# aaPanel Linux Panel

# Plug-in development documentation

**Plug-in installation location:** /www/server/panel/plugin/
**File structure:**
**# Plug-in backend main program (plugin name_main.py) If you are developing a plug-in using PHP, please put the main program in index.php**
*demo_main.py*

**# Plug-in front-end program**
*index.html*

**# Plug-in information file**
*info.json*

**# Plugin icon file**
*icon.png*

**# Install and uninstall scripts**
*install.sh*

**Plugin name_main.py content format**
Take the plug-in name as demo:

```
#!/usr/bin/python
# coding: utf-8
# The Class name must match the file name
class demo_main:
    #Your code
    #If this method needs to be called by the front end, then receive POST / GET parameters from the front end through args
    def get_list(self,args):
          pass;
```

**index.php Example:**
*Note: When using PHP to develop a plug-in, the demo_main.py file cannot be placed in the plug-in directory, otherwise index.php cannot be called.*

```
class bt_main{
    //Do not allow methods accessed by the panel to be set to public methods
    // Your code
   public function test(){
    }
}
```

**info.json Content format:**

```
{
  "title": "aaPanel Plugin DEMO",                              // Title name
  "name": "demo",                                              // The name must be the same as the
plugin directory name
  "ps": " aaPanel Plug-in development example project ",       // description
  "versions": "1.0",                                           // version number
  "checks": "/www/server/panel/plugin/demo",                   // Files or directories used to
detect if the plugin is installed
  "author":"aaPanel",                                          // Author
  "home":"https://www.bt.cn"                                   // Author Home
}
```

**Response data to the front end:**

Data types that allow return to the front end: bool, str, list, dict, int

No matter what type of data is returned, it will be automatically converted to json data format. If it is not supported by the json object, it will directly report an error.

**Demo Download：** https://www.aapanel.com/pkg/demo.zip

**Note:**

1. URLs are strictly case sensitive
2. Please use the POST method for the front-end request plugin
3. For more detailed instructions, please refer todemo
4. Please do not use the four parameter names of [action, s, name, a] in the POST parameters

# public

**Quote:** import public

**Read file**

#@param filename *File name [Required]*
#@param mode *File Open Mode Default r*
#@return bool
f_body = public.ReadFile(filename,mode='r')

**Write file**

#@param filename *File name [Required]*
#@param f_body *File content[Required]*
#@param mode *File Open Mode Default w+*
#@return bool
result = public.WriteFile(filename,f_body,mode='w+')

**Calculate MD5 of a string**

#@param strings *String to be calculated [required]*
#@return string *Lowercase MD5 results*
md5 = public.md5(strings)

**Calculate the MD5 of a file**

#@param filename *File name*
#@return string *Lowercase MD5 results*
file_md5 = public.FileMd5(filename)

**Write panel log**

#@param log_type *Log Type [Required] Example: Login Panel*
#@param log_body *Log content [required] For example: login successful*
#@return None
public.WriteLog(log_type,log_body)

**Use GET to request HTTP**

#@param url *URL Address [Required]*
#@param timeout *Timeout time 60 seconds by default*
#@return string *Successfully return http response content, fail return error code*
http_body = public.HttpGet(url,timeout=60)

**Use HTTP to request HTTP**

#@param url *URL Address [Required]*
#@param p_data *POST content, please pass in the dictionary (dict) [Required]*
#@param timeout *Timeout time 60 seconds by default*
#@return string *Successfully return http response content, fail return error code*
http_body = public.HttpPost(url,p_data,timeout=60)

**Take a random string**

#@param length *Random string length to be obtained [Required]*
#@return string
randmo_string = public.GetRandomString(length)

**Constructing generic response content**

#@param status *Response status [Required] True| False*
#@param return_msg *Response content [Required]*
return public.ReturnMsg(True,' Successful operation!')

**Get the permissions of the specified file**

#@param filename *File name [Required]*
#@return string *Authorization character device such as 755*
f_mode = public.GetFileMode(filename)

**Get the current web server**

#@return string *Server type nginx|apache*
web_server = public.GetWebServer()

**Reload the current web server**

public.ServiceReload()

**Reload the specified PHP version**

#@param version *PHP version For example, if you want to reload php7.2: 72*

public.phpReload()

**Execute the Shell command through the pipeline**

#@param shell_str *SHELL command to be executed [required]*

#@return list *Command execution result Return format: ["normal output", "abnormal output"]*

result = public.ExecShell(shell_str)

**Get the IP address of this server**

#@return string *IP address*

ip = public.GetLocalIp()

**Get user IP address**

#@return string *IP address*

client_ip = public.GetClientIp()

**Get the currently accessed HOST and port information**

#@param port *Return port [optional] True | False Default is False*

host = public.GetHost()

port = public.GetHost(True)

**Take the contents of the specified number of lines at the end of the file**

#@praam filename *File name [Required]*

#@param num *Specify the number of rows to fetch [Required]*

#@return string *Content obtained*

last_body = public.GetNumLines(filename,num)

**Byte Unit Conversion (KB, MB, GB, TB)**

#@param b_size *The number of bytes to be converted [required]*

#@return string *Conversion result*

size = public.to_size(b_size)

**Determines whether the specified process exists**

#@param process_name *Process name [Required]*

#@param exe *Execution file path [optional] Default is None No judgment, if a value is passed in, verify the execution file path*

#@return bool

is_exists = public.process_exists(process_name,exe=None)

**Get server mac address**

```
mac_address = public.get_mac_address()
```

**Restart panel**

```
public.restart_panel();
```

**Take the current format time**

```
#@param format          Format, default is %Y-%m-%d %X    2018-12-10 00:00:00
format_date = public.getDate(format='%Y-%m-%d %X')
```

**#XSS Filtering**

```
#@param data            The string to be filtered [required]
input_body    = public.checkInput(data)
```

**Get the sqlite database object of the panel's own database**

```
#@param table           Table name
#@return                Database object
db_obj = public.M('sites')
```

## Panel database object method list

(Note: The following example is only a database operation. For actual addition, deletion, and modification, please call the corresponding interface from the front end to complete the operation.)

**Get list of websites with site id 1 or 2**

```
data = public.M('sites').where('id=? or id=?',(1,2)).field('id,name,edate,path,status').select()
```

**Get list of all websites and use id descending**

```
data = public.M('sites').field('id,name,edate,path,status').order('id desc').select()
```

**Get a piece of data that meets the conditions**

```
find = public.M('sites').where('id=?',(id,)).field('id,name,edate,path,status').find()
```

**Get a piece of data that meets the conditions**

```
site_id = public.M('sites').where('name=?',('www.bt.cn',)).getField('id')
```

**Get a list of domain names by website ID and use id ascending order**

```
domains = public.M('sites').where('pid=?',(site_id,)).order('id asc').field('id,pid,name,port,addtime').select()
```

**Delete log**

```
public.M('logs').where('id=?',(id,)).delete()
```

**change the data**

```
public.M('logs').where(id=?,(id,)).save('type,log',' login ',' login successful!')
```

**Execute SQL statement to return affected rows**

```
public.M('logs').execute(sql)
```

**Execute SQL statement to return query results**

```
public.M('logs').query(sql)
```

**Constructing paginated data**

```
#@param count          Number of data rows [Required]
#@param p              Current page default = 1
#@param rows           Lines per page Default = 12
#@param callback js    Callback method name, if you are calling paginated data via JS, please pass in js callback method name
#@param result         Paging structure default = '1,2,3,4,5,8', the complete is: '1,2,3,4,5,6,7,8' Please adjust according to demand
#@return {'page': Structured release data,'shift': Offset position,'row': Offset }
page_data = public.get_page(count,p=1,rows=12,callback='',result='1,2,3,4,5,8')
```

**Example: Get website data and construct pagination**

```
# Take the number of websites
count = public.M('sites').count()
Get paging data
page_data = public.get_page(count,p=1,rows=12,callback='get_sites_list',result='1,2,3,4,5,8')

#Get the list of data by    page_data ['shift'], page_data ['ros']
site_list = public.M('sites').order('id desc')
```

```
    .limit(page_data['shift'],page_data['ros']).field('id,name,edate,path,status').select()

    #Construct return dictionary
    data = {'data':site_list,'page':page_data['page']}
    return data
```

# Task queue object

Reference module： import panelTask
Instantiate object： t = panelTask.bt_task()

## Create task queue

result = t.create_task(task_name,task_type,task_shell,other='')

## Parameter Description:

#@param task_name *String Task name [Required]*
#@param task_type *Integer Task type [Required]*
*[0]. Execute Shell*
*[1]. Download the file*
*[2]. Decompress the file*
*[3]. Compress the file*
#@param task_shell String SHELL command [Required]
*If task_type = 0, enter the SHELL command*
*If task_type = 1, please pass in the URL*
*If task_type = 2, please enter the full path of the compressed file*
*If task_type = 3, pass in the parent path of the compressed file or directory (full)*
#@param other String Other parameters [Required]
*If task_type = 0, this parameter can be omitted.*
*If task_type = 1, please enter the full path to save the download file, such as: /www/test.zip*
*If task_type = 2, please pass in a JSON string, for example: {"dfile": "full path to the unzipped directory", "password": "unzipped password (password string without password)"}*
*If task_type = 3, please pass in a JSON string, for example: {"sfile": "compressed file*

*or directory name (non-full path)", "dfile": "is used to save the full path of the compressed package, such as: /www/test.zip","z_type":"Compression type (example: tar.gz, rar, zip) "}*

## Example of executing the SHELL command:

t.create_task('View partition',0,'df -h')

## download file：

t.create_task('download file test.zip',1,'http://www.bt.cn/test.zip','/www/test.zip')

## Unzip the file to /www/test：

t.create_task('Unzip test.zip',2,'/www/test.zip','{"sfile":"/www/test","password":""}')

## Compress the test directory under / www directory to /www/test.zip：

t.create_task('compression test',3,'/www',{"sfile":"test","dfile":"/www/test.zip","z_type":"rar"})

# Plug-in dynamic routing

**Foreword:** Plug-in dynamic routing does not inherit panel session permissions. Developers need to control access permissions in the plugin (_check method). Please use it with caution.

**interview method:** http:// Panel Address: Port / Plugin_name / Accessed plugin method. Response type (html|json)

**Glossary:**

Plug-in name: Name field value in info.json
to be Accessed method： Method name in plugin controller class，Like get_logs in demo_main class
Response type: html Respond to json data as a template，If you do not add. (Html | json), the data returned by the

plug-in method is output as it is.

### Plug-in static files

It is mainly used to access the js / css / font / html used in the plug-in. Please do not put important data in the static directory.
Static directory name: static

Access example
To access the static files of the demo plugin: ./static/js/test.js
http://demo.cn:8888/demo/static/js/test.js

### Use template

The panel uses the jinja2 template engine to render the template. To use the jinja2 template in the plugin, you need to create the templates directory under the plugin directory and create a template file for the corresponding method.

Note: Jinja2 syntax is supported in the plugin template, but the extends statement cannot be used

### Template contains

Create header file ./templates/head.html
Insert include statements anywhere in the template {% include “head.html” %}

**Template response process:** User request >> check access (_check | login) >> check if template exists >> execution method >> output template variable
**About template variables**: The data returned by the plug-in method will be passed to the data variable. Please try to construct a dict and return it to the template in the plug-in method. In the template, use the data variable to access the corresponding value.

### Usage example：

Use the template to access the get_logs method in the demo plugin

1. Create template file
   ./templates/get_logs.html
   Content:
   <p> {{data [‘test’]}} </ p>
2. Write the get_logs method：
   def get_logs(self,args):
   return {‘test’:’test’}
3. Request as a template: http://demo.cn:8888/demo/get_logs.html
4. Response content test

### Develop Ajax interface (response to JSON)

To develop the ajax interface, you only need to access it with the response type of json, and the panel will automatically serialize the return value of the plugin method into a JSON string response.
Example of use:
http://demo.cn:8888/demo/get_logs.json

**Custom response**

**interview method:** http://demo.cn:8888/demo/get_logs

## Redirect

Sample code：

```
from flask import redirect    # Reference redirect object
def get_logs(self,args):
    return redirect('/login',302)    # Return redirection object redirection code support (302 | 301 | 303 | 305 | 307)
```

## Output HTML directly

Sample code：

```
def get_logs(self,args):
    html_body = "<html></html>"
    return html_body
```

## Output file

Sample code：

```
from flask import send_file    # Reference File Send Object
def get_file(self,args):
    return send_file('./test.zip')
```