

**Agilent Technologies  
Solar Array Simulator  
System Control Tools**

**14360A**

**User's Guide**

## Legal Notices

© Agilent Technologies, Inc. 2008

No part of this document may be photocopied, reproduced, or translated to another language without the prior agreement and written consent of Agilent Technologies, Inc. as governed by United States and international copyright laws.

### 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.

### Manual Editions

First Edition, June, 2008  
Printed in Malaysia.

Reprints of this manual containing minor corrections and updates may have the same printing date. Revised editions are identified by a new printing date.

### Waste Electrical and Electronic Equipment (WEEE) Directive 2002/96/EC

This product complies with the WEEE Directive 2002/96/EC marketing requirement. The affixed product label (see below) indicates that you must not discard this electrical/electronic product in domestic household waste.

**Product Category:** With reference to the equipment types in the WEEE directive Annex 1, this product is classified as “Monitoring and Control instrumentation” product.

Do not dispose in domestic household waste.

To return unwanted products, contact our local Agilent office, or see [www.agilent.com/environment/product](http://www.agilent.com/environment/product) for more information.



### Certification

Agilent Technologies certifies that this product met its published specifications at time of shipment from the factory. Agilent Technologies further certifies that its calibration measurements are traceable to the United States National Institute of Standards and Technology, to the extent allowed by the Institute's calibration facility, and to the calibration facilities of other International Standards Organization members.

### Exclusive Remedies

THE REMEDIES PROVIDED HEREIN ARE THE CUSTOMER'S SOLE AND EXCLUSIVE REMEDIES. AGILENT TECHNOLOGIES SHALL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER BASED ON CONTRACT, TORT, OR ANY OTHER LEGAL THEORY.

### Assistance

This product comes with the standard product warranty. Warranty options, extended support contacts, product maintenance agreements and customer assistance agreements are also available. Contact your nearest Agilent Technologies Sales and Service office for further information on Agilent Technologies' full line of Support Programs.

### Technologies 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.

### U.S. Government Restricted Rights

Software and technical data rights granted to the federal government include only those rights customarily provided to end user customers. Agilent provides this customary commercial license in Software and technical data pursuant to FAR 12.211 (Technical Data) and 12.212 (Computer Software) and, for the Department of Defense, DFARS 252.227-7015 (Technical Data – Commercial Items) and DFARS 227.7202-3 (Rights in Commercial Computer Software or Computer Software Documentation).

### Trademarks

Microsoft and Windows are U.S. registered trademarks of Microsoft Corporation.

## In this Book

Specific chapters in this manual contain the following information:

- Installation – Chapter 1 describes the installation of the system control tools.
- Using the System Control Tools – Chapter 2 describes how to use the Configuration Wizard, Initialize the Driver, and use the Server Control.
- System Driver Functions – Chapter 3 gives a detailed description of the driver functions including syntax, parameters, and return values.
- System Driver Enumerations – Chapter 4 describes the driver function enumerations.

### NOTE

You can contact Agilent Technologies at one of the following telephone numbers for warranty, service, or technical support information.

In the United States: (800) 829-4444

In Europe: 31 20 547 2111

In Japan: 0120-421-345

Or use our Web link for information on contacting Agilent in your country or specific location: [www.agilent.com/find/assist](http://www.agilent.com/find/assist)

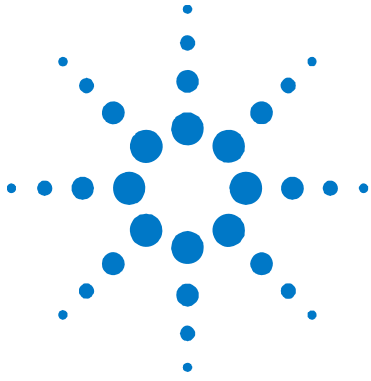
Or contact your Agilent Technologies Representative.

The web contains the most up to date version of the manual. Go to <http://www.agilent.com/find/E4360> to get the latest version of the manual.

---

# Contents

<b>1 - Installation.....</b>	<b>5</b>
Installing the Agilent 14360A System Control Tools .....	6
<b>2 - Using the System Control Tools.....</b>	<b>9</b>
System Description.....	10
Running the Configuration Wizard.....	14
Using the System Driver Locally and Remotely .....	22
Coordinating Access by Remote Clients.....	24
Example Program .....	28
System Driver Usage Guide.....	30
<b>3 - System Driver Functions.....</b>	<b>35</b>
Driver Function Hierarchy.....	36
SelectCriteria .....	37
IAgilent14360 Interface.....	39
IAgilent14360Dlog Interface.....	45
IAgilent14360DriverOperation Interface.....	47
IAgilent14360FixedMode Interface .....	53
IAgilent14360Measurement Interface.....	55
IAgilent14360Mode Interface .....	56
IAgilent14360Output Interface.....	58
IAgilent14360Protection Interface .....	59
IAgilent14360SAS Interface .....	63
IAgilent14360SASImmMode Interface .....	65
IAgilent14360SASListMode Interface .....	66
IAgilent14360Status Interface .....	70
IAgilent14360System Interface.....	72
IAgilent14360TableMode Interface.....	75
IAgilent14360Transient Interface.....	80
<b>4 - System Driver Enumerations.....</b>	<b>83</b>
System Driver Enumerations.....	84



# 1 Installation

<a href="#">Installing the Agilent 14360A System Control Tools</a> .....	6
--	---

This chapter describes the installation of the Agilent 14360A System Control Tools. It also describes the system requirements.

For complete details on installation, load connections, interface connections, and programming the instrument from the front panel, refer to the E4360 SAS Users Guide.

For complete details on programming the instrument using SCPI commands, refer to the E4360 Programmer's Reference Help file.

Both documents are included on the Agilent E4360A Product Reference CD. This CD-ROM is shipped along with your instrument.

## Installing the Agilent 14360A System Control Tools

### System Control Tools

**System Driver.** The driver for controlling the SAS system. It is in the form of a DLL with a COM interface.

**Configuration Wizard.** An application for creating and editing configuration files that describe the layout of the instruments in your SAS system. The configuration file is used to initialize the System driver.

**Server Control.** An application used when the System Driver is controlling the system from a remote application. The Server Control will listen for remote client requests to communicate with the SAS system. This application also includes a Web monitor that lets you view the measurements and settings in the system.

**Examples.** Programming examples that demonstrate the functionality of the System Driver.

### Requirements

- Windows XP or Vista
- Agilent IO Libraries Suite 15 (disc supplied with the E4360A SAS mainframe). This must be installed *before* you install the Agilent 14360 System Control Tools.
- Any COM compatible programming language such as: C++, VB, LabView, and .NET languages such as C#, VB.NET
- E4360 Series SAS (E4350B not supported)
- Not supported on Unix or Linux

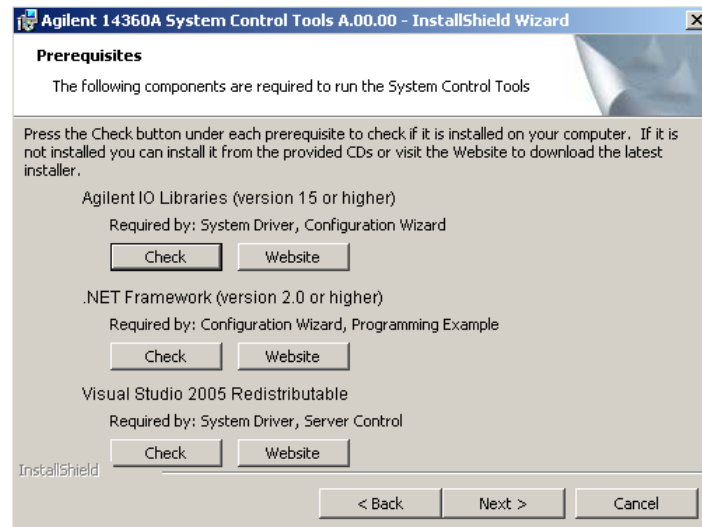
### Installation

Place the Agilent 14360 System Control Tools disc in a disc drive on your computer. Wait a few seconds for the disc to open.

Under **Software**, click **Install** to install the **Agilent 14360A System Control Tools**.

When the installer runs, it puts up the following dialog box that lets you check if your system has the following prerequisite software components installed. If you have installed the Agilent IO Libraries Suite 15, these components should have been installed on your system.

Click  **Check** to check that the software is installed. If it is not installed, you can **install** the software from Agilent IO Libraries Suite 15 disc, or click  **Website** to download the software from the web.



After you have checked for the software prerequisites, continue with the installation procedure and install the **Complete** package.

If you wish to install just the minimum necessary components follow these guidelines:

If you are installing the software on a single-controller PC, you need at minimum to install the following components:

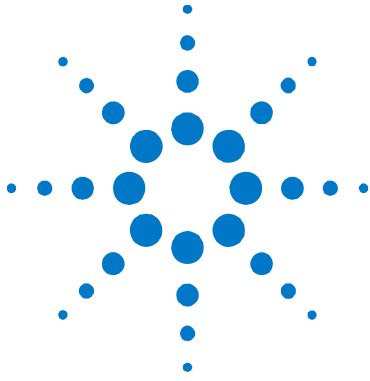
- System Driver
- Configuration Wizard

If you are installing on an Application PC connected to a Controller PC, you need at minimum to install the following components:

- System Driver - on both Controller PC and Application PC
- Server Control - on the Controller PC
- Configuration Wizard - on the Controller PC







## 2

# Using the System Control Tools

<a href="#">System Description</a> .....	10
<a href="#">Running the Configuration Wizard</a> .....	14
<a href="#">Using the System Driver Locally and Remotely</a> .....	22
<a href="#">Coordinating Access by Remote Clients</a> .....	24
<a href="#">Example Program</a> .....	28
<a href="#">System Driver Usage Guide</a> .....	30

This chapter describes how to use the 14360 System Control Tools when programming a number of individual SAS instruments configured as a large solar array simulator system.

**NOTE**

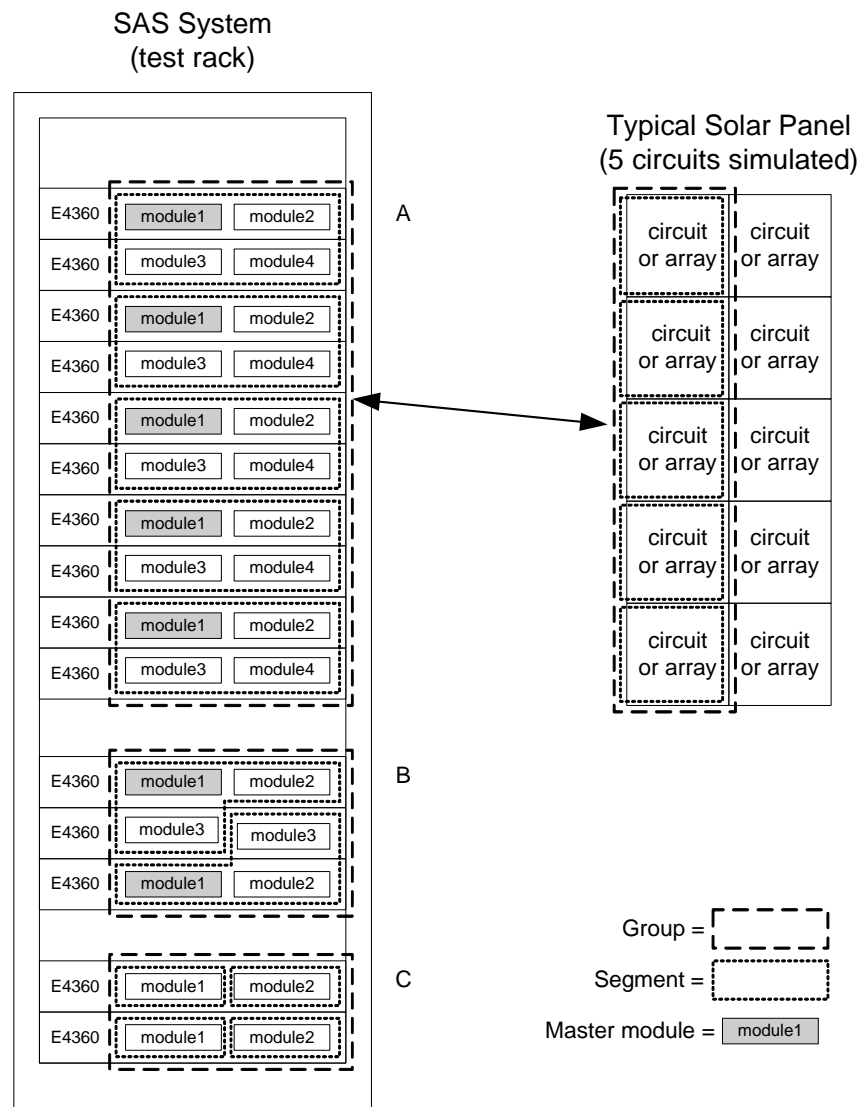
The system control tools described in this chapter are not available from the front panel or when using SCPI programming commands.

---

## System Description

A typical SAS system contains between 25 and 50 instruments, with each instrument typically containing two output power modules. The output modules can be paralleled to produce higher output power. The individual instruments and modules in an SAS system can be organized into programmable groups that are tailored to specific power requirements. The Agilent 14360 System Control Tools simplify the task of a system integrator because they provide system-level control of all instruments through a single driver interface. The initial setup of the System Driver is simplified through the use of the Configuration Wizard.

The following figure illustrates the terminology used by the System Driver software to describe the various system components. The figure also shows the correspondence of these components to an actual solar array panel.



## Glossary and Terms

### System

A System refers to a system bay or test rack, which includes at least one logical group containing a number of SAS segments. A system is not necessarily limited to one test track, but can span multiple racks, depending how the system is configured.

### Group

A Group is defined as a collection of segments. A system bay or test rack must have at least one group, but is not limited to just one. Groups provide an increased level of flexibility when emulating solar panels. Any number or arrangement of segments (circuits) can be configured into a group, which then allows all member segments to be programmed using system-level driver functions.

The example test rack in the previous figure illustrates three groups (A through C). Example A illustrates a logical group that corresponds to a grouping of circuits in a solar panel.

### Segment

A Segment corresponds to a single circuit on a solar panel (example A). A segment can contain from 1 to 4 paralleled modules installed in up to 2 mainframes. All modules in the segment must be wired in auto-parallel as described in Chapter 2 of the E4360 User's Guide. The analog connectors must be wired, with one of the modules in the channel 1 location wired as the "master" channel, and the remaining channels (2 through 4) wired as "follower" channels.

The bottom half of the test rack shows a number of alternate segment configurations. In a configuration where 3 modules are auto-parallelled, note that the master module must always be in the channel 1 location of the mainframe (example B).

### Module

A Module corresponds to a single output channel on an Agilent E4360A SAS mainframe. A module **must** be assigned to a segment even if it is not paralleled with another module (example C). This means that a system of 50 unparallelled modules, for example, still requires 50 segments to be created in the Configuration Wizard.

## IP Addresses and Hostnames

### Connecting to a site LAN

A site LAN is a local area network in which LAN-enabled instruments and computers are connected to the network through routers, hubs, and/or switches. They are typically large, centrally-managed networks with services such as DHCP and DNS servers.

Normally, when connecting the instrument to the site LAN, the factory-shipped instrument LAN settings are configured to automatically obtain an IP address from the network using a DHCP server. This is **not** recommended when you will be using a configuration file to initialize the instruments under system control.

Because the configuration file stores the instrument's IP address and hostname, the instrument's IP address and hostname must not change each time the instrument is turned on. If this occurs, the configuration file cannot initialize the SAS system.

Therefore, it is recommended that you manually assign an IP address and hostname to the instrument using the instrument's front panel. The following table gives examples of IP addresses and hostnames that could be used in an SAS system consisting of 50 instruments:

Sample IP Addresses (DHCP servers)	Sample Hostnames (DNS servers)
192.168.1.101	A-E4360-01
192.168.1.102	A-E4360-02
192.168.1.103	A-E4360-03
.	.
.	.
.	.
192.168.1.150	A-E4360-50

(Note: The first 3 values are the network portion of the address)

### Sample Subnet Mask

255.255.255.0

### Sample Default Gateway

192.168.1.1

### Configuring the LAN Parameters

To manually configure the LAN settings, press the Menu key, then use the navigation keys to select: **System** → **IO** → **LAN** → **Config**.

In the **Config** menu select **IP** to configure IP addresses; select **Name** to configure hostnames.

## IP

Select **IP** to configure the addressing of the instrument.

- Select **Manual** This parameter allows you to manually configure the addressing of the instrument by entering values in the following three fields. These fields only appear when Manual is selected.
- IP Address** This value is the Internet Protocol (IP) address of the instrument. The IP Address consists of 4 decimal numbers separated by periods. Each decimal number ranges from 0 through 255.
- Subnet Mask** This value is used to enable the instrument to determine if a client IP address is on the same local subnet. When a client IP address is on a different subnet, all packets must be sent to the Default Gateway.
- Default Gateway** This value is the IP Address of the default gateway that allows the instrument to communicate with systems that are not on the local subnet, as determined by the subnet mask setting.

## Name

Select **Name** to configure the hostname of the instrument.

- Host name** Each SAS is shipped with a default hostname with the format: A-modelnumber-serialnumber, where *modelnumber* is the mainframe's 6-character model number (e.g. E4360A), and *serialnumber* is the last five characters of the 10-character mainframe serial number located on the label on the top of the unit (e.g. 45678 if the serial number is MY12345678).

## Connecting to a private LAN

A private LAN is a network in which LAN-enabled instruments and computers are connected to a router that has DHCP for assigning addresses. These may or may not be reconfigured each time the instruments are turned on.

Basically, a private LAN has the advantage of having a smaller number of instruments (clients) and PCs, and there is usually greater control of network settings compared to a site LAN. Therefore, it is a good environment for using manual LAN settings as previously described, because usually there will be only one person responsible for setting these and less likelihood that others will be arbitrarily changing addresses – resulting in address collisions.

Manual addressing also allows you to use the same configuration file in multiple SAS systems that are each connected to a private LAN. Of course, each instrument in each SAS system must be set to the same IP address and Hostname as used in the configuration file.

### NOTE

When replacing an instrument in a previously configured SAS system, you must manually set the IP address and Hostname of the new instrument to the same address as the replaced instrument

## Running the Configuration Wizard

The purpose of the Configuration Wizard is to create a configuration file that describes the layout of the output modules in your SAS system. The configuration file contains information about the groups, segments, and modules. It also contains other information like the digital I/O pin assignments of the instruments.

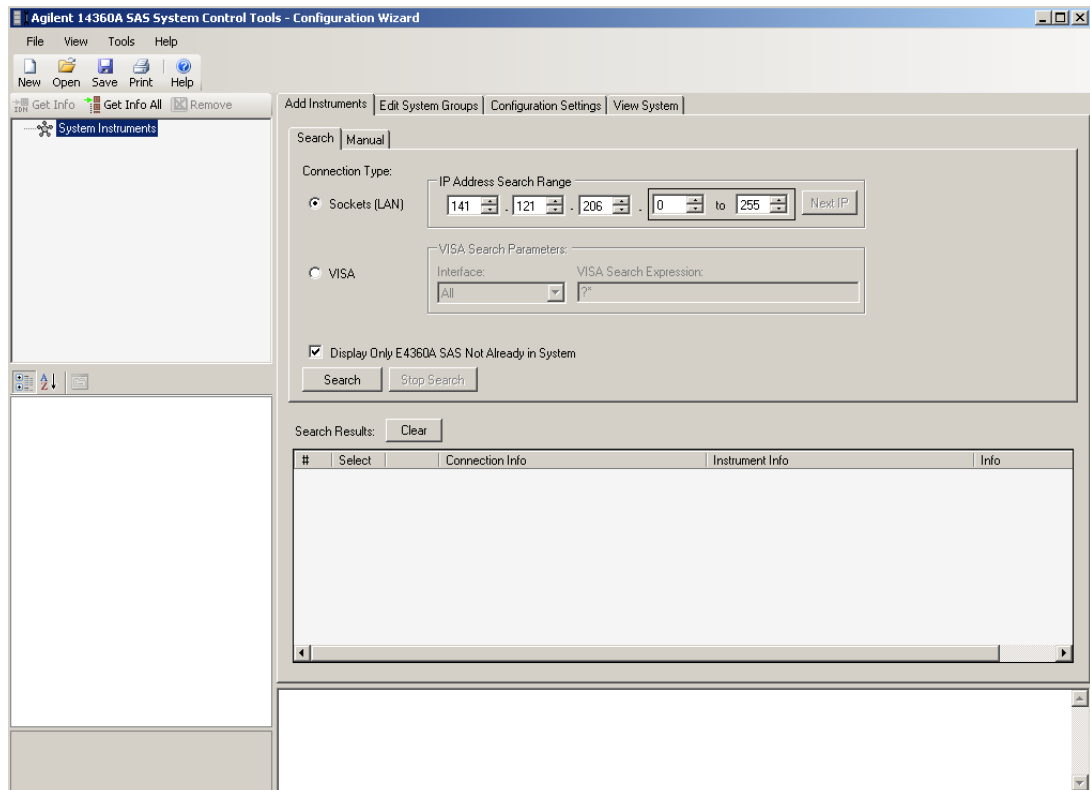
The System Configuration Wizard should be installed on the PC that has communication to the instruments via the chosen interface. This is because the Configuration Wizard will have the ability to search, find, and retrieve information about the instruments in your system.

Once the Configuration file has been created, it can be moved or copied to other controller PCs, where it will be used to initialize the System Driver on that PC.

To run the 14360 Configuration Wizard, click on **Start → Programs → Agilent 14360A System Control Tools → Configuration Wizard**.

### Adding Instrument to the Configuration

**Step 1.** Select the **Add Instruments** tab.



**Step 2.** Select the connection type; Sockets or VISA.

Add Instruments | Edit System Groups | Configuration Settings | View System

Search | Manual

Connection Type:

Sockets (LAN) IP Address Search Range: 141 . 121 . 206 . 0 to 255 Next IP

VISA VISA Search Parameters: Interface: All VISA Search Expression: ?

Display Only E4360A SAS Not Already in System

Search Stop Search

With **Sockets**, the System Driver will connect to the instrument using the operating system's sockets library (i.e. WinSock). The advantage of using sockets is a speedup over VISA when sending small amounts of data between the PC and the instruments as in most cases. The disadvantage is a minor slowdown over VISA when retrieving larger amounts of data. Under Sockets Criteria, specify the IP address. For class C addresses, the first three fields are the network portion. The last field is the host portion of the address. You can enter a range to search on for the host portion.

**NOTE**

You can view the active IP address of an instrument by pressing the **Menu** key on the instrument's front panel, and using the navigation keys to select: **System\IO\LAN\ActiveSettings**.

With **VISA**, you can specify the interface type and the search criteria for VISA connection types.

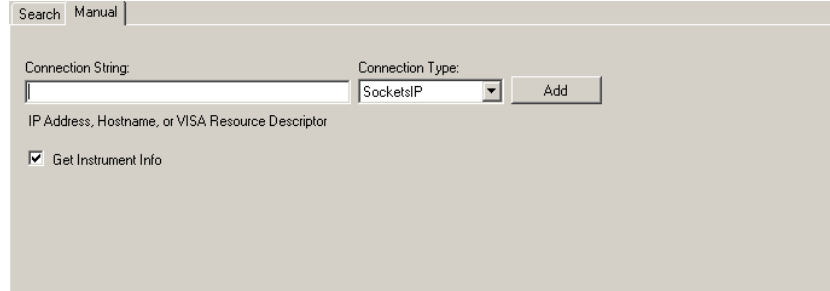
**Step 3.** Click on the **Search** button. The search will find all of the instruments and output modules that are connected to the specified interface. Once the search is complete, the instruments are added to the **Search Results**. The Search Results displays both found and not found results.

Search Results: Clear

#	Select		Connection Info	Instrument Info	Info
1	<input type="checkbox"/>	Add	IP: 141.121.202.226 Hostname: a-e4360a-00617.usa.agilent.com Comments: Warning: The instrument has an automatically assigned IP address.	E4360A (US43600617) Module 1: E4361A (US00000810) Module 2: E4361A (US00000958)	Connection Successful: Yes Already in System: No
2	<input type="checkbox"/>	Add	IP: 141.121.202.227 Hostname: a-e4360a-00313.usa.agilent.com Comments: Warning: The instrument has an automatically assigned IP address.	E4360A (US43600313) Module 1: E4361A (US00000414) Module 2: E4361A (US00000478)	Connection Successful: Yes Already in System: No
3	<input type="checkbox"/>	Add	IP: 141.121.202.228 Hostname: a-e4360a-00477.usa.agilent.com Comments: Warning: The instrument has an automatically assigned IP address.	E4360A (US43600477) Module 1: E4361A (US00000317) Module 2: E4361A (US00000548)	Connection Successful: Yes Already in System: No
4	<input type="checkbox"/>	Add	IP: 141.121.202.229 Hostname: a-e4360a-00654.usa.agilent.com Comments: Warning: The instrument has an automatically assigned IP address.	E4360A (US43600654) Module 1: E4361A (US00000615) Module 2: E4361A (US00000518)	Connection Successful: Yes Already in System: No

## 2 Using the System Control Tools

Alternatively, use the **Manual** tab to add a single instrument at a known address or an instrument that is not found by the Search function. You will need to supply the connection string and connection type. When you click **Add**, a connection with the specified connection string will be attempted.



Search Manual

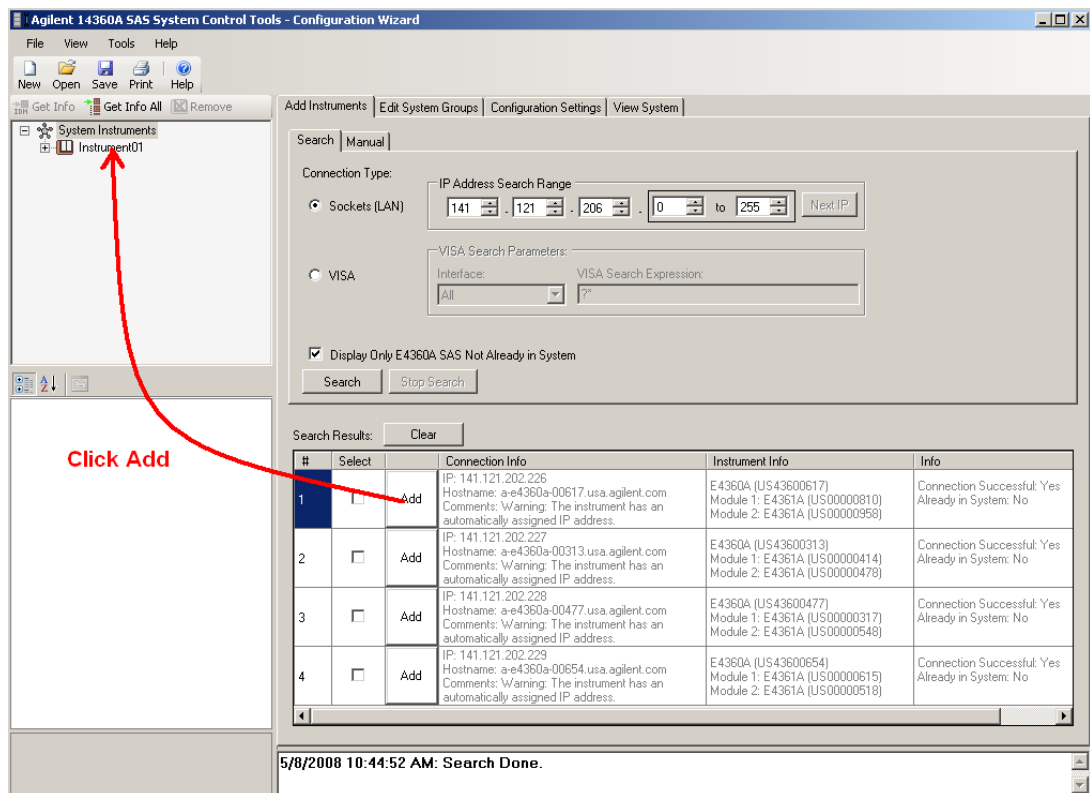
Connection String:  Connection Type: SocketsIP

IP Address, Hostname, or VISA Resource Descriptor

Get Instrument Info

Checking **Get Instrument Info** will retrieve information about the instrument and its modules. Uncheck this when working on a PC that does not have connection to the instruments.

**Step 4.** Click the **Add** buttons to then add the instruments to the System Instruments list.



Agilent 14360A SAS System Control Tools - Configuration Wizard

File View Tools Help

New Open Save Print Help

Get Info Get Info All Remove

Add Instruments Edit System Groups Configuration Settings View System

Search Manual

Connection Type:  Sockets (LAN)  VISA

IP Address Search Range: 141.121.202.10 to 255 Next IP

VISA Search Parameters: Interface: All VISA Search Expression: \*

Display Only E4360A SAS Not Already in System

Search Stop Search

Search Results: Clear

#	Select	Connection Info	Instrument Info	Info
1	<input type="checkbox"/>	IP: 141.121.202.226 Hostname: a-e4360a-00617.usa.agilent.com Comments: Warning: The instrument has an automatically assigned IP address.	E4360A (US43600617) Module 1: E4361A (US00000810) Module 2: E4361A (US00000958)	Connection Successful: Yes Already in System: No
2	<input type="checkbox"/>	IP: 141.121.202.227 Hostname: a-e4360a-00313.usa.agilent.com Comments: Warning: The instrument has an automatically assigned IP address.	E4360A (US43600313) Module 1: E4361A (US00000414) Module 2: E4361A (US00000478)	Connection Successful: Yes Already in System: No
3	<input type="checkbox"/>	IP: 141.121.202.228 Hostname: a-e4360a-00477.usa.agilent.com Comments: Warning: The instrument has an automatically assigned IP address.	E4360A (US43600477) Module 1: E4361A (US00000317) Module 2: E4361A (US00000548)	Connection Successful: Yes Already in System: No
4	<input type="checkbox"/>	IP: 141.121.202.229 Hostname: a-e4360a-00654.usa.agilent.com Comments: Warning: The instrument has an automatically assigned IP address.	E4360A (US43600654) Module 1: E4361A (US00000615) Module 2: E4361A (US00000518)	Connection Successful: Yes Already in System: No

5/8/2008 10:44:52 AM: Search Done.

System Instruments

- Instrument01

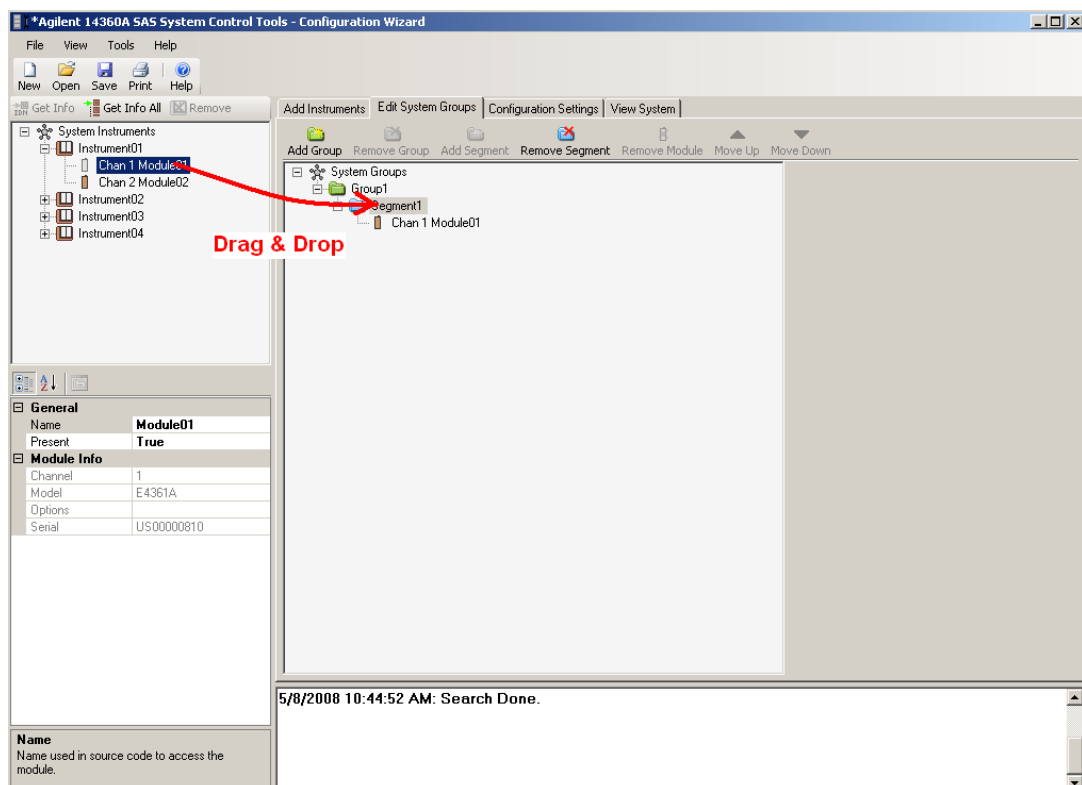
Click Add

All of the instruments you added will now appear in the System Instruments view on the left side of the window. At this point, all of the instruments that you will be using in your system have been identified, but information as to how the outputs are connected together and how they will be controlled must now be provided.



## Creating the System Groups

- Step 1.** Now you must specify the Groups, Segments, and the Modules present in the Segments. Click on the **Edit System Groups** tab. From the toolbar buttons located in the Edit tab, click on the **Add Group** button. This adds a group named “Group1” to the System Groups. Note that each system must consist of at least one Group.
- Step 2.** To add segments to the Group that has just been created, click to highlight “Group1” in the tree view, then click the **Add Segment** button, which is now active. A Segment heading named “Segment1” in this example will now be added under the group named Group1.
- Step 3.** To add modules to a Segment, simply select the module from the System Instruments list on the left side of the window and drag and drop it to the newly created Segment heading.



You can have a maximum of four (4) modules in a Segment. If a module has already been assigned to a segment, it cannot be used in another segment. By definition, if there is more than one module in a segment, the modules must be connected in parallel. The first module in the list is the master module. It **must** be located in channel one (1) of the mainframe, otherwise it cannot appear first in the list. The remaining modules in the list are the follower modules.

You can change the order of the modules in the list by selecting a module and using the Up/Down buttons to move it up or down. You can move a module from one segment to another by using the drag and drop method. You can also move a segment from one group to another.

**Important**

Multiple modules that have been assigned to a segment must have their output connectors and analog connectors wired according to the Auto-parallel connections diagram described in Chapter 2 of the E4360 User's Guide. This is because the analog wire connections from the master module to the follower modules provide accurate output synchronization between modules.

**Step 4.** To display and edit relevant information about any item in either the System Instruments list or the System Group list, simply click on an item. Information about the selected item will be displayed in the Properties area on the left side of the window.

## Specify the Configuration Settings

### Providing Configuration Information

Heading information about the Configuration file that you are now creating can be added in the Configuration Information window. Select the **General** tab to enter Configuration file heading information. Fill out the Configuration Information fields.

The screenshot shows the 'Configuration Settings' window with the 'General' tab selected. The 'Configuration Information' section includes the following fields:

Name:	big03-2
Author:	SuperTech1
Description:	System with 10 instruments and 20 modules
Company:	Agilent Technologies
Last Modified:	5/7/2008 3:43:44 PM
CRC Checksum:	2913718829 (OK)
File Version:	1.0

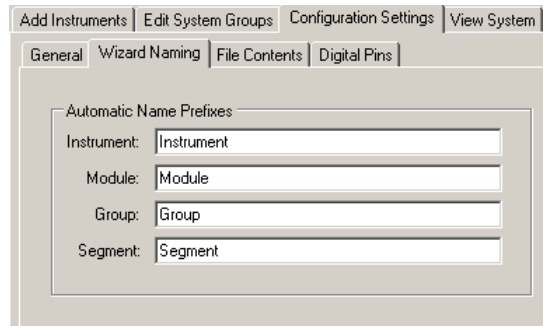
Below the fields, there is a checkbox for 'Enable Channel Groups' which is currently unchecked. Under the 'Global Connection Type' section, there are four radio buttons: 'Do Not Override' (selected), 'Sockets IP', 'Sockets Hostname', and 'VISA'.

If **Enable Channel Groups** is checked, modules that are in the same instrument and belong to the same Segment will be channel-grouped. This is the same as the instrument level SYST:GRO:DEF command. Channel grouping is explained in Appendix D of the E4360 User's Guide.

Each instrument sets its primary connection type. Under **Global Connection Type**, you can override the instrument's primary connection type and specify one of the three primary connection types that will be used when the configuration file is loaded.

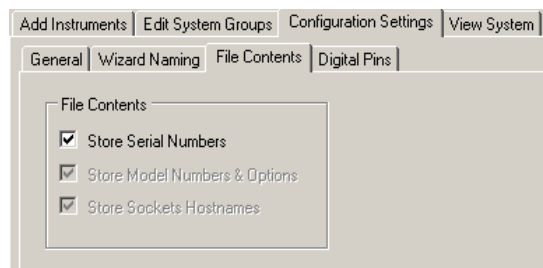
## Renaming the Name Prefixes

You can change the prefixes used when automatically generating names for an Instrument, Module, Group, or Segment. Select the **Wizard Naming** tab. Enter your own name to describe an **Instrument, Module, Group, or Segment**.



## Specifying the File Contents

You can specify what type of instrument information is stored in the configuration file. Select the **File Contents** tab. Check what type of information you wish to store in the configuration file.



**Store Serial Numbers:** Stores the serial numbers that identify the instrument.

**Store Model Numbers and Options:** If Store Serial Numbers is disabled you have the option to store the model numbers and options to identify a specific module.

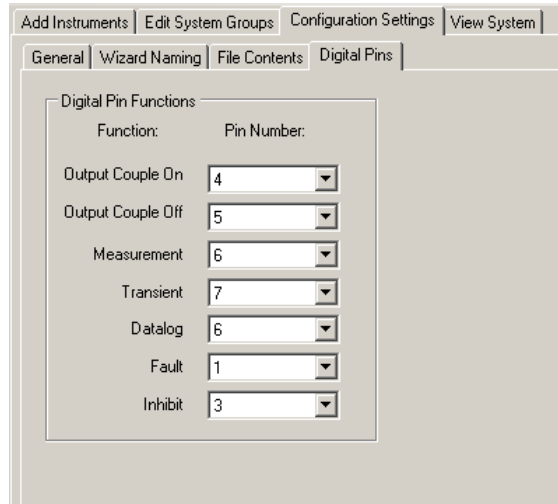
**Store Sockets Hostname:** If Store Serial Numbers is disabled you have the option to store the hostnames of the instruments. Note that automatically generated hostnames of the instruments contain the serial number of the instrument.

You can uncheck these boxes to create a configuration file that does not contain any uniquely identifying information of an instrument. This allows the same configuration file to be used in replicated systems.

## Configuring the Digital Pins

The Digital I/O pins let to configure the synchronized system functions. Synchronized system functions send a single SPCI command to an instrument/module to generate a trigger to other instruments using the digital port.

Select the **Digital Pins** tab to enter information about the digital pins into the configuration file. The digital pins must be configured across the entire system. This means that if digital pin 4 is configured to perform an Output Couple On function, digital pin 4 on each instrument is dedicated to perform this function. Pin 4 cannot be used for any other function. Note that each pin has the option of being disabled in the configuration file, so that it is not programmed using the system software.



The following pin functions can be programmed for the entire system.

Pin Function	Description	Available pins
Output Couple On Output Couple Off	This function enables or disables the outputs on of all connected mainframes. This function requires both ONC and OFFC pin connections. All designated pins and the common pins must be connected together. Can be programmed for the entire system or a specific group.  If group-wide connections have been configured, the system-wide functionality is not available.	4 - 7
Measurement	This function synchronously triggers a measurement on all outputs of all connected mainframes. All designated pins and the common pins must be connected together.	4 - 7
Transient	This function synchronously triggers an SAS list on all connected mainframes. All designated pins and the common pins must be connected together.  If multiple lists have been programmed, they must all be programmed with the exact same dwell times.	3 - 7
DataLog	This function synchronously triggers a datalog on all connected mainframes. All designated pins and the common pins must be connected together.  Only one datalog can be running at a time in instruments connected by the same datalog pin wire.	3-7

Pin Function	Description	Available pins
Fault	This function enables a fault condition on any channel to generate a fault signal on the Digital Control port. The following conditions will generate a fault event: over-voltage, over-current, over-temperature, inhibit signal, power-fail condition, or on some models, a power-limit condition. Pin 2 is the common for pin 1. Pin 2 must also be connected to the common pin.	1, 2
Inhibit	This function lets an external input signal control the output state of all of the output channels if they have previously been enabled. All designated pins and the common pins must be connected together.	3

## View the Configuration File

Select the **View System** tab to view the overall information in the configuration file. You can print this view from the **File → Print** menu

The screenshot shows the 'View System' tab in the software interface. The main window displays the following information:

**System**

Name: Author: Company: Last Modified: 5/28/2008 3:13:52 PM File Version: 1.0  
Description:

Groups

- Group: Group1
  - Segment: Segment1
    - Module: Module01
    - Module: Module02
  - Segment: Segment2
    - Module: Module03
    - Module: Module04

Instruments

Name	Connection Type	Sockets Hostname	Sockets IP	VISA Resource Descriptor	Model	Serial	Firmware Revision	Channel Groups	
Instrument01	SocketsHostname	A-E4360A-01	192.168.1.106		E4360A	USLP200009	0.01.47	'1','2'	
Modules				<b>Present</b>	<b>Channel</b>	<b>Name</b>	<b>Model</b>	<b>Serial</b>	<b>Options</b>
				true	1	Module01	E4361A	US12345678	
				true	2	Module02	E4361A	US12345677	
Instrument02	SocketsHostname	A-E4360A-00017.usa.agilent.com	141.121.202.226		E4360A	US43600017	0.01.47	'1','2'	
Modules				<b>Present</b>	<b>Channel</b>	<b>Name</b>	<b>Model</b>	<b>Serial</b>	<b>Options</b>
				true	1	Module03	E4361A	US00000010	
				true	2	Module04	E4361A	US00000018	

Variable Table

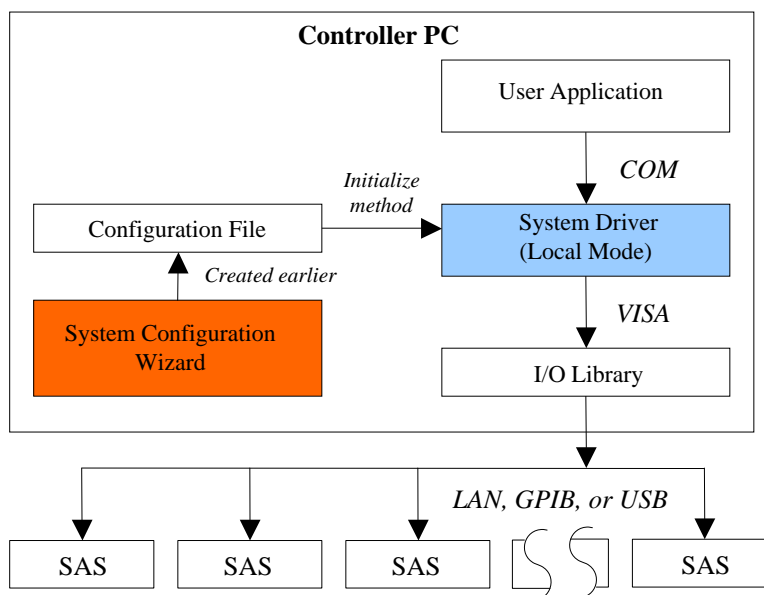
Name	Value
GlobalConnType	DoNotOverride
InstNamePrefix	Instrument
GroupNamePrefix	Group
SegmentNamePrefix	Segment

Find: (Ctrl+F)

## Using the System Driver Locally and Remotely

### Single-controller PC (Local mode)

Once installed on the single-controller PC and initialized by the Initialize method, the System Driver is set to Local mode by default. In Local mode the System Driver will communicate directly with the instruments in the system as shown in the following figure.



### Application and Controller PC (Remote mode)

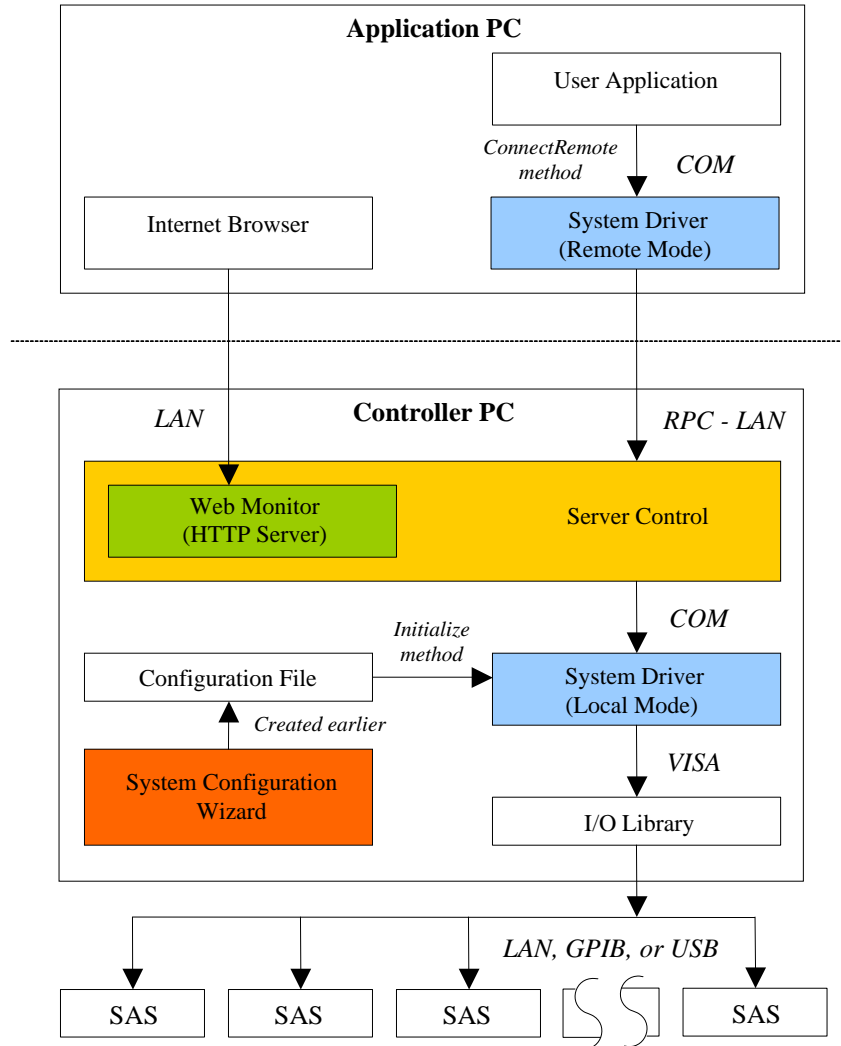
In Remote mode, the driver first goes through the Server Control and then communicates with the system using the Server Control's instance of the System Driver running in Local mode as shown in the following figure. This setup allows the SAS instruments to be isolated in a private network. The Controller PC would have two network cards with access to the SAS private network and the other to the network of the Application PC. The Application PC would only be able to access the instruments through the use of the Server Control and would not have direct access to the instruments.

#### Important

As explained in the next section, the Server Control must first be running with the RPC Server Control **Started** for the System Driver running in Remote mode to communicate with the System Driver running in Local mode.

The System Driver function calls are same in both Local and Remote modes. The only difference is that in Remote mode the ConnectRemote function is called prior to calling the Initialize function. This design allows for greater portability of your application, since it can be moved from running locally and directly communicating with the system, to running remotely and being relayed through the Server Control.

After it has been created using the Configuration Wizard, the Configuration file must reside on the PC that is controlling the System Driver in Local mode. The Configuration file cannot reside on a PC that will be used in Remote mode. There is no capability of uploading a configuration file from a Remote mode PC.



The Web Monitor illustrated in the figure above is discussed under “Using the System Web Monitor” in the next section. Use the Web Monitor to view the status of the Local mode System Driver on the Controller PC and read back measurements from the system.

#### Important

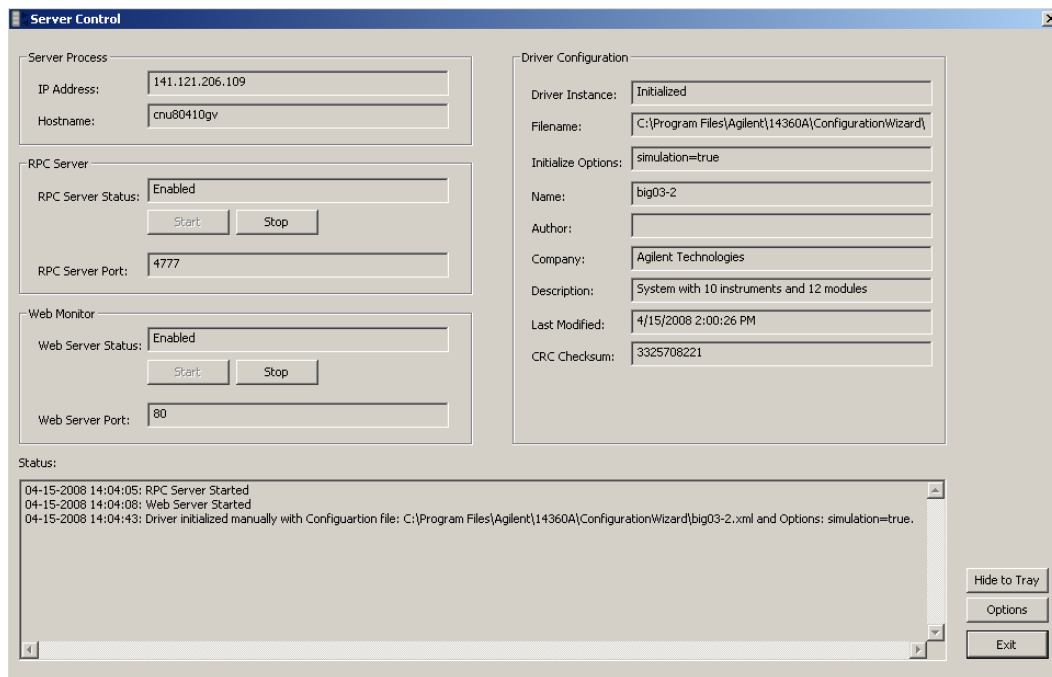
As explained in the next section, the Server Control must first be running with the RPC Server Control **Started** for the System Driver running in Remote mode to be able to communicate with the System Driver running in Local mode. The Web Monitor control must also be **Started** for the internet browser on the Application PC to access the monitor.

All of the System Driver functions that you will use to control your SAS system are described in chapter 3.

## Coordinating Access by Remote Clients

The Server Control is an application running in the background of the Controller PC. The main purpose of the Server Control is to be able to control the SAS system from a remote PC. The Server Control is also used to coordinate access to the system by remote clients.

To access the Server Control application, click on **Start** → **Programs** → **Agilent 14360A System Control Tools** → **Server Control**.



The window of the Server Process includes the following information fields and controls:

- Server Address** Displays the IP Address and Hostname of the Controller PC.
- RPC Server** Indicates the status of the RPC server (either running or stopped). Click **Start** to start the RPC Server. Click **Stop** to stop the RPC Server. Also indicates the address of the RPC server port.
- Web Monitor** Indicate the status of the Web Monitor (either running or stopped). Click **Start** to start the Web Monitor. Click **Stop** to stop the Web Monitor. Also indicates the address of the Web server port.
- Driver Configuration** Indicates the status of Local mode System Driver on the Controller PC. Also displays the configuration information stored in the Configuration file.
- Status** Displays the application status messages.
- Hide to Tray** Minimizes the Server Control to the computer's system tray.
- Options** Modify the RPC Server and Web Monitor port numbers. Enable or disable logging function calls to an external Log.txt file located in the installation folder of the Server Control.
- Exit** Closes the Server Control.

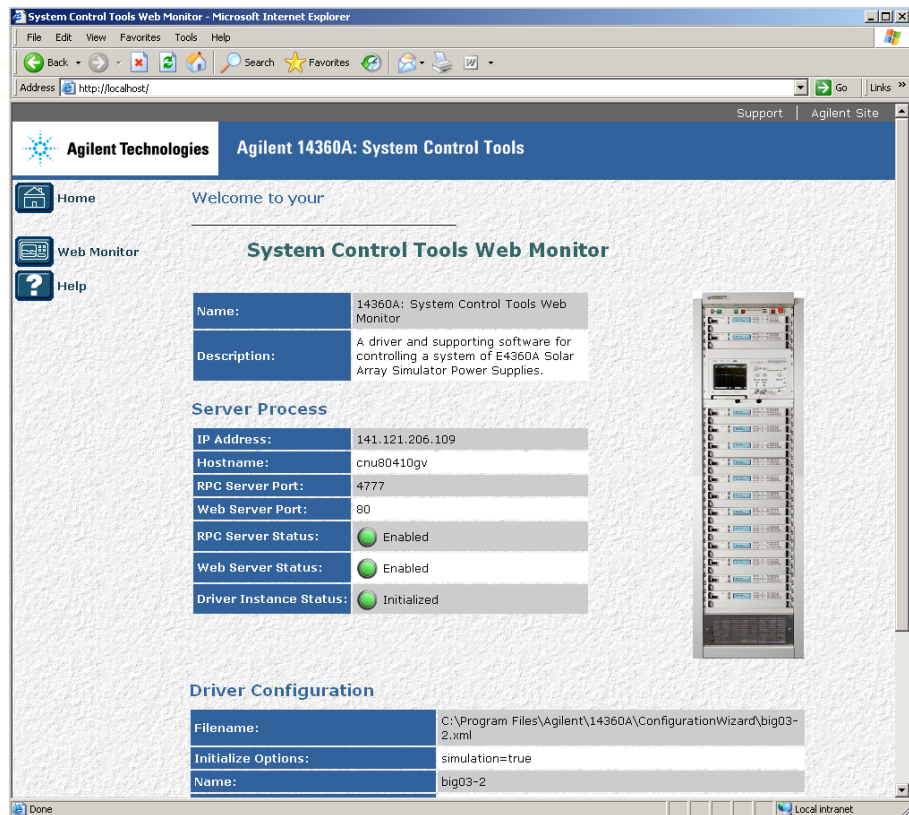


## Using the System Web Monitor

The System Web Monitor is a series of web pages hosted by the server process' Web server on the Controller PC. The Web monitor lets you view the status of the SAS system from an internet browser on a remote computer on the same network as the Server Control. It does not allow control of any of the instruments in the system.

To launch the System Web Monitor:

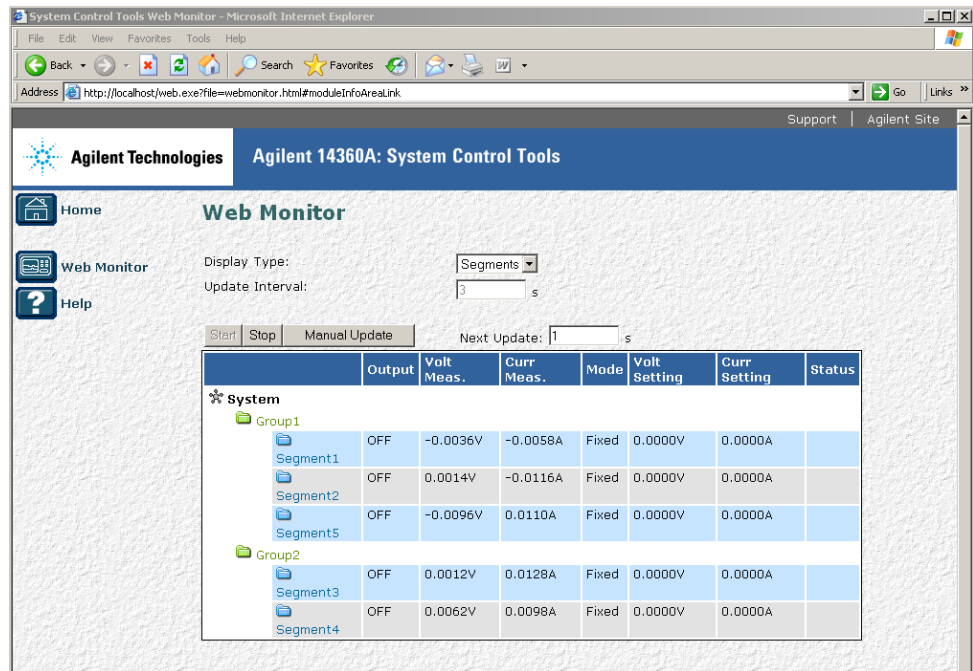
- 1 Open the internet browser on your computer.
- 2 Enter the controller PC's full hostname or IP address into the browser's Address field to launch the Web monitor. Type in "localhost" if you are using the internet browser on the Controller PC to access the Web Monitor. The following home page will appear:
- 3 Click on the Web Monitor button in the navigation bar on the left to begin monitoring your system.



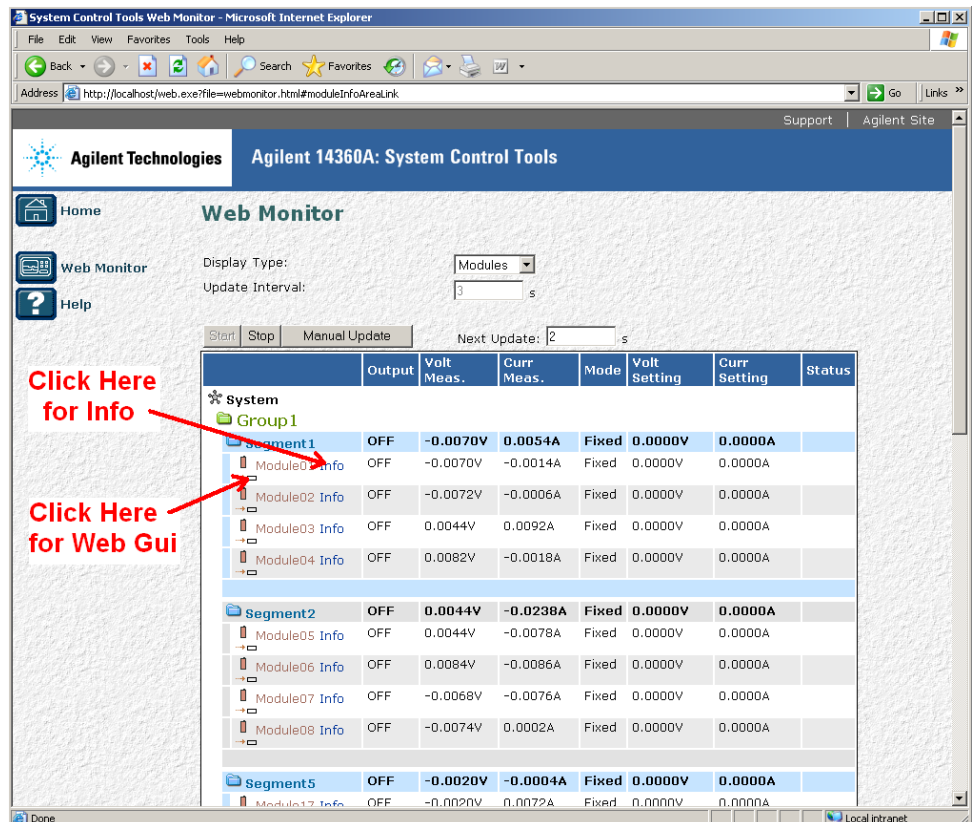
- 4 For information about any of the pages, click on the Help button.

As shown in the following figure, after clicking the Web Monitor button, the system will be displayed in a table type display. Under Display Type, you can display just the Segments, or the Segments with their included Modules. You can specify the Update Interval at which the measurements are updated on the display. Click Start to start the automatic updates. Click Stop to stop the automatic updates. Click Manual Refresh to manually update the display.

## 2 Using the System Control Tools



When Modules are selected as the Display Type, you can click the Info link that appears below each Module icon to display additional information about the module. To access the internal Web Gui of the instrument containing the module, click the small rectangle that appears to the right of the Info link. These controls are shown in the following figure.



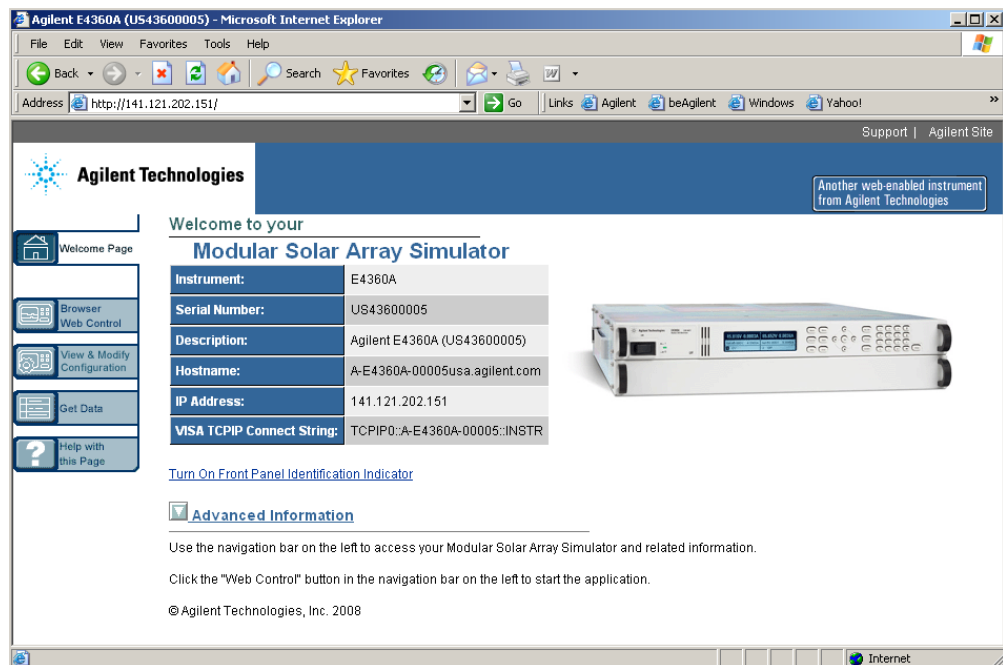
The following information is displayed at the bottom of the Web Monitor page when the Info link is pressed.

```

Module Info
Name: Module01
Model: E4361A
Serial: US12345678
Options:
Segment Member Type: Master
Channel Group Member Type: None

Instrument Info
Name: Instrument01
Model: E4361A
Serial: US12345678
Connection String: 141.121.202.1
Sockets IP: 141.121.202.1
Sockets Hostname:
VISA Resc Desc:
  
```

The following is an example of the Web Gui page of an individual instrument that appears after the small rectangle is pressed.



If the Web Gui does not appear, it may be because the remote computer is located on different network the Server Control. In this case, the Web Gui of the individual instrument cannot be accessed.

### CAUTION

In contrast to the System Web monitor, the Web Gui page of an individual instrument **does** allow you to control it from the Web Gui page. This can cause unpredictable results if the SAS system is also being controlled by an application program running on the Controller PC.

### Example Program

The Agilent 14360A Server Control Tools come with one comprehensive example program that covers System Driver initialization, monitoring, setting up Fixed mode, setting up SAS mode, protection, and data logging.

To access the Example, click on **Start → Programs → Agilent 14360A System Control Tools → Examples → UserApplication**. The following application window should appear.

To access the User application source code, open the **Examples** folder. In the SystemDriverExamples1 folder, open the solution named SystemDriverExamples.sln. Visual Studio C# 2005 or newer is required to open the solution. If you do not have Visual Studio you may still view the source code of the project by opening the .cs files located in the UserApplication folder using a text editor.

### Configure Tab

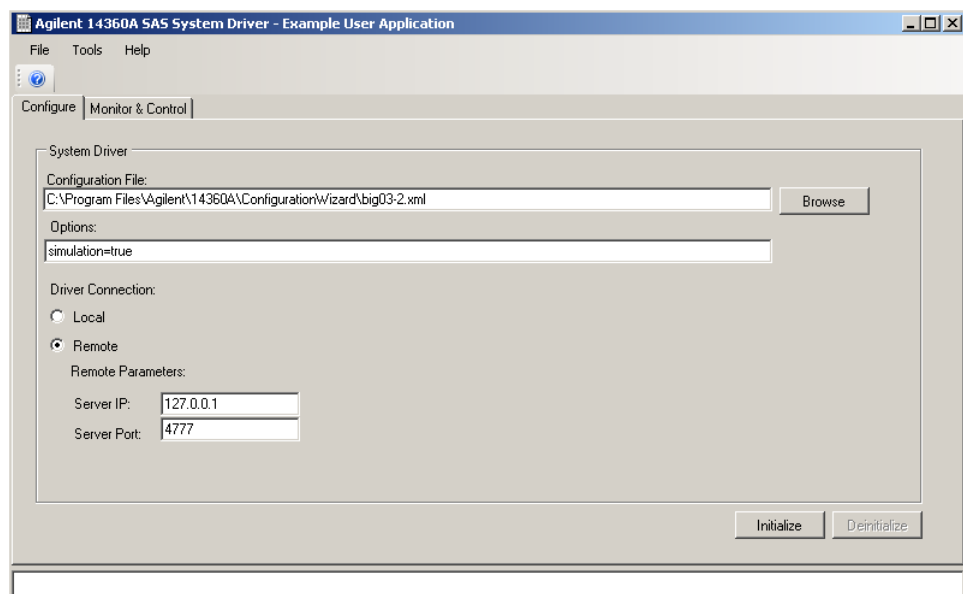
The Configure tab lets you browse for a configuration file for initializing the System Driver. Browse and select a configuration file.

Other fields let you specify initialization options and select either Local or Remote mode. Under Options, you can select from the following:

CheckInstError, CheckInstErrorOptional, CheckInstModelOnInit, CheckInstSerialOnInit, CheckModuleModelOnInit, CheckModuleSerialOnInit, CRCCheckOnInit, RebootTimeoutOnInit, ResetOnInit, ShutdownOnError, Simulation, and Timeout.

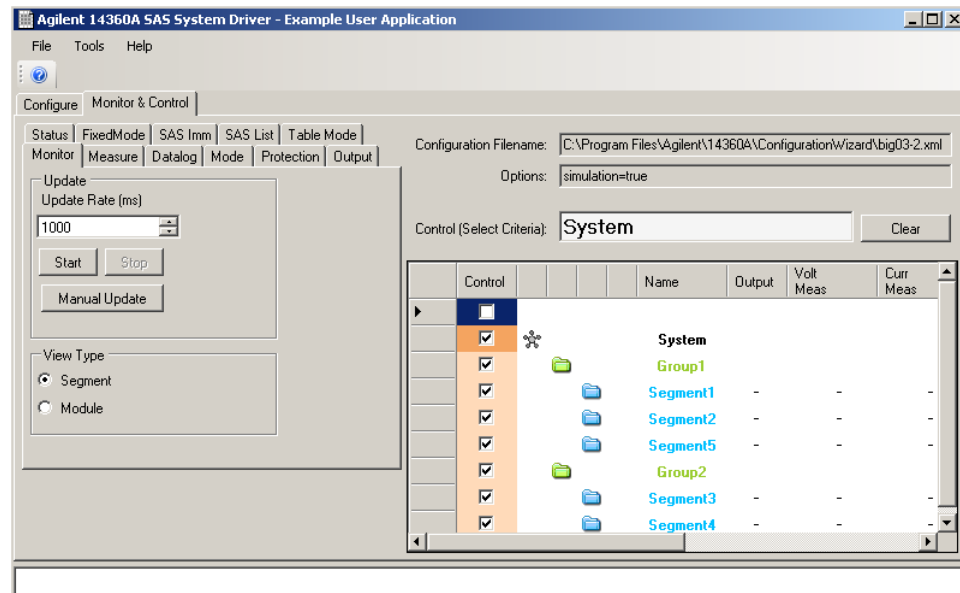
These are described in chapter 3 under Initialize Method.

After making your selections, click **Initialize** to initialize the System Driver.



## Monitor and Control Tab

After successfully initializing, the system can be monitored and controlled from this Tab window.



The tabs on the left side of the window correspond to the various functions of the system and the monitor window.

**Status** - lets you monitor the status indicators.

**Fixed Mode** - lets you set the Fixed mode voltage and current.

**SAS Imm** – lets you enter equation parameters for Immediate mode.

**SAS List** – lets you configure the SAS parameters for List mode.

**Table mode** – lets you enter voltage/current pairs for Table mode.

**Monitor** – specifies the update rate and view of the monitor window.

**Measure** - measures output voltage and current of the Results type.

**Datalog** – configures the data logger and specifies an output file.

**Mode** – selects the operating mode of the SAS.

**Protection** – configures the over-voltage, over-current and soft limits.

**Output** - enables/disable the output.

The right side of the window displays the active configuration file and options. The Control (Select Criteria) field displays the part of the system tree that is being controlled.

The spreadsheet area of the window displays the system monitor. The checkboxes let you select the Groups/Segments that will be controlled. The settings and results are filled in as they become available. Click **Clear** to clear the checkboxes.

## System Driver Usage Guide

This section describes some common tasks that are required for you to create an application program in the programming environment of your choice. All code samples are in C#. Most of the code samples are taken directly from the example program described previously.

You can view the source code of the example project by opening the .cs files located in “C:\Program Files\Agilent\14360A\Examples\SystemDriverexamples1\UserApplication” with a text editor.

### Adding a Reference

In your project, you must add a reference to the System Driver. In many development environments there is a section under Project Settings for adding references to outside dlls. The System Driver is a COM component named *Agilent 14360A 1.0 Type Library* that by default points to the file “C:\Program Files\Agilent\14360A\System Driver\Agilent14360A.dll”. In Visual Studio 2005 a project’s References can be found in the Solution Explorer under the Project Name.

The System Driver class, functions, interfaces, and enumerations are part of the Agilent14360 namespace. It is common to specify which namespaces a project uses. For example in C# programs this is done on top of source code files with the *using* keyword.

```
using Agilent14360A;
```

### Declaration and Instantiation

Next, you must declare and instantiate the System Driver in your project. Since the driver will probably be used across many functions, the driver variable should be declared as a class member variable or a global variable. When adding a COM component reference to a project, the development environment may automatically generate another layer on top of the COM interface called a “wrapper”. In .NET languages this is necessary to translate between the managed code of .NET and unmanaged code of the COM component. Thus, depending on your environment, the declaration of the System Driver variable may look a bit different. For example, in C# the System Driver is declared as the root interface of the driver:

```
private IAgilent14360 m_SystemDriver;
```

The variable is then instantiated as a new instance of CAgilent14360Class, a class generated by the .NET wrapper. In code this looks like:

```
m_SystemDriver = new CAgilent14360Class();
```

In other programming environments you may be declaring a variable of type CAgilent14360 and instantiating it as a new instance of CAgilent14360.

## Initialization

Once the System Driver is declared and instantiated, the System Driver functions are ready for use. The next step is to initialize the System Driver with a configuration file. The functions for initializing the driver can be found in the root interface `IAgilent14360`. The following are the three possible initialization scenarios. Assume that `ConfigFileTB`, `OptionsTB`, `ServerIPTB`, and `ServerPortTB` are textboxes in the application for the user to fill in with values.

### In Local Mode

```
m_SystemDriver.Initialize(ConfigFileTB.Text,
OptionsTB.Text);
```

The driver will communicate with the instruments in the system directly.

### In Remote Mode

```
m_SystemDriver.ConnectRemote(ServerIPTB.Text,
int.Parse(ServerPortTB.Text));
m_SystemDriver.Initialize(ConfigFileTB.Text,
OptionsTB.Text);
```

The driver first connects to the PC running the Server Control's RPC Server and then it initializes the Server Control's System Driver and itself. If an exception is thrown stating that "The Server Control's System Driver is already initialized with the configuration file: ..." you can forcefully close down the Server Control's System Driver by using the option "ForceRemoteClose=true" in the Initialize function.

### In Remote Mode (Monitor only)

```
m_SystemDriver.ConnectRemote(ServerIPTB.Text,
int.Parse(ServerPortTB.Text));
m_SystemDriver.Initialize("", "");
```

The driver first connects to the PC running the Server Control's RPC Server and then it initializes itself to communicate with the already initialized Server Control's System Driver. The Server Control's System Driver would have been initialized previously by another client using the preceding Remote Mode initialization scenario. You can check if the Server Control's System Driver is initialized with the use of the `GetInitialized()` Function. The Monitoring client does not have permission to call Set functions.

### Application Design

After a successful initialization, the System Driver is ready to send and receive data between the application and the SAS system. You can now control the SAS System with function calls such as:

```
m_SystemDriver.FixedMode.SetVoltLevel("GroupName=MyGroup1", 65);  
m_SystemDriver.Output.SetOutputEnabled("System", true);  
m_SystemDriver.Measurement.Measure("SegmentName=MySegment1",  
Agilent14360ResultsTypeEnum.Agilent14360ResultsTypeSegment, 1, out namesArr,  
out voltMeasArr, out currMeasArr);
```

All of these examples would work if the Configuration file the driver was initialized with contains a group named MyGroup1 and a segment named MySegment1. If the configuration file used for initialization did not contain those names, the driver would throw exceptions at run-time stating that none of the output modules in the system were selected for control.

To remedy this problem, the application could, at runtime, query the driver for names of all the groups, segments, and modules in the configuration. Refer to the [IAgilent14360DriverOperation](#) interface of the System Driver. The following code is an example of how to create in the application a data structure representing the hierarchy of Groups, Segments, and Modules.

```
public class Group {  
    public String Name;  
  
    /// <summary>  
    /// List of segments belonging to this group  
    /// </summary>  
    public List<Segment> Segments;  
  
    public Group() {  
        Segments = new List<Segment>();  
    }  
};  
  
public class Segment {  
    public String Name;  
  
    /// <summary>  
    /// List of modules belonging to this segment.  
    /// </summary>  
    public List<Module> Modules;  
  
    public Segment() {  
        Modules = new List<Module>();  
    }  
}  
  
public class Module {  
    public string Name;  
}
```



Three classes are declared, representing a group, a segment, and a module. A group has a list of member segments and a segment has a list of member modules.

```

/// <summary>
/// List of elements of type Group class. This will hold the tree
/// representation of the configuration's Groups, Segments, and Modules.
/// </summary>
private List<Group> m_Groups;

```

To represent the entire system of groups, segments, and modules, declare a list of type class Group.

```

/// <summary>
/// Populate in the application a data structure to represent
/// the tree of Groups, Segments, and Modules based on configuration file
/// passed to the system driver's initialize function.
/// </summary>
private void BuildGroupLists() {
    int groupCount, segmentCount, moduleCount;
    Group group;
    Segment segment;
    Module module;
    IAgilent14360DriverOperation Ido = m_SystemDriver.DriverOperation;

    m_Groups.Clear();

    /* get the number of groups in the configuration */
    groupCount = Ido.GetGroupCount();
    for (int g = 0; g < groupCount; g++) {
        group = new Group();
        m_Groups.Add(group);

        /* get the Name of the group at index g */
        group.Name = Ido.GetGroupNameAtIndex(g);

        segmentCount = Ido.GetGroupSegmentCount(group.Name);
        for (int s = 0; s < segmentCount; s++) {
            segment = new Segment();
            group.Segments.Add(segment);

            segment.Name = Ido.GetGroupSegmentNameAtIndex(group.Name, s);

            moduleCount = Ido.GetSegmentModuleCount(segment.Name);
            for (int m = 0; m < moduleCount; m++) {
                module = new Module();
                segment.Modules.Add(module);

                module.Name = Ido.GetSegmentModuleNameAtIndex(segment.Name,
m);
            }
        }
    }
}

```

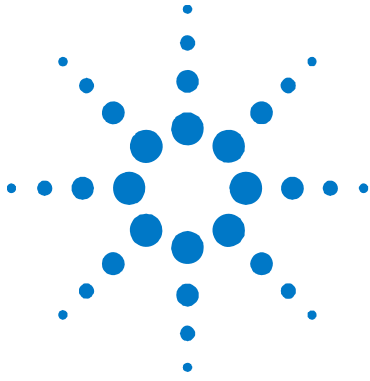
## 2 Using the System Control Tools

Finally, you must call the functions of `IAgilent14360DriverOperation` to retrieve all the group, segment, and module names.

With the application now having a representation of the hierarchy of groups, segments, and modules in the Configuration file, the System Driver function calls can have their `SelectCriteria` parameter created at runtime. Assume `bSelected` is a member variable of each of the three entities and is true if in the application the user specified that entity for control.

```
m_SystemDriver.Output.SetOutputEnabled("System", true);

foreach (Group group in m_Groups) {
    if (group.bSelected)
        m_SystemDriver.FixedMode.SetVoltLevel("GroupName=" + group.Name, 65);
    foreach (Segment segment in group.Segments) {
        if (segment.bSelected)
            m_SystemDriver.Measurement.Measure("SegmentName=" + segment.Name,
Agilent14360ResultsTypeEnum.Agilent14360ResultsTypeSegment, 1, out namesArr,
out voltMeasArr, out currMeasArr);
    }
}
```



### 3

## System Driver Functions

<a href="#">Driver Function Hierarchy</a> .....	36
<a href="#">SelectCriteria</a> .....	37
<a href="#">IAgilent14360 Interface</a> .....	39
<a href="#">IAgilent14360Dlog Interface</a> .....	45
<a href="#">IAgilent14360DriverOperation Interface</a> .....	47
<a href="#">IAgilent14360FixedMode Interface</a> .....	53
<a href="#">IAgilent14360Measurement Interface</a> .....	55
<a href="#">IAgilent14360Mode Interface</a> .....	56
<a href="#">IAgilent14360Output Interface</a> .....	58
<a href="#">IAgilent14360Protection Interface</a> .....	59
<a href="#">IAgilent14360SAS Interface</a> .....	63
<a href="#">IAgilent14360SASImmMode Interface</a> .....	65
<a href="#">IAgilent14360SASListMode Interface</a> .....	66
<a href="#">IAgilent14360Status Interface</a> .....	70
<a href="#">IAgilent14360System Interface</a> .....	72
<a href="#">IAgilent14360TableMode Interface</a> .....	75
<a href="#">IAgilent14360Transient Interface</a> .....	80

This chapter describes the System Driver functions. These driver functions are not available from the front panel or when using individual SCPI commands.

## Driver Function Hierarchy

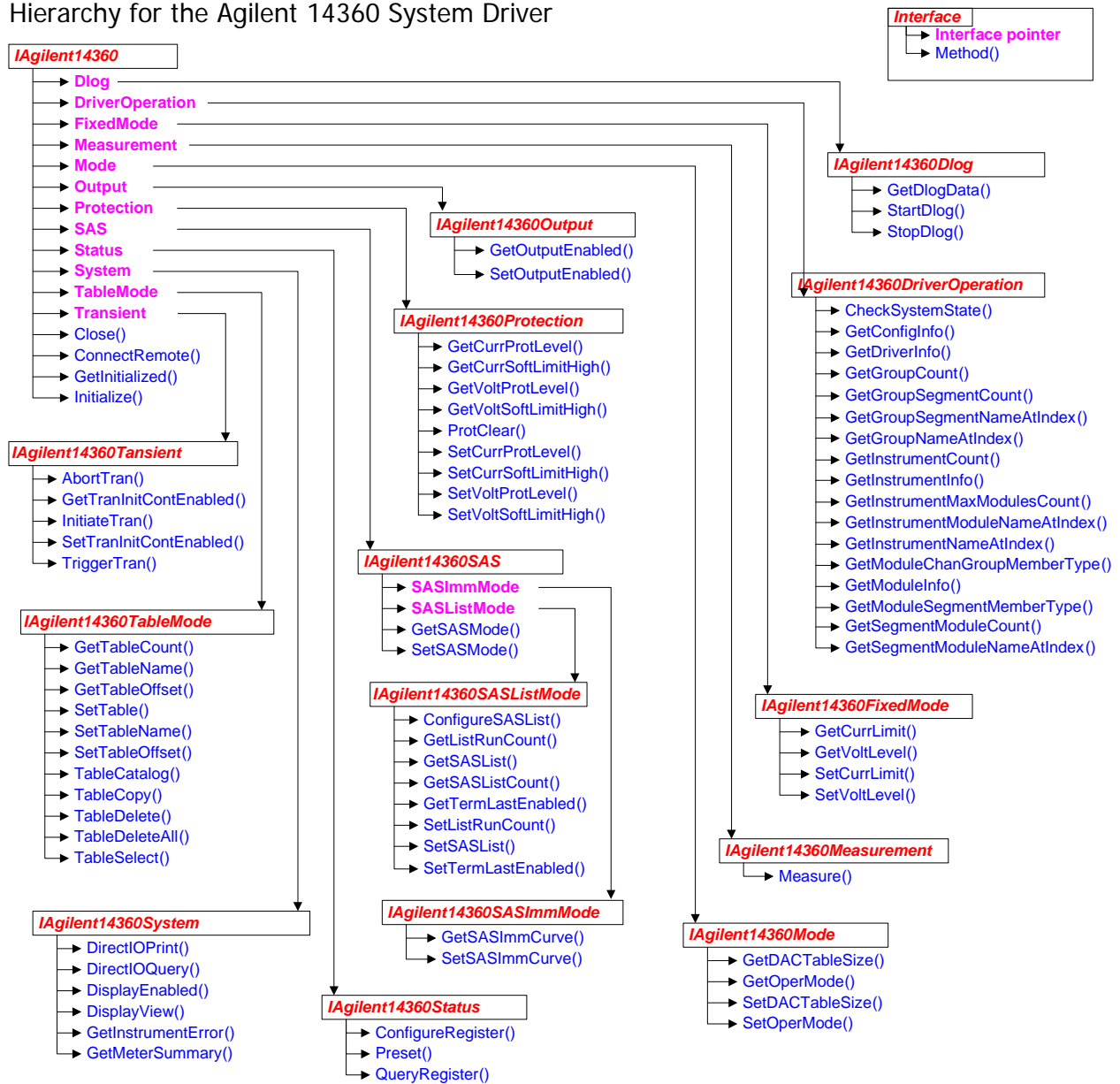
The driver functions are organized as follows:

**Interface** - An interface is a collection of related methods and properties.

**Properties** - Properties are used to access interface reference pointers that allow you to navigate the driver interface hierarchy.

**Methods** - Methods perform actions either within the System Driver or with a specific instrument. All work in the driver is done using methods.

Hierarchy for the Agilent 14360 System Driver



## SelectCriteria

### Controlling a System Subset

When using the System Driver functions, you need to specify with which configured modules you are communicating. This is done using the SelectCriteria string that is passed into the functions that require the destination of the command.

The format of the SelectCriteria string is semicolon separated key and value pairs -- “Key=Value”.

The following table summarizes the naming keys:

Key (Name)	SelectCriteria Example	Description
System	System (no value required)	Selects the entire system.
GroupName	GroupName=MyGroup1	Selects a group’s segments. Groups are referenced by name.
SegmentName	SegmentName=MySegment1	All the modules of a segment. Segments are referenced by name.
ModuleName	ModuleName=MyModule1	A single module (output channel).
InstrumentName	InstrumentName=Inst1[@<chanlist>]  Optional <chanlist> selects individual channels (@1; @2; @1,2; @1:2).	A single mainframe. To program individual modules on a mainframe, use the optional <chanlist>. Otherwise, no modules are programmed.

The following example sets the output voltage of all the modules in the group “MyGroup1” to 5V:

```
SetVoltLevel("GroupName=MyGroup1", 5.0);
```

The following example sets the output voltage of all the modules in the segment named “MySegment1” to 5V:

```
SetVoltLevel("SegmentName=MySegment1", 5.0);
```

The following example returns the error that occurred on the mainframe “Inst1”:

```
GetInstrumentError("InstrumentName=Inst1");
```

## Interpreting Function Parameters

All parameter values passed into the driver functions are interpreted based on the SelectCriteria. Most functions like SetVoltLevel or SetSASMode will not alter the passed-in parameters. However, with the functions dealing with current, like SetCurrLimit, the parameters will be altered if sent to a segment or a group but not when sent to a module directly.

The current parameter value is actually divided among the segment modules. This affects the values sent to each individual module as well as the values returned by the individual module when queried. The following is a summary of the interpretations and actions taken by the driver when SetCurrLimit and GetCurrLimit are programmed.

Action	Example	Description
Setting the current limit of a Group	SetCurrLimit ("GroupName=MyGroup1", 2.0)	The parameter value is passed to the group's segments, where it will be divided evenly between all the modules in the segment. With 4 modules, each module would be set to 0.5A.
Getting the current limit of a Group	GetCurrLimit ("GroupName=MyGroup1")	Returns the current passed to the group's segments before it was divided.
Setting the Current limit of a Segment	SetCurrLimit ("SegmentName=MySegment1", 2.0)	The parameter value is divided evenly between all the modules in the segment. If there were 4 modules, each module would be set to 0.5A.
Getting the current limit of a Segment	GetCurrLimit ("SegmentName=MySegment1")	Returns the current passed to the segment before it was divided.
Setting the current limit of a Module	SetCurrLimit ("ModuleName=Module1", 2.0)	The value is passed directly to the module (not divided).
Getting the current limit of a Module	GetCurrLimit ("ModuleName=Module1")	Returns the actual value that is passed to the module.

### Using Get functions with Groups and Segments

When Get functions are used along with either GroupName or SegmentName Select Criteria, they will return the requested value of all instruments and modules in that group or segment.

All of the values returned from the Group or Segment must be the same value. If one of the requested values is different from the others, an exception is generated. You must create an exception handler to deal with any exceptions that may occur.

## IAgilent14360 Interface

This is the root interface of the driver.

### Dlog Property

**Syntax:**

```
HRESULT Dlog([out, retval] IAgilent14360Dlog**
ppInterface);
```

**Description:**

Returns an interface reference of the driver instance.

**Return Value:**

IAgilent14360Dlog reference.

### DriverOperation Property

**Syntax:**

```
HRESULT DriverOperation([out, retval]
IAgilent14360DriverOperation** ppInterface);
```

**Description:**

Returns an interface reference of the driver instance.

**Return Value:**

IAgilent14360DriverOperation reference.

### FixedMode Property

**Syntax:**

```
HRESULT FixedMode([out, retval]
IAgilent14360FixedMode** ppInterface);
```

**Description:**

Returns an interface reference of the driver instance.

**Return Value:**

IAgilent14360FixedMode reference.

### Measurement Property

**Syntax:**

```
HRESULT Measurement([out, retval]
IAgilent14360Measurement** ppInterface);
```

**Description:**

Returns an interface reference of the driver instance.

**Return Value:**

IAgilent14360Measurement reference.

#### Mode Property

**Syntax:**

```
HRESULT Mode([out, retval] IAgilent14360Mode**  
ppInterface);
```

**Description:**

Returns an interface reference of the driver instance.

**Return Value:**

IAgilent14360Mode reference.

#### Output Property

**Syntax:**

```
HRESULT Output([out, retval] IAgilent14360Output**  
ppInterface);
```

**Description:**

Returns an interface reference of the driver instance.

**Return Value:**

IAgilent14360Output reference.

#### Protection Property

**Syntax:**

```
HRESULT Protection([out, retval]  
IAgilent14360Protection** ppInterface);
```

**Description:**

Returns an interface reference of the driver instance.

**Return Value:**

IAgilent14360Protection reference.

#### SAS Property

**Syntax:**

```
HRESULT SAS([out, retval] IAgilent14360SAS**  
ppInterface);
```

**Description:**

Returns an interface reference of the driver instance.

**Return Value:**

IAgilent14360SAS reference.



## Status Property

### Syntax:

```
HRESULT Status([out,retval] IAgilent14360Status**
ppInterface);
```

### Description:

Returns an interface reference of the driver instance.

### Return Value:

IAgilent14360Status reference.

## System Property

### Syntax:

```
HRESULT System([out, retval] IAgilent14360System**
ppInterface);
```

### Description:

Returns an interface reference of the driver instance.

### Return Value:

IAgilent14360System reference.

## TableMode Property

### Syntax:

```
HRESULT TableMode([out,retval]
IAgilent14360TableMode** ppInterface);
```

### Description:

Returns an interface reference of the driver instance.

### Return Value:

IAgilent14360TableMode reference.

## Transient Property

### Syntax:

```
HRESULT Transient([out, retval]
IAgilent14360Transient** ppInterface);
```

### Description:

Returns an interface reference of the driver instance.

### Return Value:

IAgilent14360Transient reference.

## Close Method

### Syntax:

```
HRESULT Close();
```

### Description:

Closes the driver session. In Local mode the driver disconnects from the instruments and de-initializes the driver. In Remote mode the Server Control local mode instance of the driver is de-initialized and connection to the Server Control RPC Server is closed.

### Return Value:

0: Success. Greater than 0: Error. Less than 0: Warning.

## ConnectRemote Method

### Syntax:

```
HRESULT ConnectRemote([in] BSTR ServerIP, [in] int ServerPort);
```

### Description:

Connects to the Server Control RPC Server. If ConnectRemote was called prior to calling Initialize, the driver will run in Remote mode and communicate with the instruments indirectly through the Server Control RPC Server.

### Parameters:

*ServerIP* IP address of the Server Control RPC Server.

*ServerPort* Port number of the Server Control RPC Server.

### Return Value:

0: Success. Greater than 0: Error. Less than 0: Warning

## GetInitialized Method

### Syntax:

```
HRESULT GetInitialized([out, retval] VARIANT_BOOL* pVal);
```

### Description:

Returns if the driver is initialized. In Local mode it returns if the driver initialized successfully with a configuration file. In Remote mode it returns if the Server Control local mode driver instance is initialized. You may call DriverOperation.GetConfigInfo to determine the active configuration file of the Server Control.

### Return Value:

true: Initialized. false. Not Initialized.

## Initialize Method

### Syntax:

```
HRESULT Initialize([in]BSTR ConfigurationFilename,
[in]BSTR Options);
```

### Description:

Initializes the driver. If ConnectRemote is not called prior to Initialize, the driver will run in Local mode and communicate with the instruments directly. If ConnectRemote was called prior to calling Initialize, the driver will run in Remote mode and communicate with the instruments indirectly through the Server Control RPC Server.

In case of multiple user-applications running in Remote mode and communicating through the Server Control, if the Server Control is already initialized with a Configuration file, pass in empty strings (or NULL) as the ConfigurationFilename and Options parameters for applications that will only be monitoring the system. Note that only the first Remote mode driver instance can control the instruments; additional driver instances can only monitor the instruments.

### Parameters:

*ConfigurationFilename* Filename of the configuration file used to initialize the driver.

*Options* List of options. The format of the Options string is semicolon separated key and value pairs -- "Key=Value".  
Example: "Simulation=True;Timeout=10000"

Key (Option name)	Description	Default value
CheckInstError	Check the error status register after function calls that could produce an instrument error - like a Set function.	True
CheckInstErrorOptional	Check the error status register after function calls that are unlikely to produce an instrument error - like a Get function.	False
CheckInstModelOnInit	Checks the instrument's model number and compares it with the model number stored in the configuration file.	True
CheckInstSerialOnInit	Checks the instrument's serial number and compares it with the serial number stored in the configuration file.	False
CheckModuleModelOnInit	Checks the output module's model number and compares it with the model number stored in the configuration file.	True
CheckModuleSerialOnInit	Checks the output module's serial number and compares it with the serial number stored in the configuration file.	False

### 3 System Driver Functions

Key (Option name)	Description	Default value
CRCCheckOnInit	Calculate the CRC checksum of the configuration file at initialize time and compare it to the CRC checksum stored in the configuration file. This verifies the data integrity of the file.	True
RebootTimeoutOnInit	The timeout in milliseconds after calling reboot in order to allow the instrument to restart its LAN connection. A reboot may be performed by the driver at initialize time.	25000
ShutdownOnError	Turn all outputs off when an error occurs.	False
Simulation	Enable instrument simulation. Only applies to Fixed mode.	False
Timeout	The timeout in milliseconds for instrument IO to respond	5000

**Return Value:**

0: Success. Greater than 0: Error. Less than 0: Warning.

## IAgilent14360Dlog Interface

This controls the Datalog functions.

### GetDlogData Method

**Syntax:**

```
HRESULT GetDlogData([in] BSTR SelectCriteria, [in]
Agilent14360ResultsTypeEnum ResultsType, [out,
satype("float")] SAFEARRAY** VoltArr, [out,
satype("float")] SAFEARRAY** CurrArr);
```

**Description:**

Retrieves the datalog data that is stored in the driver's buffers after a call to UpdateDlogData.

**Parameters:**

*SelectCriteria* Selection Criteria to specify the destination of the command. Only a single module can be selected.

*ResultsType* Results type: module or segment.

*VoltArr* Array that will be filled in with datalog voltage data.

*CurrArr* Array that will be filled in with datalog current data.

**Return Value:**

0: Success. Greater than 0: Error. Less than 0: Warning.

### StartDlog Method

**Syntax:**

```
HRESULT StartDlog([in] BSTR SelectCriteria, [in]
float TimeInterval);
```

**Description:**

Starts a new datalog. Any previously captured data is cleared.

**Parameters:**

*SelectCriteria* Selection Criteria to specify the destination of the command.

*TimeInterval* The time interval for capturing measurements into the datalog buffer. The range is from 0.02 seconds to 65 seconds.

**Return Value:**

0: Success. Greater than 0: Error. Less than 0: Warning.

## StopDlog Method

### Syntax:

```
HRESULT StopDlog([in] BSTR SelectCriteria);
```

### Description:

Stops a datalog that is in progress.

### Parameters:

*SelectCriteria* Selection Criteria to specify the destination of the command.

### Return Value:

0: Success. Greater than 0: Error. Less than 0: Warning.

## IAgilent14360DriverOperation Interface

These are the functions for retrieving information about the system based on the configuration.

### CheckSystemState Method

**Syntax:**

```
HRESULT CheckSystemState([in] VARIANT_BOOL
bOnErrorThrowException, [in] VARIANT_BOOL
bOnErrorShutdownSystem, [out, retval] VARIANT_BOOL*
bPassed);
```

**Description:**

Checks the instrument status registers for error conditions. The queries called are \*STB? to check for a presence of an error in the error queue, and STAT:QUES? (@chan) for indicators of OV, OC, PF, OT, OS, INH, UNR, and PROT.

**Parameters:**

*bOnErrorThrowException* Flag indicating whether or not to throw an exception if an error condition is found.

*bOnErrorShutdownSystem* Flag indicating whether or not to shutdown the system if an error condition is found.

**Return Value:**

true: Passed. false: Failed.

### GetConfigInfo Method

**Syntax:**

```
HRESULT GetConfigInfo([in]
Agilent14360ConfigInfoEnum InfoType, [out, retval]
BSTR* Info);
```

**Description:**

Gets configuration information based on InfoType.

**Parameters:**

*InfoType* Information type.

*Info* The string that will be filled in with the requested information.

**Return Value:**

0: Success. Greater than 0: Error. Less than 0: Warning.

## GetDriverInfo Method

**Syntax:**

```
HRESULT GetDriverInfo([in]  
Agilent14360DriverInfoEnum InfoType, [out, retval]  
BSTR* Info);
```

**Description:**

Gets driver information based on InfoType.

**Parameters:**

*InfoType* Information type.

*Info* The string that will be filled in with the requested information.

**Return Value:**

0: Success. Greater than 0: Error. Less than 0: Warning.

## GetGroupCount Method

**Syntax:**

```
HRESULT GetGroupCount([out, retval] int* count);
```

**Description:**

Gets the number of groups in the system.

**Return Value:**

Group count.

## GetGroupNameAtIndex Method

**Syntax:**

```
HRESULT GetGroupNameAtIndex([in] int Index, [out,  
retval] BSTR* Name);
```

**Description:**

Gets the group name at index.

**Parameters:**

*Index* The group index of system. The maximum index is GetGroupCount() - 1.

*Name* The group name.

**Return Value:**

0: Success. Greater than 0: Error. Less than 0: Warning.



## GetGroupSegmentCount Method

### Syntax:

```
HRESULT GetGroupSegmentCount([in] BSTR GroupName,
[out, retval] int* count);
```

### Description:

Gets the number of segments of a group.

### Parameters:

*GroupName* The group's name.

### Return Value:

Segment count.

## GetGroupSegmentNameAtIndex Method

### Syntax:

```
HRESULT GetGroupSegmentNameAtIndex([in] BSTR
GroupName, [in] int Index, [out, retval] BSTR*
Name);
```

### Description:

Gets the name of a segment of a group at index.

### Parameters:

*GroupName* The group's name.

*Index* The segment index of the group. The maximum index is GetGroupSegmentCount() - 1.

### Return Value:

Segment name.

## GetInstrumentCount Method

### Syntax:

```
HRESULT GetInstrumentCount([out, retval] int*
count);
```

### Description:

Gets the number of instruments in the system. Only instruments that are part of groups are included.

### Return Value:

Instrument count.

## GetInstrumentInfo Method

**Syntax:**

```
HRESULT GetInstrumentInfo([in] BSTR Name, [in] Agilent14360InstrumentInfoEnum InfoType, [out, retval] BSTR* Info);
```

**Description:**

Gets instrument information based on the InfoType.

**Parameters:**

*Name* The instrument's name

*InfoType* Information type.

*Info* The string that will be filled in with the requested information.

**Return Value:**

0: Success. Greater than 0: Error. Less than 0: Warning.

## GetInstrumentMaxModulesCount Method

**Syntax:**

```
HRESULT GetInstrumentMaxModulesCount([in] BSTR InstrumentName, [out, retval] int* count);
```

**Description:**

Gets the maximum number of modules of an instrument.

**Parameters:**

*InstrumentName* The instrument's name.

**Return Value:**

Max modules count.

## GetInstrumentModuleNameAtIndex Method

**Syntax:**

```
HRESULT GetInstrumentModuleNameAtIndex([in] BSTR InstrumentName, [in] int Index, [out, retval] BSTR* Name);
```

**Description:**

Gets the name of the module of the instrument at index.

**Parameters:**

*InstrumentName* The instrument's name.

*Index* Module index of the instrument. The maximum index is GetInstrumentMaxModulesCount() - 1.

**Return Value:**

Module name.

## GetInstrumentNameAtIndex Method

### Syntax:

```
HRESULT GetInstrumentNameAtIndex([in] int Index,
[out, retval] BSTR* Name);
```

### Description:

Get the name of the instrument at index.

### Parameters:

*Index* Instrument index of system. The max index is GetInstrumentCount() - 1.

### Return Value:

Instrument name.

## GetModuleChanGroupMemberType Method

### Syntax:

```
HRESULT GetModuleChanGroupMemberType([in] BSTR
ModuleName, [out, retval]
Agilent14360ModuleChanGroupMemberEnum* Type);
```

### Description:

Gets the channel group member type of module.

### Parameters:

*ModuleName* The module's name.

### Return Value:

Channel group member type of type Agilent14360ModuleChanGroupMemberEnum.

## GetModuleInfo Method

### Syntax:

```
HRESULT GetModuleInfo([in] BSTR Name, [in]
Agilent14360ModuleInfoEnum InfoType, [out, retval]
BSTR* Info);
```

### Description:

Gets module information based on InfoType.

### Parameters:

*Name* The module's name.

*InfoType* Information type.

*Info* The string that will be filled in with the requested information.

### Return Value:

0: Success. Greater than 0: Error. Less than 0: Warning.

## GetModuleSegmentMemberType Method

**Syntax:**

```
HRESULT GetModuleSegmentMemberType([in] BSTR  
ModuleName, [out, retval]  
Agilent14360ModuleSegmentMemberEnum* Type);
```

**Description:**

Gets the segment member type of module.

**Parameters:**

*ModuleName* The module's name.

**Return Value:**

Segment member type of type  
Agilent14360ModuleSegmentMemberEnum.

## GetSegmentModuleCount Method

**Syntax:**

```
HRESULT GetSegmentModuleCount([in] BSTR SegmentName,  
[out, retval] int* count);
```

**Description:**

Gets the number of modules of a segment.

**Parameters:**

*SegmentName* The segment's name.

**Return Value:**

Module count.

## GetSegmentModuleNameAtIndex Method

**Syntax:**

```
HRESULT GetSegmentModuleNameAtIndex([in] BSTR  
SegmentName, [in] int Index, [out, retval] BSTR*  
Name);
```

**Description:**

Gets the module name of a segment at index.

**Parameters:**

*SegmentName* The segment's name.

*Index* The module index of a segment. The maximum index is  
GetSegmentModuleCount() - 1.

**Return Value:**

Module name.

## IAgilent14360FixedMode Interface

This controls the Fixed operating mode.

### GetCurrLimit Method

**Syntax:**

```
HRESULT GetCurrLimit([in] BSTR SelectCriteria, [out, retval] float* pVal);
```

**Description:**

Gets the current limit in Fixed mode.

**Parameters:**

*SelectCriteria* Selection Criteria to specify the destination of the command. Only a single module can be selected.

**Return Value:**

Current limit.

### GetVoltLevel Method

**Syntax:**

```
HRESULT GetVoltLevel([in] BSTR SelectCriteria, [out, retval] float* pVal);
```

**Description:**

Gets the voltage level in Fixed mode.

**Parameters:**

*SelectCriteria* Selection Criteria to specify the destination of the command. Only a single module can be selected.

**Return Value:**

Voltage level.

#### SetCurrLimit Method

**Syntax:**

```
HRESULT SetCurrLimit([in] BSTR SelectCriteria, [in]  
float newVal);
```

**Description:**

Sets the current limit in Fixed mode.

**Parameters:**

*SelectCriteria* Selection Criteria to specify the destination of the command.

*newVal* The current limit.

**Return Value:**

0: Success. Greater than 0: Error. Less than 0: Warning.

#### SetVoltLevel Method

**Syntax:**

```
HRESULT SetVoltLevel([in] BSTR SelectCriteria, [in]  
float newVal);
```

**Description:**

Sets the voltage level in Fixed mode.

**Parameters:**

*SelectCriteria* Selection Criteria to specify the destination of the command.

*newVal* The voltage level.

**Return Value:**

0: Success. Greater than 0: Error. Less than 0: Warning.

## IAgilent14360Measurement Interface

This controls the measurement functions.

### Measure Method

#### Syntax:

```
HRESULT Measure([in] BSTR SelectCriteria, [in]
Agilent14360ResultsTypeEnum ResultsType, [out,
satype("BSTR")] SAFEARRAY** NameArray, [out,
satype("float")] SAFEARRAY** VoltMeasArray, [out,
satype("float")] SAFEARRAY** CurrMeasArray);
```

#### Description:

Do a voltage and current measurement in the specified modules. Each measurement result based on ResultsType is a triplet (Name, VoltMeas, and CurrMeas).

#### Parameters:

*SelectCriteria* Selection Criteria to specify the destination of the command. Only a single module can be selected.

*ResultsType* Results type of type Agilent14360ResultsTypeEnum.

*NameArray* Pointer to an array of size ResultsCount with the name of each corresponding voltage and current measurement.

*VoltMeasArray* Pointer to an array of size ResultsCount with the voltage measurements.

*CurrMeasArray* Pointer to an array of size ResultsCount with the current measurements.

#### Return Value:

0: Success. Greater than 0: Error. Less than 0: Warning.

## IAgilent14360Mode Interface

This controls the operating mode.

### GetDACTableSize Method

**Syntax:**

```
HRESULT GetDACTableSize([in] BSTR SelectCriteria,  
[out, retval] Agilent14360DACTableSizeEnum* pVal);
```

**Description:**

Gets the digital-to-analog converter table size for SAS and Table modes.

**Parameters:**

*SelectCriteria* Selection Criteria to specify the destination of the command. Only a single module can be selected.

**Return Value:**

Table size of type Agilent14360DACTableSizeEnum.

### GetOperMode Method

**Syntax:**

```
HRESULT GetOperMode([in] BSTR SelectCriteria, [out,  
retval] Agilent14360OperModeEnum* pVal);
```

**Description:**

Gets the operating mode.

**Parameters:**

*SelectCriteria* Selection Criteria to specify the destination of the command. Only a single module can be selected.

**Return Value:**

Operating mode of type Agilent14360OperModeEnum.



## SetDACTableSize Method

### Syntax:

```
HRESULT SetDACTableSize([in] BSTR SelectCriteria,
[in] Agilent14360DACTableSizeEnum newVal);
```

### Description:

Sets the digital-to-analog converter table size for SAS and Table modes.

### Parameters:

*SelectCriteria* Selection Criteria to specify the destination of the command.

*newVal* Table size of type Agilent14360DACTableSizeEnum.

### Return Value:

0: Success. Greater than 0: Error. Less than 0: Warning.

## SetOperMode Method

### Syntax:

```
HRESULT SetOperMode([in] BSTR SelectCriteria, [in]
Agilent14360OperModeEnum newVal);
```

### Description:

Sets the operating mode.

### Parameters:

*SelectCriteria* Selection Criteria to specify the destination of the command.

*newVal* Operating mode of type Agilent14360OperModeEnum.

### Return Value:

0: Success. Greater than 0: Error. Less than 0: Warning.

## IAgilent14360Output Interface

This controls the output state and related functions.

### GetOutputEnabled Method

**Syntax:**

```
HRESULT GetOutputEnabled([in] BSTR SelectCriteria,  
[out, retval] VARIANT_BOOL* pVal);
```

**Description:**

Gets the output enable state.

**Parameters:**

*SelectCriteria* Selection Criteria to specify the destination of the command. Only a single module can be selected.

**Return Value:**

True: enabled. False: disabled.

### SetOutputEnabled Method

**Syntax:**

```
HRESULT SetOutputEnabled([in] BSTR SelectCriteria,  
[in] VARIANT_BOOL Enabled);
```

**Description:**

Sets the output enabled state.

**Parameters:**

*SelectCriteria* Selection Criteria to specify the destination of the command.

*Enabled* True: enabled. False: disabled.

**Return Value:**

0: Success. Greater than 0: Error. Less than 0: Warning.

## IAgilent14360Protection Interface

This controls the protection functions of the instruments and modules.

### GetCurrProtLevel Method

**Syntax:**

```
HRESULT GetCurrProtLevel([in] BSTR SelectCriteria,  
[out, retval] float* pVal);
```

**Description:**

Gets the current protection level.

**Parameters:**

*SelectCriteria* Selection Criteria to specify the destination of the command. Only a single module can be selected.

**Return Value:**

The current protection level.

### GetCurrSoftLimitHigh Method

**Syntax:**

```
HRESULT GetCurrSoftLimitHigh([in] BSTR  
SelectCriteria, [out, retval] float* pVal);
```

**Description:**

Gets the current high soft limit.

**Parameters:**

*SelectCriteria* Selection Criteria to specify the destination of the command. Only a single module can be selected.

**Return Value:**

The current high soft limit.

## GetVoltProtLevel Method

**Syntax:**

```
HRESULT GetVoltProtLevel([in] BSTR SelectCriteria,  
[out, retval] float* pVal);
```

**Description:**

Gets the voltage protection level.

**Parameters:**

*SelectCriteria* Selection Criteria to specify the destination of the command. Only a single module can be selected.

**Return Value:**

The voltage protection level.

## GetVoltSoftLimitHigh Method

**Syntax:**

```
HRESULT GetVoltSoftLimitHigh([in] BSTR  
SelectCriteria, [out, retval] float* pVal);
```

**Description:**

Gets the voltage high soft limit.

**Parameters:**

*SelectCriteria* Selection Criteria to specify the destination of the command. Only a single module can be selected.

**Return Value:**

The voltage high soft limit.

## ProtClear Method

**Syntax:**

```
HRESULT ProtClear([in] BSTR SelectCriteria);
```

**Description:**

Clears output protection.

**Parameters:**

*SelectCriteria* Selection Criteria to specify the destination of the command.

**Return Value:**

0: Success. Greater than 0: Error. Less than 0: Warning.

## SetCurrProtLevel Method

**Syntax:**

```
HRESULT SetCurrProtLevel([in] BSTR SelectCriteria,  
[in] float newVal);
```

**Description:**

Sets the current protection level.

**Parameters:**

*SelectCriteria* Selection Criteria to specify the destination of the command.

*newVal* The current protection level.

**Return Value:**

0: Success. Greater than 0: Error. Less than 0: Warning.

## SetCurrSoftLimitHigh Method

**Syntax:**

```
HRESULT SetCurrSoftLimitHigh([in] BSTR  
SelectCriteria, [in] float newVal);
```

**Description:**

Sets the current high soft limit.

**Parameters:**

*SelectCriteria* Selection Criteria to specify the destination of the command.

*newVal* The current high soft limit.

**Return Value:**

0: Success. Greater than 0: Error. Less than 0: Warning.

#### SetVoltProtLevel Method

**Syntax:**

```
HRESULT SetVoltProtLevel([in] BSTR SelectCriteria,  
[in] float newVal);
```

**Description:**

Sets the voltage protection level.

**Parameters:**

*SelectCriteria* Selection Criteria to specify the destination of the command.

*newVal* The voltage protection level.

**Return Value:**

0: Success. Greater than 0: Error. Less than 0: Warning.

#### SetVoltSoftLimitHigh Method

**Syntax:**

```
HRESULT SetVoltSoftLimitHigh([in] BSTR  
SelectCriteria, [in] float newVal);
```

**Description:**

Sets the voltage high soft limit.

**Parameters:**

*SelectCriteria* Selection Criteria to specify the destination of the command.

*newVal* The voltage high soft limit.

**Return Value:**

0: Success. Greater than 0: Error. Less than 0: Warning.

## IAgilent14360SAS Interface

Controls the SAS Immediate and SAS List modes.

### SASImmMode Property

**Syntax:**

```
HRESULT SASImmMode([out,retval]
IAgilent14360SASImmMode** ppInterface);
```

**Description:**

Returns an interface reference of the driver instance.

**Return Value:**

IAgilent14360SASImmMode reference.

### SASListMode Property

**Syntax:**

```
HRESULT SASListMode([out,retval]
IAgilent14360SASListMode** ppInterface);
```

**Description:**

Returns an interface reference of the driver instance.

**Return Value:**

IAgilent14360SASListMode reference.

### GetSASMode Method

**Syntax:**

```
HRESULT GetSASMode([in] BSTR SelectCriteria, [out,
retval] Agilent14360SASModeEnum* pVal);
```

**Description:**

Gets the SAS mode.

**Parameters:**

*SelectCriteria* Selection Criteria to specify the destination of the command. Only a single module can be selected.

**Return Value:**

The current SAS mode of type Agilent14360SASModeEnum.

## SetSASMode Method

### Syntax:

```
HRESULT SetSASMode([in] BSTR SelectCriteria, [in]  
Agilent14360SASModeEnum newVal);
```

### Description:

Sets the SAS mode.

### Parameters:

*SelectCriteria* Selection Criteria to specify the destination of the command.

*newVal* The current SAS mode of type Agilent14360SASModeEnum.

### Return Value:

0: Success. Greater than 0: Error. Less than 0: Warning.



## IAgilent14360SASImmMode Interface

This controls the SAS Immediate mode.

### GetSASImmCurve Method

**Syntax:**

```
HRESULT GetSASImmCurve([in] BSTR SelectCriteria,
[out] float* CurrentSAS_ISC, [out] float*
CurrentSAS_IMP, [out] float* VoltageSAS_VOC, [out]
float* VoltageSAS_VMP);
```

**Description:**

Gets the SAS Immediate mode curve parameters.

**Parameters:**

*SelectCriteria* Selection Criteria to specify the destination of the command. Only a single module can be selected.

*CurrentSAS\_ISC* ISC (short-circuit current).

*CurrentSAS\_IMP* IMP (current at maximum power).

*VoltageSAS\_VOC* VOC (open-circuit voltage).

*VoltageSAS\_VMP* VMP (voltage at maximum power).

**Return Value:**

0: Success. Greater than 0: Error. Less than 0: Warning.

### SetSASImmCurve Method

**Syntax:**

```
HRESULT SetSASImmCurve([in] BSTR SelectCriteria,
[in] float CurrentSAS_ISC, [in] float
CurrentSAS_IMP, [in] float VoltageSAS_VOC, [in]
float VoltageSAS_VMP);
```

**Description:**

Sets the SAS Immediate mode curve parameters.

**Parameters:**

*SelectCriteria* Selection Criteria to specify the destination of the command. Only a single module can be selected.

*CurrentSAS\_ISC* ISC (short-circuit current).

*CurrentSAS\_IMP* IMP (current at maximum power).

*VoltageSAS\_VOC* VOC (open-circuit voltage).

*VoltageSAS\_VMP* VMP (voltage at maximum power).

**Return Value:**

0: Success. Greater than 0: Error. Less than 0: Warning.

## IAgilent14360SASListMode Interface

This controls the SAS List mode.

### ConfigureSASList

**Syntax:**

```
HRESULT ConfigureSASList([in] BSTR SelectCriteria, [in] Agilent14360ListStepEnum StepType);
```

**Description:**

Sets the SAS list trigger step type. The trigger parameters are then configured based on the step type. Call this function prior to `Transient.InitiateTran`.

**Parameters:**

*SelectCriteria* Selection Criteria to specify the destination of the command.

*StepType* Step type of type `Agilent14360ListStepEnum`.

**Return Value:**

0: Success. Greater than 0: Error. Less than 0: Warning.

### GetListRunCount Method

**Syntax:**

```
HRESULT GetListCount([in] BSTR SelectCriteria, [out, retval] int* pVal);
```

**Description:**

Gets the SAS List mode list run count. A value greater than 256 is considered to be infinity.

**Parameters:**

*SelectCriteria* Selection Criteria to specify the destination of the command. Only a single module can be selected.

**Return Value:**

Run count.

## GetSASList Method

### Syntax:

```
HRESULT GetSASList([in] BSTR SelectCriteria, [in]
Agilent14360SASListEnum ListType, [out, retval,
safetype("float")] SAFEARRAY** List);
```

### Description:

Gets the SAS List points.

### Parameters:

*SelectCriteria* Selection Criteria to specify the destination of the command. Only a single module can be selected.

*ListType* List type of type Agilent14360SASListEnum.

*List* Pointer to an array of size ElementsCount to be filled with list points.

### Return Value:

The number of list points returned.

## GetSASListCount Method

### Syntax:

```
HRESULT GetSASListCount([in] BSTR SelectCriteria,
[in] Agilent14360SASListEnum ListType, [out, retval]
int* Count);
```

### Description:

Gets the SAS List points count.

### Parameters:

*SelectCriteria* Selection Criteria to specify the destination of the command. Only a single module can be selected.

*ListType* List type of type Agilent14360SASListEnum.

### Return Value:

The list points count.

#### GetTermLastEnabled Method

**Syntax:**

```
HRESULT GetTermLastEnabled([in] BSTR SelectCriteria,  
[out, retval] VARIANT_BOOL* pVal);
```

**Description:**

Gets the SAS List Mode terminate on last list point enabled.

**Parameters:**

*SelectCriteria* Selection Criteria to specify the destination of the command. Only a single module can be selected.

**Return Value:**

True: enabled. False: disabled.

#### SetListRunCount Method

**Syntax:**

```
HRESULT SetListRunCount([in] BSTR SelectCriteria,  
[in] int newVal);
```

**Description:**

Sets the SAS List mode list run count. A value greater than 256 is considered to be infinity.

**Parameters:**

*SelectCriteria* Selection Criteria to specify the destination of the command.

*newVal* Run count.

**Return Value:**

0: Success. Greater than 0: Error. Less than 0: Warning.

## SetSASList Method

### Syntax:

```
HRESULT SetSASList([in] BSTR SelectCriteria, [in]
Agilent14360SASListEnum ListType, [in,
safetype("float")] SAFEARRAY* List);
```

### Description:

Sets the SAS List points.

### Parameters:

*SelectCriteria* Selection Criteria to specify the destination of the command.

*ListType* List type of type Agilent14360SASListEnum.

*List* Pointer to an array of size ElementsCount with the list points.

### Return Value:

0: Success. Greater than 0: Error. Less than 0: Warning.

## SetTermLastEnabled Method

### Syntax:

```
HRESULT SetTermLastEnabled([in] BSTR SelectCriteria,
[in] VARIANT_BOOL newVal);
```

### Description:

Sets the SAS List Mode terminate on last list point enabled.

### Parameters:

*SelectCriteria* Selection Criteria to specify the destination of the command.

*newVal* True: enabled. False: disabled.

### Return Value:

0: Success. Greater than 0: Error. Less than 0: Warning.

## IAgilent14360Status Interface

This controls the status functions.

### ConfigureRegister Method

**Syntax:**

```
HRESULT ConfigureRegister([in] BSTR SelectCriteria,  
[in] Agilent14360StatusRegisterEnum Register, [in]  
Agilent14360StatusSubRegisterEnum SubRegister, [in]  
int newVal);
```

**Description:**

Sets the value of the specified register's subregister. See the SCPI Programmers Reference for an explanation of the status system.

**Parameters:**

*SelectCriteria* Selection Criteria to specify the destination of the command.

*Register* The register type.

*SubRegister* The subregister type.

*newVal* Register value.

**Return Value:**

0: Success. Greater than 0: Error. Less than 0: Warning.

### Preset Method

**Syntax:**

```
HRESULT Preset([in] BSTR SelectCriteria);
```

**Description:**

Sets the status registers to their default values.

**Parameters:**

*SelectCriteria* Selection Criteria to specify the destination of the command.

**Return Value:**

0: Success. Greater than 0: Error. Less than 0: Warning.

## QueryRegister Method

### Syntax:

```
HRESULT QueryRegister([in] BSTR SelectCriteria, [in] Agilent14360StatusRegisterEnum Register, [in] Agilent14360StatusSubRegisterEnum SubRegister, [out, retval] int* pVal);
```

### Description:

Gets the value of the specified register's subregister. See the SCPI Programmers Reference for an explanation of the status system.

### Parameters:

*SelectCriteria* Selection Criteria to specify the destination of the command. Only a single module can be selected.

*Register* The register type.

*SubRegister* The subregister type.

### Return Value:

Register value.

## IAgilent14360System Interface

This controls the system functions.

### DirectIOPrint Method

**Syntax:**

```
HRESULT DirectIOPrint([in] BSTR SelectCriteria, [in] BSTR Command);
```

**Description:**

Sends a SCPI command.

**Parameters:**

*SelectCriteria* Selection Criteria to specify the destination of the command.

*Command* The SCPI command.

**Return Value:**

0: Success. Greater than 0: Error. Less than 0: Warning.

### DirectIOQuery Method

**Syntax:**

```
HRESULT DirectIOQuery([in] BSTR SelectCriteria, [in] BSTR Query, [in] int TimeoutMS, [in] int BufferSize, [out, retval] BSTR* Response);
```

**Description:**

Sends a SCPI query and gets the response.

**Parameters:**

*SelectCriteria* Selection Criteria to specify the destination of the command.

*Query* The SCPI query.

*TimeoutMS* IO timeout in milliseconds. Default is 5000 ms.

*BufferSize* The maximum string length to expect as a response from the instrument.

*Response* The string buffer that will be filled in with the response from the instrument.

**Return Value:**

The SCPI response.



## DisplayEnabled Method

### Syntax:

```
HRESULT DisplayEnabled([in] BSTR SelectCriteria,
[in] VARIANT_BOOL Enabled);
```

### Description:

Enables or disables the display on the instrument.

### Parameters:

*SelectCriteria* Selection Criteria to specify the destination of the command.

*Enabled* True: enabled. False: disabled.

### Return Value:

0: Success. Greater than 0: Error. Less than 0: Warning.

## DisplayView Method

### Syntax:

```
HRESULT DisplayView([in] BSTR SelectCriteria, [in]
Agilent14360DisplayViewEnum View);
```

### Description:

Sets the display view of an instrument.

### Parameters:

*SelectCriteria* Selection Criteria to specify the destination of the command.

*View* View type of type Agilent14360DisplayViewEnum.

### Return Value:

0: Success. Greater than 0: Error. Less than 0: Warning.

## GetInstrumentError Method

### Syntax:

```
HRESULT GetInstrumentError([in] BSTR SelectCriteria,
[out, retval] BSTR* ErrorMessage);
```

### Description:

Gets the instrument's first error from the error queue.

### Parameters:

*SelectCriteria* Selection Criteria to specify the destination of the command. Only a single module can be selected.

*ErrorMessage* The error message.

### Return Value:

0: Success. Greater than 0: Error. Less than 0: Warning.

## GetMeterSummary Method

### Syntax:

```
HRESULT GetMeterSummary([in] BSTR SelectCriteria,  
[out] VARIANT_BOOL* bOutputEnabled, [out] float*  
fVoltMeas, [out] float* fCurrMeas, [out]  
Agilent14360CurrModeEnum* Mode, [out] float*  
fVoltSetting, [out] float* fCurrSetting, [out,  
retval] BSTR* Status);
```

### Description:

Gets the summary values of a module's front panel meter. This function utilizes instrument IO bandwidth more efficiently than individual queries of each value and should be used in applications like a system meter view that are frequently refreshed.

### Parameters:

*SelectCriteria* Selection Criteria to specify the destination of the command.

*bOutputEnabled* Output enabled state.

*fVoltMeas* Front panel voltage measurement.

*fCurrMeas* Front panel current measurement.

*Mode* Current mode of type Agilent14360CurrModeEnum.

*fVoltSetting* The voltage setting of the current mode. In Fixed mode this is the volt level. In SAS and Table modes this is VMP.

*fCurrSetting* The current setting of the current mode. In Fixed mode this is the current limit. In SAS and Table modes this is IMP.

*Status*

The status of the module. Values include "" when output is off, "CV", "CC", "OV", "OC", "PF", "OT", "OS", "INH", "UNR", "PROT", and "Unknown".

### Return Value:

The SCPI response.

## IAgilent14360TableMode Interface

This controls the Table functions.

### GetTableCount Method

**Syntax:**

```
HRESULT GetTableCount([in] BSTR SelectCriteria, [in]
Agilent14360TableEnum TableType, [out, retval] float*
pVal);
```

**Description:**

Get the selected table's points count.

**Parameters:**

*SelectCriteria*

Selection Criteria to specify the destination of the command. Only a single module can be selected.

*TableType* Table type of type Agilent14360TableEnum.

**Return Value:**

The table points count.

### GetTableName Method

**Syntax:**

```
HRESULT GetTableName([in] BSTR SelectCriteria, [out,
retval] BSTR* Name);
```

**Description:**

Gets the active table name.

**Parameters:**

*SelectCriteria* Selection Criteria to specify the destination of the command. Only a single module can be selected.

*Name* The Table name.

**Return Value:**

0: Success. Greater than 0: Error. Less than 0: Warning.

#### GetTableOffset Method

**Syntax:**

```
HRESULT GetTableOffset([in] BSTR SelectCriteria,  
[in] Agilent14360TableEnum TableType, [out, retval]  
float* pVal);
```

**Description:**

Gets the selected table's offset.

**Parameters:**

*SelectCriteria* Selection Criteria to specify the destination of the command. Only a single module can be selected.

*TableType* Table type of type Agilent14360TableEnum.

**Return Value:**

The table offset.

#### SetTable Method

**Syntax:**

```
HRESULT SetTable([in] BSTR SelectCriteria, [in]  
Agilent14360TableEnum TableType, [in,  
safetype("float")] SAFEARRAY* Points);
```

**Description:**

Set the selected table's points.

**Parameters:**

*SelectCriteria* Selection Criteria to specify the destination of the command.

*TableType* Table type of type Agilent14360TableEnum.

*Values* Array with the new point values.

**Return Value:**

0: Success. Greater than 0: Error. Less than 0: Warning.

## SetTableName Method

### Syntax:

```
HRESULT SetTableName([in] BSTR SelectCriteria, [in]
BSTR Name);
```

### Description:

Set the active table name.

### Parameters:

*SelectCriteria* Selection Criteria to specify the destination of the command.

*Name* The Table name.

### Return Value:

0: Success. Greater than 0: Error. Less than 0: Warning.

## SetTableOffset Method

### Syntax:

```
HRESULT SetTableOffset([in] BSTR SelectCriteria,
[in] Agilent14360TableEnum TableType, [in] float
newVal);
```

### Description:

Set the selected table's offset.

### Parameters:

*SelectCriteria* Selection Criteria to specify the destination of the command.

*TableType* Table type of type Agilent14360TableEnum.

*newValue* The table offset.

### Return Value:

0: Success. Greater than 0: Error. Less than 0: Warning.

#### TableCatalog Method

**Syntax:**

```
HRESULT TableCatalog([in] BSTR SelectCriteria, [out, retval] BSTR* Catalog);
```

**Description:**

Returns the names of all user-defined tables in both volatile and non-volatile memory.

**Parameters:**

*SelectCriteria* Selection Criteria to specify the destination of the command.

*Catalog* A string with comma-separated table names.

**Return Value:**

0: Success. Greater than 0: Error. Less than 0: Warning.

#### TableCopy Method

**Syntax:**

```
HRESULT TableCopy([in] BSTR SelectCriteria, [in] BSTR Name);
```

**Description:**

Copies the active table to non-volatile memory with the specified name.

**Parameters:**

*SelectCriteria* Selection Criteria to specify the destination of the command.

*Name* The Table name.

**Return Value:**

0: Success. Greater than 0: Error. Less than 0: Warning.

## TableDelete Method

### Syntax:

```
HRESULT TableDelete([in] BSTR SelectCriteria, [in]
BSTR Name);
```

### Description:

Delete the table with the specified name from memory.

### Parameters:

*SelectCriteria* Selection Criteria to specify the destination of the command.

*Name* The Table name.

### Return Value:

0: Success. Greater than 0: Error. Less than 0: Warning.

## TableDeleteAll Method

### Syntax:

```
HRESULT TableDeleteAll([in] BSTR SelectCriteria);
```

### Description:

Delete all tables from memory.

### Parameters:

*SelectCriteria* Selection Criteria to specify the destination of the command.

### Return Value:

0: Success. Greater than 0: Error. Less than 0: Warning.

## TableSelect Method

### Syntax:

```
HRESULT TableSelect([in] BSTR SelectCriteria, [in]
BSTR Name);
```

### Description:

Select a table from memory for table operations.

### Parameters:

*SelectCriteria* Selection Criteria to specify the destination of the command.

*Name* The Table name.

### Return Value:

0: Success. Greater than 0: Error. Less than 0: Warning.

## IAgilent14360Transient Interface

This controls the transient functions.

### AbortTran Method

**Syntax:**

```
HRESULT AbortTran([in] BSTR SelectCriteria);
```

**Description:**

Aborts a transient trigger that is either initiated or has been triggered.

**Parameters:**

*SelectCriteria* Selection Criteria to specify the destination of the command.

**Return Value:**

0: Success. Greater than 0: Error. Less than 0: Warning.

### GetTranInitContEnabled Method

**Syntax:**

```
HRESULT GetTranInitContEnabled([in] BSTR  
SelectCriteria, [out, retval] VARIANT_BOOL* pVal);
```

**Description:**

Gets the transient initiate continuous enabled setting.

**Parameters:**

*SelectCriteria* Selection Criteria to specify the destination of the command. Only a single module can be selected.

**Return Value:**

True: enabled. False: disabled.



## InitiateTran Method

### Syntax:

```
HRESULT InitiateTran([in] BSTR SelectCriteria, [in]
VARIANT_BOOL WaitToInitiate);
```

### Description:

Initiates the transient trigger system.

### Parameters:

*SelectCriteria* Selection Criteria to specify the destination of the command.

*WaitToInitiate* Whether or not to wait for the initiate process to complete before the function returns.

### Return Value:

0: Success. Greater than 0: Error. Less than 0: Warning.

## SetTranInitContEnabled Method

### Syntax:

```
HRESULT SetTranInitContEnabled([in] BSTR
SelectCriteria, [in] VARIANT_BOOL Enabled, [in]
VARIANT_BOOL WaitToInitiate);
```

### Description:

Sets transient initiate continuous enabled.

### Parameters:

*SelectCriteria* Selection Criteria to specify the destination of the command.

*Enabled* True: enabled. False: disabled.

*WaitToInitiate* Whether or not to wait for the initiate process to complete before the function returns.

### Return Value:

0: Success. Greater than 0: Error. Less than 0: Warning.

## TriggerTran Method

**Syntax:**

```
HRESULT TriggerTran([in] BSTR SelectCriteria);
```

**Description:**

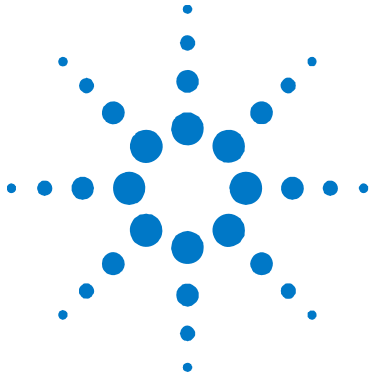
Trigger transient immediate.

**Parameters:**

*SelectCriteria* Selection Criteria to specify the destination of the command.

**Return Value:**

0: Success. Greater than 0: Error. Less than 0: Warning.



## 4 System Driver Enumerations

<a href="#">System Driver Enumerations</a> .....	84
--	----

This chapter describes the System Driver enumerations.

## System Driver Enumerations

This section lists all of the enumerations of the driver functions.

### Agilent14360ConfigInfoEnum

Configuration information.

Value	Description
Agilent14360ConfigInfoName	Name.
Agilent14360ConfigInfoAuthor	Author.
Agilent14360ConfigInfoCompany	Company.
Agilent14360ConfigInfoDescription	Description.
Agilent14360ConfigInfoLastModified	Last modified date.
Agilent14360ConfigInfoCRCChecksum	CRC checksum.
Agilent14360ConfigInfoFilename	Filename.
Agilent14360ConfigInfoInitializeOptions	Initialize options string.

### Agilent14360DACTableSizeEnum

Digital-to-analog converter table size.

Value	Description
Agilent14360DACTableSize256	256: Faster to calculate and activate, at the expense of resolution.
Agilent14360DACTableSize4096	4096: Greater resolution of the I-V characteristic, but takes longer to calculate and activate.

### Agilent14360DisplayViewEnum

Display type.

Value	Description
Agilent14360DisplayViewChan1	Channel 1.
Agilent14360DisplayViewChan2	Channel 2.
Agilent14360DisplayViewAll	All channels.

**Agilent14360DriverInfoEnum**

Driver information.

Value	Description
Agilent14360DriverInfoName	Name.
Agilent14360DriverInfoVersion	Version.
Agilent14360DriverInfoDescription	Description.

**Agilent14360InstrumentInfoEnum**

Instrument information.

Value	Description
Agilent14360InstrumentInfoModel	Model.
Agilent14360InstrumentInfoConnStr	Connection string. The actual string used to successfully connect to the instrument.
Agilent14360InstrumentInfoSerial	Serial number.
Agilent14360InstrumentInfoSocketsIP	Sockets IP address.
Agilent14360InstrumentInfoSocketsHostname	Sockets hostname.
Agilent14360InstrumentInfoResDesc	VISA resource descriptor.

**Agilent14360ListStepEnum**

List step.

Value	Description
Agilent14360ListStepOnce	Once.
Agilent14360ListStepAuto	Auto.

**Agilent14360ModuleChanGroupMemberEnum**

Module channel group member.

Value	Description
Agilent14360ModuleChanGroupMemberNone	None. A module not part of a channel group.
Agilent14360ModuleChanGroupMemberMaster	Master. The first module in a channel group.
Agilent14360ModuleChanGroupMemberFollower	Follower. Modules in a channel group other than the first master module.

**Agilent14360ModuleInfoEnum**

Module information.

Value	Description
Agilent14360ModuleInfoModel	Model.
Agilent14360ModuleInfoSerial	Serial number.
Agilent14360ModuleInfoOptions	Options.
Agilent14360ModuleInfoParentInstrumentName	Parent instrument name.
Agilent14360ModuleInfoChannel	Channel number.
Agilent14360ModuleInfoChannelGroupCount	Channel group count.

**Agilent14360ModuleSegmentMemberEnum**

Module segment member.

Value	Description
Agilent14360ModuleSegmentMemberMaster	Master. The first module in a segment.
Agilent14360ModuleSegmentMemberFollower	Follower. Modules in a segment other than the first master module.

**Agilent14360OperModeEnum**

The operating current mode.

Value	Description
Agilent14360OperModeFixed	Fixed mode.
Agilent14360OperModeSAS	SAS mode. Set the SAS mode to Immediate or List using SAS.SetSASMode.
Agilent14360OperModeTable	Table mode.

**Agilent14360OutputPONStateEnum**

Power on state.

Value	Description
Agilent14360OutputPONStateRST	Reset values.
Agilent14360OutputPONStateRCL0	Recall state 0.

**Agilent14360ResultsTypeEnum**

Measurement results type.

Value	Description
Agilent14360ResultsTypeModule	Return results per module.
Agilent14360ResultsTypeSegment	Return results per segment. Segment results are calculated from member modules.

**Agilent14360SASListEnum**

SAS list mode.

Value	Description
Agilent14360SASListISC	ISC (short circuit current).
Agilent14360SASListIMP	IMP (current at max power).
Agilent14360SASListVOC	VOC (open circuit voltage).
Agilent14360SASListVMP	VMP (Voltage at max power).
Agilent14360SASListDwell	Dwell.

**Agilent14360SASModeEnum**

SAS mode.

Value	Description
Agilent14360SASModeImm	SAS Immediate mode.
Agilent14360SASModeList	SAS List mode.

**Agilent14360StatusRegisterEnum**

Status register.

Value	Description
Agilent14360StatusRegisterStatusByte	Status byte.
Agilent14360StatusRegisterStandardEvent	Standard event.
Agilent14360StatusRegisterOperation	Operation.
Agilent14360StatusRegisterQuestionable	Questionable.

### Agilent14360StatusSubRegisterEnum

Sub Register.

Value	Description
Agilent14360StatusSubRegisterCondition	Condition.
Agilent14360StatusSubRegisterNegativeTransition	Negative Transition.
Agilent14360StatusSubRegisterPositiveTransition	Positive Transition.
Agilent14360StatusSubRegisterEvent	Event.
Agilent14360StatusSubRegisterEnable	Enable.

### Agilent14360TableEnum

SAS mode.

Value	Description
Agilent14360TableCurr	Current table.
Agilent14360TableVolt	Voltage table.



# Index

<b>C</b>	
configuration	
add instruments .....	14
create groups.....	17
specify settings .....	18
configuration wizard .....	6, 14
<b>D</b>	
default gateway .....	13
DHCP server .....	12
digital pins .....	20
DNS server.....	12
<b>E</b>	
Enumerations .....	84
Agilent14360ConfigInfoEnum .....	84
Agilent14360DACTableSizeEnum .....	84
Agilent14360DisplayViewEnum .....	84
Agilent14360DriverInfoEnum.....	85
Agilent14360InstrumentInfoEnum.....	85
Agilent14360ListStepEnum.....	85
Agilent14360ModuleChanGroupMemberEnum .....	85
Agilent14360ModuleInfoEnum.....	86
Agilent14360ModuleSegmentMemberEnum .....	86
Agilent14360OperModeEnum.....	86
Agilent14360OutputPONStateEnum .....	86
Agilent14360ResultsTypeEnum .....	87
Agilent14360SASListEnum .....	87
Agilent14360SASModeEnum .....	87
Agilent14360StatusRegisterEnum.....	87
Agilent14360StatusSubRegisterEnum .....	88
Agilent14360TableEnum.....	88
example program .....	28
<b>F</b>	
function parameters.....	38
<b>G</b>	
glossary .....	11
group.....	11
<b>H</b>	
history.....	2
Hostname.....	13

<b>I</b>	
IAgilent14360 Interface .....	39
Close .....	42
ConnectRemote .....	42
Dlog.....	39
DriverOperation .....	39
FixedMode .....	39
GetInitialized .....	42
Initialize.....	43
Measurement.....	39
Mode .....	40
Output.....	40
Protection .....	40
SAS .....	40
Status .....	41
System.....	41
TableMode.....	41
Transient.....	41
IAgilent14360Dlog Interface .....	45
GetDlogData.....	45
StartDlog .....	45
StopDlog .....	46
IAgilent14360DriverOperation Interface.....	47
CheckSystemState .....	47
GetConfigInfo.....	47
GetDriverInfo.....	48
GetGroupCount.....	48
GetGroupSegmentCount.....	49
GetGroupSegmentUIDAtIndex.....	49
GetGroupUIDAtIndex .....	48
GetInstrumentCount.....	49
GetInstrumentInfo.....	50
GetInstrumentMaxModulesCount .....	50
GetInstrumentModuleUIDAtIndex .....	50
GetInstrumentUIDAtIndex.....	51
GetModuleChanGroupMemberType.....	51
GetModuleInfo.....	51
GetModuleSegmentMemberType .....	52
GetSegmentModuleCount.....	52
GetSegmentModuleUIDAtIndex .....	52
IAgilent14360FixedMode Interface .....	53
GetCurrLimit.....	53
GetVoltLevelMethod.....	53
SetCurrLimit .....	54
SetVoltLevel .....	54

## Index

IAgilent14360Measurement Interface.....	55	IAgilent14360TableMode Interface.....	75
Measure .....	55	GetTableCount.....	75
IAgilent14360Mode Interface.....	56	GetTableName.....	75
GetDACTableSize.....	56	GetTableOffset.....	76
GetOperMode.....	56	SetTable.....	76
SetDACTableSize.....	57	SetTableName.....	77
SetOperMode.....	57	SetTableOffset.....	77
IAgilent14360Output Interface.....	58	TableCatalog.....	78
GetOutputEnabled.....	58	TableCopy.....	78
SetOutputEnabled.....	58	TableDelete.....	79
IAgilent14360Protection Interface.....	59	TableDeleteAll.....	79
GetCurrProtLevel.....	59	TableSelect.....	79
GetCurrSoftLimitHigh.....	59	IAgilent14360Transient Interface.....	80
GetVoltProtLevel.....	60	AbortTran.....	80
GetVoltSoftLimitHigh.....	60	GetTranInitContEnabled.....	80
ProtClear.....	60	InitiateTran.....	81
SetCurrProtLevel.....	61	SetTranInitContEnabled.....	81
SetCurrSoftLimitHigh.....	61	TriggerTran.....	82
SetVoltProtLevel.....	62	IP address.....	13
SetVoltSoftLimitHigh.....	62		
IAgilent14360SAS Interface.....	63	L	
GetSASMode.....	63	LAN interface	
SASFixedMode.....	63	private.....	13
SASListMode.....	63	site.....	12
SetSASMode.....	64		
IAgilent14360SASFixedMode Interface.....	65	M	
GetSASFixedCurve.....	65	module.....	11
SetSASFixedCurve.....	65	multiple client	
IAgilent14360SASListMode Interface.....	66	access.....	24
ConfigureSASList.....	66		
GetListRunCount.....	66	P	
GetSASList.....	67	print date.....	2
GetSASListCount.....	67		
GetTermLastEnabled.....	68	S	
SetListRunCount.....	68	segment.....	11
SetSASList.....	69	select criteria.....	37
SetTermLastEnabled.....	69	server control.....	6, 24
IAgilent14360Status Interface.....	70	subnet mask.....	13
ConfigureRegister.....	70	system	
Preset.....	70	installation requirements.....	6
QueryRegister.....	71	requirements.....	6
IAgilent14360System Interface.....	72	system control tools.....	6
DirectIOPrint.....	72	system description.....	10
DirectIOQuery.....	72	system driver.....	6, 22
DisplayEnabled.....	73	adding a reference.....	30
DisplayView.....	73	application.....	32
GetInstrumentError.....	73	declaration.....	30
GetMeterSummary.....	74	functions.....	36

initialization..... 31  
instantiation..... 30  
local mode..... 22  
remote mode..... 22  
Web monitor..... 25

W

Web monitor..... 25  
Web URL's ..... 3