PUSR®
www.pusr.com

Communication Expert of

Industrial IOT

USR-G809 Secondary Development

Product description

The G809/G807 series is a new generation of flagship routers for industrial applications. Equipped with high-end Qualcomm main control, it offers two series: 5 Gigabit Ethernet ports and Gigabit（2*SFP+8*RJ45）. It supports optional 4G cellular network boards, with 1GB RAM, 8GB storage, and Python secondary development capabilities. Featuring Qualcomm dual-band Wi-Fi 6 and rich hardware interfaces, it includes: 1*RS232 + 1*RS485 dual serial ports, 1*DI/DO, dual SIM redundancy, GNSS positioning, OLED display, USB port, and SD interface.

Designed to industrial standards, the product operates in -40°C~75°C temperatures, accepts DC 9-60V power, and has robust hardware protection. Tested in harsh environments, it features dual hardware/software watchdogs and self-recovery for reliable performance across industries.

The product features standard DIN-rail mounting and a compact design. It applies to smart factories, intelligent manufacturing, industrial robotics, smart warehousing, drones, smart healthcare, vehicle networking, industrial automation, smart transportation, smart cities, smart industries, security & criminal investigation, and driverless scenarios.

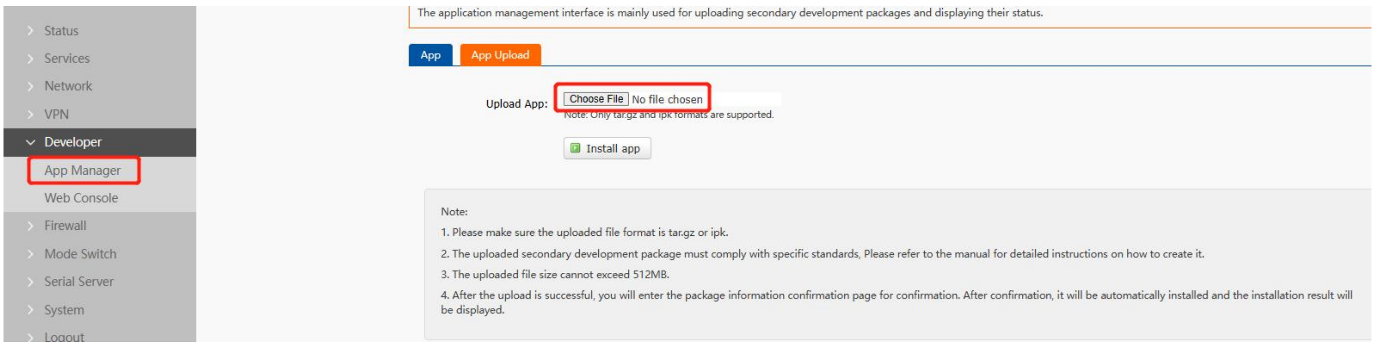
1. For Developers

1.1 Application Management

Customers can install and run secondary development C programs / Python programs / shell scripts on the router through Application Management.

1.1.1 Uploading Secondary Development Programs

On the upload page, click 'Choose File' - then click 'Install Application'



The application management interface is mainly used for uploading secondary development packages and displaying their status.

Upload App: No file chosen

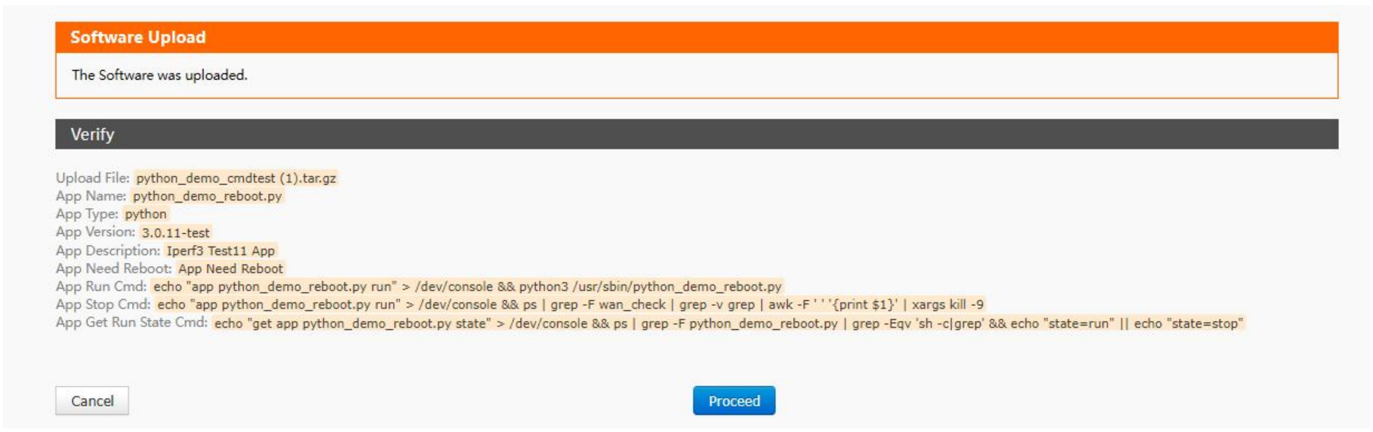
Note: Only tar.gz and ipk formats are supported.

Note:

1. Please make sure the uploaded file format is tar.gz or ipk.
2. The uploaded secondary development package must comply with specific standards, Please refer to the manual for detailed instructions on how to create it.
3. The uploaded file size cannot exceed 512MB.
4. After the upload is successful, you will enter the package information confirmation page for confirmation. After confirmation, it will be automatically installed and the installation result will be displayed.

Fig. 1 Upload files

After uploading the application, the properties of this program are displayed.



Software Upload

The Software was uploaded.

Verify

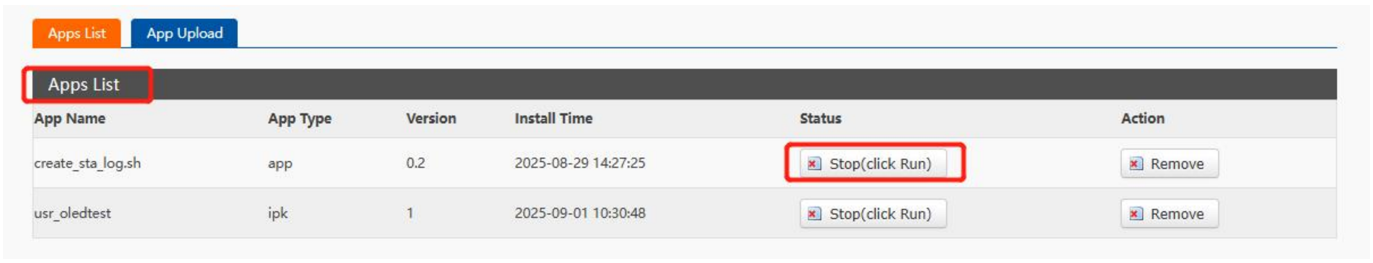
Upload File: python_demo_cmdtest (1).tar.gz
App Name: python_demo_reboot.py
App Type: python
App Version: 3.0.11-test
App Description: Iperf3 Test11 App
App Need Reboot: App Need Reboot
App Run Cmd: `echo "app python_demo_reboot.py run" > /dev/console && python3 /usr/sbin/python_demo_reboot.py`
App Stop Cmd: `echo "app python_demo_reboot.py run" > /dev/console && ps | grep -F wan_check | grep -v grep | awk -F ' ' '{print $1}' | xargs kill -9`
App Get Run State Cmd: `echo "get app python_demo_reboot.py state" > /dev/console && ps | grep -F python_demo_reboot.py | grep -Eqv 'sh -c|grep' && echo "state=run" || echo "state=stop"`

Fig. 2 Check properties

After confirming the package properties, click 'Proceed' to complete the program installation.

View the list of installed programs in the Application List. You can click 'Run' to execute the secondary development program (if the package property has 'Auto-start at boot' enabled, you only need to click 'Run' once after installation; it will start automatically on subsequent boots).

If an application package is no longer needed, you can click 'Remove' to uninstall it.



Apps List

App Name	App Type	Version	Install Time	Status	Action
create_sta_log.sh	app	0.2	2025-08-29 14:27:25	Stop(click Run)	Remove
usr_oledtest	ipk	1	2025-09-01 10:30:48	Stop(click Run)	Remove

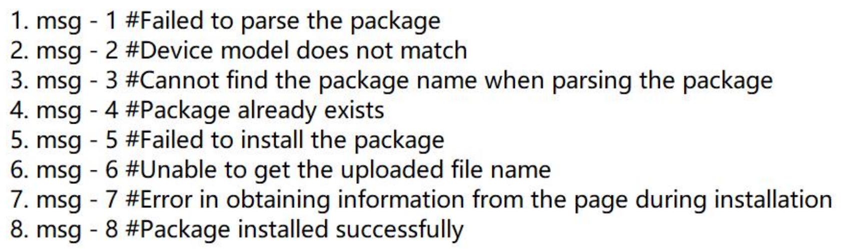
Fig. 3 Apps list

1.2 Back-end implementation logic

1.2.1. Webpage logic

1. Click upload file, first check whether the file format selected is normal, abnormal pop-up prompt.

2. After the format check is completed, the attribute will be parsed and checked. If the attribute does not conform to the specification, the parsing fails and an error is prompted. The error code is as follows:



1. msg - 1 #Failed to parse the package
2. msg - 2 #Device model does not match
3. msg - 3 #Cannot find the package name when parsing the package
4. msg - 4 #Package already exists
5. msg - 5 #Failed to install the package
6. msg - 6 #Unable to get the uploaded file name
7. msg - 7 #Error in obtaining information from the page during installation
8. msg - 8 #Package installed successfully

Fig. 4 error code

3. After the application is checked, install it into the file system and give it permission to run.

4. After the application is installed, it will not run. After clickingRun on the page, it will call the backend program. The backend program will find the corresponding package according to the package name and then run it in an appropriate way. The running application status is listed as "Running (click Stop)"

5. When stopping the application, after clicking on the page, the backend program will be called. The backend program will find the corresponding package according to the package nameand then stop it in an appropriate way. The status of the stopped application is listed as "Not running (click to run)"

6. To remove an installed app, click the Delete button and the backend program is invoked. The backend program finds the corresponding package according to the package name and deletes all related files and information.

1.2.2 Two-pack design logic

1. Supports uploading data packages in ipk and tar.gz formats. ipk is the openwrt native support package format, and tar.gz is a custom package that needs to be created in the specified format.

2. ipk is an openwrt standard package, which is made using ope nWRT standard package method.

3. Tar.gz package for linux executable programs and libraries, which need to contain two files.

① The control file, which contains various information for the installation script to identify the attributes of this package.The information contained is as follows:

②data.tar.gz is an app package in compressed format. The directory structure of this package should be prepared in advance, such as/usr/bin/app1/usr/lib/aaa.so 2, etc.

③ The installation program sets configuration files based on the properties in control (such as package type, whether to auto-start, package name, etc.).

④ During installation, all installed content is recorded.

4. After the package is installed successfully, click Run.

① For IPK, there are three ways to start, the priority is as follows:

I. Check if there is an init.d script, if there is, execute this script restart

Ⅱ. Check if there isRunCmd attribute, if there is, execute this attribute RunIII. Package name Run (package name must be the same asapp)

III. Run by package name (the package name must be the same as the app name).

② For custom packages, there are two ways to start, with the following priorities:

Ⅰ. Check if there isRunCmd attribute, if there is, execute this attribute RunII. Package name Run (package name must be the same asapp)

II. Run by package name (the package name must be the same as the app name).

5. Package is running, click Stop when:

① For IPK, there are three ways to stop, the priority is as follows:

I. Check if there is an init.d script, if there is, execute this script stop

Ⅱ. Check if there is StopCmd attribute, if there is, execute this attribute Run

III. Stop by package name (the package name must be the same as the app name).

② For custom packages, there are two ways to start, with the following priorities:

Ⅰ. Check if there is StopCmd attribute, if there is, execute this attribute Run

II. package name stop (package name must be the same as app)

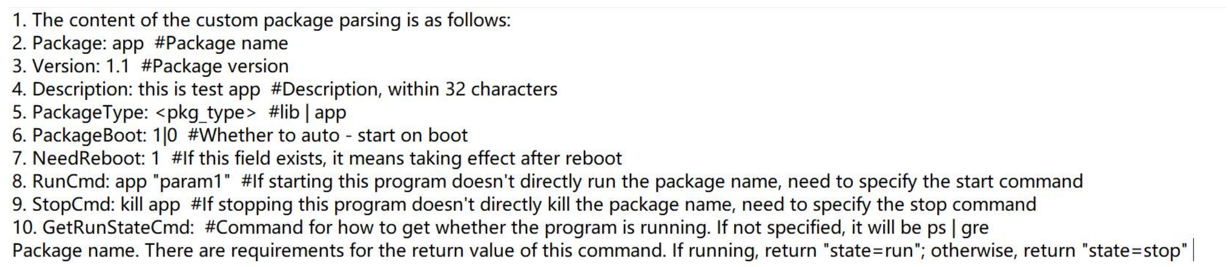
6. Get program running status:

① Check whether GetRunStateCmd attribute exists. If yes, call the command specified by this attribute to obtain it. The return value of this command is required. If running,return "state=run", otherwise return "state=stop".

②Check whether it is runningby package name>(package name must be the same as app

7. Auto-start at boot

① After the system is started, it will traverse the installed packages, find the ones that need to be started, and then run them using the above operation logic.



```
1. The content of the custom package parsing is as follows:
2. Package: app  #Package name
3. Version: 1.1  #Package version
4. Description: this is test app  #Description, within 32 characters
5. PackageType: <pkg_type>  #lib | app
6. PackageBoot: 1|0  #Whether to auto - start on boot
7. NeedReboot: 1  #If this field exists, it means taking effect after reboot
8. RunCmd: app "param1"  #If starting this program doesn't directly run the package name, need to specify the start command
9. StopCmd: kill app  #If stopping this program doesn't directly kill the package name, need to specify the stop command
10. GetRunStateCmd:  #Command for how to get whether the program is running. If not specified, it will be ps | gre
Package name. There are requirements for the return value of this command. If running, return "state=run"; otherwise, return "state=stop"
```

Fig. 5 control package property file format

1.2.3 C API Library

For details, please refer to: 2D Toolkit\C Language 2D API-demo + Dynamic Library + Compiler Toolchain\op_usr_basic. tar. gz\libusr_basic\include\usr_basic. h file description

Link: https://www.pusr.com/support/download/API-C-language-API-demo-dll-Cross-Compile-tool-chain.html

1.2.4 Python API Library

For details, please refer to: Two-Open Toolkit\Python Two-Open API\python Two-Open Interface Description

Link: https://www.pusr.com/support/download/API-python-API-interface-description.html

1.2.5 Demo from PUSR

①usr_oledtest_1_ipq.ipk：

Link: https://www.pusr.com/support/download/demo-demo-ipk.html

OLED secondary development program. After installation and running, the following content will be displayed on the

OLED:

Hello World

Hi OLED

After logging into the web console, you can execute:

usr_oledtest"<Custom Page>""Lines>"<Contents>"or test_oled.sh"<Custom Page>""<Lines>"<Contents>"

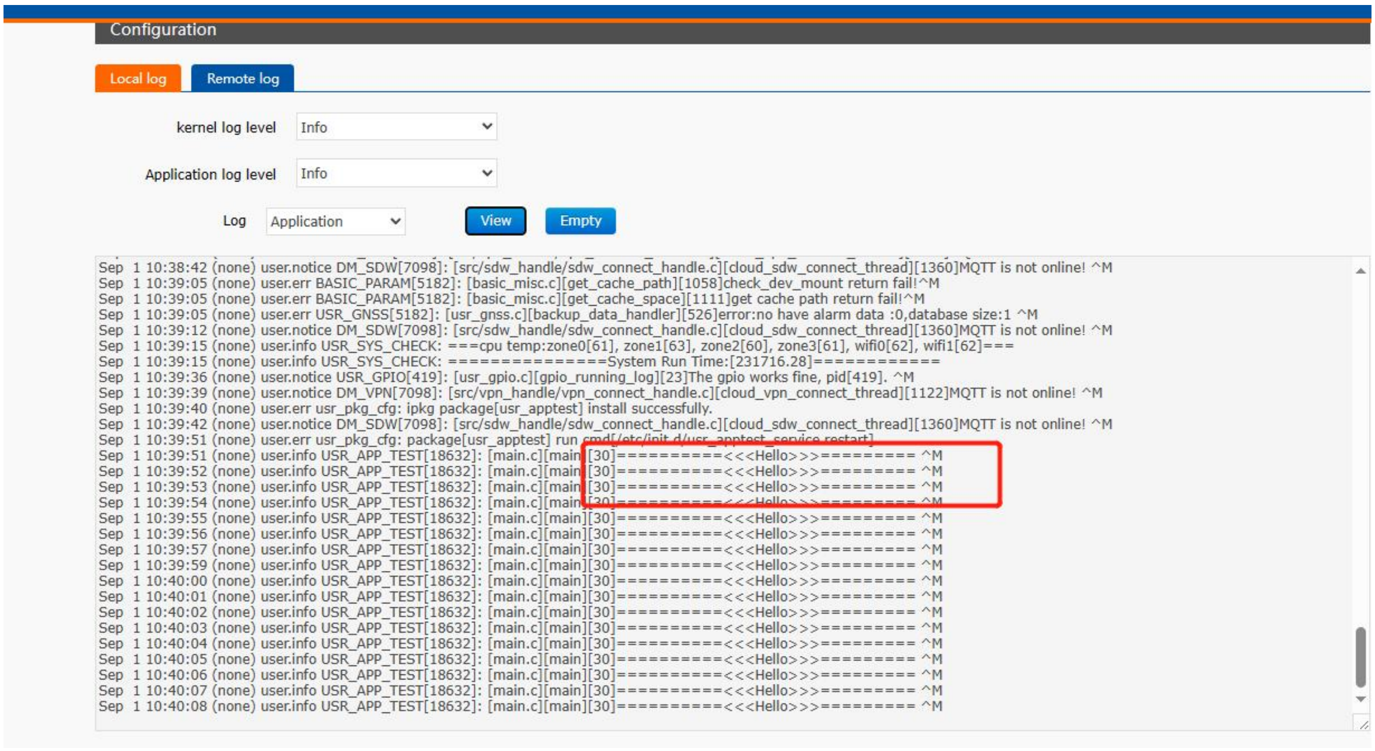
Example:

usr_oledtest "2" "1" "aaa"　　aaa will be displayed on the first line of customization page 1

②usr_apptest　1　ipq.ipk：

Link: https://www.pusr.com/support/download/demo-demo-ipk.html

will always print =====<<<Hello>>>=== ======



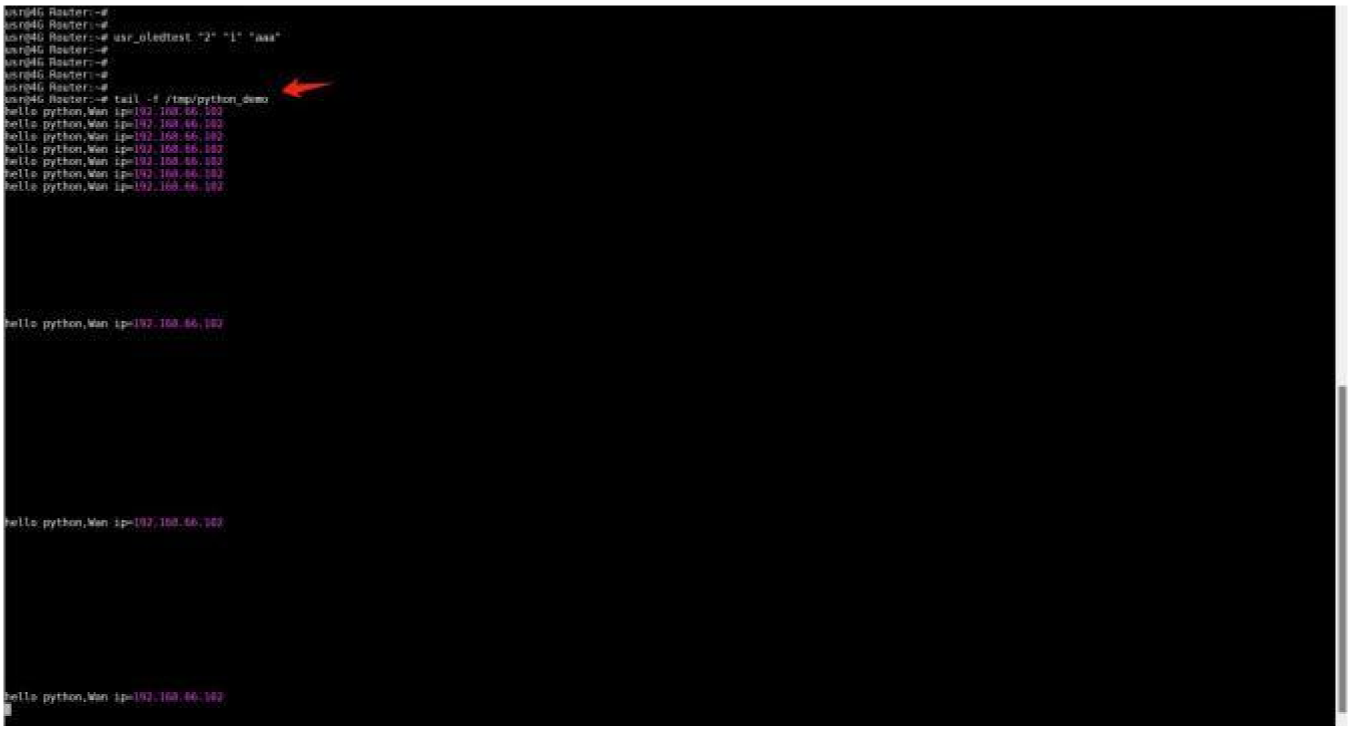
```
Sep  1 10:38:42 (none) user.notice DM_SDW[7098]: [src/sdw_handle/sdw_connect_handle.c][cloud_sdw_connect_thread][1360]MQTT is not online! ^M
Sep  1 10:39:05 (none) user.err BASIC_PARAM[5182]: [basic_misc.c][get_cache_path][1058]check_dev_mount return fail!^M
Sep  1 10:39:05 (none) user.err BASIC_PARAM[5182]: [basic_misc.c][get_cache_space][1111]get cache path return fail!^M
Sep  1 10:39:05 (none) user.err USR_GNSS[5182]: [usr_gnss.c][backup_data_handler][526]error:no have alarm data :0,database size:1 ^M
Sep  1 10:39:12 (none) user.notice DM_SDW[7098]: [src/sdw_handle/sdw_connect_handle.c][cloud_sdw_connect_thread][1360]MQTT is not online! ^M
Sep  1 10:39:15 (none) user.info USR_SYS_CHECK: ===cpu temp:zone0[61], zone1[63], zone2[60], zone3[61], wifi0[62], wifi1[62]===
Sep  1 10:39:15 (none) user.info USR_SYS_CHECK: ===============System Run Time:[231716.28]============
Sep  1 10:39:36 (none) user.notice USR_GPIO[419]: [usr_gpio.c][gpio_running_log][23]The gpio works fine, pid[419]. ^M
Sep  1 10:39:39 (none) user.notice DM_VPN[7098]: [src/vpn_handle/vpn_connect_handle.c][cloud_vpn_connect_thread][1122]MQTT is not online! ^M
Sep  1 10:39:40 (none) user.err usr_pkg_cfg: ipkg package[usr_apptest] install successfully.
Sep  1 10:39:42 (none) user.notice DM_SDW[7098]: [src/sdw_handle/sdw_connect_handle.c][cloud_sdw_connect_thread][1360]MQTT is not online! ^M
Sep  1 10:39:51 (none) user.err usr_pkg_cfg: package[usr_apptest] run cmd[/etc/init.d/usr_apptest service restart]
Sep  1 10:39:51 (none) user.info USR_APP_TEST[18632]: [main.c][main][30]==========<<<Hello>>>========== ^M
Sep  1 10:39:52 (none) user.info USR_APP_TEST[18632]: [main.c][main][30]==========<<<Hello>>>========== ^M
Sep  1 10:39:53 (none) user.info USR_APP_TEST[18632]: [main.c][main][30]==========<<<Hello>>>========== ^M
Sep  1 10:39:54 (none) user.info USR_APP_TEST[18632]: [main.c][main][30]==========<<<Hello>>>========== ^M
Sep  1 10:39:55 (none) user.info USR_APP_TEST[18632]: [main.c][main][30]==========<<<Hello>>>========== ^M
Sep  1 10:39:56 (none) user.info USR_APP_TEST[18632]: [main.c][main][30]==========<<<Hello>>>========== ^M
Sep  1 10:39:57 (none) user.info USR_APP_TEST[18632]: [main.c][main][30]==========<<<Hello>>>========== ^M
Sep  1 10:39:59 (none) user.info USR_APP_TEST[18632]: [main.c][main][30]==========<<<Hello>>>========== ^M
Sep  1 10:40:00 (none) user.info USR_APP_TEST[18632]: [main.c][main][30]==========<<<Hello>>>========== ^M
Sep  1 10:40:01 (none) user.info USR_APP_TEST[18632]: [main.c][main][30]==========<<<Hello>>>========== ^M
Sep  1 10:40:02 (none) user.info USR_APP_TEST[18632]: [main.c][main][30]==========<<<Hello>>>========== ^M
Sep  1 10:40:03 (none) user.info USR_APP_TEST[18632]: [main.c][main][30]==========<<<Hello>>>========== ^M
Sep  1 10:40:04 (none) user.info USR_APP_TEST[18632]: [main.c][main][30]==========<<<Hello>>>========== ^M
Sep  1 10:40:05 (none) user.info USR_APP_TEST[18632]: [main.c][main][30]==========<<<Hello>>>========== ^M
Sep  1 10:40:06 (none) user.info USR_APP_TEST[18632]: [main.c][main][30]==========<<<Hello>>>========== ^M
Sep  1 10:40:07 (none) user.info USR_APP_TEST[18632]: [main.c][main][30]==========<<<Hello>>>========== ^M
Sep  1 10:40:08 (none) user.info USR_APP_TEST[18632]: [main.c][main][30]==========<<<Hello>>>========== ^M
```

Fig. 6　　print content

③python_demo_cmdtest.tar.gz：

Link: https://www.pusr.com/support/download/demo-demo-python.html

After installation and operation, continuously print the IP of WAN1 network card to the file/tmp/python_demo (Note: insert WAN1 network cable, wait for WAN1 network card to have IP and test)

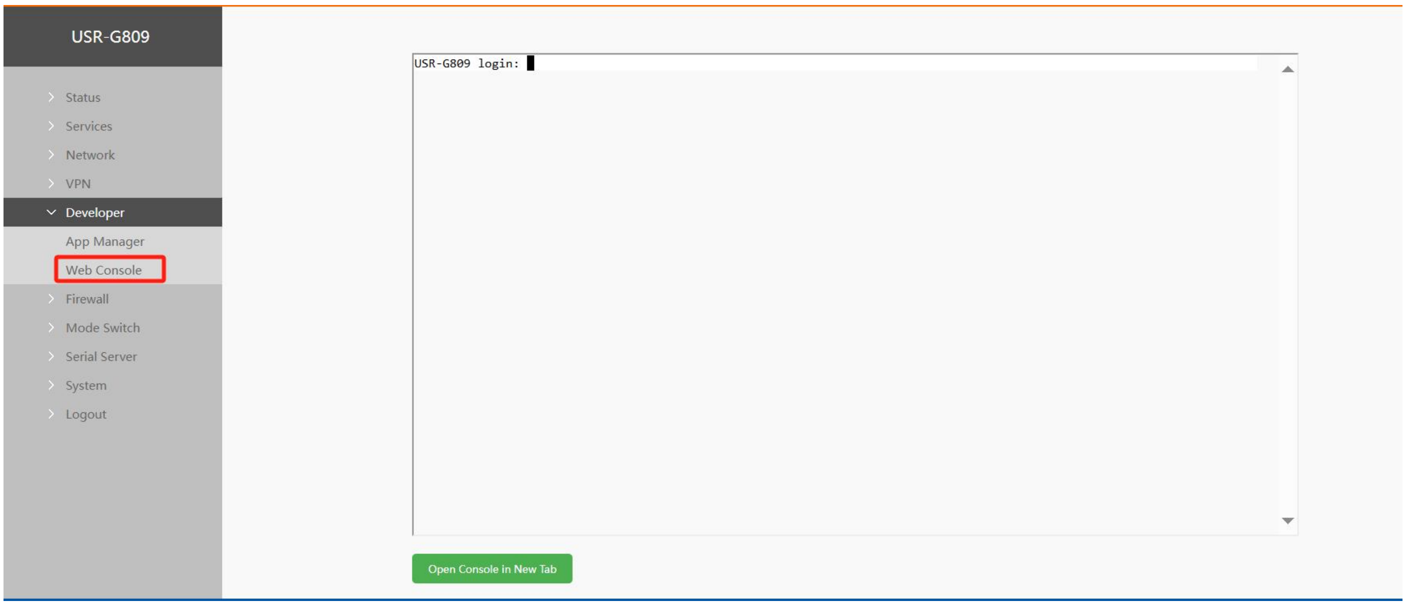


```
usr@4G Router:~#
usr@4G Router:~#
usr@4G Router:~# usr_oledtest "2" "1" "aaa"
usr@4G Router:~#
usr@4G Router:~#
usr@4G Router:~#
usr@4G Router:~#
usr@4G Router:~# tail -f /tmp/python_demo
hello python,Wan ip=192.168.66.102
hello python,Wan ip=192.168.66.102
hello python,Wan ip=192.168.66.102
hello python,Wan ip=192.168.66.102
hello python,Wan ip=192.168.66.102
hello python,Wan ip=192.168.66.102
hello python,Wan ip=192.168.66.102



hello python,Wan ip=192.168.66.102



hello python,Wan ip=192.168.66.102



hello python,Wan ip=192.168.66.102
```

Fig. 7 Print WAN1IP to File

1.3 Web console

Use account/password: root/root to log in to the router management background to debug the second open program.



```
USR-G809 login:
```

Fig. 8 web console