Date : 02/06/2015

Scilab VISA
User Manual

Scilab Enterprises S.A.S | 143 bis rue Yves Le Coz - 78000 Versailles - Tél. : 01.80.77.04.60 - contact@scilab-enterprises.com
Capital €245,200 | 523 289 536 RCS Evry | SIRET 523 289 536 00022 | VAT FR56523289536 | Siège social Parc Orsay Université - 2 rue Jean Rostand - 91893 Orsay Cedex -France

summary

Scilab Enterprises S.A.S | 143 bis rue Yves Le Coz - 78000 Versailles - Tél. : 01.80.77.04.60 - contact@scilab-enterprises.com
Capital €245,200 | 523 289 536 RCS Evry | SIRET 523 289 536 00022 | VAT FR56523289536 | Siège social Parc Orsay Université - 2 rue Jean Rostand - 91893 Orsay Cedex -France

Installation

Pre-requisites

- Appropriate hardware, in the form of a National Instruments GPIB,GPIB-VXI, MXI/VXI or serial interface board.
- For GPIB applications, install NI-488. For VXI applications, install NI-VXI. For other hardware interfaces, NI-VISA uses the system's standard drivers.
- NI-VISA distribution media.

To use this module, you have to:
- Install NI-VISA,
- Install Scilab 5.5.1,
- Run in Scilab: atomsInstall("<path to attached file>\scilab_visa_<version>.zip"); (once)
- Run in Scilab: atomsLoad("scilab_visa"); (when starting a Scilab session).

Agilent

You can use both National Instruments (GPIB) and Agilent/HP (HPIB) Controllers on the same system.

To activate the interface between NI_VISA and Agilent, you must enable the NiTulip mode of NI_VISA.
Under Windows :

- Search "NI MAX" in order to start it.
- In "NI MAX" go to "System"->"Software" and select "NI-VISA".
- In "NI-VISA" go to "My System"->"General Settings" and select "Passports".
- Check NiTulip.dll.

You can find this information here.

Scilab Enterprises S.A.S | 143 bis rue Yves Le Coz - 78000 Versailles - Tél. : 01.80.77.04.60 - contact@scilab-enterprises.com
Capital €245,200 | 523 289 536 RCS Evry | SIRET 523 289 536 00022 | VAT FR56523289536 | Siège social Parc Orsay Université - 2 rue Jean Rostand - 91893 Orsay Cedex -France

Quick start

This section information on using the VISA Library Application Programmer's Interface (API) under Scilab.

VISA

The Virtual Instrument Software Architecture (VISA) is a standard for configuring, programming, and troubleshooting instrumentation systems comprising GPIB, VXI, PXI, Serial, Ethernet, and/or USB interfaces. VISA provides the programming interface between the hardware and development environments.

GPIB

GPIB (General Purpose Interface Bus) comes from IEEE-488 standard. It is a short-range digital communications bus specification. It was created for use with automated test equipment.

Scilab Enterprises S.A.S | 143 bis rue Yves Le Coz - 78000 Versailles - Tél. : 01.80.77.04.60 - contact@scilab-enterprises.com
Capital €245,200 | 523 289 536 RCS Evry | SIRET 523 289 536 00022 | VAT FR56523289536 | Siège social Parc Orsay Université - 2 rue Jean Rostand - 91893 Orsay Cedex -France

Tutorial

Get the list of connected devices

To connect between your computer and your device you need the address of the instrument.
For this, run the findAllInstruments() which gives you the list of all connected instruments :

```
[status, deviceAddrs] = findAllInstruments()
```

If findAllInstruments() hasn't found any device, it returns [].
- status contains the return code of the operation.
- deviceAddrs contains the list of descriptors (or adresses) of all connected devices.

Open a session

viOpenDefaultRM() creates a session and returns its identifier:

```
[status, defaultRM] = viOpenDefaultRM();
```

- status contains the return code of the operation.
- defaultRM is a unique logical identifier of the Default Resource Manager session.

Connect to a device

To open a communication with the first device above:

```
[status, idDevice] = viOpen( defaultRM,  deviceAddrs(1), viGetDefinition("VI_NULL"), viGetDefinition("VI_NULL"));
```

- defaultRM is Resource Manager session-should always be a session returned from viOpenDefaultRM().
- deviceAddrs(1) is the address of the first connected device.
- The first viGetDefinition("VI_NULL") is specifies the mode by which the resource is to be accessed, here the session uses VISA-supplied default values (please Refer to the Description section for valid values).
- The second viGetDefinition("VI_NULL") is specifies the maximum time period (in milliseconds) the operation waits before returning an error. (this does not set the I/O timeout—to do that you must call viSetAttribute() with the attribute VI_ATTR_TMO_VALUE).
- status contains the return code of the operation.
- idDevice is the device identifier.

Send commands

Once connected, you can send commands to your instrument using the function viWrite.

```
[status, count] = viWrite(idDevice, "*IDN?");
```

- idDevice is the device identifier.
- "*IDN?" is a command to get a device identification string (a command ending with "?" will return a value). Please refer to your instrument manual to get the list of its supported instructions.
- status contains the return code of the operation.
- count is the number of bytes actually transferred.

Scilab Enterprises S.A.S | 143 bis rue Yves Le Coz - 78000 Versailles - Tél. : 01.80.77.04.60 - contact@scilab-enterprises.com
Capital €245,200 | 523 289 536 RCS Evry | SIRET 523 289 536 00022 | VAT FR56523289536 | Siège social Parc Orsay Université - 2 rue Jean Rostand - 91893 Orsay Cedex -France

Read data

The viRead() is used for this purpose:

```
[status, bufferOut, count] = viRead(idDevice, 255);
```

- idDevice is the device identifier.
- 255 is number of bytes to be read.
- status contains the return code of the operation.
- bufferOut contains the read data.
- count is the number of bytes actually read.

Read attributes

You can also read one of the instruments attributes with the viGetAttribute() function.
This function needs a pointer on the attribute, so you need to create a pointer of the attribute type, then pass it to the function.

Then, you will need to read the pointer value, use the dedicated function for this:

```
pQueueLength = new_ViPUInt16();
status = viGetAttribute(idDevice, viGetDefinition("VI_ATTR_MAX_QUEUE_LENGTH"), pQueueLength);
QueueLength = ViPUInt16_value(pQueueLength);
```

- idDevice is the device identifier.
- viGetDefinition("VI_ATTR_MAX_QUEUE_LENGTH") is resource attribute for which the state query is made.
- pQueueLengh is the pointer to the value of the attribute.
- status is the pointer to the value of the attribute.

Disconnect from the device

The communication with the device is over, you can close it using the function viClose.

```
viClose(idDevice);
```

- idDevice is the device identifier.

Close the session

The same command is used to close the session.

```
viClose(defaultRM);
```

- defaultRM is a unique logical identifier of the Default Resource Manager session.

Scilab Enterprises S.A.S | 143 bis rue Yves Le Coz - 78000 Versailles - Tél. : 01.80.77.04.60 - contact@scilab-enterprises.com
Capital €245,200 | 523 289 536 RCS Evry | SIRET 523 289 536 00022 | VAT FR56523289536 | Siège social Parc Orsay Université - 2 rue Jean Rostand - 91893 Orsay Cedex -France

Tutorial full script

```
[status, deviceAddrs] = findAllInstruments();
[status, defaultRM] = viOpenDefaultRM();

// if no device is connected, use NI FTP as device
if deviceAddrs == [] then
   deviceAddrs = "TCPIP0::ftp.ni.com::21::SOCKET";
end

[status, idDevice] = viOpen(defaultRM, deviceAddrs(1), viGetDefinition("VI_NULL"),viGetDefinition("VI_NULL"));

[status, count] = viWrite(idDevice, "*IDN?");
[status, bufferOut, count] = viRead(idDevice, 255)

 pMaxQueueLength = new_ViPUInt16();
 status = viGetAttribute(idDevice, viGetDefinition("VI_ATTR_MAX_QUEUE_LENGTH"), pMaxQueueLength);
 maxQueueLength = ViPUInt16_value(pMaxQueueLength)
 delete_ViPUInt16(pMaxQueueLength);

viClose(idDevice);
viClose(defaultRM);
```


Scilab Enterprises S.A.S | 143 bis rue Yves Le Coz - 78000 Versailles - Tél. : 01.80.77.04.60 - contact@scilab-enterprises.com
Capital €245,200 | 523 289 536 RCS Evry | SIRET 523 289 536 00022 | VAT FR56523289536 | Siège social Parc Orsay Université - 2 rue Jean Rostand - 91893 Orsay Cedex -France

Module overview

This section describes gives an overwiew of the functions, types, and other entities of the Scilab VISA interface.

The Scilab VISA interface is based on the VISA C interface (described in the NI-VISA Programmer Manual).
It means that for example, all Scilab functions have the same names as in the C API, and their signatures are the same, as much as possible. Differences are described here.

Functions

Scilab functions have mostly the same name as functions in the C API. Only the prototypes of functions differ. Some functions with a pointer as parameter can be returned in the Scilab equivalent function. These are some examples:

VISA C function	**Equivalent in Scilab**
ViStatus viOpenDefaultRM(ViPSession sesn);	[status, sesn] = viOpenDefaultRM();
ViStatus viOpen(ViSession sesn, ViRsrc rsrcName, … ViAccessMode accessMode, ViUInt32 openTimeout, ViPSession vi)	[status, vi] = viOpen(sesn, rsrcName, ... accessMode, openTimeout);
ViStatus viRead(ViSession vi, ViPBuf buf, ViUInt32 count, ViPUInt32 retCount);	[status, buf, retCount] = viParseRsrc(vi, count);
ViStatus viReadAsync(ViSession vi, ViPBuf buf, … ViUInt32 count, ViPJobId jobId);	[status, buf, jobId] = viReadSTB(vi, count);
ViStatus viParseRsrc(ViSession sesn, ViRsrc rsrcName, … ViPUInt16 intfType, ViPUInt16 intfNum);	[status, intfType, intfNum] = viParseRsrc(sesn, … rsrcName);
ViStatus viReadSTB(ViSession vi, ViPUInt16 status);	[status, ViPUInt16_status] = viReadSTB(vi);
ViStatus viVxiCommandQuery(ViSession vi, ViUInt16 mode, … ViUInt32 cmd, ViPUInt32 response);	[status, response] = viReadSTB(vi, mode, cmd);

Example :

```
[status, sesn] = viOpenDefaultRM()
viClose (sesn);
```


Scilab Enterprises S.A.S | 143 bis rue Yves Le Coz - 78000 Versailles - Tél. : 01.80.77.04.60 - contact@scilab-enterprises.com
Capital €245,200 | 523 289 536 RCS Evry | SIRET 523 289 536 00022 | VAT FR56523289536 | Siège social Parc Orsay Université - 2 rue Jean Rostand - 91893 Orsay Cedex -France

Some other functions accept buffer pointers in inputs, and Scilab simplifies their use. As we can see below, a simple string can be passed instead of buffer pointers, and count is automatically calculated.

VISA C function	**Equivalent in Scilab**
ViStatus viWrite(ViSession vi, ViBuf buf, ViUInt32 count, … ViPUInt32 retCount);	[status, retCount] = viWrite(vi, buf);
ViStatus viWriteAsync(ViSession vi, ViBuf buf, ViUInt32 count, … ViPJobId jobId);	[status, jobId] = viWriteAsync(vi, buf);
ViStatus viBufWrite(ViSession vi, ViBuf buf, ViUInt32 count, … ViPUInt32 retCount);	[status, retCount] = viBufWrite(vi, buf);
ViStatus viGpibCommand(ViSession vi, ViBuf buf, … ViUInt32 count, ... ViPUInt32 retCount);	[status, retCount] = viGpibCommand(vi, buf);

Example :

```
[status, count] = viWrite(vi, "*IDN?");
```

Some functions use pointers to void , as the output type is not defined and therefore it is the user's responsibility to define this type.

VISA C function	**Equivalent in Scilab**
ViStatus viGetAttribute(ViObject vi, ViAttr attribute, … void * attrState);	status = viGetAttribute(vi, attribute, attrState);

Example :

```
attrState = new_ViPUInt16();
attribute = viGetDefinition("VI_ATTR_MAX_QUEUE_LENGTH");
status = viGetAttribute(vi, attribute, attrState);
State = ViPUInt16_value(attrState)
```


Scilab Enterprises S.A.S | 143 bis rue Yves Le Coz - 78000 Versailles - Tél. : 01.80.77.04.60 - contact@scilab-enterprises.com
Capital €245,200 | 523 289 536 RCS Evry | SIRET 523 289 536 00022 | VAT FR56523289536 | Siège social Parc Orsay Université - 2 rue Jean Rostand - 91893 Orsay Cedex -France

Types

The VISA C API redefines the primitive types. For example the VISA C type for an integer 16 bits is ViUint16.
These primitive types are automatically mapped to Scilab types. The functions using that types can be used transparently, without any conversion.
Some functions require pointers (to a VISA primitive type) as arguments (that is the case of the function viWrite() for example).
VISA also defines also types for these pointers, following is the list:

ViPUInt32	ViPAddr
ViAUInt32	ViAAddr
ViPInt32	ViPReal32
ViAInt32	ViAReal32
ViPUInt16	ViPReal64
ViAUInt16	ViAReal64
ViPInt16	ViPBoolean
ViAInt16	ViABoolean
ViPUInt8	ViPStatus
ViAUInt8	ViAStatus
ViPInt8	ViPVersion
ViAInt8	ViAVersion
ViPChar	ViPObject
ViAChar	ViAObject
ViPByte	ViASession
ViAByte	

A set of dedicated functions are provided to create and manipulate pointers on each of these types.

Create a pointer on a given type <Type> is done with `new_<Type>()`. For example to create a pointer on a ViUint16:

```
pData = new_ViPUInt16();
```

To dereference a pointer, use `<Type>_value()` to read, and `<Type>_assign()` to write, with the pointer as argument:

```
ViPUInt16_assign(pData, valueIn);
valueOut = ViPUInt16_value(pData);
```

Copying a pointer is done with `copy_<Type>()`.

Finally, you have to delete the pointer (to free the allocated memory) with `delete_<Type>()`

```
delete_ViPUInt16(pData);
```


Scilab Enterprises S.A.S | 143 bis rue Yves Le Coz - 78000 Versailles - Tél. : 01.80.77.04.60 - contact@scilab-enterprises.com
Capital €245,200 | 523 289 536 RCS Evry | SIRET 523 289 536 00022 | VAT FR56523289536 | Siège social Parc Orsay Université - 2 rue Jean Rostand - 91893 Orsay Cedex -France

Constants

The VISA C API defines a lot a constants, like attribute IDs (ex: VI_ATTR_JOB_ID), event IDs

(ex: VI_EVENT_IO_COMPLETION), and so on.

These constants are accessed in Scilab via the viGetDefinition() function.

Note on instructions

When you send a command with viWrite(), a carriage return ("\n") is automatically added by the function:

```
[status, count] = viWrite(instr, ":WAVeform:DATA?");
```


Scilab Enterprises S.A.S | 143 bis rue Yves Le Coz - 78000 Versailles - Tél. : 01.80.77.04.60 - contact@scilab-enterprises.com
Capital €245,200 | 523 289 536 RCS Evry | SIRET 523 289 536 00022 | VAT FR56523289536 | Siège social Parc Orsay Université - 2 rue Jean Rostand - 91893 Orsay Cedex -France

Functions

viGetDefinition

Calling Sequence

res = viGetDefinition(string_in)

Arguments

string_in :

a character string.

res :

a real .

Description

During the VISA library integration into the module, names of definitions which were over 24 characters had to be abbreviated.
This macro match the abbreviated name with the original one.

Examples :

viGetDefinition("VI_SUCCESS")

Scilab Enterprises S.A.S | 143 bis rue Yves Le Coz - 78000 Versailles - Tél. : 01.80.77.04.60 - contact@scilab-enterprises.com
Capital €245,200 | 523 289 536 RCS Evry | SIRET 523 289 536 00022 | VAT FR56523289536 | Siège social Parc Orsay Université - 2 rue Jean Rostand - 91893 Orsay Cedex -France

findAllInstruments

Returns the addresses of all connected devices.

Calling Sequence

[status, deviceAddrs] = findAllInstruments()

Arguments

status :

a real containing the status of the operation.

deviceAddrs :

a matrix of string containing the addresses of all connected devices.

Description

This macro returns the adress (or descriptor) of each connected device. An empty matrix [] is returned if no device is connected.

Examples :

```
[status, deviceDescriptors] = findAllInstruments()
```

output example:

```
!TCPIP0::127.0.0.1::TEST::INSTR    !
!                                  !
!ASRL1::INSTR                      !
!                                  !
!ASRL4::INSTR                      !
```


Scilab Enterprises S.A.S | 143 bis rue Yves Le Coz - 78000 Versailles - Tél. : 01.80.77.04.60 - contact@scilab-enterprises.com
Capital €245,200 | 523 289 536 RCS Evry | SIRET 523 289 536 00022 | VAT FR56523289536 | Siège social Parc Orsay Université - 2 rue Jean Rostand - 91893 Orsay Cedex -France

viOpenDefaultRM

This function returns a session to the Default Resource Manager resource.

Calling Sequence

[status, sesn] = viOpenDefaultRM()

Arguments

status :

contains the return code of the operation.

sesn :

unique logical identifier to a Default Resource Manager session.

Description

The viOpenDefaultRM() function must be called before any VISA operations can be invoked. The first call to this function initializes the VISA system, including the Default Resource Manager resource, and also returns a session to that resource. Subsequent calls to this function return unique sessions to the same Default Resource Manager resource. When a Resource Manager session is passed to viClose(), not only is that session closed, but also all find lists and device sessions (which that Resource Manager session was used to create) are closed.

Examples :

```
[status, defaultRM] = viOpenDefaultRM();
viClose(defaultRM);
```


Scilab Enterprises S.A.S | 143 bis rue Yves Le Coz - 78000 Versailles - Tél. : 01.80.77.04.60 - contact@scilab-enterprises.com
Capital €245,200 | 523 289 536 RCS Evry | SIRET 523 289 536 00022 | VAT FR56523289536 | Siège social Parc Orsay Université - 2 rue Jean Rostand - 91893 Orsay Cedex -France

viOpen

Opens a session to the specified resource .

Calling Sequence

[status, vi] = viOpen (sesn, rsrcName, accessMode, openTimeout)

Arguments

status :
contains the return code of the operation.

vi :
will be the computer's identifier for other functions.

sesn :
resource Manager session-should always be a session returned from viOpenDefaultRM().

rsrcName :
unique symbolic name of a resource (please see the ViOpen() official help page for details) .

accessMode :
specifies the mode by which the resource is accessed, here the session uses VISA-supplied default values (please Refer to the Description section for valid values).

openTimeout :
specifies the maximum time period (in milliseconds) the operation waits before returning an error. (this does not set the I/O timeout-to do that you must call viSetAttribute() with the attribute VI_ATTR_TMO_VALUE).

Description

The viOpen() operation opens a session to the specified resource.

Examples :

```
[status, defaultRM] = viOpenDefaultRM();

// Connect to device located at TCPIP0::ftp.ni.com::21::SOCKET
[status, instr] = viOpen(defaultRM, "TCPIP0::ftp.ni.com::21::SOCKET", viGetDefinition("VI_NULL"), …
 viGetDefinition("VI_NULL"));

viClose(instr);
viClose(defaultRM);
```


Scilab Enterprises S.A.S | 143 bis rue Yves Le Coz - 78000 Versailles - Tél. : 01.80.77.04.60 - contact@scilab-enterprises.com
Capital €245,200 | 523 289 536 RCS Evry | SIRET 523 289 536 00022 | VAT FR56523289536 | Siège social Parc Orsay Université - 2 rue Jean Rostand - 91893 Orsay Cedex -France

viClose

Closes the specified session, or the connection to a device.

Calling Sequence

status = viClose(vi)

Arguments

status :

contains the return code of the operation.

vi :

unique logical identifier to a session, event, or find list.

Description

The viClose() operation closes a session or a connection to a device (or an event, or a find list). In this process all the data structures that had been allocated for the specified vi are freed. Calling viClose() on a VISA Resource Manager session will also close all I/O sessions associated with that resource manager session.

Examples :

```
[status, defaultRM] = viOpenDefaultRM();

// Connect to device located at TCPIP0::ftp.ni.com::21::SOCKET
[status, instr] = viOpen(defaultRM, "TCPIP0::ftp.ni.com::21::SOCKET", viGetDefinition("VI_NULL"), …
 viGetDefinition("VI_NULL"));

viClose(instr);
viClose(defaultRM);
```


Scilab Enterprises S.A.S | 143 bis rue Yves Le Coz - 78000 Versailles - Tél. : 01.80.77.04.60 - contact@scilab-enterprises.com
Capital €245,200 | 523 289 536 RCS Evry | SIRET 523 289 536 00022 | VAT FR56523289536 | Siège social Parc Orsay Université - 2 rue Jean Rostand - 91893 Orsay Cedex -France

viWrite

Writes synchronously data to device or interface

Calling Sequence

[status, writeCount] = viWrite(session, buf) = viWrite(session, buf)

Arguments

status :

a real.

writeCount :

number of bytes actually transferred.

session :

unique logical identifier of a session.

buf :

a character string.

location of a data block to be sent to a device.

Description

The viWrite() operation synchronously transfers data. The data to be written is in the buffer represented by buf. This operation returns only when the transfer is terminated. Only one synchronous write operation can occur at a time.

Examples :

```
[status, defaultRM] = viOpenDefaultRM();

    // Write a command to device located at TCPIP0::ftp.ni.com::21::SOCKET
    [status, instr] = viOpen(defaultRM, "TCPIP0::ftp.ni.com::21::SOCKET", viGetDefinition("VI_NULL"), …
viGetDefinition("VI_NULL"));
    [status, count] = viWrite(instr, ":AUT")

    viClose(instr);
    viClose(defaultRM);
```


Scilab Enterprises S.A.S | 143 bis rue Yves Le Coz - 78000 Versailles - Tél. : 01.80.77.04.60 - contact@scilab-enterprises.com
Capital €245,200 | 523 289 536 RCS Evry | SIRET 523 289 536 00022 | VAT FR56523289536 | Siège social Parc Orsay Université - 2 rue Jean Rostand - 91893 Orsay Cedex -France

viRead

Synchronously reads data from device or interface .

Calling Sequence

[status, buf, readCount] = viRead(session, count)

Arguments

status :

a real.

buf :

a character string.
location of the buffer receiving data from device.

readCount :

number of bytes actually transferred.

session :

unique logical identifier of a session.

count :

number of bytes to be read .

Description

The viRead() operation synchronously transfers data. The data read is stored in the buffer represented by buf. This operation returns only when the transfer is terminated. Only one synchronous read operation can occur at a time.

Examples :

```
[status, defaultRM] = viOpenDefaultRM();

    // Send command to get identifier of device located at TCPIP0::ftp.ni.com::21::SOCKET
    [status, instr] = viOpen(defaultRM, "TCPIP0::ftp.ni.com::21::SOCKET", viGetDefinition("VI_NULL"), …
viGetDefinition("VI_NULL"));
    [status, count] = viWrite(instr, "*IDN?");

    // Read command answer (device identifier)
    [status, bufferOut, count] = viRead(instr, 255);
    disp(bufferOut);

    viClose(instr);
    viClose(defaultRM);
```


Scilab Enterprises S.A.S | 143 bis rue Yves Le Coz - 78000 Versailles - Tél. : 01.80.77.04.60 - contact@scilab-enterprises.com
Capital €245,200 | 523 289 536 RCS Evry | SIRET 523 289 536 00022 | VAT FR56523289536 | Siège social Parc Orsay Université - 2 rue Jean Rostand - 91893 Orsay Cedex -France

viRequest

Synchronously Read and Write data in device or interface.

Calling Sequence

[status, bufOut] = viRequest(session, buf)

Arguments

status :

a real.

bufOut :

a character string.
location of a buffer to receiving data from device.

session :

unique logical identifier of a session.

buf :

a character string.
location of the data block to be sent to the device .

Description

The viRequest() operation synchronously transfers data. The data to be written is in the buffer represented by buf and the data read to be stored is in the buffer represented by bufOut. This operation returns only when the transfer is terminated. Only one synchronous query operation can occur at a time.

Examples :

```
[status, defaultRM] = viOpenDefaultRM();

disp("write :");
dips("Adress = "" device adress "" exemple Adress = """TCPIP0::ftp.ni.com::21::SOCKET""");

if execstr("Adress", 'errcatch') == 4 then
   Adress = "TCPIP0::ftp.ni.com::21::SOCKET";
end

[status, instr] = viOpen (defaultRM, Adress, viGetDefinition("VI_NULL"),viGetDefinition("VI_NULL"));
[status, bufferOut] = viRequest(instr, "*IDN?");
bufferOut

viClose (instr);
viClose (defaultRM);
```


Scilab Enterprises S.A.S | 143 bis rue Yves Le Coz - 78000 Versailles - Tél. : 01.80.77.04.60 - contact@scilab-enterprises.com
Capital €245,200 | 523 289 536 RCS Evry | SIRET 523 289 536 00022 | VAT FR56523289536 | Siège social Parc Orsay Université - 2 rue Jean Rostand - 91893 Orsay Cedex -France

viGetAttribute

Gets the value of a resource attribute.

Calling Sequence

status = viGetAttribute(vi, attribute, pData)

Arguments

vi :

unique logical identifier to a session, event, or find list.

attribute :

identifier of the resource attribute.

pData :

pointer to the attribute value.

Description

The viGetAttribute() operation is used to get the value of an attribute for the specified session, event, or find list.

The pData output parameter is a pointer which type depends on the attribute. For boolean attributes, you have to create a pointer to a ViPBoolean variable and pass it to the function. Use the functions dedicated to pointer creation, as following:

```
pDataViBoolean = new_ViPBoolean();
```

Another example for ViUInt32 attributes:

```
pDataViUInt32 = new_ViPUInt32();
```

`ViGetAttribute()` writes the value of the attribute to the address pointed by the pData pointer argument. To read this value, use another dedicated function (following is for ViUInt32 attributes):

```
data = ViPUInt32_value(pDataViUInt32);
```


Scilab Enterprises S.A.S | 143 bis rue Yves Le Coz - 78000 Versailles - Tél. : 01.80.77.04.60 - contact@scilab-enterprises.com
Capital €245,200 | 523 289 536 RCS Evry | SIRET 523 289 536 00022 | VAT FR56523289536 | Siège social Parc Orsay Université - 2 rue Jean Rostand - 91893 Orsay Cedex -France

Examples :

```
[status, defaultRM] = viOpenDefaultRM();

    // Connect to device located at TCPIP0::ftp.ni.com::21::SOCKET
    [status, instr] = viOpen(defaultRM, "TCPIP0::ftp.ni.com::21::SOCKET", viGetDefinition("VI_NULL"), …
viGetDefinition("VI_NULL"));

    // Get Max Queue Length attribute
    pMaxQueueLength = new_ViPUInt16();
    status = viGetAttribute(instr, viGetDefinition("VI_ATTR_MAX_QUEUE_LENGTH"), pMaxQueueLength);
    maxQueueLength = ViPUInt16_value(pMaxQueueLength)

    delete_ViPUInt16(pMaxQueueLength);
    viClose(instr);
    viClose(defaultRM);
```


Scilab Enterprises S.A.S | 143 bis rue Yves Le Coz - 78000 Versailles - Tél. : 01.80.77.04.60 - contact@scilab-enterprises.com
Capital €245,200 | 523 289 536 RCS Evry | SIRET 523 289 536 00022 | VAT FR56523289536 | Siège social Parc Orsay Université - 2 rue Jean Rostand - 91893 Orsay Cedex -France