# ACTi WAN Daemon

|  |  |  |
| --- | --- | --- |
| Change log | | |
| Version | Item | Date |
| V1.0.00 | Create this document. | 2009/05/15 |
| V1.0.01 | Remove API “WdSetLinkSpeed”.  Add API “WdSetHostName”.  Remove “SPEED\_WAN” and “SPEED\_LAN” in “Config file”  Remove “WanSpeed” and “LanSpeed” in “Struct of tWanInfo”  Add “HostName” in “Struct of tWanInfo”. | 2009/05/25 |
| V1.0.02 | Add API “WdExitNotify”. | 2009/6/8 |
| V1.0.03 | Update User flow.  Update Config File.  Update WAN Daemon API.  Add Example | 2009/6/11 |
| V1.0.04 | Update User flow.  Add API “WdSetBonjour”.  Update “Struct of tWanInfo”  Modify the input arguments of “WdSetDNS”, “WdSetDDNS” and “WdSetHostName” | 2009/11/9 |
| V1.0.05 | Wording amendment by MingYoung | 2010/01/06 |
| V1.0.06 | Arranging The Wan Daemon API | 2012/04/16 |
| V1.0.07 | Updata “WdGetInfo” & “WdGetStatus” API | 2012/06/01 |
| V1.0.08 | Updata non block “WdGetInfo” & “WdGetStatus” API | 2012/08/17 |

[ACTi WAN Daemon 1](#_Toc333313365)

[Change log 1](#_Toc333313366)

[I. Introduction 3](#_Toc333313367)

[II. User flow 3](#_Toc333313368)

[III. Config file 7](#_Toc333313369)

[IV. Define Struct 8](#_Toc333313370)

[V. WAN Daemon API 10](#_Toc333313371)

[ WdSetDns 10](#_Toc333313372)

[ WdSetDdns 10](#_Toc333313373)

[ WdSetBonjour 11](#_Toc333313374)

[ WdSetHostName 12](#_Toc333313375)

[ WdGetInfo 13](#_Toc333313376)

[ WdGetStatus 15](#_Toc333313377)

[ WdWait4Notify 18](#_Toc333313378)

[ WdExitNotify(not using…) 19](#_Toc333313379)

[VI. How to Integrate Wan Daemon API 20](#_Toc333313380)

## Introduction

ACTi WAN Daemon is designed for ACTi devices (IP camera, Video Server….). This Daemon can help to manage the network interface of device. The method of getting IP address in the WAN port could be “**Static**” “**DHCP**” or “**PPPoE**”. The management of “**Bonjour**”, “**DNS**”, “**DDNS**” and the **Host Name** are handled by this daemon as well .

WAN Daemon should be bringing up by the system init script(/etc/init.d/service). It will read the configuration file, from **/etc/config/update.conf** , for the configurations of network(**/etc/WanDaemon.conf**). The WAN Daemon APIs helps the applications to communicate with WAN Daemon. Through these APIs, the applications could configure the device network settings and get the runtime networking status from the WAN daemon.

## User flow

1. System init script prepare configuration file, from **/etc/config/update.conf** for WAN Daemon.
2. System init script brings up WAN Daemon. WAN Daemon reads the configuration file(**/etc/WanDaemon.conf**) and setup network.
3. Application gets networking status by calling WdGetInfo() API.
4. Setup Bonjour with “enable”, “FriendlyName” , “PortHttp”, “PortHttps” and “LLIP” configurations if the Bonjour is enabled.
5. When needs to know about the network connect status (ex. URL command “WAN\_STATUS”), Get the status with API “WdGetStatus”.
6. When user changes the network config (DNS, DDNS, Bonjour, HostName), Application can notify WAN Daemon through the APIs (WDSetDNS, WDSetDDNS, WDSetBonjour, WDSetHostName).
7. Application should provide a thread for API “WDGetNotify”. This API will block on a message queue receive for the WAN status change information. When the IP address of device change, WAN Daemon will sends these information to the message queue. WDGetNotify can helps to collect the WAN status change information.

**User flow of WAN Daemon**

**Flow diagram between WNA Daemon and the APIs**

**WAN Daemon State Diagram**

## Config file

When WAN Daemon starts, it will read config file “/etc/WanDaemon.conf”. The config items are list below:

LAN\_HOSTNAME=’ACTi’  
 WAN\_TYPE=’0’  
 WAN\_IP=’192.168.0.100’  
 WAN\_NETMASK=’255.255.255.0’  
 WAN\_GATEWAY=192.168.0.254’  
 LAN\_IP=’192.168.1.100’  
 LAN\_NETMASK=’255.255.255.0’  
 WAN\_PPPOE\_USERNAME=’’  
 WAN\_PPPOE\_PASSWORD=’’  
 DNS\_PRIMARY=’’   
 DNS\_SECONDARY=’’  
 DDNS\_TYPE=’1’ #1=Disable, 2=Enable  
 DDNS\_HOSTNAME=’’  
 DDNS\_SERVICE=’dyndns’  
 DDNS\_USERNAME=’’  
 DDNS\_PASSWORD=’’

If item “LAN\_IP” is empty, Wan Daemon will ignore LAN interface.

## Define Struct

/\*\*\*\*\*\*\* Def tBonjourInfo \*\*\*\*\*\*/

typedef struct{

int Enable;

char FriendlyName[64];

int PortHttp;

int PortHttps;

char LLIP[16];

char IfName[16];

int ChgMask;

}tBonjourInfo;

/\*\*\*\*\*\*\* Def tWD\_DDNS \*\*\*\*\*\*/

typedef struct {

int DdnsType; /\* 1)Disable 2)Enable \*/

char DdnsService[16];

char DdnsHostName[128];

char DdnsUser[64];

char DdnsPwd[64];

char DdnsServer[16];

int DdnsPort;

} tWD\_DDNS;

/\*\*\*\*\*\*\* Def tWD\_DNS \*\*\*\*\*\*/

typedef struct {

char DnsPrimary[16]; /\* Primary DNS server IP address \*/

char DnsSecondary[16]; /\* Secondary DNS server IP address \*/

int ChgMask;

} tWD\_DNS;

/\*\*\*\*\*\*\* Def of tWanInof \*\*\*\*\*\*/

typedef struct {

char WanConnected; /\* 0)Disconnect 1)Connected, No use \*/

char DnsConnected; /\* 0)Disconnect 1)Connected, No use \*/

char DdnsConnected; /\* 0)Disconnect 1)Connected \*/

char Reserve;

} tStatusInfo;

/\*\*\*\*\*\*\* Def tWAN \*\*\*\*\*\*/

typedef struct {

char HostName[32];

char WanIp[16];

char WanNetmask[16];

char WanGateway[16];

char WanMac[20];

int ChgMask;

} tWAN;

/\*\*\*\*\*\*\* Def tWanInof \*\*\*\*\*\*/

typedef struct {

int status; /\* Used by Wan Daemon \*/

int ChgMask;

char DhcpNtpServer[32]; /\* NTP server provide by DHCP server \*/

tWAN WAN;

tStatusInfo StatusInfo;

tWD\_DNS DNS;

tWD\_DDNS DDNS;

tBonjourInfo BonjourInfo;

} tWanInfo;

## WAN Daemon API

### WdSetDns

Update DNS setting. The setting will take effect after apply. The Dns Network Information is transfer to Wan Daemon in “pDNS” carry by ‘pWanInfo’. If the parameters pDNS ->DnsPrimary and pDNS->DnsSecondary are empty or not, WAN Daemon will accept this setting after apply.

#### <sysopsis>

#include “wan\_daemon\_api.h”  
 Int WdSetDns(tDNS \*pDNS);

#### <parameters>

**pDNS ->DnsPrimary –** Primary DNS server IP address **pDNS ->DnsSecondary–** Secondary DNS server IP address

#### <return value>

WD\_SUCCESS – Setup DNS successfully.  
WD\_FAIL – Failure.

#### <How to use this API>

WdSetDns(tWD\_DNS \*pDNS)

printf("\n--- Set DNS ---\n");

strncpy(pWanInfo->DNS.DnsPrimary, "168.95.192.1", 16);

strncpy(pWanInfo->DNS.DnsSecondary, "168.95.1.1", 16);

rc = WdSetDns(&pWanInfo->DNS);

if(rc == WD\_SUCCESS) {

printf("\n--- Set DNS ok! ---\n");

}

else {

printf("\n--- Set DNS error! ---\n");

}

### WdSetDdns

Update DDNS setting. The setting will take effect after apply. The Ddns Network Information is transfer to Wan Daemon in “pDDNS” carry by “pWanInfo”. If the parameters must not empty, WAN Daemon will accept this setting after apply.

#### <sysopsis>

#include “wan\_daemon\_api.h”  
 Int WdSetDdns(tDDNS \*pDDNS);

#### <parameters>

**pDDNS ->DdnsType** – 1)Disable 2)Enable

**pDDNS ->DdnsService** – DDNS service server configuration.

**pDDNS ->DdnsHostName** – DDNS host name configuration.

**pDDNS ->DdnsUser** – DDNS server logon name.

**pDDNS ->DdnsPwd** – DDNS server logon password.

#### <return value>

WD\_SUCCESS – Setup DDNS successfully.  
WD\_FAIL – Failure.

#### <How to use this API>

WdSetDdns(tWD\_DDNS \*pDDNS)

printf("\n--- Set DDNS ---\n");

pWanInfo->DDNS.DdnsType=2;

strncpy(pWanInfo->DDNS.DdnsService, "dyndns", 16);

strncpy(pWanInfo->DDNS.DdnsHostName, "hqao1.dyndns.info", 128);

strncpy(pWanInfo->DDNS.DdnsUser, "acti-hqao", 64);

strncpy(pWanInfo->DDNS.DdnsPwd, "123456", 64);

rc = WdSetDdns(&pWanInfo->DDNS);

if(rc == WD\_SUCCESS) {

printf("\n--- Set Hostname ok! ---\n");

}

else {

printf("\n--- Set Hostname error! ---\n");

}

### WdSetBonjour

Update Bonjour setting. The setting will take effect after apply. The Bonjour Information is transfer to Wan Daemon in “pBonjourInfo” carry by “pWanInfo”. The Bonjour must enable then setting will take effect after apply. If Bonjour executed the this setting will update the Bonjour after apply.

#### <sysopsis>

#include “wan\_daemon\_api.h”  
 int WdSetBonjour(tBonjourInfo\*pBonjourInfo);

#### <parameters>

**pBonjourInfo -> Enable**– 0)Disable 1)Enable

**pBonjourInfo -> FriendlyName**– The friendly name of this device. Length <64

**pBonjourInfo -> PortHttp**– HTTP port of this device.

**pBonjourInfo -> PortHttps**–HTTPS port of this device.

**pBonjourInfo -> LLIP**– Link-Local IP of this device. Give “” if there is not pre-defined link local IP address.

#### <return value>

WD\_SUCCESS – Setup Bonjour successfully.  
WD\_FAIL – Failure.

#### <How to use this API>

WdSetBonjour(tBonjourInfo \*pBonjourInfo)

printf("\n--- Set Bonjour ---\n");

strncpy(pWanInfo->BonjourInfo.FriendlyName, "ACTI\_WAN\_DAEMON\_TEST", 32);

strcpy(pWanInfo->BonjourInfo.LLIP, "169.254.64.213");

pWanInfo->BonjourInfo.Enable=1;

pWanInfo->BonjourInfo.PortHttp=80;

pWanInfo->BonjourInfo.PortHttps=8080;

rc = WdSetBonjour(&pWanInfo->BonjourInfo);

if(rc == WD\_SUCCESS) {

printf("\n--- Set Bonjour ok! ---\n");

}

else {

printf("\n--- Set Bonjour error! ---\n");

}

### WdSetHostName

Set host name to Linux system. The setting will take effect after apply. The Set Host Name function is transfer to Wan Daemon in “HostName” carry by “pWanInfo”.

The Maximum “HostName” Length = 68.

#### <sysopsis>

#include “wan\_daemon\_api.h”  
 Int WdSetHostName(char \*HostName);

#### <parameters>

**HostName** – Host Name.

#### <return value>

WD\_SUCCESS – Setup Host Name successfully.  
WD\_FAIL – Failure.

#### <How to use this API>

WdSetHostName(char \*HostName)

printf("\n--- Set Hostname ---\n");

strncpy(pWanInfo->WAN.HostName, "ACTI\_WAN\_DAEMON\_TEST", 32);

rc = WdSetHostName(pWanInfo->WAN.HostName);

if(rc == WD\_SUCCESS) {

printf("\n--- Set Hostname ok! ---\n");

}

else {

printf("\n--- Set Hostname error! ---\n");

}

### WdGetInfo

Get network information from WAN Daemon. The WAN Information is returned in “pWanInfo”. The " time\_out” is the variable value which to check the times of message Q receiving loop or hold the loop until message Q received. This function can select **block** or **non block**. **Suggestion:** Check the return value which you want.

#### <sysopsis>

#include “wan\_daemon\_api.h”  
 Int WdGetInfo(tWanInfo \*pWanInfo, unsigned int time\_out);

#### <parameters>

pWanInfo– Return the WAN informations.

time\_out – Loop Holding time, the time unit is usec.

if time\_out=0, **block**

#### <return value>

WD\_SUCCESS – Get the WAN information successfully.  
WD\_NODATA – Get the WAN information Time Out & NO Data.

WD\_NOT\_READY – Wan Daemon IP notify not yet.

WD\_FAIL – Failure.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| WdGetInfo | Return Value | | | |
| Block | WD\_FAIL | WD\_SUCCESS |  |  |
| Non Block | WD\_FAIL | WD\_SUCCESS | WD\_NODATA | WD\_NOT\_READY |

#### <How to use this API>

WdGetInfo(tWanInfo \*pWanInfo, unsigned int time\_out)

* 1. Block:

printf("\n--- Get WAN INFO ---\n");

rc = WdGetInfo(pWanInfo, 0); //block

if(rc == WD\_SUCCESS) {

printf("\n--- Get WAN Info successfully ---\n");

print\_wan\_info(pWanInfo);

}

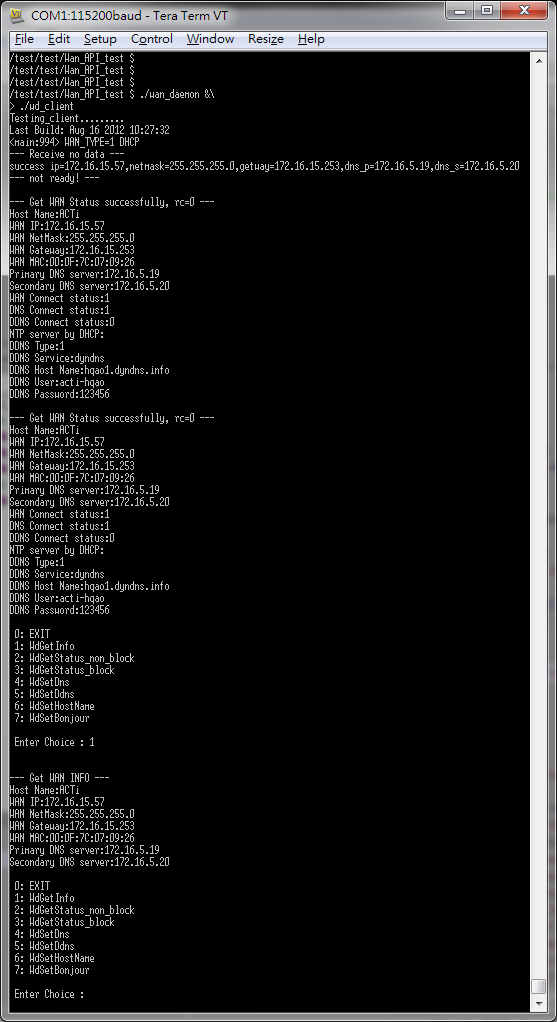
else {

printf("\n--- Receive error! ---\n");

}

The “WdGetInfo” in Block Mode:

* + - 1. Hold and Wait, Until Wan Daemon Respone.
      2. Received the “WanInfo” assured, Uless Msg Q API fail.
      3. Return Value, “WD\_FAIL” & “WD\_SUCCESS”.
  1. Result:



Success

* 1. Non Block:

printf("\n--- Get WAN INFO ---\n");

rc = WdGetInfo(pWanInfo, 2\*WAIT\_UNIT); //non block

if(rc == WD\_SUCCESS) {

printf("\n--- Get WAN Info successfully ---\n");

print\_wan\_info(pWanInfo);

}

else if(rc == WD\_NODATA) {

printf("\n--- Receive no data ---\n");

}

else if(rc == WD\_NOT\_READY) {

printf("--- not ready! ---\n");

}

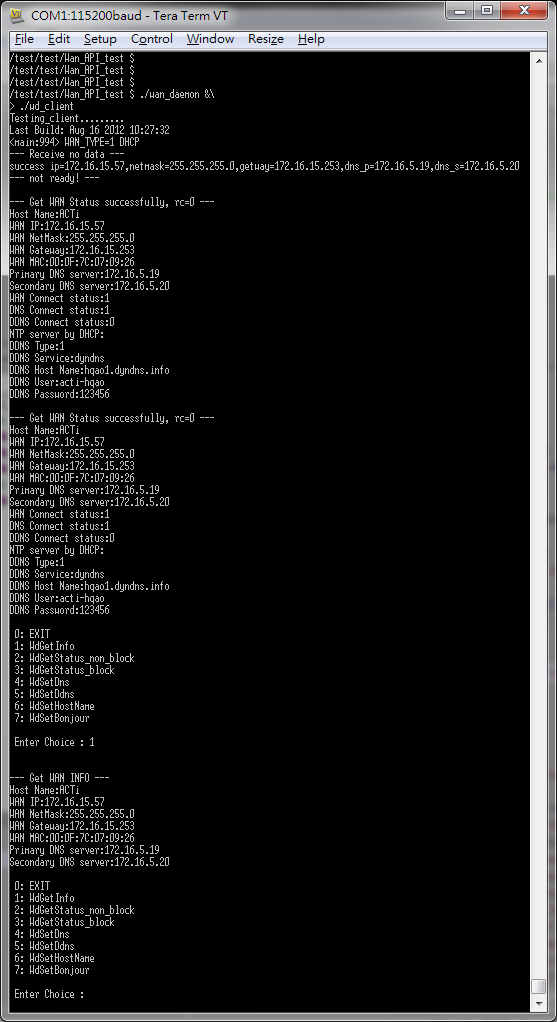
else {

printf("\n--- Receive error! ---\n");

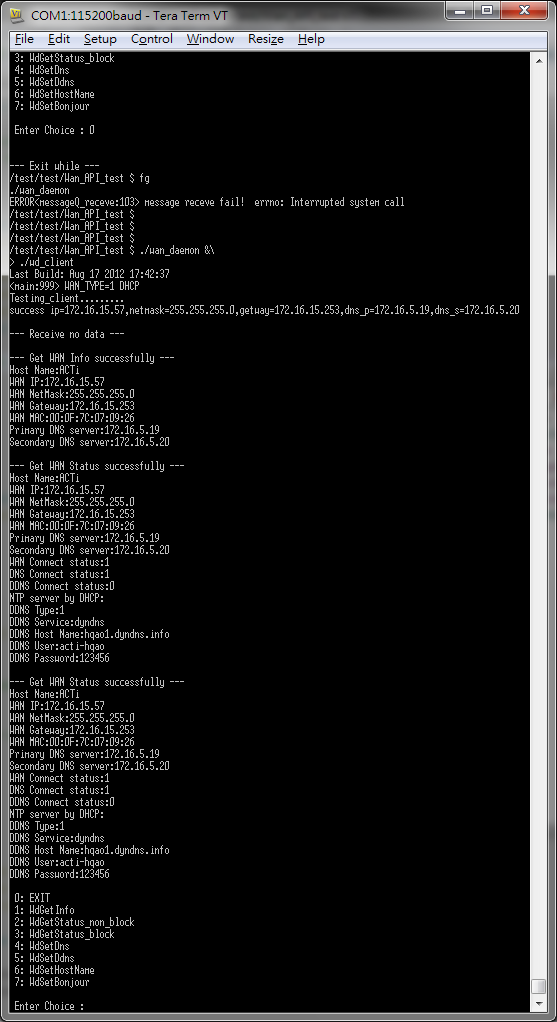
}

The “WdGetInfo” in Non Block Mode:

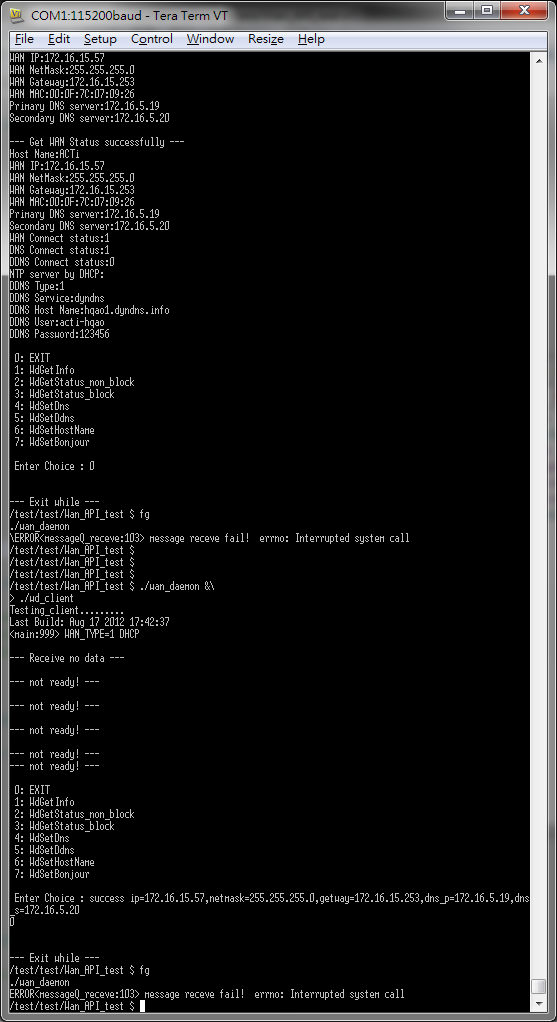
* + - 1. Just Wait to time out which you set.
      2. May or may not got response from Wan Daemon.
      3. Return Value, “WD\_FAIL” & “WD\_SUCCESS” & “WD\_NODATA” & “WD\_NOT\_READY”.
  1. Result:



Success



Msg Channel Receive Time out(Msg Channel not ready)



Wan Daemon IP notification not broadcast out.

### WdGetStatus

Get network Sataus from WAN Daemon. The network status is returned in ‘pWanInfo’. The Status Info includes “WanConnected”, “DnsConnected” and “DdnsConnected” for URL Command “WAN\_STATUS” to show the network connect status. WAN Daemon will probe the status when receive WdGetStatus Request. The " time\_out” is the variable value which to check the times of message Q receiving loop or hold the loop until message Q received. This function can select **block** or **non block**, Suggestion: Check the return value which you want.

#### <sysopsis>

#include “wan\_daemon\_api.h”  
 int WdGetStatus(tWanInfo \*pWanInfo, unsigned int time\_out);

#### <parameters>

pWanInfo– Return the WAN informations.

time\_out – Loop Holding time, Loop Holding time, the time unit is usec.

if time\_out=0, **block**

#### <return value>

WD\_SUCCESS – Get the WAN information successfully.  
WD\_NODATA – Get the WAN information Time Out & NO Data.

WD\_NOT\_READY – Wan Daemon IP notify not yet.

WD\_PRE\_DATA –Wan Daemon Still in check connection status. Get the last time WAN information.

WD\_FAIL – Failure.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| WdGetStatus | Return Value | | | | |
| Block | WD\_FAIL | WD\_SUCCESS |  |  |  |
| Non Block | WD\_FAIL | WD\_SUCCESS | WD\_NODATA | WD\_NOT\_READY | WD\_PRE\_DATA |

#### <How to use this API>

WdGetStatus(tWanInfo \*pWanInfo, unsigned int time\_out)

* 1. Block:

printf("\n--- Get WAN Status BLOCK ---\n");

rc = WdGetStatus(pWanInfo, 0);

if(rc == WD\_SUCCESS) {

printf("\n--- Get WAN Status successfully ---\n");

print\_wan\_info(pWanInfo);

print\_wan\_remaining\_info(pWanInfo);

}

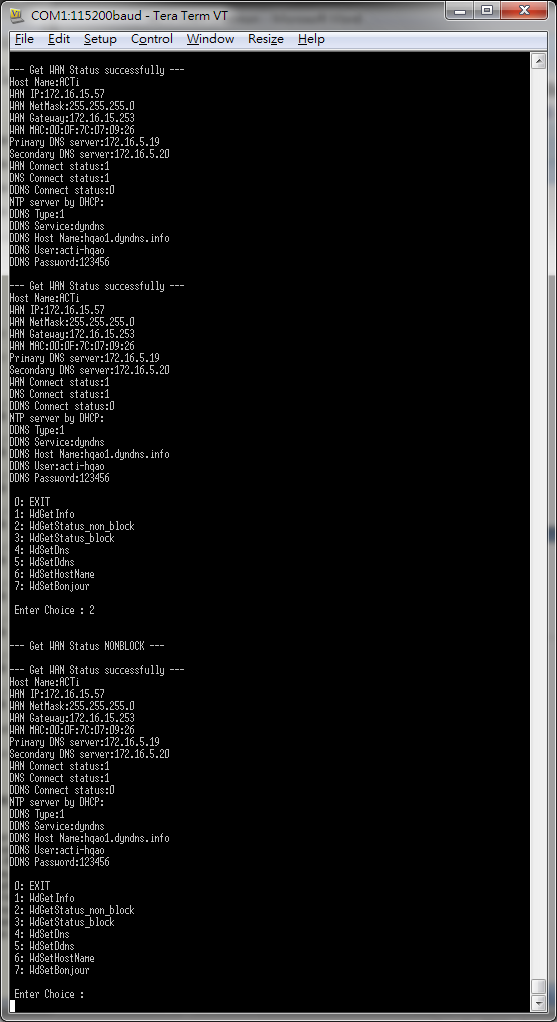
else {

printf("--- Receive error! ---\n");

}

The “WdGetStatus” in Block Mode:

* + - 1. Hold and Wait, Until Wan Daemon Respone.
      2. Received the complete “WanInfo” assured, Unless Msg Q API fail.
      3. Return Value, “WD\_FAIL” & “WD\_SUCCESS”.
  1. Result:



Success

* 1. Non Block:

printf("\n--- Get WAN Status NONBLOCK ---\n");

rc = WdGetStatus(pWanInfo, WAIT\_UNIT\*40);

if(rc == WD\_SUCCESS) {

printf("\n--- Get WAN Status successfully ---\n");

print\_wan\_info(pWanInfo);

print\_wan\_remaining\_info(pWanInfo);

}

else if(rc == WD\_PRE\_DATA) {

printf("\n--- Get WAN Pre Load Status ---\n");

print\_wan\_info(pWanInfo);

print\_wan\_remaining\_info(pWanInfo);

}

else if(rc == WD\_NODATA) {

printf("--- Receive no data ---\n");

}

else if(rc == WD\_NOT\_READY) {

printf("--- not ready! ---\n");

}

else {

printf("--- Receive error! ---\n");

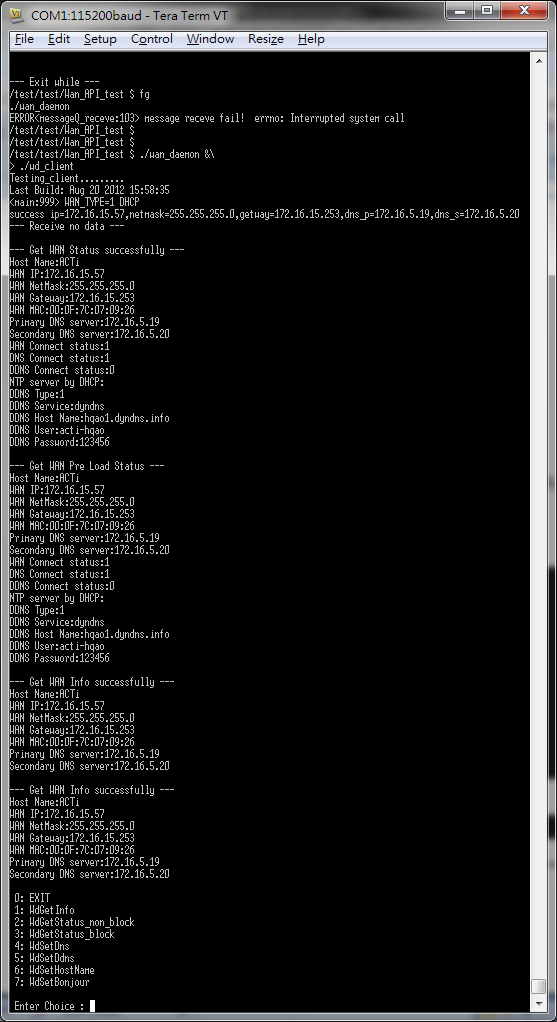
}

The “WdGetStatus” in Non Block Mode:

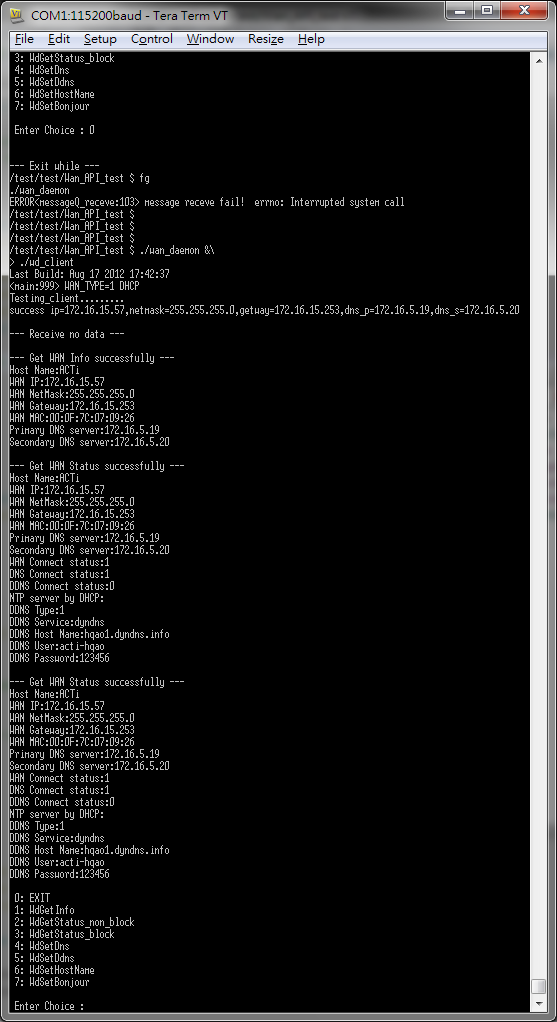
* + - 1. Just Wait to time out which you set.
      2. May or may not got response from Wan Daemon.
      3. Return Value, “WD\_FAIL” & “WD\_SUCCESS” & “WD\_NODATA” & “WD\_PRE\_DATA” & “WD\_NOT\_READY”.
      4. “WD\_PRE\_DATA” is mean, the Wan Daemon Still in check connection status. This information you got, was Wan Daemon update last time. So if you want to use this Pre load information, you must check after.
  1. Result:



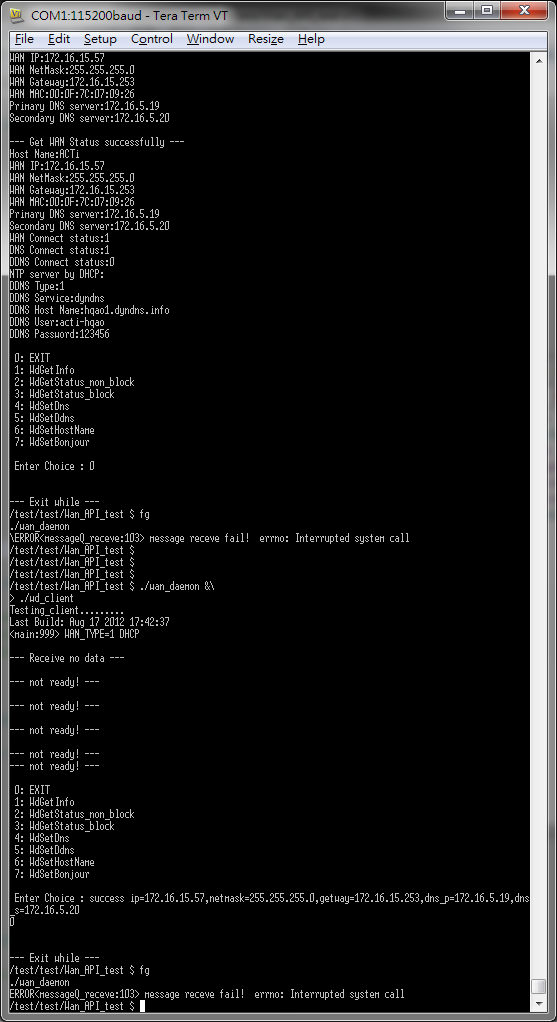
Success



Receive Pre Load Wan Info from WD(WD Ping Status not return yet)



Msg Channel Receive Time out(Msg Channel not ready)



Wan Daemon IP notification not broadcast out.

### WdWait4Notify

Check IP change notification from wan\_daemon. The WAN Information is returned in “pWanInfo”. It will return the useful information and some status of IP change, such as Bonjour, DHCP\_success, DHCP\_fail, pppoe\_success, pppoe\_fail and static IP setup.

#### <sysopsis>

#include “wan\_daemon\_api.h”  
 Int WdExitNotify(void);

#### <parameters>

pWanInfo– Return the WAN informations.

#### <return value>

WD\_SUCCESS – WdWait4Notify successfully.

WD\_FAIL – Failure.

WD\_NODATA – Receive data is zero.

### WdExitNotify(not using…)

~~Force leave WDGetNotify().~~

#### ~~<sysopsis>~~

~~#include “wan\_daemon\_api.h”  
 Int WdExitNotify(void);~~

#### ~~<return value>~~

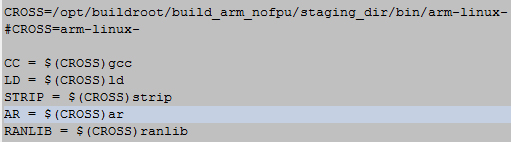
~~WD\_SUCCESS – WdExitNotify successfully.  
WD\_FAIL – Failure.~~

## How to Integrate Wan Daemon API

1. Build Wan Daemon API.
   1. Check the “**Makefile**” in “Wan Daemon API” folder.
   2. Find the “Rule.make” which is definition the cross compile rule in your platform.

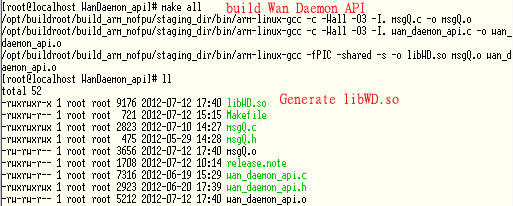
未命名 -1.jpg

Makefile in Wan Daemon API folder, EX: WanDaemon\_api/Makefile.



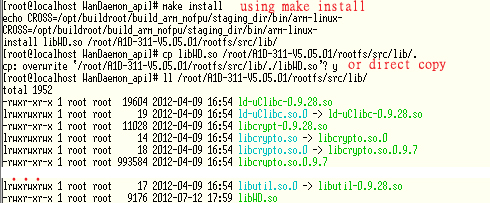
Rule.make in K platform “applications” folder, EX: applications/Rule.make.

* 1. Build Wan Daemon API to share library.
  2. Wan Daemon API share library “libWD.so” will generate after build.



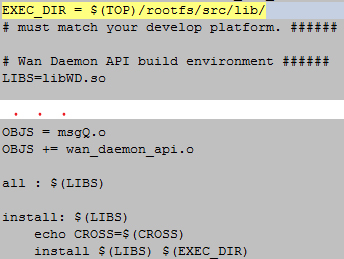
Build & generate libWD.so in K platform.

* 1. Put the Wan Daemon API share library “libWD.so” to platform target RFS.



Using make install or direct copy to platform target RFS.

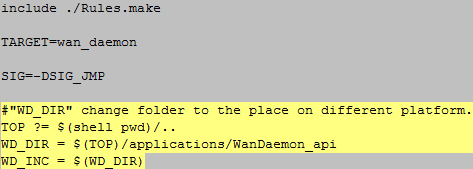
* 1. If using make install, please Note “EXEC\_DIR” link location. Must platform target RFS



please Note “EXEC\_DIR” link location

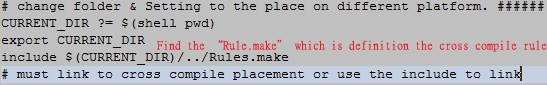
* 1. Direct copy to platform target RFS, please Note platform target RFS location.

1. Build Wan Daemon.
   1. Check the “**Makefile**” in “Wan Daemon” folder.
   2. Check the “WD\_DIR” is link to correct “Wan Daemon API” folder.

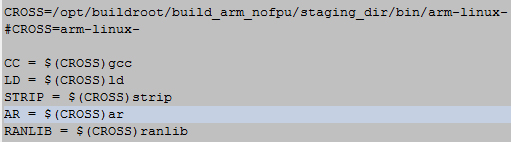


please check “WD\_DIR” link location

* 1. Check the “**Rule.make**” in “Wan Daemon” folder.
  2. Find the “Rule.make” which is definition the cross compile rule in your platform.



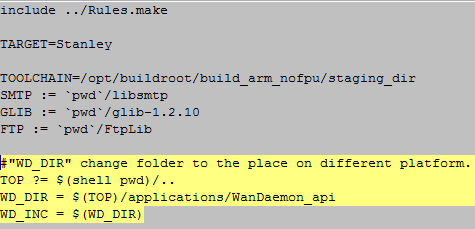
please check cross compile rule link location



Rule.make in K platform “applications” folder, EX: applications/Rule.make.

* 1. Build Wan Daemon.

1. Build Streamer.
   1. Check the “**Makefile**” in “Streamer” folder.
   2. Check the “WD\_DIR” is link to correct “Wan Daemon API” folder.

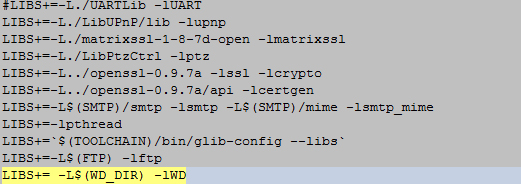


please check “WD\_DIR” link location

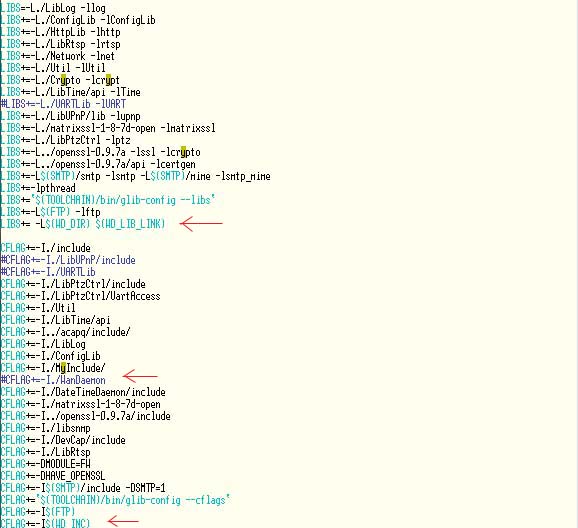
* 1. Disable ‘#OBJECTS+=WanDaemon/wan\_daemon\_api.o’
  2. Disable ’#OBJECTS+=WanDaemon/msgQ.o’



* 1. Add ‘LIBS+= -L$(WD\_DIR) -lWD’



* 1. Disable ‘#CFLAG+=-I./WanDaemon’
  2. Add ‘CFLAG+=-I$(WD\_INC)’



* 1. Remove the Residual files and folders of Wan Daemon in the ‘Streamer’ folder, And build streamer.
* Example for How to use the Wan Daemon

Please Check out from SVN server in “/SSPO/MyLibs/tags/WanDaemon\_api/version.x.x.x/Wan\_API\_test”.

The each platform owner can using those source code. After compiler it for platform test Wan Daemon and get Wan Info from Wan Daemon.