Velox Python Interface (VPI) Version 2.6

This is a guide to the Velox Python Interface (VPI).  
Author: Bob Puckett, Form Factor, Inc.  
Last Update: 1/10/2019  
Velox Version Targeted: Velox 2.6

Contents

[Introduction 1](#_Toc500140717)

[Getting Started 2](#_Toc500140718)

[Installation 2](#_Toc500140719)

[Samples 2](#_Toc500140720)

[Sample Code Walkthrough 3](#_Toc500140721)

[Example 1: Commands and Return Values – vpisample.py 3](#_Toc500140722)

[Example 2: A minimal example – vpisampleminimal.py 4](#_Toc500140723)

[SCI Command Exception Handling 5](#_Toc500140724)

[Example 3: Exception Handling – vpiexceptionsample.py 5](#_Toc500140725)

[Hints and Tips 5](#_Toc500140726)

# Introduction

Velox provides an extensive API for scripting Velox applications such as WaferMap, Spectrum, etc.

The interface uses SCI Commands documented in the Velox Remote Interface Guide.

The Velox Scripting Console, installed with the Velox product, allows communication with a locally installed Velox probe station. Commands may be sent as simple text commands or as Python scripts. The version of Python provided in Scripting Console is Iron Python Version 2.7 from Microsoft.

The package described in this document, The Velox Python Interface (VPI), provides Python modules making the Velox SCI command API easily accessible through a standard Python installation. Python 2.7 and Python 3.5+ are both supported. This allows communication with a local installation of Velox running on the same PC, or a remote installation communicating over network sockets.

This document describes using this Python library. Users should be familiar with the Velox Integration Toolkit, supplied separately, in addition to the information presented here.

# Getting Started

Using the VPI requires that a version of Python be installed. For the latest versions of Python, see [www.python.org](http://www.python.org).

VPI will work with Python 2.75 or above and Python 3.5 and above. Python 3.6 or later is recommended. Python version 3.5 and above will provide the best code completion and ‘Intellisense’ experience for the developer.

The basic steps to use VPI are:

1. Ensure that a working copy of Python 2.75+ or 3.5+ is installed. This guide assumes that the user is familiar with the Python development environment.
2. Install the velox.py module using the python pip tool.
3. Use the provided samples as a starting point.

## Installation

The velox.py package is not currently distributed through the standard Python Package Index, PyPI. There is an option to install via PIP from a command line.

1. Save the velox.tar.gz file to a convenient temporary location.
2. Open a command prompt at that location.
3. Type the command: pip install velox.tar.gz

NOTE: You must use the full file extension to avoid getting any modules named Velox from the public Python repository.

You should see several installation messages as the process proceeds.

The contents of the velox.tar.gz installation file is this document, a setup.py file, and a folder named ‘velox’. There is no need to extract the contents of the .gz file before installing.

The velox folder contains the Python module and a samples folder.

# Samples

A minimal sample can be as short as 3 lines of code:

import velox

with velox.MessageServerInterface() as msgServer:

**print**(velox.ReportKernelVersion())

All velox python programs should use the first two lines above to open a connection to the Velox Message Server. If the Velox Message Server is running on a remote PC, supply the IP address and port number as:

with velox.MessageServerInterface(ipaddr=’x.x.x.x’, targetSocket=n) as msgServer:

Replace ‘x.x.x.x’ with the IP Address of the probe station and n with the port number for the message server. The default address is ‘localhost’ and the port number is 1412. If the port number is 1412, it does not need to be specified.

## Sample Code Walkthrough

The samples below are in the velox folder in a subfolder named Samples.

### Example 1: Commands and Return Values – vpisample.py

*# import velox will choose the correct version of the library*

*# based on the version of Python being executed*

import velox

*# import sys for this example, but it is not required for VPI*

import sys

*# print the version of Python that is running, just to see that it is working*

**print**('Using Python Version ' + sys.version)

*# Connect to the message server using a 'with' statement.*

*# The application name registered with the Velox Message Server*

*# will be the name of our script. Keep a reference to*

*# the object returned by velox.MessageServerInterface*

*# - in this example, we call it msgServer*

*# the parameters to MessageServerInterface are optional.*

*# For a local connection, you can leave them blank. If*

*# connecting to a remote machine, add keyword parameters and set the*

*# address and socket*

*# velox.MessageServerInterface(ipaddr='localhost',targetSocket=1412)*

with velox.MessageServerInterface() as msgServer:

*# as long as the connection was successful, we can send some commands.*

*# if it was not successful, an exception is thrown.*

*# try some SCI Command Examples*

*# The SCI commands in the velox module have docstrings to describe*

*# their function, inputs, and outputs.*

*# You can print them from a Python command to see what the function does.*

*# print the docstring for the ReportKernelVersion command*

*# ReportKernelVersion returns 2 values.*

*# here is the description of the command*

**print**(velox.ReportKernelVersion.\_\_doc\_\_)

*# SCI commands return a namedtuple if multiple values are returned.*

*# ReportKernelVersion returns a version number and a description.*

*# You can acess the return values by name or by indexing the tuple.*

v = velox.ReportKernelVersion()

**print**('The kernel version is', v.Version, 'and', v.Description)

**print**('The kernel version array is', v[0], 'and', v[1])

*# a more convenient example is to unpack the tuple into variables*

*# as part of the function call.*

*# here we get variables named ver and desc and use them directly*

ver, desc = velox.ReportKernelVersion()

**print**('Got ', ver, 'and', desc)

*# The Echo Data command just returns its own first input.*

*# Echo data only returns the first string in the parameter list.*

*# To get a string with spaces to display, we need to add double quotes.*

*# Try it both ways to see the difference. The first one below does*

*# what we expect. The second only prints the first word.*

*# The third assigns a return value to a variable*

**print**(velox.EchoData('"Hi World"'))

**print**(velox.EchoData("Hi World"))

evalue = velox.EchoData("Echo\_This")

*# ReadChuckPosition returns 3 values. Here we print the entire tuple*

*# to show the tuple name and the names of the 3 return values*

**print**(velox.ReadChuckPosition("Microns"))

*# it is possible to send raw command strings with the*

*# sendSciCommand function.*

*# it isn't recommended, but in some cases, it might be necessary.*

*# it is much easier and 'safer' to use the defined SCI functions.*

*# note that this is a function of MessageServerInterface,*

*# not a velox module function*

**print**(msgServer.sendSciCommand("echodata", rparams='"RawHello Velox"'))

### Example 2: A minimal example – vpisampleminimal.py

Communication with Velox is very simple using the VPI. As simple as the following:

import velox

with velox.MessageServerInterface() as msgServer:

ver, desc = velox.ReportKernelVersion()

**print**('Got ', ver, 'and', desc)

x,y,z = velox.ReadChuckPosition("Microns")

**print**(x,y,z)

# SCI Command Exception Handling

## Example 3: Exception Handling – vpiexceptionsample.py

This code will raise an exception if Spectrum is not running. Velox command errors raise a velox.SciException.

import velox

with velox.MessageServerInterface() as msgServer:

*# FindFocus will throw an exception because we aren't properly set up yet.*

*# This is to demonstrate exception handling.*

try:

**print**(velox.FindFocus())

except velox.SciException as e:

**print**('We caught an exception as expected:')

**print**(e)

# Hints and Tips

* Remember to prefix SCI functions with velox, as in velox.ReportKernelVersion()
* Python functions are often named using lower case, but the velox library uses mixed case names to match the SCI command documentation. Use mixed case when calling SCI commands.
* Use print(velox.commandname.\_\_doc\_\_) to get information about what a command does and what inputs and outputs to expect.
* SCI commands return Python namedtuples for commands that return multiple values. Sci sommands that return a single value return a simple variable type.
* Use multiple return values for convenience. See the example using ReportKernelVesion.
* Use exception handling and catch the velox.SciException for SCI command errors.
* EchoData returns only the first word of a string passed to it unless it is contained in double quotes surrounded by single quotes. Example EchoData(‘”Hello World”’)