apos;0&apos;
      Row=&apos;0&apos; /&gt; &lt;/
      Velocity11&gt;' />
   </VolumeChanges>
   </VolumeChanges>
</Velocity11>
```

Related topics

For information about...	See...
Example of a command XML block	“Command XML block” on page 42
<Velocity11> element attributes	“<Velocity11> element” on page 25
GetMetaData method	“GetMetaData method” on page 56

For information about...	See...
Wells Selection XML block	"Wells Selection XML block" on page 33

ImplementsPipetteWizards method

Description

The ImplementsPipetteWizards method is called when VWorks software queries device to determine if it implements fill plate or serial dilution wizards.

The driver plug-in informs VWorks software whether or not it supports the Serial Dilution and Fill Plate tasks.

Syntax

```
HRESULT ImplementsPipetteWizard([out,retval]
BSTR* WizardXML);
```

Parameters

The ImplementPipetteWizards method has the following parameter:

- [out,retval] BSTR *WizardXML

  A PipetteWizardQuery XML block that provides information about the plug-in's wizard support.

PipetteWizardQuery XML block

The PipetteWizardQuery XML block provides which pipette wizards the plug-in supports and parameter labels so that VWorks software can match fluid task parameters (such as volume and well selection) with the fields in the wizard dialog box.

XML structure

```
<Velocity11>
   <PipetteWizardQuery />
</Velocity11>
```

<PipetteWizardQuery> element

The <PipetteWizardQuery> element has the following attributes:

Attribute Name	Description
ImplementsFillPlate	Whether the plug-in supports fill plate wizard. The valid values are: 0 = Fill Plate not supported 1 = Fill Plate supported
ImplementsSerialDilution	Whether the plug-in supports a serial dilution wizard. The valid values are: 0 = Serial dilution not supported 1 = Serial dilution supported

Attribute Name	Description
PlateLabel	The name that describes the current location of the plate.
VolumeLabel	The name of the parameter for the volume field in the device's aspirate and dispense commands metadata.
WellSelectionLabel	The name of the parameter for the well selection field in the aspirate, dispense, and mix commands metadata.

Example

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData'
md5sum='7a13f0c6aeb406f23556bc208a77ecf6'
version='1.0' >
   <PipetteWizardQuery ImplementsFillPlate='0'
   ImplementsSerialDilution='1'
   PlateLabel='Location' VolumeLabel='Volume'
   WellSelectionLabel='Well selection' />
</Velocity11>
```

Related topics

For information about...	See...
<Velocity11> element attributes	"<Velocity11> element" on page 25

IRobotDriver interface reference

8

This chapter describes the IRobotDriver interface and its methods.

Note: You must also implement the IWorksDriver interface methods if you implement the IRobotDriver methods.

This chapter covers the following topics:

❑ "Method list" on page 136
❑ "GetTeachPoints method" on page 137
❑ "Move method" on page 137

Method list

About this topic This topic lists the IRobotDriver interface methods.

IRobotDriver methods The following table provides an alphabetical listing of the IRobotDriver interface methods.

Method	Description	See...
CheckBarCode	This is a placeholder for possible implementation in a future release.	
CheckPlatePresent	This is a placeholder for possible implementation in a future release.	
Delid	This is a placeholder for possible implementation in a future release.	
GetSimulationTimes	This is a placeholder for possible implementation in a future release.	
GetSpeed	This is a placeholder for possible implementation in a future release.	
GetTeachPoints	Provide a list of teachpoint names	“GetTeachPoints method” on page 137
Move	Request for robot to pick and place	“Move method” on page 137
PushdownAtLastPlaceLocation	This is a placeholder for possible implementation in a future release.	
Relid	This is a placeholder for possible implementation in a future release.	
SetSpeed	This is a placeholder for possible implementation in a future release.	
Stir	This is a placeholder for possible implementation in a future release.	

GetTeachPoints method

Description

The GetTeachPoints method is called when VWorks software requests a list of teachpoint names.

Syntax

```
HRESULT GetTeachPoints([out,retval] VARIANT*
Teachpoints);
```

Parameters

The GetTeachPoints method has the following parameter:

- [out,retval] VARIANT*Teachpoints

  A list of teachpoint names.

Move method

Description

The Move method is called automatically during a protocol run as necessary.

Syntax

```
HRESULT Move([in] BSTR PickupLocation, [in]
FLOAT PickupHeightOffset, [in] FLOAT
PickupDepartureHeight, [in] BSTR
DropoffLocation, [in] FLOAT
DropoffOffsetDistance, [in] FLOAT
DropoffDepartureHeight, [in] FLOAT
PushdownOverextend, [in] BSTR labware,
[out,retval] ReturnCode* RetVal);
```

Parameters

The Move method has the following parameters:

- [in] BSTR PickupLocation
- [in] FLOAT PickupHeightOffset
- [in] FLOAT PickupDepartureHeight
- [in] BSTR DropoffLocation
- [in] FLOAT DropoffOffsetDistance
- [in] FLOAT DropoffDepartureHeight
- [in] FLOAT PushdownOverextend
- [in] BSTR labware

  The name of the labware.

- [out,retval] ReturnCode* RetVal

  Indicates whether or not the request was completed successfully.

Related topics

For information about...	See...
Return codes	“Return Code values” on page 47
Testing the Move method	“Testing IRobotDriver methods” on page 196

IStorageDriver interface reference

9

This chapter describes the StorageDriver interface and its methods. These methods are used for a device that can provide random access to a set of plates.

Note: You must also implement the IWorksDriver interface methods if you implement the IStorageDriver methods.

This chapter covers the following topics:

- ❑ "Method list" on page 140
- ❑ "GetStorageLocations method" on page 140
- ❑ "LoadPlate method" on page 143
- ❑ "LookupLocations method" on page 145
- ❑ "QueryStorageLocations method" on page 147
- ❑ "UnloadPlate method" on page 148

Method list

About this topic

This topic lists the IStorageDriver interface methods.

IStorageDriver methods

The following table provides an alphabetical listing of the IStorageDriver interface methods.

Method	Description	See...
GetStorageLocations	Provide the capacity of the storage device.	"GetStorageLocations method" on page 140
LoadPlate	Accept a plate from the robot and place it to a specific location	"LoadPlate method" on page 143
LookupLocations	Provide external location and offset to given plate coordinates	"LookupLocations method" on page 145
QueryStorageLocations	Provide inventory on a range of plates	"QueryStorageLocations method" on page 147
UnloadPlate	Present plate to robot at a specific location	"UnloadPlate method" on page 148

GetStorageLocations method

Description

The GetStorageLocations method is called when the device is created in VWorks software.

Syntax

```
HRESULT GetStorageLocations([out] BSTR
StorageLocationsXML, [out,retval] ReturnCode*
retVal);
```

Parameters

The GetStorageLocations method has the following parameters:

- [out] BSTR StorageLocationsXML

  A StorageDimensions XML block that provides names for the coordinates and the capacity of the storage device.

- [out,retval] ReturnCode *retVal

  Indicates whether or not the request was completed.

StorageDimensions XML block

Provide information about the storage device capacity in the StorageDimensions XML block.

XML structure

```
<Velocity11>
   <StorageDimensions>
      <Dimensions>
         <StorageDimension />
         ...
      </Dimensions>
   </StorageDimensions>
</Velocity11>
```

<StorageDimensions> element

The <StorageDimensions> element has the following attributes:

Attribute Name	Description
DirectStorageAccess	Specifies whether the robot reaches into the device's storage area (for example a multiple-sided storage carousel) or if the device has an external staging area (for example, STX incubator or Cytomat storage devices). 1 = Robot accesses plates directly in the device 0 = Robot accesses plates on an external staging area
Name0	Name used to describe a column (cassette) of plates.
Name1	Name used to describe what holds each plate.

<Dimensions> element

The <Dimensions> element has no attributes and groups one or more <StorageDimension> elements.

<StorageDimension> element

The <StorageDimension> element describes the capacity of the storage device. You will have a <StorageDimension> element for each column of plates. The <StorageDimension> element has the following attribute:

Attribute Name	Description
Size	Specifies the number of plates per column.

Example

The following example is for a Platehub Carousel which is a twelve-sided plate storage device that has 16 slots per cassette.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData'
md5sum='07885f2690dba0d44ca74c70133981d7'
version='1.0' >
   <StorageDimensions DirectStorageAccess='1'
   Name0='Cassette' Name1='Slot' >
      <Dimensions>
         <StorageDimension Size='16' />
         <StorageDimension Size='16' />
         <StorageDimension Size='16' />
         <StorageDimension Size='16' />
         <StorageDimension Size='16' />
         <StorageDimension Size='16' />
         <StorageDimension Size='16' />
         <StorageDimension Size='16' />
         <StorageDimension Size='16' />
         <StorageDimension Size='16' />
         <StorageDimension Size='16' />
         <StorageDimension Size='16' />
      </Dimensions>
   </StorageDimensions>
</Velocity11>
```

Related topics

For information about...	See...
Return codes	“Return Code values” on page 47

LoadPlate method

Description

The LoadPlate method is called each time a load task occurs in a protocol run.

Syntax

```
HRESULT LoadPlate([in] BSTR Labware, [in]
PlateFlagsType PlateFlags, [in] BSTR
LoadPlateLocationXML, [out,retval] ReturnCode*
retVal);
```

Parameters

The LoadPlate method has the following parameters:

- [in] BSTR Labware

  The labware name.

- [in] PlateFlagsType PlateFlags

  Indicates whether plate type is normal, lidded, or sealed.

- [in] BSTR LoadPlateLocationXML

  A StorageLocation XML block that describes the specific location where VWorks software will load the plate.

- [out,retval] ReturnCode *retVal

  Indicates whether or not the request was completed.

StorageLocation XML block

The coordinates for the plate load is described in the StorageLocation XML block.

XML structure

```
<Velocity11>
   <StorageLocation>
      <Coordinates>
         <StorageLocationCoordinate />
         ...
      </Coordinates>
      <Location />
   </StorageLocation>
</Velocity11>
```

<StorageLocation> element

The <StorageLocation> element has no attributes. This element contains the child elements <Coordinates> and <Location>.

<Coordinates> element

The <Coordinates> element has no attributes and groups two <StorageLocationCoordinate> elements.

<StorageLocationCoordinate> element

Two <StorageLocationCoordinate> elements specify the coordinate for the plate load and has the following attributes:

Attribute Name	Description
Name	Name given by the plug-in in the GetStorageLocations method for the column of plates or the name for what holds each plate.
Value	A number to indicate which column of plates or which slot.

Example

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='MetaData'
md5sum='07885f2690dba0d44ca74c70133981d7'
version='1.0' >
   <StorageLocation>
      <Coordinates>
         <StorageLocationCoordinate
         Name='Cassette' Value='1' />
         <StorageLocationCoordinate Name='Slot'
         Value='1' />
      </Coordinates>
      <Location Group='0', MaxStackHeight='500'
      Offset='0' Type='1' />
   </StorageLocation>
</Velocity11>
```

Related topics

For information about...	See...
PlateFlagsType values	“PlateFlagsType values” on page 47
Return codes	“Return Code values” on page 47
<Location> element attributes	“<Location> element” on page 29

LookupLocations method

Description

The LookupLocations method is called before all calls to the LoadPlate and UnLoadPlate methods.

Syntax

```
HRESULT LookupLocations([in] VARIANT_BOOL*
Load, [in] BSTR Labware,[in] PlateFlagsType
PlateFlags, [in] BSTR
LoadPlateLocationXML,[out]
ExternalLocationsXML,[out,retval] ReturnCode*
retVal);
```

Parameters

The LookupLocations method has the following parameters:

- [in] VARIANT_BOOL* Load

  Indicates if the plate is about to be loaded or unloaded.

  1 = Plate is about to be loaded

  0 = Plate is about to be unloaded

- [in] BSTR Labware

  Name of the labware.

- [in] PlateFlagsType PlateFlags

  Indicates whether plate type is normal, lidded, or sealed.

- [in] BSTR LoadPlateLocationXML

  A StorageLocation XML block that describes the specific coordinates for the location VWorks software wants to access.

- [out] BSTR ExternalLocationsXML

  A LocationVector XML block that describes the specific location where VWorks software needs to move the plate to access the given coordinates.

- [out,retval] ReturnCode* retVal

  Indicate whether or not the operation completed successfully.

LocationVector XML block

You need to provide a LocationVector XML block to give coordinates to VWorks software so it can access the given location.

XML structure

```
<Velocity11>
   <LocationVector>
      <Locations>
         <Location />
         ...
      </Locations>
   </LocationVector>
</Velocity11>
```

<LocationVector> element

The <LocationVector> element has no attributes. This element contains the child elements <Locations> and <Location>.

<Locations> element

The <Locations> element has no attributes and contains one <Location> elements.

<Location> element

The <Location> element has the following attributes:

Attribute Name	Description
Name	The external location name.
Offset	The vertical offset. This provides the storage coordinate (cassette, slot) translated into a storage access point (location plus offset).
MaxStackHeight	Maximum height of the stack (used only by stacker devices that are returning <Location> elements to define the stack locations)

Related topics

For information about...	See...
PlateFlagsType values	“PlateFlagsType values” on page 47
StorageLocation XML block	“StorageLocation XML block” on page 143
Return codes	“Return Code values” on page 47
LoadPlate method	“LoadPlate method” on page 143
UnLoadPlate method	“UnloadPlate method” on page 148

QueryStorageLocations method

Description

The QueryStorageLocations method is called when the user requests an inventory of the plates in a storage device. This is performed with plate groups in the inventory editor.

After this method is called, VWorks software expects the results to be returned with an IWorksController interface Update method call from the plug-in.

Syntax

```
HRESULT QueryStorageLocations([in] BSTR
QueryXML, [out,retval] ReturnCode* retVal);
```

Parameters

The QueryStorageLocations method has the following parameters:

- [in] BSTR QueryXML

  A QueryLocation XML block that provides the coordinates of range of plates to be inventoried.

- [out,retval] ReturnCode* retVal

  Indicates whether or not the operation completed successfully.

QueryLocation XML block

VWorks software provides the location of the plates requiring inventory in the QueryLocation XML block.

XML structure

Note: This XML block has a different structure than used in most IWorksDriver methods. The attributes are all included in the <Velocity11> element for this method.

<Velocity11> element

The <Velocity11> element for the QueryLocation XML block has the following attributes:

Attribute Name	Description
file	`PlateStorageInventory`
md5sum	A 128-bit hash value that can be used to verify the integrity of the XML block.
version	`1.0`
StartCassette	The cassette for the beginning of the plate range.
StartSlot	The slot for the beginning of the plate range.
EndCassette	The cassette for the end of the plate range.
EndSlot	The slot for the end of the plate range.

Example

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='PlateStorageInventory'
md5sum='c8a87934c31fc8d1e5751a09a1f1b396'
version='1.0' StartCassette='1' StartSlot='1'
EndCassette='1' EndSlot='4'/>
```

Related topics

For information about...	See...
Returning the bar codes for the requested plates	"Inventory Plate Barcodes update XML block" on page 103
Return codes	"Return Code values" on page 47
VWorks software inventory editor	*VWorks Version 3 Automation Control User Guide*

UnloadPlate method

Description

The UnloadPlate method is called each time an unload task occurs in a protocol run.

Syntax

```
HRESULT UnloadPlate([in] BSTR Labware, [in]
PlateFlagsType PlateFlags, [in] BSTR
LoadPlateLocationXML, [out,retval] ReturnCode*
retVal);
```

Parameters

The UnloadPlate method has the following parameters:

- [in] BSTR Labware

  The labware name.

- [in] PlateFlagsType PlateFlags

  Indicates whether plate type is normal, lidded, or sealed.

- [in] BSTR UnLoadPlateLocationXML

  A StorageLocation XML block that describes the specific location where VWorks software will unload the plate.

- [out,retval] ReturnCode *retVal

  Indicates whether or not the request was completed.

Related topics

For information about...	See...
PlateFlagsType values	“PlateFlagsType values” on page 47
StorageLocation XML block	“StorageLocation XML block” on page 143
Return codes	“Return Code values” on page 47

VHooks software interface reference 10

This chapter describes the VHooksInterface COM interface methods. The VHooks interface provides methods for acting on events in VWorks software.

This chapter contains the following topics:

- "VHooks XML parameters" on page 153
- "Method list" on page 154
- "Aborted method" on page 156
- "AvailablePlateList method" on page 157
- "BarCodeRead method" on page 157
- "BarCodeMisread method" on page 159
- "CompileComplete method" on page 161
- "CustomMenuClick method" on page 162
- "Deadlock method" on page 164
- "Error method" on page 164
- "FileOpened method" on page 166
- "FileSaved method" on page 167
- "GetUserInterface method" on page 168
- "LiquidTransferComplete method" on page 168
- "PipetProcessFinished method" on page 171
- "PipetProcessStarting method" on page 173
- "PipetTaskFinished method" on page 174
- "PipetTaskStarting method" on page 176
- "PlateGroupMapping method" on page 177
- "ProcessFinished method" on page 178
- "ProcessStarting method" on page 180
- "ProtocolFinished method" on page 181
- "ProtocolStarted method" on page 182
- "RobotMove method" on page 184
- "RobotPickComplete method" on page 185

- ❑ “RobotPlaceComplete method” on page 186
- ❑ “ScriptPlateError method” on page 187
- ❑ “TaskFinished method” on page 188
- ❑ “TaskStarting method” on page 190
- ❑ “UserLoggedIn method” on page 191
- ❑ “UserLoggedOut method” on page 192

VHooks XML parameters

About this topic

This topic describes the syntax for the standard VHooks XML string used in many of the method parameters.

Standard VHooks XML

A standard VHooks XML string. is used to pass information between VWorks software and your VHooks plug-in. Your VHooks plug-in needs to:

- ❑ Parse the standard VHooks XML string from VWorks software provided as the input parameter in the COM interface methods
- ❑ Construct a well-formed standard VHooks XML string as the output parameter that is sent back to VWorks software in the COM interface methods

The first line of the XML string is always the following XML declaration.

```
<?XML version='1.0' encoding='ASCII' ?>
```

The next line is the always the document element <velocity11 />. The <velocity11 /> element includes a specific set of attributes depending on the method used. The set of attributes is described with each method.

Common VHooks output parameter

Many of the VHooks methods use a common VHooks XML string for the output parameter. This output string is described here and when a method uses this output string, it is referred to as the common VHooks output parameter.

The common VHooks XML output parameter consists of the standard VHooks XML string with the following attributes.

Attribute	Description
version	The method version number.
LogMessage	Writes the specified message to the VWorks software log
PauseExecution	Requests VWorks software to pause the protocol run. You can specify any value for this attribute. If the attribute exists, then the request to pause is enabled.
PauseMessage	Message to display to the operator with the request to pause the protocol run. If no pause message is specified, the following message is displayed to the user: “Pausing protocol execution at plugin’s request”

Attribute	Description
AbortExecution	Abort the protocol run. You can specify any value for this attribute. If the attribute exists, then the request to abort execution is enabled.
AbortMessage	Message to display to the operator with the request to abort the protocol run. If no abort message is specified, the following message is displayed to the user: "Aborting protocol execution at plugin's request"

The following example output XML string makes a request to VWorks software to abort the protocol run with an abort message.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 version='1.0' AbortExecution='1'
AbortMessage='Protocol aborted due to
unrecoverable errors.' />
```

Related topics

For information about...	See...
Methods for VHooks software interface	"Method list" on page 154

Method list

About this topic

This topic lists the VHooksInterface COM interface methods.

Methods

The following table provides an alphabetical listing of the VHooks interface methods.

Method	Description	See...
Aborted	A user aborted the protocol	"Aborted method" on page 156
AvailablePlateList	Provides a list of plates after a protocol is compiled	"AvailablePlateList method" on page 157
BarCodeMisread	A bar code misread occurred	"BarCodeMisread method" on page 159
BarCodeRead	A bar code has been read	"BarCodeRead method" on page 157

Method	Description	See...
CompileComplete	Compile of a protocol is complete	"CompileComplete method" on page 161
CustomMenuClick	A custom context menu item in the inventory editor was selected	"CustomMenuClick method" on page 162
Deadlock	A non-recoverable error occurred	"Deadlock method" on page 164
Error	An error occurred	"Error method" on page 164
FileOpened	A protocol, run-set, or plug-in file was opened	"FileOpened method" on page 166
FileSaved	A protocol, run-set, or plug-in file was saved	"FileSaved method" on page 167
GetUserInterface	User selected **Tools > Show Plugins** command	"GetUserInterface method" on page 168
LiquidTransferComplete	Provides information for source and destination plates upon the completion of liquid transfer	"LiquidTransferComplete method" on page 168
PipetTaskFinished	❑ A pipette task completed (VPrep) ❑ A sub-process task completed	"ProcessFinished method" on page 178
PipetTaskStarting	❑ A pipette task started (VPrep only) ❑ A sub-process task started	"PipetTaskStarting method" on page 176
PipetProcessFinished	❑ A pipette process completed (VPrep only) ❑ A sub-process completed	"PipetProcessFinished method" on page 171
PipetProcessStarting	❑ A pipette process started (VPrep only) ❑ A sub-process started	"PipetProcessStarting method" on page 173
PlateGroupMapping	Provides a plate group name and the list of barcodes associated with that plate group	"PlateGroupMapping method" on page 177
ProcessFinished	A process completed	"ProcessFinished method" on page 178
ProcessStarting	A process started	"ProcessStarting method" on page 180
ProtocolFinished	A protocol has completed	"ProcessFinished method" on page 178
ProtocolStarted	A protocol has started	"ProtocolStarted method" on page 182

Method	Description	See...
RobotMove	The robot moved to away from the current location and/or to a new location	"RobotMove method" on page 184
RobotPickComplete	The robot has picked up a plate	"RobotPickComplete method" on page 185
RobotPlaceComplete	The robot has placed a plate	"RobotPlaceComplete method" on page 186
ScriptPlateError	Request from JavaScript to send a message to the plug-in	"ScriptPlateError method" on page 187
TaskFinished	A task completed	"TaskFinished method" on page 188
TaskStarting	A task started	"TaskStarting method" on page 190
UserLoggedIn	A user has logged into VWorks software	"UserLoggedIn method" on page 191
UserLoggedOut	A user has logged off of VWorks software	"UserLoggedOut method" on page 192

Aborted method

Description

The Abort method is called after the user has aborted a protocol.

Syntax

```
HRESULT Aborted([in] BSTR sXML, [in, out] BSTR
*sResultXML);
```

Parameters

The Aborted method has the following parameters:

- [in] BSTR sXML

  The sXML parameter is always the empty string.

- [in, out] BSTR* pResultXML

  Use the common XML output parameter.

Related topics

For information about...	See...
Deadlock event	"Deadlock method" on page 164
Error event	"Error method" on page 164
Common VHooks output parameter	"Common VHooks output parameter" on page 153

AvailablePlateList method

Description

The AvailablePlateList method is called after a protocol has been compiled. VWorks software provides the plate names from the current loaded protocol.

Syntax

```
HRESULT AvailablePlateList([in] VARIANT
PlateNames, [in, out] BSTR *pResultXML);
```

Parameters

The AvailablePlateList method has the following parameters:

- [in] VARIANT PlateNames

  A list of plate names.

- [in, out] BSTR* pResultXML

  Use the common XML output parameter.

Related topics

For information about...	See...
Common VHooks output parameter	"Common VHooks output parameter" on page 153

BarCodeRead method

Description

The BarCodeRead method is called after a plate bar code was read. The plug-in must indicate whether or not to request that VWorks software place the misread plate into quarantine. The user can specify devices to use as a quarantine station when setting up a plate icon in the Task Parameters toolbar.

Syntax

```
HRESULT BarCodeRead([in] BSTR sXML, [in, out]
BSTR *sResultXML);
```

Parameters

The BarCodeRead method has the following parameters:

- [in] BSTR sXML

  Standard VHooks XML string with the following attributes:

Attribute	Description/Value
file	`BarCodeRead`
version	The method version number.
NorthSideBarcode	The barcode if located on the north side of the plate. Otherwise, the value is the empty string.
SouthSideBarcode	The barcode if located on the south side of the plate. Otherwise, the value is the empty string.
WestSideBarcode	The barcode if located on the south side of the plate. Otherwise, the value is the empty string.
EastSideBarcode	The barcode if located on the south side of the plate. Otherwise, the value is the empty string.
PlateName	The name user has specified for the plate.
InstanceNumber	The number indicating the instance of this process in the number of cycles specified for the protocol.
Plugin	The name of a plug-in (DLL file) located in the VWorks software plugins folder.
Labware	The labware name (in VWorks software, this is referred to as the Plate type in the Task Parameters toolbar)
Device	The device name (can be empty).

  Example:

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='BarCodeRead'
md5sum='849392019ca47102839e845113d11840'
version='1.0' NorthSideBarcode='XUB89893-
909' SouthSideBarcode='' WestSideBarcode=''
EastSideBarcode='' PlateName=''
InstanceNumber='1' Plugin='' Labware='384
V11 ST10 Tip Box 10734.102' Device='' />
```

- [in, out] BSTR* sResultXML

  Use the common XML output parameter. In addition to the common output XML parameter attributes, there is one additional required attribute.

Attribute	Description
Action	❑ `BCR_QUARANTINE` Send the plate to the quarantine location. ❑ `BCR_IGNORE` No action is taken.

Related topics

For information about...	See...
Barcode misread event	“BarCodeMisread method” on page 159
Common VHooks output parameter	“Common VHooks output parameter” on page 153

BarCodeMisread method

Description

The BarCodeMisread method is called after a plate barcode was misread. The plug-in must indicate whether or not to request that VWorks software place the misread plate into quarantine. The user can specify devices to use as a quarantine station when setting up a plate icon in the Task Parameters toolbar.

Syntax

```
HRESULT BarCodeMisread([in] BSTR sXML, [in,
out] BSTR *sResultXML);
```

Parameters

The BarCodeMisread method has the following parameters:

❑ [in] BSTR sXML

Standard VHooks XML string with the following attributes:

Attribute	Description/Value
file	`BarCodeMisread`
version	The method version number.
PlateName	The plate name.
InstanceNumber	The number indicating the instance of this process in the number of cycles specified for the protocol.
Plugin	The name of a plug-in (DLL file) located in the VWorks software plugins folder.

Attribute	Description/Value
Labware	The labware name (in VWorks software, this is referred to as the Plate type in the Task Parameters toolbar)
BarcodeSide	Indicates on which side the bar code read failed. ❑ NORTH_SIDE ❑ SOUTH_SIDE ❑ EAST_SIDE, ❑ WEST_SIDE
Device	The device name (can be empty).

Example:

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='BarCodeMisread'
md5sum='849392019ca47102839e845113d11840'
version='1.0' BarcodeSide='SOUTH_SIDE'
PlateName='NameofPlate' Plugin='' '384 V11
ST10 Tip Box 10734.102' Device='' />
```

❑ [in, out] BSTR* sResultXML

Use the common XML output parameter. In addition to the common output XML parameter attributes, there is one additional required attribute.

Attribute	Description
Action	❑ BCR_QUARANTINE Send the plate to the quarantine location. ❑ BCR_IGNORE No action is taken.

The following example requests that VWorks software write a message to the Log and quarantine the plate.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 version='1.0' LogMessage='Bar
code misread. Moved to quarantine.'
Action='BCR_QUARANTINE' />
```

Related topics

For information about...	See...
Common VHooks output parameter	“Common VHooks output parameter” on page 153

CompileComplete method

Description

The CompileComplete method is called after a protocol has been compiled.

Syntax

```
HRESULT CompileComplete([in] BSTR sXML, [in,
out] BSTR *sResultXML);
```

Parameters

The CompileComplete method has the following parameters:

- [in] BSTR sXML

  Standard VHooks XML string with the following attributes.

Attribute	Description
file	`CompileComplete`
md5sum	A 128-bit hash value that can be used to verify the integrity of the XML block.
version	The method version number.
Errors	The number of errors encountered during the compile.
Warnings	The number of warnings encountered during the compile.

  Example:

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='CompileComplete'
md5sum='849392019ca47102839e845113d11840'
version='1.0' Errors='0' Warnings='0' />
```

- [in, out] BSTR* sResultXML

  The standard VHooks XML string with the following attributes.

Attribute	Description
version	The method version number.
Action	❑ `AllowErrors` Compile errors are not written to the error log. ❑ Any non-empty string Compile errors are written to the error log.

  The following example XML string is required allow compile errors and request that errors are not logged.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 version='1.0' Errors='0'
Action='AllowErrors' />
```

Related topics

For information about...	See...
Barcode read event	"BarCodeRead method" on page 157
VHooks software methods	"Methods" on page 154

CustomMenuClick method

Description

Custom menu items can be added to the VWorks software inventory editor's Inventory Management tab using the IWorksController interface. When the user selects one of these custom menu items, the CustomMenuClick method is called. VWorks software provides which custom menu item was selected and the selected plates from the inventory editor.

Syntax

```
HRESULT CustomMenuClick([in] BSTR sXML, BSTR
*sResultXML);
```

Parameters

The CustomMenuClick method has the following parameters:

- [in] BSTR sXML

  An XML string containing the selected custom menu and the selected plates from the Inventory Editor. The string is a standard VHooks XML string with the following attributes.

Attribute	Description
file	`CustomMenuClick`
version	The method version number.
md5sum	A 128-bit hash value that can be used to verify the integrity of the XML block.
MenuName	The name of the menu.
MenuItem	The selected menu item.

  In addition to the attributes listed above for the root element <Velocity11 />, the following child elements are included.

Element	Attribute	Description
Rows	RowCount	The number of rows.
	DataCount	The total number of data items.
Row	The child element of `<Rows>` for each row of data.	
Data	The child element of `<Row>` that includes the data in the current row.	
	Name	The name of the data item.
	Value	The value of the data item.

Example:

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='CustomMenuClick'
md5sum='849392019ca47102839e845113d11840'
version='1.0' MenuName='Inventory'
MenuItem='custom-menu'
   <Rows RowCount='2' DataCount='8'
      <Row>
         <Data Name='ItemID' Value='0'>
         <Data Name='cassette' Value='1'>
         <Data Name='device' Value='PlateHub'>
         <Data Name='slot' Value='1'>
      </Row>
      <Row>
         <Data Name='ItemID' Value='1'>
         <Data Name='cassette' Value='1'>
         <Data Name='device' Value='PlateHub'>
         <Data Name='slot' Value='2'>
      </Row>
   </Rows> />
</Velocity11>
```

- ❑ [in, out] BSTR* sResultXML

  Use the common XML output parameter.

Related topics

For information about...	See...
Common VHooks output parameter	“Common VHooks output parameter” on page 153
Adding custom menu items to the inventory editor	“Menu update XML block” on page 105

Deadlock method

Description

The Deadlock method is called when there are remaining tasks to perform but it is impossible to perform them. For example, if there are too many plates in the system and there is no way to move plates around further, this can result in a deadlock. When a deadlock occurs, the protocol stops.

Syntax

```
HRESULT Deadlock([in] BSTR sXML, [in, out] BSTR
*sResultXML);
```

Parameters

The Deadlock method has the following parameters:

- [in] BSTR sXML

  The sXML parameter is always the empty string.

- [in, out] BSTR* sResultXML

  Use the common XML output parameter.

Related topics

For information about...	See...
Abort event	“Aborted method” on page 156
Error event	“Error method” on page 164
Common VHooks output parameter	“Common VHooks output parameter” on page 153

Error method

Description

The Error method is called whenever an error occurs in VWorks software.

Syntax

```
HRESULT Error([in] BSTR sXML,[in, out] BSTR
*sResultXML);
```

Parameters

The Error method has the following parameters:

- [in] BSTR sXML

  Standard VHooks XML string with the following attributes.

Attribute	Description
file	`Error`

Attribute	Description
version	The method version number.
Message	VWorks software error message.
If the error is associated with a device, the following attributes are included...	
DeviceName	The name of the device.
DeviceParent	The name of the device's parent (for example, VPrep for VPrep shelves, or BenchCel for stackers.
Labware	The labware name (in VWorks software, this is referred to as the Plate type in the Task Parameters toolbar).
Profile	The profile name.
If there is a plate on the device, the following attributes are included...	
PlateName	The barcode if located on the south side of the plate. Otherwise, the value is the empty string.
NorthSideBarcode	The barcode if located on the north side of the plate. Otherwise, the value is the empty string.
SouthSideBarcode	The barcode if located on the south side of the plate. Otherwise, the value is the empty string.
WestSideBarcode	The barcode if located on the south side of the plate. Otherwise, the value is the empty string.
EastSideBarcode	The barcode if located on the south side of the plate. Otherwise, the value is the empty string.

- [in, out] BSTR* sResultXML

  Use the common XML output parameter.

Related topics

For information about...	See...
Abort event	"Aborted method" on page 156
Deadlock event	"Deadlock method" on page 164
Common VHooks output parameter	"Common VHooks output parameter" on page 153

FileOpened method

Description

The FileOpened method is called when a protocol, a run-set, or a plugin file has been opened. This method is also called when a new protocol file is created.

Syntax

```
HRESULT FileOpened([in] BSTR sXML, [in, out]
BSTR *sResultXML);
```

Parameters

The FileOpened method has the following parameters:

- [in] BSTR sXML

  Standard VHooks XML string with the following attributes:

Attribute	Description/Value
file	`FileOpened`
version	The method version number.
Type	The type of file that was opened. One of the following values: ❑ `PluginFile` ❑ `ProtocolFile` ❑ `RunsetFile`
Path	The path to the file.

  Example:

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='FileOpened'
md5sum='849392019ca47102839e845113d11840'
version='1.0' Type='ProtocolFile'
Path='C:\VWorks
Workspace\Protocols\protocol1.pro' />
```

- [in, out] BSTR* sResultXML

  Use the common XML output parameter.

Related topics

For information about...	See...
File saved event	“FileSaved method” on page 167
Common VHooks output parameter	“Common VHooks output parameter” on page 153

FileSaved method

Description

The FileSaved method is called when either protocol, a run-set, or a plugin file has been saved.

Syntax

```
HRESULT FileSaved([in] BSTR sXML, [in, out]
BSTR *sResultXML);
```

Parameters

The FileSaved method has the following parameters:

- [in] BSTR sXML

  Standard VHooks XML string with the following attributes:

Attribute	Description/Value
file	`FileSaved`
version	The method version number.
Type	The type of file that was saved as one of the following values: ❑ `PluginFile` ❑ `ProtocolFile` ❑ `RunsetFile`
Path	The path to the file.

  Example:

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='FileSaved'
md5sum='849392019ca47102839e845113d11840'
version='1.0' Type='ProtocolFile'
Path='C:\VWorks
Workspace\Protocols\file1.pro' />
```

- [in, out] BSTR* sResultXML

  Use the common XML output parameter.

Related topics

For information about...	See...
File opened event	“FileOpened method” on page 166
Common VHooks output parameter	“Common VHooks output parameter” on page 153

GetUserInterface method

Description

To integrate your application GUI in VWorks software, you provide the Windows handle for your application. The GetUserInterface method is called when a user chooses **Tools > Show Plugins** to launch your application.

Syntax

```
HRESULT GetUserInterface([in, out] BSTR
*pResultXML);
```

Parameters

The GetUserInterface method has the following parameter:

- [in, out] BSTR* pResultXML

  Standard VHooks XML string with the following attributes:

Attribute	Description
pHWND	A windows handle of your GUI application.

  The following example XML string will open the GUI application using the Windows handle provided.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 pHWND='12345' />
```

Related topics

For information about...	See...
Common VHooks output parameter	"Common VHooks output parameter" on page 153

LiquidTransferComplete method

Description

The LiquidTransferComplete method is called upon the completion of a liquid transfer. This method can be used to keep a liquid management system updated in real-time about the locations of compounds in the system.

Syntax

```
HRESULT LiquidTransferComplete([in] BSTR
SourcePlateInfoXML, [in] BSTR
DestinationPlateInfoXML, [in, out] BSTR
*pResultXML);
```

Parameters

The LiquidTransferComplete method has the following parameters:

- [in] BSTR SourcePlateInfoXML

  Standard VHooks XML string with the following attributes:

Attribute	Description
file	`SourcePlateInfo`
version	The method version number.
Timestamp	The date and time of the liquid transfer in the following format: “m/d/yy - H:M:S.MS am/pm” where m = month (1–12), d = day(1–31), yy = year (00–99), H = hour (1–12) M = minutes (00–59) S = seconds (00–59) MS = miliseconds (00–59)
NorthSideBarcode	The barcode if located on the north side of the plate. Otherwise, the value is the empty string.
SouthSideBarcode	The barcode if located on the south side of the plate. Otherwise, the value is the empty string.
WestSideBarcode	The barcode if located on the south side of the plate. Otherwise, the value is the empty string.
EastSideBarcode	The barcode if located on the south side of the plate. Otherwise, the value is the empty string.
Quadrant	The set quadrant which depends on the plate to labware mapping (1, 1–4, or 1–16).
Volume	The volume in microliters.
Plate	The plate name.
Device	The device name.

Example:

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='SourcePlateInfo'
md5sum='849392019ca47102839e845113d11840'
version='1.0' TimeStamp='2/2/07 -
5:00:00:00' NorthSideBarcode='XUB90-3344'
SouthSideBarcode='' WestSideBarcode=''
```

```
EastSideBarcode='' Quadrant='4' Volume='10'
Plate='PlateA' Device='Device1A' />
```

❑ [in] BSTR DestinationPlateInfoXML

Standard VHooks XML string with the following attributes:

Attribute	Description
file	`DestinationPlateInfo`
version	The method version number.
Timestamp	The date and time of the liquid transfer in the following format: "m/d/yy - H:M:S.MS am/pm" where m = month (1–12), d = day(1–31), yy = year (00–99), H = hour (1–12) M = minutes (00–59) S = seconds (00–59) MS = miliseconds (00–59)
NorthSideBarcode	The barcode if located on the north side of the plate. Otherwise, the value is the empty string.
SouthSideBarcode	The barcode if located on the south side of the plate. Otherwise, the value is the empty string.
WestSideBarcode	The barcode if located on the south side of the plate. Otherwise, the value is the empty string.
EastSideBarcode	The barcode if located on the south side of the plate. Otherwise, the value is the empty string.
Quadrant	The set quadrant which depends on the plate to labware mapping (1, 1–4, or 1–16).
Volume	The volume in microliters.
Plate	The plate name.
Device	The device name.

Example:

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='DestinationPlateInfo'
md5sum='849392019ca47102839e845113d11840'
version='1.0' TimeStamp='2/2/07 -
5:00:00:00' NorthSideBarcode='XUB90-3346'
SouthSideBarcode='' WestSideBarcode=''
```

```
EastSideBarcode='' Quadrant='2' Volume='10'
Plate='PlateA' Device='Device1A' />
```

- [in, out] BSTR* pResultXML

  Use the common XML output parameter.

Related topics

For information about...	See...
Common VHooks output parameter	"Common VHooks output parameter" on page 153

PipetProcessFinished method

Description

The PipetProcessFinished method is called as each pipette process is completed.

Note: The PipetProcessFinished method provides two different input parameters—one for a Pipette Process (VPrep Pipettor only) and one for a sub-process. The `file` attribute indicates the type of process.

Syntax

```
HRESULT PipetProcessFinished([in] BSTR sXML,
[in, out] BSTR *sResultXML);
```

Parameters

The PipetProcessFinished method has the following parameters:

- [in] BSTR sXML

  Pipette process (VPrep Pipettor only). Standard VHooks XML string with the following attributes.

Attribute	Description/Value
file	`PipetProcessFinished`
version	The method version number.
TaskCount	The number of tasks in the process.

  Example:

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='PipetProcessFinished'
```

```
md5sum='849392019ca47102839e845113d11840'
version='1.0' TaskCount='2' />
```

Sub-process. Standard VHooks XML string with the following attributes.

Attribute	Description/Value
file	`SubProcessFinished`
version	The method version number.
TaskCount	The number of tasks in the process.
Description	The task description.

Example:

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='SubProcessFinished'
md5sum='849392019ca47102839e845113d11840'
version='1.0' TaskCount='2'
Description='Aspirate (Bravo)' />
```

- [in, out] BSTR* sResultXML

  Use the common XML output parameter.

Related topics

For information about...	See...
Pipette process started event	"PipetProcessStarting method" on page 173
Process started event	"ProcessStarting method" on page 180
Process finished event	"ProcessFinished method" on page 178
Protocol started event	"ProtocolStarted method" on page 182
Protocol finished event	"ProtocolFinished method" on page 181
Common VHooks output parameter	"Common VHooks output parameter" on page 153
Pipette processes	"VWorks software basics" on page 4

PipetProcessStarting method

Description

The PipetProcessStarting method is called as each pipette process starts. A pipette process is a sequence of pipette tasks that perform liquid handling procedures.

Note: The PipetProcessStarting method provides two different input parameters—one for a Pipette Process (VPrep Pipettor only) and one for a sub-process. The `file` attribute indicates the type of process.

Syntax

```
HRESULT PipetProcessStarting([in] BSTR sXML,
[in, out] BSTR *sResultXML);
```

Parameters

The PipetProcessStarting method has the following parameters:

- [in] BSTR sXML

  Pipette process (VPrep Pipettor only). Standard VHooks XML string with the following attributes.

Attribute	Description/Value
file	`PipetProcessStarting`
version	The method version number.
TaskCount	The number of tasks in the process.

  Example:

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='PipetProcessStarting'
md5sum='849392019ca47102839e845113d11840'
version='1.0' TaskCount='2' />
```

  Sub-process. Standard VHooks XML string with the following attributes (sub-process).

Attribute	Description/Value
file	`SubProcessStarting`
version	The method version number.
TaskCount	The number of tasks in the process.
Description	The task description.

  Example:

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='SubProcessStarting'
md5sum='849392019ca47102839e845113d11840'
version='1.0' TaskCount='2'
Description='Aspirate (Bravo)' />
```

- [in, out] BSTR* sResultXML

  Use the common XML output parameter.

Related topics

For information about...	See...
Pipette process finished event	“PipetProcessFinished method” on page 171
Process started event	“ProcessStarting method” on page 180
Process finished event	“ProcessFinished method” on page 178
Protocol started event	“ProtocolStarted method” on page 182
Protocol finished event	“ProtocolFinished method” on page 181
Common VHooks output parameter	“Common VHooks output parameter” on page 153
Pipette processes	“VWorks software basics” on page 4

PipetTaskFinished method

Description

The PipetTaskFinished method is called as each pipette task is completed.

Note: The PipetTaskFinished method provides two different input parameters—one for a Pipette task (VPrep Pipettor only) and one for a sub-process task. The `file` attribute indicates the type of task.

Syntax

```
HRESULT PipetTaskFinished([in] BSTR sXML, [in,
out] BSTR *sResultXML);
```

Parameters

The PipetTaskFinished method has the following parameters:

- [in] BSTR sXML

  Pipette task (VPrep Pipettor only). Standard VHooks XML string with the following attributes.

Attribute	Description/Value
file	`PipetTaskFinished`
version	The method version number.
PlateName	The name of the plate.
Description	The task description displayed under the icon in the protocol editor.

  Example:

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='PipetTaskFinished'
```

```
md5sum='849392019ca47102839e845113d11840'
version='1.0' PlateName='Plate1'
Description='Mix 10.00 ul 3 times at Plate1
quadrant 1' />
```

Sub-process task. Standard VHooks XML string with the following attributes.

Attribute	Description/Value
file	`PipetTaskFinished`
version	The method version number.
Description	The task description displayed under the icon in the protocol editor.

Example:

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='SubTaskFinished'
md5sum='849392019ca47102839e845113d11840'
version='1.0' Description='' />
```

- ❑ [in, out] BSTR* sResultXML

  Use the common XML output parameter.

Related topics

For information about...	See...
Pipette task started event	"PipetTaskStarting method" on page 176
Task started event	"TaskStarting method" on page 190
Task finished event	"TaskFinished method" on page 188
Common VHooks output parameter	"Common VHooks output parameter" on page 153
Pipette tasks	"VWorks software basics" on page 4

PipetTaskStarting method

Description

The PipetTaskStarting method is called as each pipette task starts. A pipette task is an operation that is performed on one or more plates by a liquid handler.

Note: The PipetTaskStarting method provides two different input parameters—one for a Pipette task (VPrep Pipettor only) and one for a sub-process task. The `file` attribute indicates the type of task.

Syntax

```
HRESULT PipetTaskStarting([in] BSTR sXML, [in,
out] BSTR *sResultXML);
```

Parameters

The PipetTaskStarting method has the following parameters:

- [in] BSTR sXML

  Pipette process (VPrep Pipettor only). Standard VHooks XML string with the following attributes.

Attribute	Description/Value
file	`PipetTaskStarting`
version	The method version number.
PlateName	The name of the plate.
Description	The task description displayed under the icon in the protocol editor.

  Example:

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='PipetTaskStarting'
md5sum='849392019ca47102839e845113d11840'
version='1.0' PlateName='Plate1'
Description='Mix 10.00 ul 3 times at Plate1
quadrant 1' />
```

  Sub-process. Standard VHooks XML string with the following attributes.

Attribute	Description/Value
file	`SubTaskStarting`
version	The method version number.
Description	The task description displayed under the icon in the protocol editor.

  Example:

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='SubTaskStarting'
```

```
md5sum='849392019ca47102839e845113d11840'
version='1.0' Description='' />
```

- [in, out] BSTR* sResultXML

  Use the common XML output parameter.

Related topics

For information about...	See...
Pipette task finished event	"PipetTaskFinished method" on page 174
Task started event	"TaskStarting method" on page 190
Task finished event	"TaskFinished method" on page 188
Common VHooks output parameter	"Common VHooks output parameter" on page 153
Pipette tasks	"VWorks software basics" on page 4

PlateGroupMapping method

Description

The PlateGroupMapping method is called when the user clicks **OK** to close the user interface for the plug-in provided with the GetUserInterface method.

This method is also called when the user saves a plug-in file using the **File > Plugin File** command.

Syntax

```
HRESULT PlateGroupMapping([in] BSTR
PlateGroupName, [in] VARIANT Barcodes, [in,
out] BSTR *pResultXML);
```

Parameters

The PlateGroupMapping method has the following parameters:

- [in] BSTR PlateGroupName

  A plate group name.

- [in] VARIANT Barcodes

  A list of barcodes in the plate group (array of strings).

- [in, out] BSTR* sResultXML

  Use the common XML output parameter.

Related topics

For information about...	See...
Plate groups	*VWorks Version 3 Automation Control User Guide*
Method to display plug-in user interface	"GetUserInterface method" on page 168
Common VHooks output parameter	"Common VHooks output parameter" on page 153

ProcessFinished method

Description

The ProcessFinished method is called when each protocol process is completed.

Syntax

```
HRESULT ProcessFinished([in] BSTR sXML, [in,
out] BSTR *sResultXML);
```

Parameters

The ProcessFinished method has the following parameters:

- [in] BSTR sXML

  Standard VHooks XML string with the following attributes.

Attribute	Description/Value
file	`ProcessFinished`
version	The method version number.
Plugin	The name of the plug-in.
Labware	The labware name (in VWorks software, this is referred to as the Plate type in the Task Parameters toolbar).
TaskCount	The number of tasks in the process.
NorthSideBarcode	The barcode if located on the north side of the plate. Otherwise, the value is the empty string.
SouthSideBarcode	The barcode if located on the south side of the plate. Otherwise, the value is the empty string.
WestSideBarcode	The barcode if located on the south side of the plate. Otherwise, the value is the empty string.
EastSideBarcode	The barcode if located on the south side of the plate. Otherwise, the value is the empty string.

Attribute	Description/Value
PlateName	The name of the plate.
InstanceNumber	The number indicating the instance of this process in the number of cycles specified by the operator for the protocol.

Example:

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='ProcessFinished'
md5sum='849392019ca47102839e845113d11840'
version='1.0' Plugin='NameofPlugin'
Labware='UsePlugin' TaskCount='3'
NorthSideBarcode='' SouthSideBarcode=''
WestSideBarcode='XJA123456'
EastSideBarcode='' PlateName='PlateA'
InstanceNumber='1' />
```

- [in, out] BSTR* sResultXML

  Use the common XML output parameter.

Related topics

For information about...	See...
Process started event	“ProcessStarting method” on page 180
Protocol started event	“ProtocolStarted method” on page 182
Protocol finished event	“ProtocolFinished method” on page 181
Pipette process started event	“PipetProcessStarting method” on page 173
Pipette process finished event	“PipetProcessFinished method” on page 171
Common VHooks output parameter	“Common VHooks output parameter” on page 153
Processes	“VWorks software basics” on page 4

ProcessStarting method

Description

The ProcessStarting method is called when each protocol process starts. A process is the sequence of tasks that are performed on a plate icon.

Syntax

```
HRESULT ProcessStarting([in] BSTR sXML, [in,
out] BSTR *sResultXML);
```

Parameters

The ProcessStarting method has the following parameters:

- [in] BSTR sXML

  Standard VHooks XML string with the following attributes.

Attribute	Description/Value
file	`ProcessStarting`
version	The method version number.
Plugin	The name of the plug-in.
Labware	The labware name (in VWorks software, this is referred to as the Plate type in the Task Parameters toolbar)
TaskCount	The number of tasks in the process.
NorthSideBarcode	The barcode if located on the north side of the plate. Otherwise, the value is the empty string.
SouthSideBarcode	The barcode if located on the south side of the plate. Otherwise, the value is the empty string.
WestSideBarcode	The barcode if located on the south side of the plate. Otherwise, the value is the empty string.
EastSideBarcode	The barcode if located on the south side of the plate. Otherwise, the value is the empty string.
PlateName	The name of the plate.
InstanceNumber	The number indicating the instance of this process in the number of cycles specified for the protocol.

Example:

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='ProcessStarting'
md5sum='849392019ca47102839e845113d11840'
version='1.0' Plugin='NameofPlugin'
Labware='UsePlugin' TaskCount='3'
NorthSideBarcode='' SouthSideBarcode=''
WestSideBarcode='XJA123456'
```

```
EastSideBarcode='' PlateName='PlateA'
InstanceNumber='1' />
```

- [in, out] BSTR* sResultXML

  Use the common XML output parameter.

Related topics

For information about...	See...
Process finished event	“ProcessFinished method” on page 178
Protocol started event	“ProtocolStarted method” on page 182
Protocol finished event	“ProtocolFinished method” on page 181
Pipette process started event	“PipetProcessStarting method” on page 173
Pipette process finished event	“PipetProcessFinished method” on page 171
Common VHooks output parameter	“Common VHooks output parameter” on page 153
Processes	“VWorks software basics” on page 4

ProtocolFinished method

Description

The ProtocolFinished method is called after a protocol has completed.

Syntax

```
HRESULT ProtocolFinished([in] BSTR sXML, [in,
out] BSTR *sResultXML);
```

Parameters

The ProtocolFinished method has the following parameters:

- [in] BSTR sXML

  Standard VHooks XML string with the following attributes.

Attribute	Description/Value
file	`ProtocolFinished`
version	The method version number.
Path	The path to the protocol file or `<not specified>` if the path is empty.

  Example:

```
<?xml version='.0' encoding='ASCII' ?>
<Velocity11 file='ProtocolFinished'
md5sum='849392019ca47102839e845113d11840'
version='1.0'
Path='C:\VWorks Workspace\Protocol1.pro' />
```

- [in, out] BSTR* sResultXML

  Use the common XML output parameter.

Related topics

For information about...	See...
Protocol started event	"ProtocolStarted method" on page 182
Process started event	"ProcessStarting method" on page 180
Process finished event	"ProcessFinished method" on page 178
Pipette process started event	"PipetProcessStarting method" on page 173
Pipette process finished event	"PipetProcessFinished method" on page 171
Common VHooks output parameter	"Common VHooks output parameter" on page 153
Protocols	"VWorks software basics" on page 4

ProtocolStarted method

Description

The ProtocolStarted method is called after a protocol has started. A protocol is a collection of processes that run at the same time.

Syntax

```
HRESULT ProtocolStarted([in] BSTR sXML, [in,
out] BSTR *sResultXML);
```

Parameters

The ProtocolStarted method has the following parameters:

- [in] BSTR sXML

  Standard VHooks XML string with the following attributes.

Attribute	Description/Value
file	`ProtocolStarted`
version	The method version number.
Path	The path to the protocol file or `<not specified>` if the path is empty.
Notes	The notes associated with the protocol when the protocol was saved.

  Example:

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='ProtocolStarted'
md5sum='849392019ca47102839e845113d11840'
version='1.0'
Path='C:\VWorks Workspace\Protocol1.pro'
Notes='' />
```

- [in, out] BSTR* sResultXML

  Use the common XML output parameter.

Related topics

For information about...	See...
Protocol finished event	“ProtocolFinished method” on page 181
Process started event	“ProcessStarting method” on page 180
Process finished event	“ProcessFinished method” on page 178
Pipette process started event	“PipetProcessStarting method” on page 173
Pipette process finished event	“PipetProcessFinished method” on page 171
Common VHooks output parameter	“Common VHooks output parameter” on page 153
Protocols	“VWorks software basics” on page 4

RobotMove method

Description

The RobotMove method is called just before a robot moves.

Syntax

```
HRESULT RobotMove([in] BSTR sXML, [in, out]
BSTR *sResultXML);
```

Parameters

The RobotMove method has the following parameters:

- [in] BSTR sXML

  Standard VHooks XML string with the following attributes.

Attribute	Description/Value
file	`RobotMove`
version	The method version number.
RobotName	The name for the robot.
FromDeviceName	The name of device robot moved from.
FromLocationName	The name of location robot moved from.
ToDeviceName	The name of device robot moved to.
ToLocationName	The name of location robot moved to.

  Example:

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='RobotMove'
md5sum='849392019ca47102839e845113d11840'
version='1.0' RobotName='Robot-1'
FromDeviceName='Device1A'
FromLocationName='Stage1' />
```

- [in, out] BSTR* sResultXML

  Use the common XML output parameter.

Related topics

For information about...	See...
Robot pick completed event	“RobotPickComplete method” on page 185
Robot place completed event	“RobotPlaceComplete method” on page 186
Common VHooks output parameter	“Common VHooks output parameter” on page 153

RobotPickComplete method

Description

The RobotPickComplete method is called when the robot has completed picking up a plate.

Syntax

```
HRESULT RobotPickComplete([in] BSTR sXML, [in,
out] BSTR *sResultXML);
```

Parameters

The RobotPickComplete method has the following parameters:

- [in] BSTR sXML

  Standard VHooks XML string with the following attributes.

Attribute	Description/Value
file	`RobotPickComplete`
version	The method version number.

  Example:

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='RobotPickComplete'
md5sum='849392019ca47102839e845113d11840'
version='1.0' />
```

- [in, out] BSTR* sResultXML

  Use the common XML output parameter.

Related topics

For information about...	See...
Robot place completed event	“RobotPlaceComplete method” on page 186
Robot move event	“RobotMove method” on page 184
Common VHooks output parameter	“Common VHooks output parameter” on page 153

RobotPlaceComplete method

Description

The RobotPlaceComplete method is called when the robot has placed a plate.

Syntax

```
HRESULT RobotPlaceComplete([in] BSTR sXML, [in,
out] BSTR *sResultXML);
```

Parameters

The RobotPlaceComplete method has the following parameters:

- [in] BSTR sXML

  Standard VHooks XML string with the following attributes.

Attribute	Description/Value
file	`RobotPlaceComplete`
version	The method version number.

  Example:

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='RobotPlaceComplete'
md5sum='849392019ca47102839e845113d11840'
version='1.0' />
```

- [in, out] BSTR* sResultXML

  Use the common XML output parameter.

Related topics

For information about...	See...
Robot pick completed event	“RobotPickComplete method” on page 185
Robot move event	“RobotMove method” on page 184
Common VHooks output parameter	“Common VHooks output parameter” on page 153

ScriptPlateError method

Description

The ScriptPlateError method is used to send a message to the plug-in from a script associated with a protocol. Use the following JavaScript method in the JavaScript to invoke this VHooks method.

```
plate.reportErrorToPlugin("message to plugin")
```

Syntax

```
HRESULT ScriptPlateError([in, out] BSTR
*pResultXML);
```

Parameters

The ScriptPlateError method has the following parameters:

- [in, out] BSTR* pResultXML

  Standard VHooks XML string with the following attributes:

Attribute	Description/Value
file	`ScriptPlateError`
version	The method version number.
Message	The error message from the script.
PlateName	The name of plate.
NorthSideBarcode	The barcode if located on the north side of the plate. Otherwise, the value is the empty string.
SouthSideBarcode	The barcode if located on the south side of the plate. Otherwise, the value is the empty string.
WestSideBarcode	The barcode if located on the south side of the plate. Otherwise, the value is the empty string.
EastSideBarcode	The barcode if located on the south side of the plate. Otherwise, the value is the empty string.
Labware	The labware name (in VWorks software, this is referred to as the Plate type in the Task Parameters toolbar).

The following example XML string sends the message from the script to the plugin.

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 version='1.0'
file='ScriptPlateError'
Message='errorMessageText'
PlateName='PlateA' NorthSideBarcode=''
SouthSideBarcode='' WestSideBarcode=''
```

```
EastSideBarcode='XJW9992009B' Labware='1536
Greiner 782076 blk sqr well flt btm' />
```

- [in, out] BSTR* sResultXML

  Use the common XML output parameter.

Related topics

For information about...	See...
Common VHooks output parameter	"Common VHooks output parameter" on page 153

TaskFinished method

Description

The TaskFinished method is called just after each task is completed.

Syntax

```
HRESULT TaskFinished([in] BSTR sXML, [in, out]
BSTR *sResultXML);
```

Parameters

The TaskFinished method has the following parameters:

- [in] BSTR sXML

  Standard VHooks XML string with the following attributes.

Attribute	Description/Value
file	`TaskFinished`
version	The method version number.
Plugin	The name of the plug-in.
Labware	The labware name (this is referred to as the Plate type in the Task Parameters toolbar).
TaskCount	The number of tasks in the process.
PlateName	The name of the plate.
InstanceNumber	The number indicating the instance of this process in the number of cycles specified by the operator for the protocol.
Description	The task description.
Device	The first device from the list of selected devices.
NorthSideBarcode	The barcode if located on the north side of the plate. Otherwise, the value is the empty string.

Attribute	Description/Value
SouthSideBarcode	The barcode if located on the south side of the plate. Otherwise, the value is the empty string.
WestSideBarcode	The barcode if located on the south side of the plate. Otherwise, the value is the empty string.
EastSideBarcode	The barcode if located on the south side of the plate. Otherwise, the value is the empty string.

Example:

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='TaskFinished'
md5sum='849392019ca47102839e845113d11840'
version='1.0' Labware='UsePlugin'
Plugin='pluginname' TaskCount='3'
PlateName='plateA' InstanceNumber='1'
Description='Apply Label' Device='Device1A'
NorthSideBarcode='9383-X8B'
SouthSideBarcode='' WestSideBarcode=''
EastSideBarcode='' />
```

- [in, out] BSTR* sResultXML

  Use the common XML output parameter.

Related topics

For information about...	See...
Task starting event	“TaskStarting method” on page 190
Common VHooks output parameter	“Common VHooks output parameter” on page 153
Protocol tasks	“VWorks software basics” on page 4

TaskStarting method

Description

The TaskStarting method is called as each task starts. A task is an operation that is performed on one plate.

Syntax

```
HRESULT TaskStarting([in] BSTR sXML, [in, out]
BSTR *sResultXML);
```

Parameters

The TaskStarting method has the following parameters:

- [in] BSTR sXML

  Standard VHooks XML string with the following attributes.

Attribute	Description/Value
file	`TaskStarting`
version	The method version number.
Plugin	The name of the plug-in.
Labware	The labware name (in VWorks software, this is referred to as the Plate type in the Task Parameters toolbar).
TaskCount	The number of tasks in the process.
PlateName	The name of the plate.
InstanceNumber	The number indicating the instance of this process in the number of cycles specified by the operator for the protocol.
Description	The task description.
Device	The first device from the list of selected devices.
NorthSideBarcode	The barcode if located on the north side of the plate. Otherwise, the value is the empty string.
SouthSideBarcode	The barcode if located on the south side of the plate. Otherwise, the value is the empty string.
WestSideBarcode	The barcode if located on the south side of the plate. Otherwise, the value is the empty string.
EastSideBarcode	The barcode if located on the south side of the plate. Otherwise, the value is the empty string.

Example:

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='TaskStarting'
md5sum='849392019ca47102839e845113d11840'
```

```
version='1.0' Labware='UsePlugin'
Plugin='pluginname' TaskCount='3'
PlateName='plateA' InstanceNumber='1'
Description='Apply Label' Device='Device1A'
NorthSideBarcode='9383-X8B'
SouthSideBarcode='' WestSideBarcode=''
EastSideBarcode='' />
```

- [in, out] BSTR* sResultXML

  Use the common XML output parameter.

Related topics

For information about...	See...
Task finished event	"TaskFinished method" on page 188
Common VHooks output parameter	"Common VHooks output parameter" on page 153
Protocol tasks	"VWorks software basics" on page 4

UserLoggedIn method

Description

The UserLoggedIn method is called when a user has logged into VWorks software.

Syntax

```
HRESULT UserLoggedIn([in] BSTR sXML, [in, out]
BSTR *sResultXML);
```

Parameters

The UserLoggedIn method has the following parameters:

- [in] BSTR sXML

  Standard VHooks XML string with the following attributes:

Attribute	Description/Value
file	`UserLoggedIn`
version	The method version number.
Account	The account user name.
SecurityLevel	The security level using one of the following values: 0 = Administrator 1 = Technician 2 = Operator 3 = Guest

Example:

```
<?xml version='1.0' encoding='ASCII' ?>
<Velocity11 file='UserLoggedIn'
md5sum='849392019ca47102839e845113d11840'
version='1.0' Account='Administrator'
SecurityLevel='0' />
```

- [in, out] BSTR* sResultXML

  Use the common XML output parameter.

Related topics

For information about...	See...
User logged out event	"UserLoggedOut method" on page 192
Common VHooks output parameter	"Common VHooks output parameter" on page 153

UserLoggedOut method

Description

The UserLoggedOut method is called when a user has logged out of VWorks software.

Syntax

```
HRESULT UserLoggedOut([in] BSTR sXML, [in, out]
BSTR *sResultXML);
```

Parameters

The UserLoggedOut method has the following parameters:

- [in] BSTR sXML

  The sXML parameter is always the empty string.

- [in, out] BSTR* sResultXML

  Use the common XML output parameter.

Related topics

For information about...	See...
User logged in event	"UserLoggedIn method" on page 191
Common VHooks output parameter	"Common VHooks output parameter" on page 153

Testing and debugging 11

This chapter describes ways to test your device driver (IWorks) or VHooks plug-in and covers the following topics:

❑ “Testing your device driver plug-in” on page 194

❑ “Testing your VHooks plug-in” on page 197

Testing your device driver plug-in

About this topic

This topic describes how to use the IWorksTest utility to test your device driver plug-in that you have developed using the VWorks software Device Driver API (IWorks).

The IWorksTest utility is a diagnostics tool designed for initial debugging and testing basic functionality. You should always thoroughly test your plug-in using VWorks software before putting your plug-in into production.

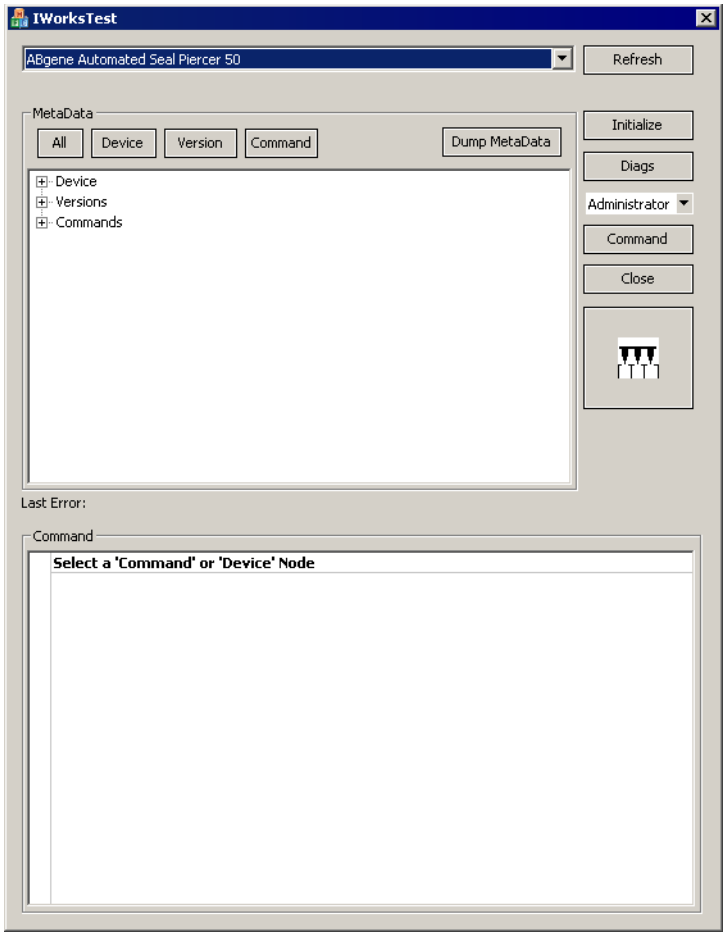
Before you start

Compile your device driver plug-in and copy the appropriate DLL files into the C:*VWorks software_install_dir*\plugins folder.

Starting IWorkTest

To start IWorksTest:

1. Open the C:*VWorks software_install_dir* folder.
2. Double-click IWorksTest.exe to launch the IWorksTest utility.



3. Select your plug-in from the top list box.

   The names displayed in this list box are from the Device Description attribute provided in the XML metadata.

   If your plug-in is not listed in the list box, make sure you have copied all the proper DLL and TLB files to the VWorks software plugins folder. The plug-in will not load if the XML metadata is not well-formed. You can add breakpoints to your code to determine if the plug-in is invoked when IWorksTest is started.

Verifying your XML metadata

To review your XML metadata:

1. Review your XML metadata in the tree hierarchy. To limit the sections that are shown, select the section you want to see from the following buttons:
   - All
   - Device
   - Version
   - Command
2. To save a copy of your XML metadata to a file:
   a. Click **Dump MetaData**.
   b. Enter a **File name** with an .xml extension so you can open the file in an XML editor or a Web browser.
   c. Click **Save**.
3. If you have provided a 32x32 bitmap icon for the device, it should be displayed on the right side of the window.

Testing device initialization

To test the device initialization:

1. Select the **Device** entry in the **MetaData** area.
2. Click **Initialize**.

   This calls the Initialize method. If the device is powered on, you can verify that your plug-in 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 plug-in's ability to handle that error condition.

Testing the diagnostics dialog

To test the diagnostics dialog box for your device:

1. Select a user privilege from the menu under the **Diags** button.
2. Click **Diags**.

   This calls the ShowDiagsDialog method. By selecting different user privileges, you can test the behavior of the diagnostics dialog box under different security levels without having to log in as different user in VWorks software.

Testing commands

To test individual commands:

1. Select the command you want to test in the **MetaData** area.
2. Click **Command**.

   The Command method is called with a command XML block from the command selected in the MetaData area.

Testing IRobotDriver methods

If your plug-in implements the IRobotDriver interface, you can check two methods using IWorksTest.

To test IRobotDriver methods:

1. In the **MetaData** area, select the method you want to test:
   - Check plate presence

     Note: This method will be available in a future release.
   - Move plate
2. Click **Command**.

   The selected method (CheckPlatePresent or Move) is called.

Related topics

For information about...	See...
Device initialization method	“Initialize method” on page 60
Device bitmap icon method	“Get32x32Bitmap method” on page 53
Perform command method	“Command method” on page 49
Display diagnostics dialog box method	“ShowDiagsDialog method” on page 70
Move plate method	“Move method” on page 137
Testing your VHooks plug-in	“Testing your VHooks plug-in” on page 197

Testing your VHooks plug-in

About this topic

This topic describes a way to test your VHooks plug-in.

Testing your plug-in

To test your new plug-in:

1. In your project code for the ProtocolStarted method, add the following code.

Programming language	Code

Visual Basic	`Public Sub ProtocolStarted(ByVal sXML As String, byRef sResultXML As String)`
	
	`  MsgBox("Hello from VWorks VHooks Plug-in")`
	
	`End Sub`

C#	`Public void ProtocolStarted(String sXML, Ref String sResultXML)`
	`{`
	
	`  System.Windows.Forms.MessageBox.Show("Hello from VWorks VHooks Plug-in");`
	
	`}`

1. Compile the code and copy the project DLL, TLB, and the interop DLL files to the C:\ *VWorks software_install_dir*\plugins folder.

   Note: For Visual Basic, you do not need to copy the interop DLL file to the VWorks software plugins folder. For C#, the interop DLL file must be copied to the plugins folder.

   Note: To save time during testing, you can temporarily move all the other plug-ins from the VWorks software plugins folder. Move the contents of the VWorks software plugins folder to a temporary location such as plugins_backup. This speeds up the plug-in loading process in VWorks software and reduces message in the log file so it is easier to see the messages related to your new plug-in.

   !! IMPORTANT !! Restore your original plug-in files from the temporary location to the VWorks plugins folder when you have finished testing.

2. Run VWorks software.
3. Check the log to verify that your plug-in loads successfully.

   If you do not see your plug-in 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 in the VWorks software folder and not in the VWorks software plugins folder.

4. Create a simple protocol and run it.

   You should see a message box open when the protocol starts with the message from your plug-in.

5. To launch your plug-in interface, in VWorks software choose **Tools > Show Plugins**.

Related topics

For information about...	See...
VHooks software interface methods	“Method list” on page 154
XML paramters	“VHooks XML parameters” on page 153

Index

Note: You can search our technical documentation on our website at www.velocity11.com/support/support.html.

A

Abort method, *49*
Aborted method (VHooks), *156*
AbortExecution VHooks return XML, *154*
AbortMessage VHooks return XML, *154*
API
  VWorks Device Driver, *3*, *13*, *45*
  VWorks Event Monitoring, *3*, *14*, *151*

B

bar code
  query, *75*, *77*
  query response, *77*
  scanner query, *95*
  update, *102*
BarCodeMisread method, *159*
BarCodeRead method, *157*
bitmap icon, *53*

C

<Channels> element, *33*
CheckBarCode method, *136*
CheckPlatePresent method, *136*
<Command> element, *40*
command icon, *53*
Command method, *49*
  testing, *196*
Command XML block, *42*
<Commands> element, *39*
Compile method, *50*
CompileComplete method (VHooks), *161*
CompilerErrors XML block, *51*
CompileType values, *50*
ControllerQuery method, *52*
custom menu update, *105*
CustomMenuClick method, *162*

D

Deadlock method, *164*
debugging
  device driver plug-in, *8*, *194*
  VHooks plug-in, *197*
Delid method, *136*
device
  diagnostics, *6*
  driver plug-in testing, *194*
  icon, *53*
  locations, *5*
  properties, *5*
  show diagnostics, *70*
device driver
  defined, *2*
<Device> element, *27*
device manager modified query, *89*
Device XML block, *42*
DeviceLocationTeachpoints query, *75*, *79*
devices
  defined, *4*

E

error handling (IWorks)
  Abort method, *49*
  GetErrorInfo method, *55*
  Ignore method, *59*
  Retry method, *69*
  return codes, *47*
Error method (VHooks), *164*
ErrorType values, *51*

F

FileOpened method, *166*
FileSaved method, *167*
fill plate task support, *133*

G

Get32x32Bitmap method, *53*
  testing, *195*
GetDescription method, *54*
GetErrorInfo method, *55*
GetHeadChange method, *124*
GetJavascriptVariable query, *75*, *82*
GetLayoutBitmap method, *57*
GetMetaData method, *56*
  metadata XML example, *23*
  MetaDataType values, *56*
GetProfileHeadType method, *126*
GetRunSetStatus query, *75*
GetSimulationTimes method, *136*
GetSpeed method, *136*
GetStorageLocations method, *140*
GetTaskType method, *127*
GetTeachPoints method, *137*
GetTipChange method, *129*

GetUserInterface method, *168*
GetVolumeChange method, *131*

H

head change. *see* GetHeadChange method
head type. *see* GetProfileHeadType method
human robot, *8*

I

icon, device or command, *53*
IControllerClient interface, *71*
    SetController, *73*
Ignore method, *59*
ImplementsPipetteWizards method, *133*
Initialize method, *60*
    testing, *195*
inter plug-in communication, *52*, *86*
InterPlugin query, *75*
InventoryPlateBarcodes update, *102*, *103*
IPipetteDriver interface, *13*, *123*
    GetHeadChange, *124*
    GetProfileHeadType, *126*
    GetTaskType, *127*
    GetTipChange, *129*
    GetVolumeChange, *131*
    ImplementsPipetteWizards, *133*
    list of methods, *124*
IRobotDriver interface, *13*, *135*
    CheckBarCode, *136*
    CheckPlatePresent, *136*
    Delid, *136*
    GetSimulationTimes, *136*
    GetSpeed, *136*
    GetTeachPoints, *137*
    list of methods, *136*
    Move, *137*
    PushdownAtLastPlaceLocation, *136*
    Relid, *136*
    SetSpeed, *136*
    Stir, *136*
IsDeviceManagerDirty query, *75*
IsLocationAvailable method, *61*
IsStackEmpty method, *116*
IsStackFull method, *117*
IStackerDriver interface, *13*, *115*
    IsStackEmpty, *116*
    IsStackFull, *117*
    list of methods, *116*
    LoadStack, *118*
    SinkPlate, *119*
    SourcePlate, *120*
    UnloadStack, *121*
IStorageDriver interface, *13*, *139*
    GetStorageLocations, *140*
    list of methods, *140*
    LoadPlate, *143*
    LookupLocations, *145*
    QueryStorageLocations, *147*
    UnloadPlate, *148*
IWorksController interface, *13*, *71*
    PrintToLog, *73*
    Query, *74*
    Update, *101*
IWorksDriver interface, *13*, *45*
    Abort, *49*
    Command, *49*
    Compile, *50*
    ControllerQuery, *52*
    Get32x32Bitmap, *53*
    GetDescription, *54*
    GetErrorInfo, *55*
    GetLayoutBitmap, *57*
    GetMetaData, *56*
    Ignore, *59*
    Initialize, *60*
    IsLocationAvailable, *61*
    list of methods, *46*
    MakeLocationAvailable, *62*
    PlateDroppedOff, *65*
    PlatePickedUp, *66*
    PlateTransferAborted, *67*
    PrepareForRun, *68*
    Retry, *69*
    ShowDiagsDialog, *70*
IWorksTest, *194*

J

JavaScript
    RunScript update, *107*
    ScriptPlateError method, *187*
    variable query, *81*
    variable query response, *82*

L

labware
    defined, *6*
    query, *90*
    query response, *90*
Labware query, *75*
LiquidTransferComplete method, *168*
LoadPlate method, *143*
LoadStack method, *118*
<LocationAvailable> element, *63*
<Location> element, *29*
LocationAvailable XML block, *62*
<Locations> element, *28*

LocationToTeachpoints query, *75*, *93*
LocationVector XML block, *145*, *147*
LogMessage VHooks return XML, *153*
LookupLocations method, *145*

M

MakeLocationAvailable method, *62*
md5sum attribute, *25*
menu click, custom (VHooks), *162*
Menu update, *102*, *105*
<MetaData> element, *26*
metadata elements
  <Command>, *40*
  <Commands>, *39*
  <Device>, *27*
  <Location>, *29*
  <Locations>, *28*
  <MetaData>, *26*
  <Parameter>, *31*
  <Parameters>, *31*
  <Plates>, *43*
  <Range>, *35*
  <Ranges>, *35*
  <StorageDimensions>, *36*
  <Velocity11>, *25*
  <Version>, *38*
  <Versions>, *38*
MetaDataType values, *56*
method lists
  IPipetteDriver, *124*
  IRobotDriver, *136*
  IStackerDriver, *116*
  IStorageDriver, *140*
  IWorksDriver, *46*
  VHooks, *154*
Move method, *137*

O

online help, *9*
opening a protocol, *7*

P

<Parameter> element, *31*
<Parameters> element, *31*
PauseExecution VHooks return XML, *153*
PauseMessage VHooks return XML, *153*
PDF guide, *9*, *10*
pipette methods (IWorks)
  GetHeadChange method, *124*
  GetProfileHead method, *126*
  GetTaskType method, *127*
  GetTipChange method, *129*
  GetVolumeChange method, *131*
  ImplementsPipetteWizards method, *133*
pipette methods (VHooks)
  PipetProcessFinished method, *171*
  PipetProcessStarting method, *173*
  PipetTaskFinished method, *174*
  PipetTaskStarting method, *176*
pipette process task, *7*
pipette task, *7*
pipette wizard. *see* ImplementsPipetteWizards method
PipetteHead XML block, *125*
PipetteTaskQuery XML block, *127*
PipetteWizardQuery XML block, *133*
plate information
  query, *97*
  query response, *98*
PlateDroppedOff method, *65*
PlateFlagsType values, *47*
PlateGroupMapping method, *177*
PlateInfoXML block, *43*
  <Plates> element, *43*
  example, *43*
PlatePickedUp method, *66*
<Plates> element, *43*
PlateTransferAborted method, *67*
plug-in (device driver)
  testing, *194*
plug-in (VHooks)
  defined, *3*
  testing, *197*
plug-in to plug-in communication, *52*, *86*
plug-in, defined, *2*
plug-in, location for, *4*, *18*
PrepareForRun method, *68*
PrintToLog method, *73*
process
  defined, *6*
  pipette, *7*
ProcessFinished method, *178*
ProcessStarting method, *180*
profile, *6*
protocol
  compiling, *7*
  defined, *6*
  opening, *7*
  running, *8*
  saving, *7*
ProtocolFinished method, *181*
ProtocolStarted method, *182*
PushdownAtLastPlaceLocation method, *136*

Q

<Quadrant> element, *33*

Query method, *74*
    Barcode, *75*, *77*
    DeviceLocationTeachpoints, *75*, *79*
    GetJavascriptVariable, *75*, *81*
    GetRunSetStatus, *75*, *84*
    InterPlugin, *75*, *86*
    IsDeviceManagerDirty, *75*, *89*
    Labware, *75*, *90*
    list of types, *75*
    LocationToTeachpoints, *75*, *93*
    ScanBarcode, *75*, *95*
    SystemPlateInformation, *75*, *97*
    TeachpointInformation, *75*, *99*
query response
    Barcode, *77*
    DeviceLocationTeachpoints, *79*
    GetJavascriptVariable, *82*
    GetRunSetStatus, *84*
    InterPlugin, *87*
    IsDeviceManagerDirty, *89*
    Labware, *90*
    LocationToTeachpoints, *94*
    ScanBarcode, *96*
    SystemPlateInformation, *98*
    TeachpointInformation, *99*
QueryStorageLocations method, *147*

R

RackInfo update, *102*, *106*
<Range> element, *35*
<Ranges> element, *35*
Relid method, *136*
Retry method, *69*
ReturnCode values, *47*
RobotMove method, *184*
RobotPickComplete method, *185*
RobotPlaceComplete method, *186*
running a protocol, *8*
RunScript update, *102*, *107*
run-set
    status query, *75*, *84*
    status query response, *84*
    update, *102*, *108*

S

saving a protocol, *7*
scan bar code
    query, *95*
    query response, *96*
ScanBarcode query, *75*
ScriptPlateError method, *187*
SecurityLevel values, *70*
serial dilution task, *133*
SetController method, *73*
SetSpeed method, *136*
ShowDiagsDialog method, *70*
    SecurityLevel values, *70*
    testing, *195*
SinkPlate method, *119*
SourcePlate method, *120*
StackHeight update, *102*, *110*
starting IWorksTest, *194*
Stir method, *136*
<StorageDimensions> element, *36*
StorageDimensions XML block, *140*
<StorageLocation> element, *63*
StorageLocation XML block, *143*
system plate information
    query, *97*
    query response, *98*
SystemPlateInformation query, *75*

T

TaskFinished method, *188*
tasks
    defined, *6*
    provide description, *54*
tasks (pipette)
    defined, *7*
    provide type, *127*
TaskStarting method, *190*
TeachpointInformation query, *75*
teachpoints
    information query, *75*, *99*
    information query response, *99*
    query for all locations, *75*, *79*
    query for location, *75*, *93*
    query response for all locations, *79*
    query response for location, *94*
TermSuccess update, *102*, *111*
testing, *194*
    Command method, *196*
    device driver plug-in, *8*, *194*
    Get32x32Bitmap method, *195*
    Initialize method, *195*
    ShowDiagsDialog method, *195*
    VHooks plug-in, *197*
tip change. *see* GetTipChange method
TipChange XML block, *129*
transfer liquid complete (VHooks), *168*

U

UnloadPlate method, *148*
UnloadStack method, *121*
Update method, *101*
    Barcode, *102*

InventoryPlateBarcodes, *102*, *103*
list of types, *102*
Menu, *102*, *105*
RackInfo, *102*, *106*
RunScript, *102*, *107*
RunsetAdd, *102*, *108*
StackHeight, *102*, *110*
TermSuccess, *102*, *111*
Volume, *102*, *111*
UserLoggedIn method, *191*
UserLoggedOut method, *192*

V

<Velocity11> element, *25*
md5sum attribute, *25*
<Version> element, *38*
<Versions> element, *38*
VHooks interface
Aborted, *156*
BarCodeMisread, *159*
BarCodeRead, *157*
CompileComplete, *161*
CustomMenuClick, *162*
Deadlock, *164*
Error, *164*
FileOpened, *166*
FileSaved, *167*
GetUserInterface, *168*
LiquidTransferComplete, *168*
list of methods, *154*
PipetProcessFinished, *171*
PipetProcessStarting, *173*
PipetTaskFinished, *174*
PipetTaskStarting, *176*
PlateGroupMapping, *177*
ProcessFinished, *178*
ProcessStarting, *180*
ProtocolFinished, *181*
ProtocolStarted, *182*
RobotMove, *184*
RobotPickComplete, *185*
RobotPlaceComplete, *186*
ScriptPlateError, *187*
TaskFinished, *188*
TaskStarting, *190*
UserLoggedIn, *191*
UserLoggedOut, *192*
VHooks plug-in testing, *197*
VHooks XML output attributes, *154*
AbortMessage, *154*
LogMessage, *153*
PauseExecution, *153*
PauseMessage, *153*
volume change. *see* GetVolumeChange method
Volume update, *102*, *111*
VolumeChange XML block, *131*
VWorks Device Driver API, *3*, *13*, *45*
VWorks Event Monitoring API, *3*, *14*, *151*

W

Wells Selection XML block, *33*

X

XML block, *42*
Command, *42*
CompilerErrors, *51*
defined, *20*
Device, *42*
LocationAvailable, *62*
LocationVector, *145*, *147*
PipetteHead, *125*
PipetteTaskQuery, *127*
PipetteWizardQuery, *133*
PlateInfo, *43*
StorageDimensions, *140*
StorageLocation, *143*
TipChange, *129*
VolumeChange, *131*
Wells Selection, *33*
XML elements
<Command>, *40*
<Commands>, *39*
<Device>, *27*
<Location>, *29*
<Locations>, *28*
<MetaData>, *26*
<Parameter>, *31*
<Parameters>, *31*
<Plates>, *43*
<Range>, *35*
<Ranges>, *35*
<StorageDimensions>, *36*
<Velocity11>, *25*
<Version>, *38*
<Versions>, *38*
XML parameter string (VHooks), *153*

Agilent Technologies

Developer Guide

G5415-90007