VIRTUAL ACCESS

Service Provider Solutions

www.virtualaccess.com

WatchdogADSL: Watchdog Monitoring of an ADSL Interface

Issue 1.1

Date 26 February 2014

1 Introduction

The watchdogADSL script is designed to alleviate issues which can result in the ADSL interface going down and not recovering - mainly due to line issues or interoperability issues between vendor ADSL chipsets.

The script periodically monitors the state of an ADSL interface and takes the following recovery steps to bring the interface back up again when it's found to be down.

- Forces a retrain of the interface
- Resets the interface
- Optionally reboots the router

The script is commonly used in a scenario shown in the figure below. The WAN connection is via a bridged ADSL link using a Ethernet encapsulation. If using PPP encapsulation then the watchdogPPP script will be more appropriate since it will also catch PPP network issues.

Note: this script can also be run on multiple interfaces at the same time. However, if you are using it on a multi WAN scenario (ADSL with 3G backup) it is normally only desirable to reload the router as a last resort on the lowest priority interface.

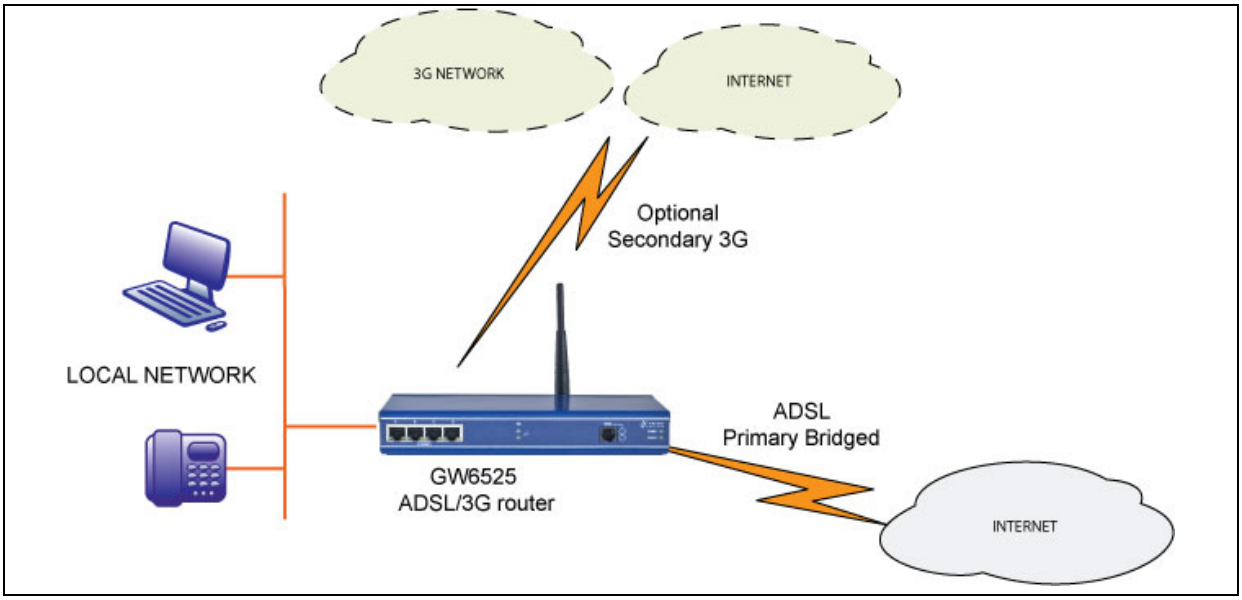

Figure 1: Network architecture

2 Configuring the watchdogADSL script

2.1 Script overview

The script is designed to be run on boot. On boot:

- The script first waits for ADSL to establish on the monitored link.
- The script can be configured for two types of operation:
    1. The script will wait indefinitely for the ADSL to come up before proceeding to interface state checking.
    2. The script will wait for a configurable period of time before proceeding to interface state checking
- Once the ADSL comes up (or the configurable initial timeout period elapses) the script will start checking the state of the ADSL interface at configurable intervals.
- If the script detects that the ADSL interface is down for a configurable number of checks a recovery procedure is started (see Section 2.2).
- If the ADSL interface comes back up on the monitored link then the recovery procedure is terminated.

2.2 Recovery procedure

ADSL interface:

The script first forces a manual retrain of the ADSL connection. If this does not fix the issue, then the ADSL chip is reset. Finally, if this fails to solve the problem, the script will optionally reboot the router.

2.3 Script parameters

The script name is watchdogADSL and it takes in five optional parameters.

watchdogADSL	[adsl_port] [check_timer] [max_fails] [reload] [init_wait]

These parameters are described in the example and table below.

watchdogADSL adsl-0, 300, 2, 1, 3600

Parameter	Description	Default
adsl-0	The ADSL interface to monitor	adsl-0
300	The frequency of the ADSL state check in seconds	300
2	The number of ADSL interface checks before moving to recovery procedure. This number of checks is also used by the recovery procedure before moving to the next stage of recovery.	2
1	Whether to reload the router as a last resort. [0 for no reload, 1 to reload]	1
3600	The time to wait in seconds for the ADSL interface to come up on boot before initiating ADSL state checks. If set to 0 the script will not proceed to ADSL state checks until the ADSL interface has come up.	3600

Table 1: Six optional parameters and their descriptions

2.4 Configuring the script

The script is embedded in firmware images greater than 9.08.27 and 10.00.21. To use the script on older firmware versions first paste the script from Section5 'WatchdogADSL script' into the script editor and then use the scheduler to run the script at boot up.

From the start page, click **Advanced** to open the Expert View menu.

2.4.1 Pasting the script into the script editor

If using 9.09.xx firmware, in the Expert View menu, click **system > scripts->script editor**. The Script Editor page appears.

If using 10.00.xx firmware in the Expert View menu, click **system > management > scripts > script editor**. The Script Editor page appears.

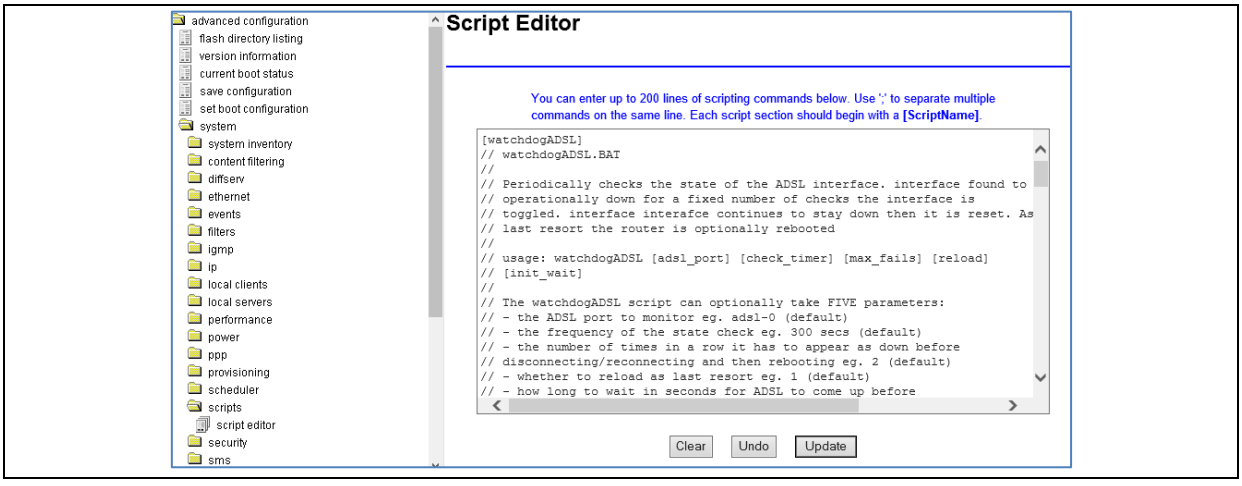

Figure 2: The script editor page in version 9.09.xx

Paste in the script from Section5 'WatchdogADSL script' from this document. The first line of the script should begin with the script name in square brackets, [watchdogADSL]. This name will be used to call the script using the scheduler.

You can omit any line beginning with // (denotes a comment tag) if the number of script lines needs to be reduced. Also multiple script lines can be entered onto the same script

editor line separated by ';' (semi colon). When you have completed the script, click **Update**.

2.4.2 Scheduling the script to run on boot

If using 9.09.xx firmware, in the Expert View menu, click **system > scheduler > scheduler tasks**.

Click **Add** or **modify/delete** to access the scheduler Task Entry Page.

If using 10.00.xx firmware, in the Expert View menu, click **system > management > scheduler > scheduler tasks**. The Scheduler Task Entry page appears.

Click **Add** or **modify/delete** to access the scheduler Task Entry Page.

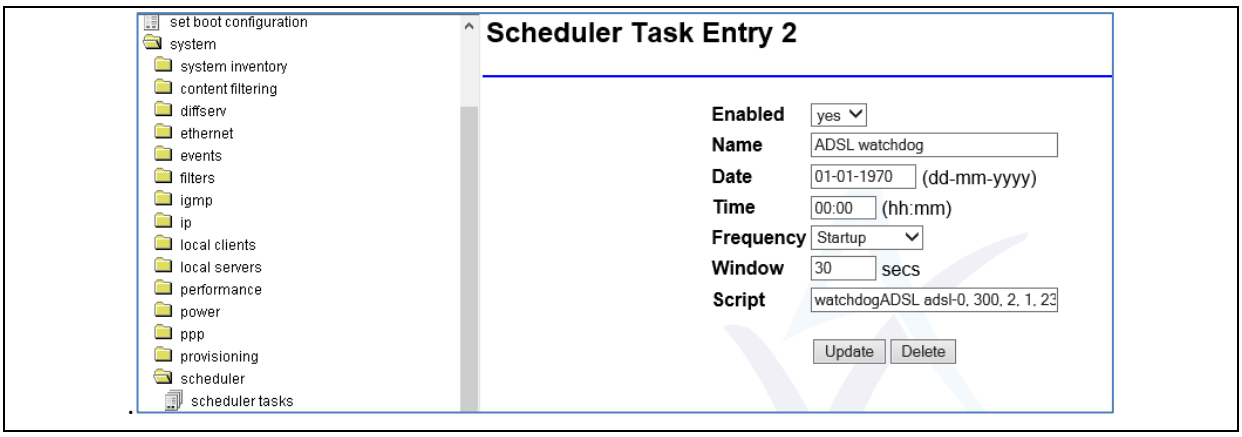

Figure 3: The scheduler task entry page in version 9.09.xx

Field	Description
Enabled	Enables or disables a particular schedule. Set to **Yes**.
Name	The name associated with the schedule. Enter a descriptive name
Date	The date the script initiates. This field is ignored when frequency is set to start up. Leave at **default**.
Time	The time the script initiates. This field is ignored when frequency is set to start up. Leave at **default**.
Frequency	Sets the frequency the script executes. Set to **startup**.
Window	This parameter sets how long the system will wait if it is busy before executing the script. For example if the script is set to execute at 10:00 and the window is set to 30 seconds, the system will try executing the script within this window only. Set to **30**.
Script	The name of the script to be executed. Enter the script name, followed by the relevant parameters as shown in the above image. Separate the parameters by commas. Example: **watchdogADSL adsl-0, 300, 2, 1, 3600**.

Table 2: The scheduler task fields and their descriptions

3 Debugging commands

Useful debug commands via command line are described in the table below.

Diagnostic Command	Description
Show tasks	Displays all running tasks.
Show task <tasknum>	Displays running task. Also indicates position task is currently at.
Show task vars <tasknum>	Displays variables and variable values associated with task.
Show ip route	Displays routing table.
Show events	Displays event log.
Show change log	Displays recent configuration changes.
Dir scripts	Displays all scripts embedded in the firmware.
Show config script ALL	Displays all scripts in the script editor.
Show config script <scriptname>	Displays the <scriptname> script as configured in script editor. Includes line numbers.
Show config script –n <scriptname>	Displays the <scriptname> script as configured in the script editor. Does not include line numbers.
sh state adsl <adsl_port>	Displays state of an ADSL port. (Reports: Admin status, operational status, error status, error message)
Sh stats adsl	Shows ADSL line statistics

Table 3: Debug command lines and their descriptions

Useful trace commands via the command line are described in the table below.

Trace command	Description
++All 6	Traces all INFO events
++ip	Traces IP traffic
++adsl	Traces ADSL events
++script	Traces script events
--script	Stops script event tracing
--	Stops all event tracing
Trace on <script_name>	Traces each line in a script as it executes
Trace off <script_name>	Turns off tracing for script

Table 4: Trace command lines and their descriptions

4 Script events

Severity	Class	Subclass	Text
INFO	49	40	watchdogADSL running parameters <adsl_port>, <check_timer>, <max_fails>, <force_reload>,<init_wait>
INFO	49	40	watchdogADSL <adsl_port> down forcing ADSL retrain
INFO	49	40	watchdogADSL <adsl_port> down resetting ADSL interface
INFO	49	40	watchdogADSL <adsl_port> still down after ADSL reset - rebooting
INFO	49	40	watchdogADSL <adsl_port> still down after ADSL reset - restarting checks

Table 5: Script events

5 WatchdogADSL script

```
[watchdogADSL]
// watchdogADSL.BAT
//
// Periodically checks the state of the ADSL interface. If found to be
// operationally down for a fixed number of checks the interface is
// toggled. If interface continues to stay down then it is reset. As a
// last resort the router is optionally rebooted
//
// Usage: watchdogADSL [adsl_port] [check_timer] [max_fails] [reload]
//                     [init_wait]
//
// The watchdogADSL script can optionally take FIVE parameters:
//    - the ADSL port to monitor eg. adsl-0 (default)
//    - the frequency of the state check eg. 300 secs (default)
//    - the number of times in a row it has to appear as down before
//      disconnecting/reconnecting and then rebooting eg. 2 (default)
//    - whether to reload as last resort eg. 1 (default)
//    - how long to wait in seconds for ADSL to come up before
//      initiating checks eg. 3600 (default). Set to 0 to wait
//      indefinitely for ADSL to come up before initiating checks.
//
// EXAMPLES
// --------
//    watchdogADSL
//    (Checks adsl-0 state every 300 secs. If found to be down 2 times
//    in a row, manually disconnects and reconnects adsl-0. If adsl-0
//    does not come up within another 2 checks then adsl-0 is reset.
//    If down after a further 2 checks then the router is rebooted.
//    If ADSL fails to come up after 1 hour on boot then checks are
//    initiated)
//
//    watchdogADSL adsl-1, 30, 3, 1,  600
//    (Checks adsl-1 state every 30 secs. If found to be down 3 times
//    in a row adsl-1 interface is toggled. If adsl-1 does not come up
```

```
//    within another 3 the interface is reset. If this does not fix in a
//    further 3 checks then the router is rebooted. Checks do not begin
//    until ADSL interface has come up)

!echo off

$adsl_port = $1
$adsl_timer = $2
$max_fails = $3
$force_reload = $4
$init_adsl_wait = $5

//set defaults
!if adsl_port = ''
  $adsl_port = adsl-0
!endif
!if adsl_timer = ''
  $adsl_timer = 300
!endif
!if max_fails = ''
  $max_fails = 2
!endif
!if force_reload = ''
  $force_reload = 1
!endif
!if init_adsl_wait = ''
  $init_adsl_wait = 3600
!endif

!log watchdogADSL running parameters $adsl_port, $adsl_timer, $max_fails, $force_reload, $init_adsl_wait

$z = `sh state adsl $adsl_port`
!if $z[2] = Up
  !goto ADSL_RESTART_CLEAN
!endif
```

```
//initial wait until ADSL up before starting main script
!if $init_adsl_wait <> 0
  !waitevent 66.01:$adsl_port&Kbps $init_adsl_wait
  !endevent
!else
  !waitevent 66.01:$adsl_port&Kbps
  !endevent
!endif

!label ADSL_RESTART_CLEAN
$phy_toggle = 0
$phy_reset = 0
$fail_count = 0

!label ADSL_CK
!while $fail_count < max_fails
  !pause $adsl_timer
  $z = `sh state adsl $adsl_port`
  !if $z[2] = Up
    //reset counters
    $phy_toggle = 0
    $phy_reset = 0
    $fail_count = 0
  !else
    !inc fail_count
  !endif
!endwhile

!label ADSL_DOWN
!if $phy_toggle <> 0

  //manual disconnect did not work, try to reset chip if fails then reload
  !if $phy_reset <> 0

    !if $force_reload <> 0
      !log watchdogADSL $adsl_port still down after ADSL reset - rebooting
```

```
    !goto ADSL_RESET
  !else
    !log watchdogADSL $adsl_port still down after ADSL reset - restarting checks
    !goto ADSL_RESTART_CLEAN
  !endif

 !else
  !log watchdogADSL $adsl_port down resetting ADSL interface
  $z = `reset adsl $adsl_port`
  //give DSL chip some time to sort itself out
  !pause 60
  $fail_count = 0
  $phy_reset = 1
  !goto ADSL_CK
 !endif
!else

 //disconnect and reconnect physical
 !log watchdogADSL $adsl_port down forcing ADSL retrain
 $z = `phy_toggle $adsl_port`
 //give the physical some time to come back
 !pause 30
 $fail_count = 0
 $phy_toggle = 1
 !goto ADSL_CK
!endif

!label ADSL_RESET
reload
```