



# National Centre for Radio Astrophysics

Internal Technical Report  
GMRT/ R-267

## Python Based Interactive command line & GUI environment for OnlineV2

Deepak Bhong

*Email: [deepak@gmrt.ncra.tifr.res.in](mailto:deepak@gmrt.ncra.tifr.res.in)*

Revision	Date: Aug. 2015	Modification/ Change
Ver. 1		Initial Version R-267

## Abstract

This report is about the design and working of interactive command line environment for OnlineV2. OnlineV2 is control and monitor software for upgraded **GMRT** antennas and other sub systems like **correlator**, **GAB** etc developed at GMRT. OnlineV2 is generalized framework to support future expansion.

Interactive command line environment uses **Ipython** as interactive command line shell, so it is called Python environment. It can have as many as 32 controlling command line terminals, all communicate with **OnlineV2** through **Hub** using TCP/IP protocol.

This report also gives description about wrapper functions created on top of **NOVAS** astronomical library. The algorithms used by **NOVAS** are based on vector astrometry theories and the **IAU** resolutions. The wrapper functions are written for **C** programming language as well as scripting language i.e. PHP, PYTHON and PERL.

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	OnlineV2 . . . . .	3
1.2	Interactive command line environment for OnlineV2 . . . . .	3
1.3	Why Python? . . . . .	3
<b>2</b>	<b>Components of Python Environment</b>	<b>5</b>
2.1	User input terminal . . . . .	5
2.2	Python Environment Hub . . . . .	5
2.3	Application Programming Interface <b>API</b> in Python for OnlineV2 . . . . .	6
<b>3</b>	<b>Design</b>	<b>7</b>
3.1	Design of Interactive Scripting environment . . . . .	7
3.2	Communication with OnlineV2 . . . . .	8
3.3	Application Programming Interface(API) and Procedures . . . . .	9
<b>4</b>	<b>Astronomical Calculations</b>	<b>11</b>
4.1	NOVAS . . . . .	11
4.2	Wrapper functions for NOVAS . . . . .	11
4.3	Wrapper functions for PHP, PERL and PYTHON . . . . .	11
<b>5</b>	<b>Under Development</b>	<b>13</b>
5.1	Monitoring <b>API</b> in Python for OnlineV2 . . . . .	13
5.2	Background task manager . . . . .	13
5.3	GUI . . . . .	13
<b>6</b>	<b>References</b>	<b>14</b>

# 1 Introduction

## 1.1 OnlineV2

**OnlineV2** is an upgraded control-monitor software for **GMRT**. **OnlineV2** can communicate with the antennas and antenna subsystems using the fast Ethernet enabled over the optical fiber. The design and development of new monitor and control modules which use a Rabbit RCM 4300 processor and of software running on this module and on the central control computer (OnlineV2).

OnlineV2 does not require an antenna-base computer like the ABC in the current system since the Rabbit MCM cards can easily handle both antenna-base operations and the communication traffic. In this upgraded system, the Rabbit MCM directly communicates with OnlineV2 through the Layer 2 network switch at the antenna base and Layer 3 switch in the receiver room and can play the supervisory role at the antenna base, when required.

The Linux, Apache, MySQL, PHP (LAMP) model of free and open software is used in the development of OnlineV2. MCM cards use Dynamic C and OnlineV2 is coded in C.

## 1.2 Interactive command line environment for OnlineV2

To fulfill the need of command line input, grouping of commands to create custom procedure and automation in execution of science observations and system tests, interactive command line environment is developed. Logical execution of OnlineV2 commands is achieved by using **IPython** as command line input shell for control terminal, so power of **PYTHON** can be utilized in decision making on basis of complex logic of tasks. A **python package** is developed to control all GMRT subsystems namely Servo, FPS, Front-End, GAB(GMRT Analog Backend) and Sentinel.

These libraries are developed using **Object Oriented Programming** in Python, which can act as API(Application Programming Interface) for controlling the systems.

The format of a command follows **Object Oriented Design**, some example command are listed below.

- move servo of C00 antenna in elevation by +10 degree  
\$ C00.servo.mvelev(10d)
- default setting of GAB of C00 antenna  
\$ C00.gab.set()
- Send GAB setting command to defined sub array of antennas.  
\$ gab.set()

## 1.3 Why Python?

Python is a very-high-level dynamic object-oriented programming language. It is designed to be easy to program and easy to read. It is very popular in scientific community. Python is an advanced scripting language that is being used successfully to glue together large software components.

List of software where **PYTHON** is used.

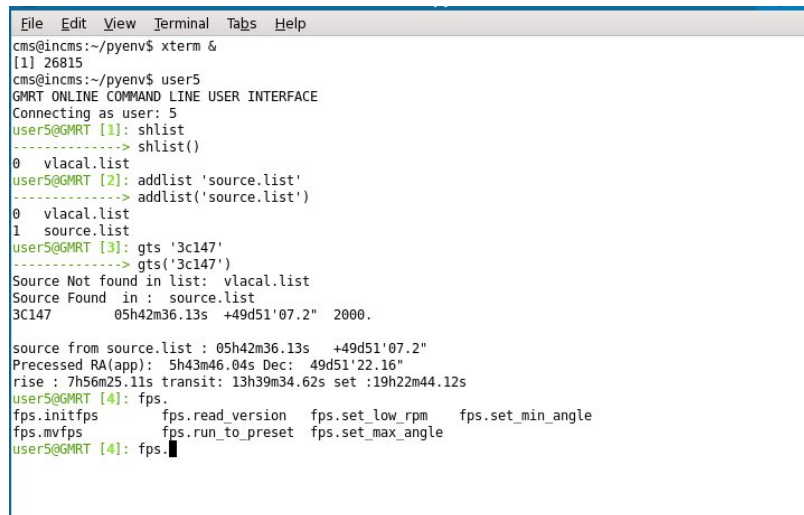
- **CASA** uses **PYTHON** for standard scripting interface, with **IPYTHON** extension for interactive command line use.
- **Taurus** is a python framework for control and data acquisition CLIs and GUIs in scientific/industrial environments. It supports multiple control systems or data sources: Tango, EPICS etc. New control system libraries can be integrated through plugins.
- **Sagemath** is free, open source math software that supports research and teaching in algebra, geometry, number theory, cryptography, numerical computation, and related areas.
- **Spacepy** is Python Based Tools for the Space Science Community. SpacePy is being used for major space science projects and at several government and educational institutions.

## 2 Components of Python Environment

### 2.1 User input terminal

It is based on IPython shell. IPython is an interactive shell built on top of **Python** interpreter. It has support for **terminal, GUI and WEB** interface. Qt library is used for GUI interface.

One can create multiple number of users, currently this number is limited to **32**.



```
File Edit View Terminal Tabs Help
cms@incms:~/pyenv$ xterm &
[1] 26815
cms@incms:~/pyenv$ user5
GMRT ONLINE COMMAND LINE USER INTERFACE
Connecting as user: 5
user5@GMRT [1]: shlist
-----> shlist()
0 vlocal.list
user5@GMRT [2]: addlist 'source.list'
-----> addlist('source.list')
0 vlocal.list
1 source.list
user5@GMRT [3]: gts '3c147'
-----> gts('3c147')
Source Not found in list: vlocal.list
Source Found in : source.list
3C147      05h42m36.13s +49d51'07.2" 2000.

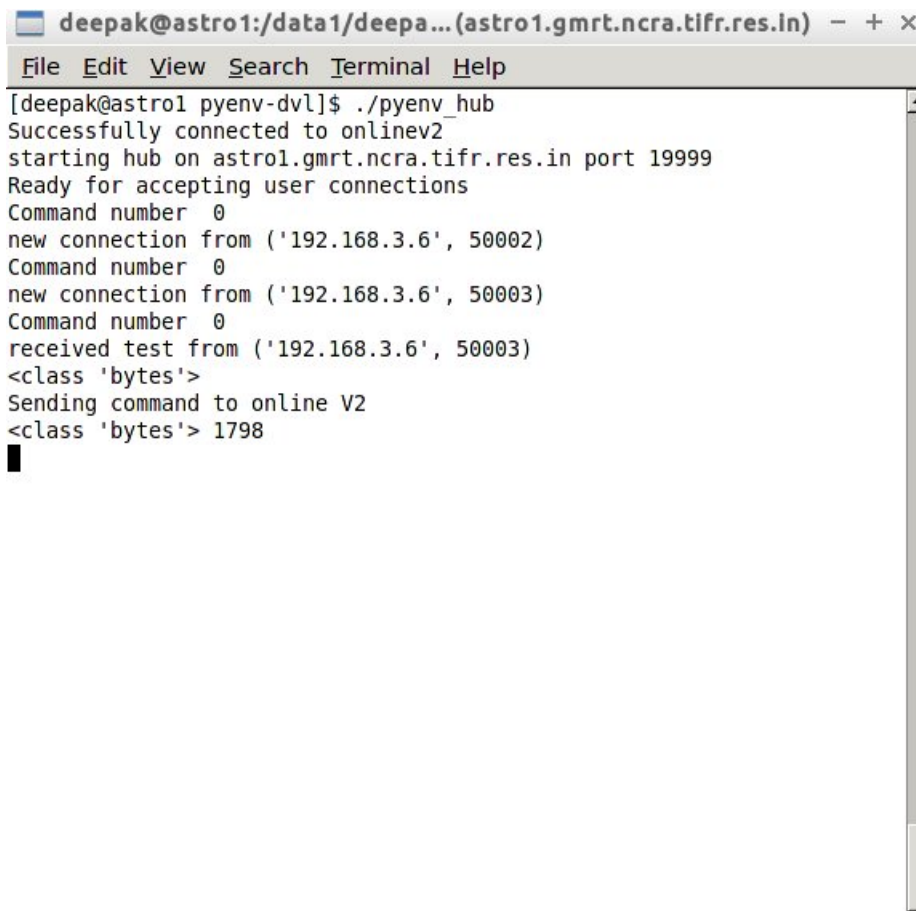
source from source.list : 05h42m36.13s +49d51'07.2"
Precessed RA(app): 5h43m46.04s Dec: 49d51'22.16"
rise : 7h56m25.11s transit: 13h39m34.62s set :19h22m44.12s
user5@GMRT [4]: fps.
fps.initfps      fps.read_version  fps.set_low_rpm   fps.set_min_angle
fps.mvfps        fps.run_to_preset fps.set_max_angle
user5@GMRT [4]: fps.
```

Figure 2.1: Python Control Terminal

Figure 2.1 shows OnlineV2 user input in gnome terminal.

### 2.2 Python Environment Hub

As its name suggest it acts as router to **OnlineV2** for all user input terminals. It is written in Python language. All user input terminals gets connected as client to HUB through socket. Also Hub is connected using TCP/IP socket onlineV2. Figure 2.2 shows output from **HUB** while it is in running mode.



```
deepak@astro1:/data1/deepa... (astro1.gmrt.ncra.tifr.res.in) - + x
File Edit View Search Terminal Help
[deepak@astro1 pyenv-dvl]$ ./pyenv_hub
Successfully connected to onlinev2
starting hub on astro1.gmrt.ncra.tifr.res.in port 19999
Ready for accepting user connections
Command number 0
new connection from ('192.168.3.6', 50002)
Command number 0
new connection from ('192.168.3.6', 50003)
Command number 0
received test from ('192.168.3.6', 50003)
<class 'bytes'>
Sending command to online V2
<class 'bytes'> 1798
█
```

Figure 2.2: Hub

### 2.3 Application Programming Interface API in Python for OnlineV2

It has two components.

- Control API
- Monitor API

Control API is used to send control commands to GMRT antenna or subsystems. And Monitor API will be used to read parameters related to GMRT antennas or systems, which are set or changing with time like servo position, temperature etc. Both these package follow Object Oriented design and are fully compatible as third party PYTHON package, which can be imported in IPython shell.

Monitoring API is Object Oriented interface to read shared memory created by OnlineV2. The monitoring API is under final development.

User input terminals are uses both Control and Monitor API. They are imported at start of IPython shell as Python package.

### 3 Design

#### 3.1 Design of Interactive Scripting environment

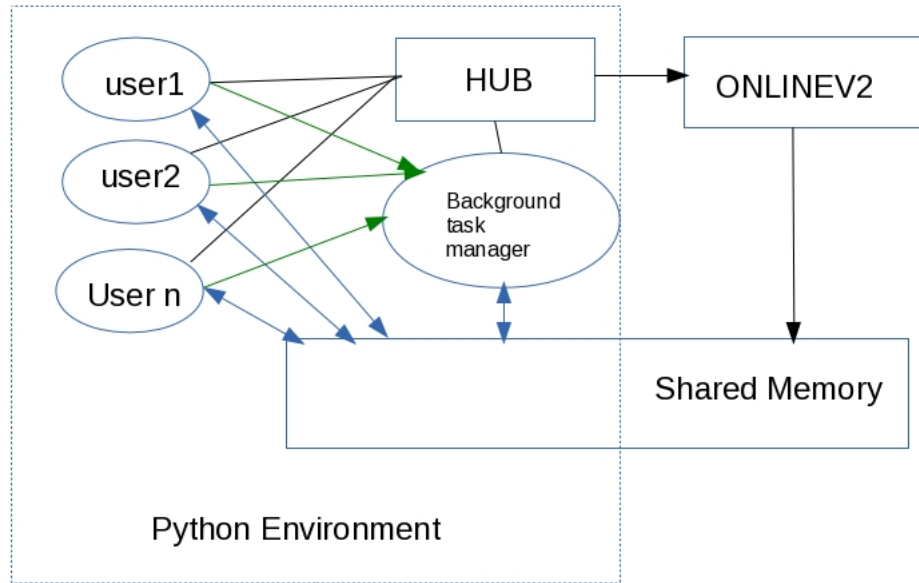


Figure 3.1: Design of Python scripting environment for OnlineV2

Figure 3.1 shows the architectural block diagram of interaction scripting environment for OnlineV2 that has been successfully implemented and tested with several antennas. In Figure 3.1, 'user1', 'user2'.....represent the user terminals which are used by telescope operators to interface with OnlineV2 which in turn will interface with all the GMRT subsystems at the antenna base and in the CEB. The user terminals, Hub and background task manager - all part of the Python environment interface with the shared memory which has all the control and responses from OnlineV2 recorded.



### 3.2 Communication with OnlineV2

Interactive user input terminals are connected to **HUB** through **TCP/IP** sockets. And **HUB** is connected to **OnlineV2** through a socket. Communication between **USER** and **OnlineV2** is handled by **HUB**. User communicates with **OnlineV2** through **HUB** using protocol based on **C** structure.

The listing 1 shows common structure for all commands send to **OnlineV2** from user.

Listing 1: Common command structure

```
typedef struct {
    int user_no;
    int cmd_no;
    int array ;
    int no_of_antennas;
    char antenna_list[4][32];
    char system[16];
} common;
```

The listing 2 shows structure used for servo commands. It is same structure which is used by **OnlineV2** for communication between **OnlineV2** and servo system of antennas.

Listing 2: Servo command structure

```
typedef struct {
    int seq;
    char timestamp[64];
    char system_name[16];
    char op_name[16];
    short int number_param;
    char para_name[32][16];
    char para_value[32][16];
} servocmd;
```

Structure shown in listing 3 is used to send commands which are connected through **RABBIT MCM**. This is same structure used for communication between **OnlineV2** and systems connected through **rabbit MCM**.

Listing 3: Rabbit MCM command structure

```
typedef struct {
    int seq;
    char timestamp[64];
    char system_name[16];
    char op_name[16];
    short int number_param;
    char parameter_name[32][16];
    char Argument_Ch1[32][16];
    char Argument_Ch2[32][16];
} cmd;
```

### 3.3 Application Programming Interface(API) and Procedures

It is OBJECT ORIENTED PYTHON package to give control and monitor commands to OnlineV2. Architecture overview of API with PROCEDURES are shown in figure 3.2

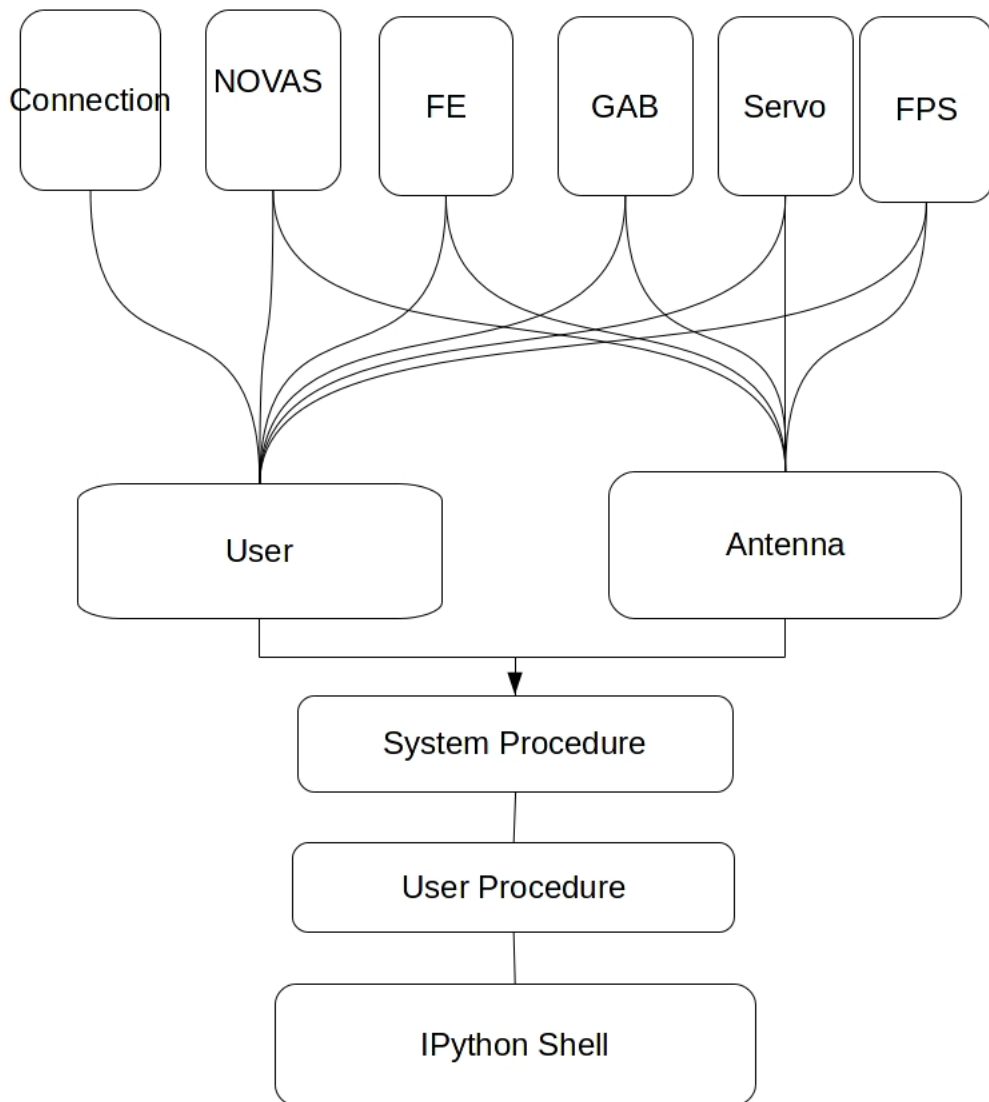


Figure 3.2: Object Oriented design of API

Figure 3.2 shows class hierarchy of different systems. **Connection, GAB, NOVAS, FE, Servo and FPS** represent base class for handling socket connection to

OnlineV2, commands related to GAB, commands related to NOVAS, commands related to FE , commands related to servo and commands related to FPS respectively.

**User and Antenna** represent another derived class which has relationship with above mentioned classes. Upto this part it called as Control API.

Using this API system procedures are written and again these are used in user procedure layer, which are directly used in **IPython** shell.

## 4 Astronomical Calculations

### 4.1 NOVAS

The NAVAL OBSERVATORY VECTOR ASTROMETRY SOFTWARE (**NOVAS**) is a software library for astrometry-related numerical computations. It is developed by the Astronomical Applications Department, United States Naval Observatory. We have studied different libraries i.e. STARLINK, IAUSOFA, astropy and NOVAS for use of astronomical calculations. NOVAS has capability to get precessed position of celestial objects as well solar system objects like SUN, Jupiter and other planets and was selected by us for use in OnlineV2.

NOVAS consist of several astronomical utilities in C language. To call these several wrapper functions were developed as explained below.

### 4.2 Wrapper functions for NOVAS

Following are wrapper functions implemented in C.

1. *precess\_radec\_mean*: It returns mean value of precessed right ascension and declination to given epoch.
2. *precess\_radec\_app* : It returns apparent value of precessed right ascension and declination to given epoch.
3. *radec2azel* : It returns astronomical azimuth and elevation for given RA and DEC, at given time, for given location i.e. latitude and longitude.
4. *riseset* : It returns rise, transit and set time of stars for given time.
5. *get\_lst* : It returns **local sidereal time** for given time in seconds. Depending of option it gives mean or apparent LST

Following is list of C function prototypes.

```
Hms hrs_hms(double hrs);
Dms degs_dms(double degrees);
int precess_radec_mean( double ra_hrs ,double dec_degs , time_t epoch2 ,double *pra ,double *pdec);
int precess_radec_app( double ra_hrs ,double dec_degs , time_t epoch2 ,double *pra ,double *pdec);
int radec2azel(time_t secs, double ra, double dec, int ref_opt,on_surface *location, double *az, double *el);
int riseset(double ra, double dec, time_t secs, double *risetime , double *settime, double *transittime);
double secs2jdutc(time_t secs);
double get_lst(time_t curr_time_secs, int lst_type);
```

### 4.3 Wrapper functions for PHP, PERL and PYTHON

Native interface functions in *PHP, PYTHON and PERL* for different functions described in subsection 4.2 are implemented using **SWIG**. SWIG is a software development tool that connects programs written in C and C++ with a variety of scripting languages.

*Examples code in PERL and PHP are given below*

```
#!/usr/bin/perl -w
# use gnovas package
use gnovas;

$secs = time();
# To calculate local sidereal time
$lst=gnovas::get_lst(int($secs),1);
printf( "\nLocal sidereal time: $lst\n");
# To get rise, transit set time of
$ra = 1; # in hours
$dec = 1; # in degrees
($stat,$rise,$set)=gnovas::riset($ra,$dec,int($secs));
printf( "\nRise and Set time: $rise $set");
($stat,$ra,$dec)=gnovas::precess_radec_app($ra,$dec,int($secs));
printf( "\napparent precession $stat\n $ra $dec");
($stat,$ra,$dec)=gnovas::precess_radec_mean($ra,$dec,int($secs));
printf( "\nmean precession $stat\n $ra $dec");
```

```
<?php
# gnovas interface file
require('gnovas.php');
# Get LST
$lst = get_lst(time(NULL),1);
printf( "LST in hours @ GMRT $lst\n");
$RA = 1; # hours
$DEC = 1; # degree
# Calculate rise, transit and set time
$output = riset($RA,$DEC,time(NULL));
printf("status $output[0] rise:$output[1] set:$output[2], \n");
# mean precession
$output = precess_radec_mean($RA,$DEC,time(NULL));
printf("status $output[0]mean ra(hrs):$output[1] dec(deg):$output[2], \n");
# apparent precession
$output = precess_radec_app($RA,$DEC,time(NULL));
printf("status $output[0]app ra(hrs):$output[1] dec(deg):$output[2], \n");
?>
```

While the wrapper function in C are developed so that OnlineV2 software can call the NOVAS routine for precession, time conversion and other astronomical calculations, the PERL and PHP wrapper are provided to be able to interface the NOVAS library for web-based operations.

## 5 Under Development

### 5.1 Monitoring API in Python for OnlineV2

Monitoring API will be python package to read status of command, parameters related to antenna, correlator, GAB and other systems. These monitored parameters can be used in command file, since command file is itself PYTHON script. And using these parameters as input, logical decisions can be taken while running background task. Which will help in automation of science observations and systems tests.

For example if any antenna applies brake, hold command can be issued without human intervention. Similar known problems can be handled by software itself.

### 5.2 Background task manager

It will handle background task, such as science observation or system test without affecting user input terminal. That is task will run in background and input terminals will be freed to accept other commands from standard input.

### 5.3 GUI

GUI using python and QT is under development. We will make use of the GUI developed for direct interface to OnlineV2 ( see [report](#) by Naresh Sisodiya in this OnlineV2 report series).

## 6 References

[www.python.org](http://www.python.org)

[www.ipython.org](http://www.ipython.org)

Explanatory Supplement to the Astronomical Almanac - by Sean Urban (Editor), P.  
Kenneth Seidelmann (Editor)

[www.aa.usno.navy.mil/software/novas/novas\\_info.php](http://www.aa.usno.navy.mil/software/novas/novas_info.php)

[www.swig.org](http://www.swig.org)