

VWorks3 Software

ActiveX Guide

Notices

© Agilent Technologies, Inc. 2009

No part of this manual may be reproduced in any form or by any means (including electronic storage and retrieval or translation into a foreign language) without prior agreement and written consent from Agilent Technologies, Inc. as governed by United States and international copyright laws.

User Guide Part Number

G5415-90059

April 2009

Contact Information

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

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

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

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

Documentation feedback:
documentation.automation@agilent.com

Web:
www.agilent.com/lifesciences/automation

Acknowledgements

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 sub-contract, 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

(June 1987) or DFAR 252.227-7015 (b)(2) (November 1995), as applicable in any technical data.

Safety Notices

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

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



VWorks ActiveX control

This guide contains the following topics:

- “About this guide” on page 2
- “About the VWorks ActiveX control” on page 3
- “Starting the VWorks software” on page 5
- “Methods” on page 6
- “Properties” on page 30
- “Events” on page 30
- “Enumerated types” on page 35

About this guide

Who should read this guide

This user guide is for VWorks software users who are integrating third-party applications with the VWorks software.

Read this topic if you have administrator or technician privileges. This guide assumes that you know how to write programs in C# and have basic programming knowledge.

What this guide covers

This guide explains how to use VWorks ActiveX control to enable third-party applications to interface with the VWorks software, version 3.0.

Where to find user information

You can search the online knowledge base or download the latest version of any PDF file from the Agilent Technologies website at www.agilent.com/lifesciences/automation.

This ActiveX control guide should be used in conjunction with the following user documents:

- *VWorks3 User Guide*
- *IWorks Device Driver Programming Interface and VWorks Hooks Interface Developer Guide*
- Automation Solutions device guides
- Third-party device driver guides

Related information

For information about...	See...
VWorks ActiveX control	“About the VWorks ActiveX control” on page 3
Starting the VWorks software through ActiveX control	“Starting the VWorks software” on page 5
VWorks ActiveX methods	“Methods” on page 6
VWorks ActiveX properties	“Properties” on page 30
VWorks ActiveX events	“Events” on page 30
VWorks ActiveX enumerated types	“Enumerated types” on page 35

About the VWorks ActiveX control

What is the VWorks ActiveX control

The VWorks ActiveX control is the software component that allows the VWorks software to interact with a third-party lab automation system.

How the VWorks ActiveX control is used

In an Agilent Technologies automation system, the VWorks software runs in standalone mode, and the ActiveX control is not used. However, some integrations, such as those with LIMS, require that a third-party application control the VWorks software. The VWorks ActiveX control enables third-party applications to interface with the VWorks software. Through the ActiveX control, the third-party application can create an instance of the VWorks software, schedule protocol runs, and display various VWorks dialog boxes.

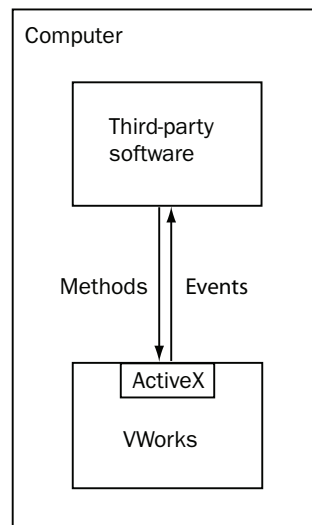
Each ActiveX control consists of a collection of the following:

- *Methods*. Functions that can be called to invoke individual operations
- *Properties*. Variables that are used in methods (for example, speed = fast)
- *Events*. Notifications that methods have completed or resulted in errors

For proper integration, you need to know the available methods and properties for the ActiveX control.

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

Note: Although the VWorks Active X control generates events, the third-party application must implement handlers for them.



Integrating the VWorks ActiveX control

When integrating the VWorks ActiveX control in third-party software:

- 1** Install the VWorks ActiveX control. To install the VWorks ActiveX control:
 - a** Insert the VWorks software CD into the controlling computer CD-ROM drive.
 - b** In the CD folder, double-click VWorks Installer.exe.
 - c** Follow the directions in the installation wizard window.
- 2** To register the application program interface, run the VWorks software as a standalone application.
- 3** Refer to the description of the methods, properties, and events in this guide.

Related information

For information about..	See...
Starting the VWorks software through ActiveX control	“Starting the VWorks software” on page 5
VWorks ActiveX methods	“Methods” on page 6
VWorks ActiveX properties	“Properties” on page 30
VWorks ActiveX events	“Events” on page 30
VWorks ActiveX enumerated types	“Enumerated types” on page 35

Starting the VWorks software

Procedure

You can start the VWorks software using the VWorks ActiveX control. The following C# example shows how to start the VWorks software.

```
private VWorksXMLLib.ApplicationClass vApp = new  
VWorksXMLLib.ApplicationClass();
```

The C# project must include a reference to the VWorksXML 1.0 Type Library COM component before creating a new instance of `VWorksXMLLib.ApplicationClass()`.

If the software is already running, the preceding call will cause an exception: `System.Runtime.InteropServices.COMException`.

Related information

For information about..	See...
Starting the VWorks software through ActiveX control	“Starting the VWorks software” on page 5
VWorks ActiveX methods	“Methods” on page 6
VWorks ActiveX properties	“Properties” on page 30
VWorks ActiveX events	“Events” on page 30
VWorks ActiveX enumerated types	“Enumerated types” on page 35

Methods

AbortProtocol

Description

Aborts the protocol run that is in progress.

Parameters

None

Returns

Name	Type	Description
returnCode	V11ReturnCode	See “Enumerated types” on page 35.

C# example

```
vApp.AbortProtocol(out retcode);  
if (VWorksXMLLib.RetCode.RETURN_SUCCESS != retcode){  
    MessageBox.Show("VWorks [AbortProtocol] function failed!");  
}
```

AddRunsetXML

Description

Adds protocols to a runset using the specified XML file.

A runset is a collection of runs that are scheduled in advance to run without operator intervention. The VWorks Runset Manager enables scheduling of a series of runs using different protocols. See the *VWorks3 User Guide* or search the [knowledge base](#) for more information on runsets.

Parameters

Name	Type	Description
sRunsetXML	BSTR	Gets an XML block of information that specifies the runs to be added to the Runset Manager. See schema and examples below.

Returns

Name	Type	Description
returnCode	V11ReturnCode	See “Enumerated types” on page 35. The returnCode is RETURN_SUCCESS if the entries were successfully added to the Runset Manager, or RETURN_FAIL if the XML was invalid or if an invalid value was provided.

C# example

```

VWorksXMLLib.RetCode retcode;
string fileName = "C:\\somerunsetfile.rst";
StreamReader streamReader = new StreamReader(fileName);
try{
    string strXML = streamReader.ReadToEnd();
    //At this point, strXML has XML for processing.

    //Call AddRunsetXML with contents of specified runset file.
    vApp.AddRunsetXML(strXML, out retcode);
    if (VWorksXMLLib.RetCode.RETURN_SUCCESS != retcode){
        MessageBox.Show("VWorks [AddRunsetXML] function failed!");
    }
}finally{
    streamReader.Close();
}

```

Example XML (sRunsetXML) to add two protocols to a runset

```

<?xml version='1.0' encoding='us-ascii' ?>
<runset_data file='' md5sum='b2138525cefa2114db0c2e8d35c32fee' version='1.0' >
  <runset_entry Cycles='2' Day='8' Hour='16' Minute='26' Month='12' Plugin=''
  Protocol='C:\VWorks Workspace\protocols\jsTasks.pro' Second='28' Status='Scheduled'
  Year='2020'>
    <notes>
      <barcodes group="groupname1" side='1'>
        <barcode>SRC0001</barcode>
        <barcode>SRC0002</barcode>
      </barcodes>
      <barcodes group="groupname2" side='1'>
        <barcode>DST0001</barcode>
        <barcode>DST0002</barcode>
      </barcodes>
    </notes>
  </runset_entry>

  <runset_entry Cycles='3' Day='8' Hour='16' Minute='26' Month='12' Plugin=''
  Protocol='C:\VWorks Workspace\protocols\jsTasks2.pro' Second='31' Status='Scheduled'
  Year='2020'>
    </runset_entry>
</runset_data>

```

Note: Side is specified by number and has the following mapping.

SOUTH-0
 WEST-1
 EAST-2
 NORTH-3

XML schema

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="runset_data">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" name="runset_entry">
          <xs:complexType mixed="true">
            <xs:sequence minOccurs="0" maxOccurs="1">
              <xs:element minOccurs="0" maxOccurs="1" name="notes">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element minOccurs="0" maxOccurs="unbounded" name="barcodes">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element minOccurs="0" maxOccurs="unbounded"
name="barcode" type="xs:string" />
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                    <xs:attribute name="group" type="xs:string" use="optional" />
                    <xs:attribute name="side" type="xs:string" use="optional" />
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="Cycles" type="xs:string" use="required" />
      <xs:attribute name="Day" type="xs:string" use="required" />
      <xs:attribute name="Hour" type="xs:string" use="required" />
      <xs:attribute name="Minute" type="xs:string" use="required" />
      <xs:attribute name="Month" type="xs:string" use="required" />
      <xs:attribute name="Plugin" type="xs:string" use="required" />
      <xs:attribute name="Protocol" type="xs:string" use="required" />
      <xs:attribute name="Second" type="xs:string" use="required" />
      <xs:attribute name="Status" type="xs:string" use="required" />
      <xs:attribute name="Year" type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>
</xs:sequence>
<xs:attribute name="file" type="xs:string" use="required" />
<xs:attribute name="md5sum" type="xs:string" use="required" />
<xs:attribute name="version" type="xs:string" use="required" />
</xs:complexType>
</xs:element>
</xs:schema>
```

Note: The Notes node is optional and the VWorks software ignores it. This XML block can be retrieved from within the protocol execution through the JavaScript API.

AppendRunset

Description

Appends the specified file to the runset file. Existing data is retained in the runset file.

A runset is a collection of runs that can be scheduled in advance to run without operator intervention. Each run in the runset can use a different protocol. See the *VWorks3 User Guide* or search the [knowledge base](#) for more information on runsets.

Parameters

Name	Type	Description
sRunset	BSTR	The file path to the runset file (*.rst).

Returns

Name	Type	Description
returnCode	V11ReturnCode	See “Enumerated types” on page 35. <i>Note:</i> returnCode is RETURN_SUCCESS if the XML is valid. However, a protocol might fail to get scheduled if the actual data is invalid, for example, the protocol specified does not exist. In this case, a failure will occur at the scheduled run time.

C# example

```
string strRunsetPath = "C:\\VWorks Workspace\\runsets\\runset.rst";
vApp.AppendRunset(strRunsetPath, out retcode);
if (VWorksXMLLib.RetCode.RETURN_SUCCESS != retcode){
    MessageBox.Show("VWorks [AppendRunset] function failed!");
}
```

ClearRunset

Description

Clears all runset entries from the Runset Manager.

ClearRunset fails if a protocol is running. In this case, you can use AbortProtocol to cancel the protocol before you call ClearRunset.

Parameters

None

Returns

Name	Type	Description
returnCode	V11ReturnCode	See “Enumerated types” on page 35.

C# example

```
VWorksXMLLib.RetCode retcode;  
vApp.ClearRunset(out retcode);  
if (VWorksXMLLib.RetCode.RETURN_SUCCESS != retcode){  
    MessageBox.Show("VWorks [ClearRunset] function failed!");  
}
```

CompileProtocol

Description

Compiles the currently open protocol and is used with the LogMessage event. CompileProtocol checks the syntax and logic of the protocol and associated JavaScript.

Parameters

None.

Returns

Name	Type	Description
iErrors	LONG*	The number of errors found. Compiler errors are logged, and the VWorks software generates a MessageBoxAction event with the following message: XX errors found in protocol. It is not recommended that this protocol be run. Continue anyway? If the client sets actionToTake = 6 (Yes), the protocol is executed. If the client sets actionToTake = 7 (No), the protocol is aborted and the next protocol in the Runset Manager, if any, is executed at its scheduled time.
IWarnings	LONG*	The number of warnings found.
returnCode	V11ReturnCode	See “Enumerated types” on page 35.

Note: For details on the MessageBoxAction event, see “MessageBoxAction” on page 32.

C# example

```
int iNumErrors = 0;
int iNumWarnings = 0;
vApp.CompileProtocol(out iNumErrors, out iNumWarnings, out
retcode);
if (VWorksXMLLib.RetCode.RETURN_SUCCESS != retcode){
    MessageBox.Show("VWorks [CompileProtocol function failed!];
}
MessageBox.Show("compiled: Errors:" + iNumErrors + "Warnings: " +
iNumWarnings);
```

CustomHook

Description

For Agilent use only.

Parameters

Name	Type	Description
sIn	BSTR	For Agilent use only

Returns

Name	Type	Description
returnCode	V11ReturnCode	See “Enumerated types” on page 35.

EnableRunsetManager

Description

Enables and disables the Runset Manager.

The VWorks Runset Manager enables you to schedule a series of protocols that can be run without operator intervention. See the *VWorks3 User Guide* or search the [knowledge base](#) for more information on runsets.

Parameters

Name	Type	Description
bEnable	VARIANT_BOOL	The value that indicates the state: <ul style="list-style-type: none"> • True = Enabled. • False = Disabled.

Returns

Name	Type	Description
returnCode	V11ReturnCode	See “Enumerated types” on page 35.

C# example

```
bool bEnable = true;
vApp.EnableRunsetManager(bEnable, out retcode);
if (VWorksXMLLib.RetCode.RETURN_SUCCESS != retcode){
    MessageBox.Show("VWorks [EnableRunsetManager] function
failed!");
}
```

EnumerateUsers

Description

Returns the list of users who have VWorks accounts.

Parameters

None.

Returns

Name	Type	Description
user	VARIANT*	An array containing the user names.
returnCode	V11ReturnCode	See “Enumerated types” on page 35.

C# example

```
object oUsers;
vApp.EnumerateUsers(out oUsers, out retcode);
if (VWorksXMLLib.RetCode.RETURN_SUCCESS != retcode){
    MessageBox.Show("VWorks [EnumerateUsers] function failed!");
}
System.String[] sUsers= (System.String[])oUsers;
foreach (System.String sUser in sUsers){
    MessageBox.Show(sUser);
}
```

GetRunsetProtocolsInfo

Description

Returns the XML block that corresponds to information currently in the Runset Manager.

See the *VWorks3 User Guide* or search the [knowledge base](#) for more information on runsets and the Runset Manager.

Parameters

None

Returns

Name	Type	Description
sRunsetXML	BSTR	Returns an XML block of information from the Runset Manager.
returnCode	V11ReturnCode	See “Enumerated types” on page 35.

C# example

```
string strInfo;
VWorksXMLLib.RetCode retcode;
vApp.GetRunsetProtocolsInfo(out strInfo, out retcode);
if (VWorksXMLLib.RetCode.RETURN_SUCCESS != retcode) {
    MessageBox.Show("VWorks [GetRunsetProtocolsInfo] function
        failed!");
} else {
    MessageBox.Show("GetRunsetProtocolsInfo returned: " + strInfo);
}
```

Example of XML block

```
<?xml version='1.0' encoding='ASCII' ?>
<runset_data
file=''
md5sum='b9adb11c1c9242b83d34b333e1f0da1f' version='1.0' >

<runset_entry
Cycles='2'
Day='5' Hour='11'
JobID='483cb376-5dcc-406f-ba07-d1ed3b911764'
Minute='38'
Month='12'
Plugin=''
Protocol='C:\VWorks Workspace\protocols\jsTasks.pro'
Second='5'
Status='Running'
Year='2008' />

<runset_entry
Cycles='3'
Day='5' Hour='11'
JobID='025c0bda-10ca-4676-826b-d70807546272'
Minute='38'
Month='12'
Plugin=''
Protocol='C:\VWorks Workspace\protocols\jsTasks2.pro'
Second='8'
Status='Scheduled'
Year='2008' />

</runset_data>
```

XML attribute descriptions

XML attribute	Description
JobID	ID, generated by the Runset Manager, that uniquely identifies the job in the Runset Manager. The JobID is used in <code>RemoveFromRunset</code> to identify the scheduled protocol to remove from runsets.
Protocol	Full path to the protocol being scheduled.
Cycles	Number of times the protocol should execute.
Year	YYYY year field of the scheduled time/date at which the protocol will execute.
Month	MM month field of the scheduled time/date at which the protocol will execute.
Day	DD day field of the scheduled time/date at which the protocol will execute.
Hour	hh hour field (24-hour clock) of the scheduled time/date at which the protocol will execute.
Minute	mm minute field of the scheduled time/date at which the protocol will execute.
Second	ss second field of the scheduled time/date at which the protocol will execute.
Plugin	Not used.
Status	Indicates the state of the protocol. Possible Status states are: <ul style="list-style-type: none"> • <i>Scheduled</i>. The protocol has been scheduled to run at a time that has not yet occurred. • <i>Pending</i>. The run has been paused. • <i>Running</i>. The protocol run is currently in progress. • <i>Completed</i>. The protocol has finished running. • <i>Aborted</i>. The run has been canceled, either manually or as the result of a non-recoverable error. • <i>Expired</i>. The time for the run has past, but the run has not yet started. The run must be rescheduled.
file	Name of the runset file, if applicable.
md5sum	Checksum value used to verify the runset XML.
version	Number that indicates the version of the XML schema.

Example of schema for XML returned by GetRunsetProtocolsInfo

```
<?xml version="1.0" encoding="ASCII" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="runset_data">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" name="runset_entry">
          <xs:complexType>
            <xs:attribute name="JobID" type="xs:string" use="required" />
            <xs:attribute name="Protocol" type="xs:string" use="required" />
            <xs:attribute name="Cycles" type="xs:string" use="required" />
            <xs:attribute name="Year" type="xs:string" use="required" />
            <xs:attribute name="Month" type="xs:string" use="required" />
            <xs:attribute name="Day" type="xs:string" use="required" />
            <xs:attribute name="Hour" type="xs:string" use="required" />
            <xs:attribute name="Minute" type="xs:string" use="required" />
            <xs:attribute name="Second" type="xs:string" use="required" />
            <xs:attribute name="Plugin" type="xs:string" use="required" />
            <xs:attribute name="Status" use="required">
              <xs:simpleType>
                <xs:restriction base="xs:string">
                  <xs:enumeration value="None" />
                  <xs:enumeration value="Expired" />
                  <xs:enumeration value="Scheduled" />
                  <xs:enumeration value="Pending" />
                  <xs:enumeration value="Running" />
                  <xs:enumeration value="Completed" />
                  <xs:enumeration value="Aborted" />
                </xs:restriction>
              </xs:simpleType>
            </xs:attribute>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="file" type="xs:string" use="required" />
      <xs:attribute name="md5sum" type="xs:string" use="required" />
      <xs:attribute name="version" type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>
</xs:schema>
```

LoadProtocol

Description

Loads the protocol for a run.

Parameters

Name	Type	Description
sProtocol	BSTR	The protocol file path.

Returns

Name	Type	Description
returnCode	V11ReturnCode	See “Enumerated types” on page 35.

C# example

```
vApp.LoadProtocol("C:\\Vworks
Workspace\\protocols\\MyProtocol.pro", out retcode);
```

LoadRunsetFile

Description

Opens the runset file, loads contents to the current runset, and removes the existing contents.

See the *VWorks3 User Guide* or search the [knowledge base](#) for more information on runsets.

Parameters

Name	Type	Description
sRunset	BSTR	The runset file path.

Returns

Name	Type	Description
returnCode	V11ReturnCode	See “Enumerated types” on page 35.

C# example

```
string strRunsetPath = "C:\\VWorks Workspace\\runsets\\runset.rst";
vApp.LoadRunsetFile(strRunsetPath, out retcode);
if (VWorksXMLLib.RetCode.RETURN_SUCCESS != retcode) {
    MessageBox.Show("VWorks [LoadRunsetFile] function failed!");
}
```

Login

Description

Logs into the VWorks software using the provided user name and password.

The VWorks software requires a login to run protocols. Client applications should always log in before making any VWorks method calls. For details on the login process, the types of user accounts, and administrator tasks, see the [VWorks3 User Guide](#) or search the [knowledge base](#).

Parameters

Name	Type	Description
userName	BSTR	The user name.
password	BSTR	The password.

Returns

Name	Type	Description
returnCode	V11ReturnCode	See “Enumerated types” on page 35.

C# example

```
vApp.Login("username", "password", out retcode);  
if (VWorksXMLLib.RetCode.RETURN_SUCCESS != retcode) {  
    MessageBox.Show("VWorks [Login] function failed!");  
}
```

Logout

Description

Logs out the current user session.

Parameters

None.

Returns

Name	Type	Description
returnCode	V11ReturnCode	See “Enumerated types” on page 35.

C# example

```
vApp.Logout(out retcode);
if (VWorksXMLLib.RetCode.RETURN_SUCCESS != retcode) {
    MessageBox.Show("VWorks [Logout] function failed!");
}
```

PauseProtocol

Description

Pauses the protocol run that is in progress. The tasks that are in progress will be finished. No new tasks will be started.

Parameters

None.

Returns

Name	Type	Description
returnCode	V11ReturnCode	See “Enumerated types” on page 35.

C# example

```
vApp.PauseProtocol(out retcode);
if (VWorksXMLLib.RetCode.RETURN_SUCCESS != retcode){
    MessageBox.Show("VWorks [PauseProtocol] function failed!");
}
```

ReinitializeDevices

Description

Reinitializes devices. For example, a device must be reinitialized if a different profile is selected or in the event of a communication error between the VWorks software and the device. For more details, see the the [VWorks3 User Guide](#) or search the [knowledge base](#).

Note: The ReinitializeDevices method returns immediately, so other methods may be called while the devices are initializing. However, the method will fail if called while a protocol is running.

Parameters

None

Returns

Name	Type	Description
returnCode	V11ReturnCode	See “Enumerated types” on page 35. <i>Note:</i> returnCode is RETURN_FAIL if the method is called when a protocol is running. The InitializationComplete event occurs when initialization of the system is complete.

Note: For details on the InitializationComplete event, see “InitializationComplete” on page 31.

C# example

```
vApp.ReinitializeDevices(out retcode);  
if (VWorksXMLLib.RetCode.RETURN_SUCCESS != retcode){  
    MessageBox.Show("VWorks [ReinitializeDevices] function  
    failed!");  
}
```

RemoveFromRunset

Description

Removes a scheduled protocol from a runset.

Currently running protocols are not affected. To remove a protocol that is running, you must first use AbortProtocol to cancel the protocol.

The scheduled protocol is identified by its job ID. In the C# example, the scheduled protocol to be removed has the job ID 483cb376-5dcc-406f-ba07-d1ed3b911764.

Note: To acquire a job ID, use [GetRunsetProtocolsInfo](#).

See the *VWorks3 User Guide* or search the [knowledge base](#) for more information on runsets.

Parameters

Name	Type	Description
sJobID	BSTR	The scheduled protocol in the runset.

Returns

Name	Type	Description
returnCode	V11ReturnCode	See “Enumerated types” on page 35. <i>Note:</i> returnCode is RETURN_SUCCESS if the job was successfully removed, or RETURN_FAIL if sJobID does not exist or refers to the currently running protocol.

C# example

```
VWorksXMLLib.RetCode retcode;  
string strID = "483cb376-5dcc-406f-ba07-d1ed3b911764";  
vApp.RemoveFromRunset(strID, out retcode);  
if (VWorksXMLLib.RetCode.RETURN_SUCCESS != retcode){  
    MessageBox.Show("VWorks [RemoveFromRunset] function failed!");  
}
```

ResumeProtocol

Description

Resumes the paused protocol run.

Note: This method is ignored if executed when the protocol is not paused.

Parameters

None.

Returns

Name	Type	Description
returnCode	V11ReturnCode	See “Enumerated types” on page 35.

C# example

```
VWorksXMLLib.RetCode retcode;
vApp.ResumeProtocol(out retcode);
if (VWorksXMLLib.RetCode.RETURN_SUCCESS != retcode){
    MessageBox.Show("VWorks [ResumeProtocol] function failed!");
}
```

RunProtocol

Description

Starts the run for the currently open protocol.

Note: This method is ignored if the protocol is already running.

Parameters

Name	Type	Description
IRuns	LONG	The number of plates to run.

Returns

Name	Type	Description
returnCode	V11ReturnCode	See “Enumerated types” on page 35.

C# example

```
vApp.RunProtocol(3, out retcode);
if (VWorksXMLLib.RetCode.RETURN_SUCCESS != retcode){
    MessageBox.Show("VWorks [RunProtocol] function failed!");
}
```

ShowDiagsDialog

Description

Displays the device diagnostics dialog box. This action is equivalent to using the Device diagnostics button in the VWorks software window.



Parameters

None

Returns

Name	Type	Description
returnCode	V11ReturnCode	See “Enumerated types” on page 35.

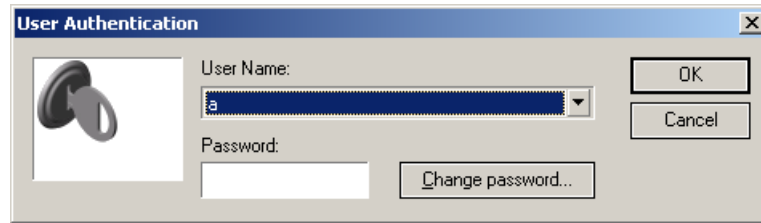
C# example

```
vApp.ShowDiagsDialog(out retcode);  
if (VWorksXMLLib.RetCode.RETURN_SUCCESS != retcode){  
    MessageBox.Show("VWorks [ShowDiagsDialog] function failed!");  
}
```

ShowLoginDialog

Description

Displays the User Authentication (or login) dialog box.



Parameters

None

Returns

Name	Type	Description
returnCode	V11ReturnCode	See “Enumerated types” on page 35.

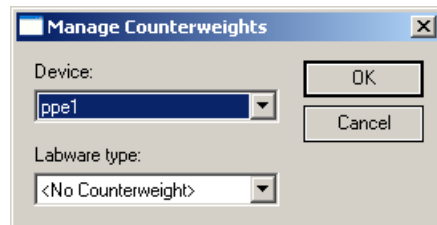
C# example

```
vApp.ShowLoginDialog(out retcode);
if (VWorksXMLLib.RetCode.RETURN_SUCCESS != retcode){
    MessageBox.Show("VWorks [ShowLoginDialog] function failed!");
}
```

ShowManageCounterweightsDialog

Description

Displays the Manage Counterweights dialog box.



Parameters

None

Returns

Name	Type	Description
returnCode	V11ReturnCode	See “Enumerated types” on page 35.

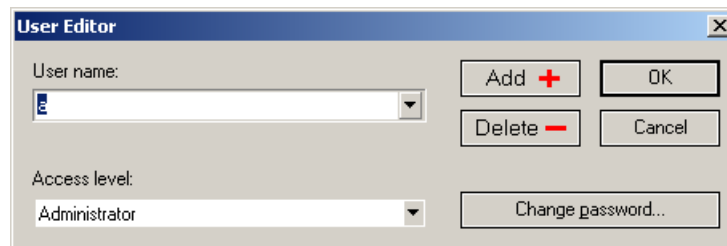
C# example

```
vApp.ShowManageCounterweightsDialog(out retcode);  
if (VWorksXMLLib.RetCode.RETURN_SUCCESS != retcode){  
    MessageBox.Show("VWorks [ShowManageCounterweightsDialog]  
    function failed!");  
}
```

ShowManageUserDialog

Description

Displays the User Editor dialog box.



Parameters

None

Returns

Name	Type	Description
returnCode	V11ReturnCode	See “Enumerated types” on page 35.

C# example

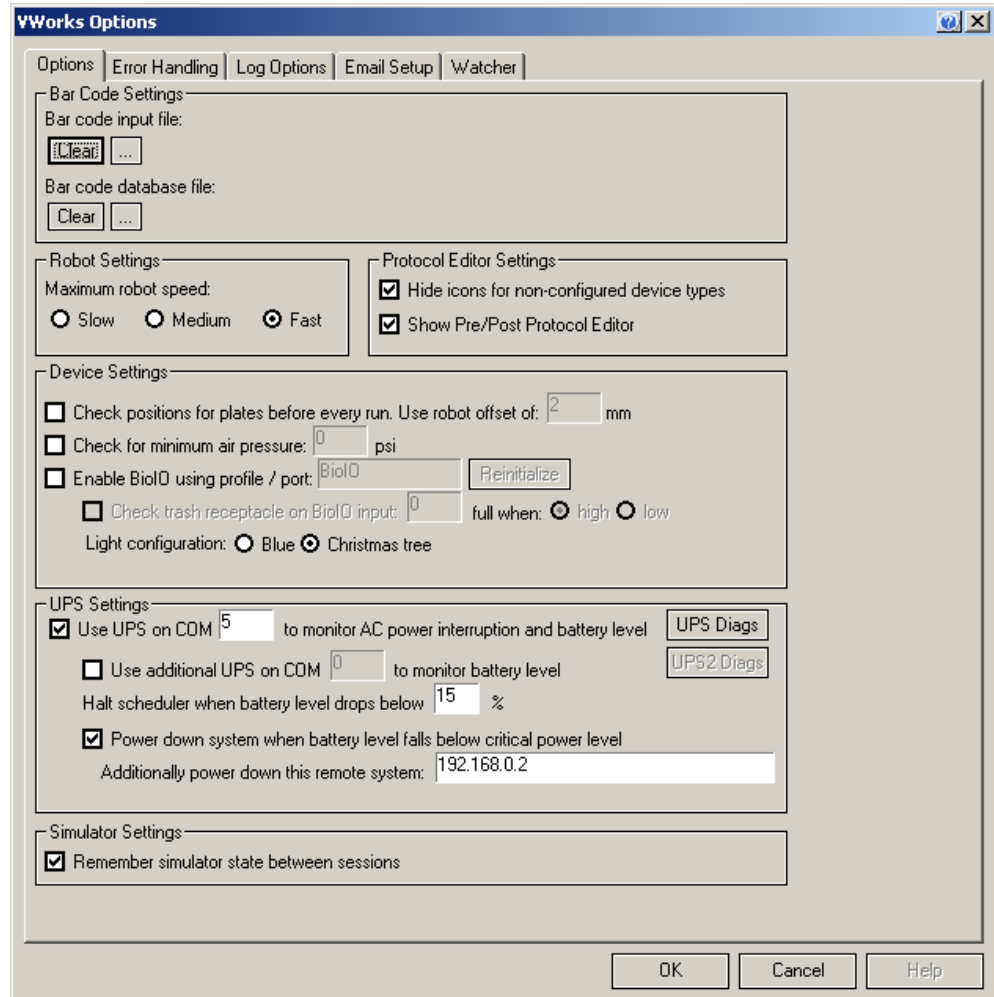
```
vApp.ShowManageUserDialog(out retcode);  
if (VWorksXMLLib.RetCode.RETURN_SUCCESS != retcode){  
    MessageBox.Show("VWorks [ShowManageUserDialog] function  
    failed!");  
}
```

ShowOptionsDialog

Description

Displays the VWorks Options dialog box.

Note: The dialog box cannot be displayed while a protocol is running, and the function will fail (RetCode != RETURN_SUCCESS) when the method is called.



Parameters

None

Returns

Name	Type	Description
returnCode	V11ReturnCode	See “Enumerated types” on page 35.

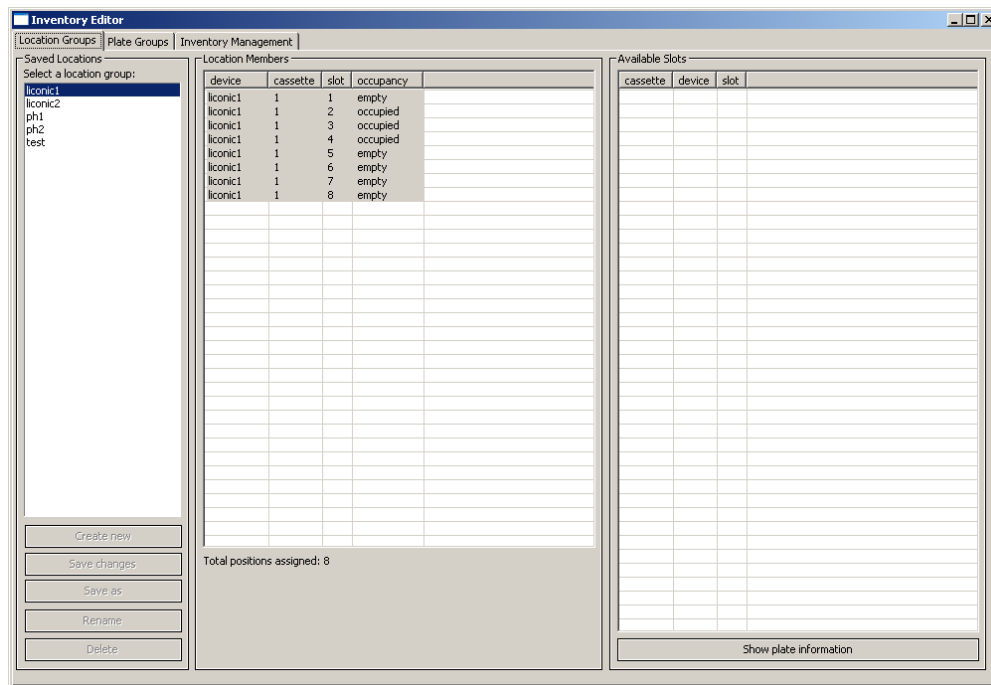
C# example

```
vApp.ShowOptionsDialog(out retcode);
if (VWorksXMLLib.RetCode.RETURN_SUCCESS != retcode){
    MessageBox.Show("VWorks [ShowOptionsDialog] function failed!");
}
```

ShowPlateGroupEditorDialog

Description

Displays the Inventory Editor dialog box, where the Plate Group tab can be accessed.



Parameters

None

Returns

Name	Type	Description
returnCode	V11ReturnCode	See “Enumerated types” on page 35.

C# example

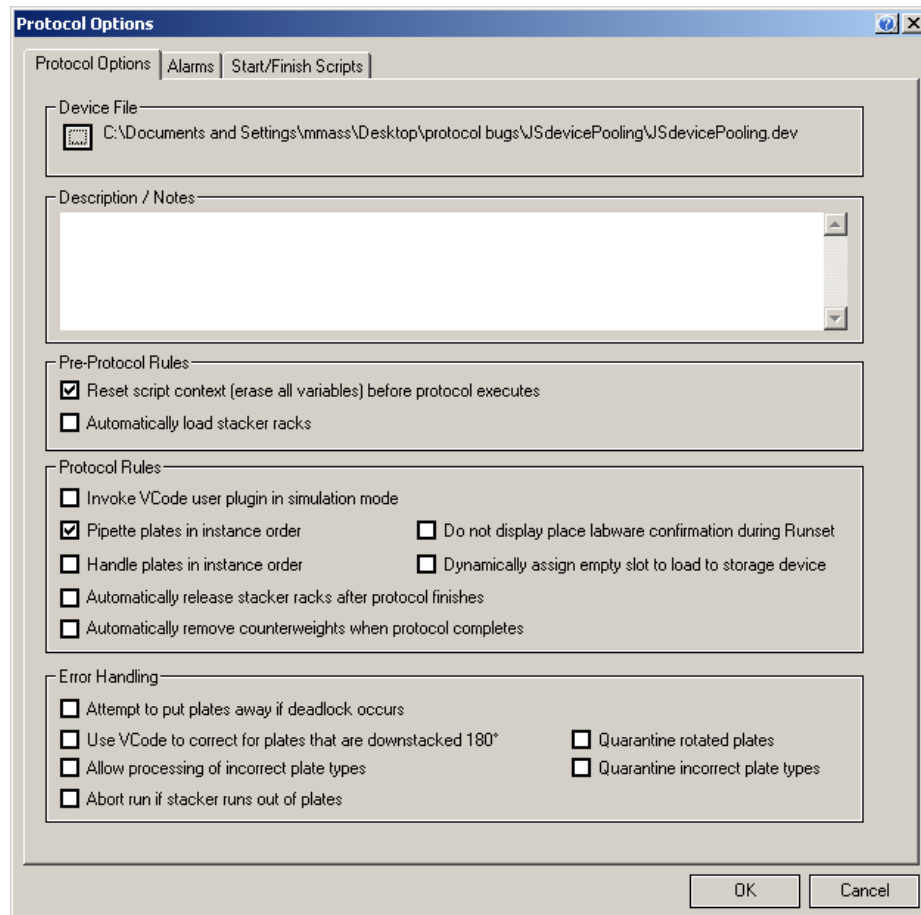
```
vApp.ShowPlateGroupEditorDialog(out retcode);
if (VWorksXMLLib.RetCode.RETURN_SUCCESS != retcode){
    MessageBox.Show("VWorks [ShowPlateGroupEditorDialog] function
failed!");
}
```

ShowProtocolOptionsDialog

Description

Displays the Protocol Options dialog box.

Note: The dialog box cannot be displayed while a protocol is running, and the function will fail (RetCode != RETURN_SUCCESS) when the method is called.



Parameters

None

Returns

Name	Type	Description
returnCode	V11ReturnCode	See “Enumerated types” on page 35.

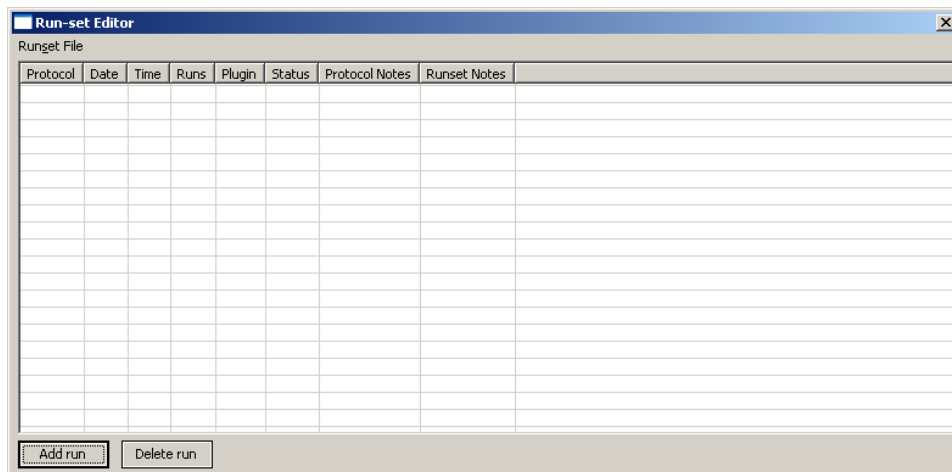
C# example

```
vApp.ShowProtocolOptionsDialog(out retcode);
if (VWorksXMLLib.RetCode.RETURN_SUCCESS != retcode){
    MessageBox.Show("VWorks [ShowProtocolOptionsDialog] function
failed!");
}
```

ShowRunsetEditorDialog

Description

Displays the Run-set Editor dialog box.



See the [VWorks3 User Guide](#) or search the [knowledge base](#) for more information on runsets.

Parameters

None

Returns

Name	Type	Description
returnCode	V11ReturnCode	See “Enumerated types” on page 35.

C# example

```
vApp.ShowRunsetEditorDialog(out retcode);
if (VWorksXMLLib.RetCode.RETURN_SUCCESS != retcode){
    MessageBox.Show("VWorks [ShowRunsetEditorDialog] function
failed!");
}
```

Related information

For information about...	See...
VWorks ActiveX control	“About the VWorks ActiveX control” on page 3
Starting the VWorks software ActiveX control	“Starting the VWorks software” on page 5
VWorks ActiveX properties	“Properties” on page 30
VWorks ActiveX events	“Events” on page 30
VWorks ActiveX enumerated types	“Enumerated types” on page 35

Properties

SimulationMode

Description

Gets or sets the simulation mode state:

- *True*. The simulation mode is on.
- *False*. The simulation mode is off.

C# example

```
vApp.SimulationMode = true;  
MessageBox.Show("Simulation mode = " + vApp.SimulationMode);
```

VWorksUIVisible

Description

Gets or sets whether the VWorks software window is hidden or displayed.

- *True*. The window is displayed.
- *False*. The window is hidden.

While the VWorks software is under COM control, the VWorks software window can be hidden. Use this property to determine whether the window is hidden.

C# example

```
vApp.VWorksUIVisible = true;  
MessageBox.Show("VWorksUIVisible = " + vApp.VWorksUIVisible);
```

Events

AbortProtocolComplete

Description

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

Parameters

None.

Returns

None.

InitializationComplete

Description

Occurs when initialization of the system is complete.

Parameters

None.

Returns

None.

LogMessage

Description

Occurs every time a message is posted to the log.

Parameters

Name	Type	Description
device	BSTR	The device name. An empty string is permitted.
loglevel	BSTR	The type of error: <ul style="list-style-type: none"> <i>Info</i>. General information. For example, System start up, or Simulation mode toggled on. <i>Event</i>. A software action. For example, Scheduler started, or Move plate. <i>Error</i>. An error that can stop the software and must be resolved. For example, Location incompatible with labware. <i>Warning</i>. An error that might permit the software to continue. For example, Task requires that tips be on the pipette head.
message	BSTR	The error message text.
plate	BSTR	The plate name (<i>processname instance</i>).
timeStamp	BSTR	The time at which the error occurred.

Returns

None.

MessageBoxAction

Description

Occurs when user response is required.

Parameters

Name	Type	Description
actionToTake	LONG*	The value that indicates the action to take: 1 = OK 2 = CANCEL 3 = ABORT 4 = RETRY 5 = IGNORE 6 = YES 7 = NO
caption	BSTR	The title of the message.
message	BSTR	The message body.
type	LONG	The type of message: 0 = MB_OK 1 = MB_OKCANCEL 2 = MB_ABORTRETRYIGNORE 3 = MB_YESNOCANCEL 4 = MB_YESNO 5 = MB_RETRYCANCEL

Returns

None.

ProtocolComplete

Description

Occurs after startup, cleanup, and main protocols are finished.

Parameters

None.

Returns

None.

RecoverableError

Description

Occurs whenever an error is displayed and expects the operator to abort, retry, or ignore the error.

Parameters

Name	Type	Description
actionToTake	LONG*	The value that indicates the action to take: 0 = Retry 1 = Ignore 2 = Abort
source	BSTR	The device that caused the error.
description	BSTR	The error description.

Returns

None.

UnrecoverableError

Description

Occurs when an error is displayed and does not expect the operator to respond with a decision.

Parameters

Name	Type	Description
description	BSTR	The description of the error.

Returns

None.

UserMessage

Description

Occurs when a User Message task occurs.

Parameters

Name	Type	Description
caption	BSTR	The title of the message.
message	BSTR	The body of the message.
userdata	BSTR	Allows user to enter variable name if wantsData = True.

Returns

None.

Related information

For information about...	See...
VWorks ActiveX control	“About the VWorks ActiveX control” on page 3
VWorks ActiveX properties	“Properties” on page 30
VWorks ActiveX methods	“Methods” on page 6
VWorks ActiveX enumerated types	“Enumerated types” on page 35

Enumerated types

RetCode

Description

Indicates the method call status.

Constants

Name	Value	Description
RETURN_SUCCESS	0	The method was called successfully.
RETURN_BAD_ARGS	1	The method contains bad arguments.
RETURN_FAIL	2	The method call failed.

Related information

For information about...	See...
VWorks ActiveX control	“About the VWorks ActiveX control” on page 3
VWorks ActiveX methods	“Methods” on page 6
VWorks ActiveX events	“Events” on page 30



VWorks3 ActiveX Guide
G5415-90059