
TZOLogin

An IP address monitor for dynamic connections

Author: Jon Barnett

Document version: 1.0

Table of contents

1	Network management for dynamic addresses	1-1
1.1	Dynamic DNS	1-1
1.2	Firewalls and routing.....	1-1
1.3	Existing solutions	1-1
1.4	Requirements.....	1-1
1.4.1	Address change detection.....	1-1
1.4.2	Registration.....	1-1
1.4.3	Status information for external programs	1-2
1.4.4	Constraints	1-2
1.5	Design.....	1-2
1.6	Operation	1-2
1.7	Installing on Linux	1-3
1.7.1	Prerequisites.....	1-3
1.7.2	Installation	1-3
1.7.3	Security	1-3
1.7.4	Detection and registration configuration	1-3
1.7.5	Environment configuration	1-5
1.7.6	Cron job installation	1-7
1.7.7	Standalone operation	1-8

1 Network management for dynamic addresses

It is a fairly cheap and simple task to obtain a broadband connection that gives you a dynamic IP address. However, it means that when you lose your connection, whether it is because of a network outage or whether you have disconnected from the broadband network, the next time you connect, you will probably receive a new IP address.

This can cause problems for a variety of reasons. If you have an internal network connected to the Internet via a Linux machine, you may have your firewall rules dependent on the IP address of your broadband connection. You may also have a dynamic DNS entry that you need to update. So there are plenty of reasons to want to monitor the status of your broadband connection.

With cable modem there are fewer problems with dynamic IP addresses as the allocation is performed via DHCP server. So even if you lose your connection for a short period, as long as your DHCP lease has not expired, when you connect back up you should retain your IP address. With ADSL, you are connecting over Ethernet via PPP. So like a dial-up line, you will most likely obtain a new IP address each time you reconnect.

1.1 Dynamic DNS

When you want to host your own website but don't want to pay for a static IP broadband service then you will need to use a dynamic DNS service. Every time your IP changes, you need to notify the dynamic DNS provider that the IP address for your domain name has changed.

1.2 Firewalls and routing

With more bandwidth from your broadband connection, people often consider adding more systems to utilise the connection. Although you can pay for more Internet IP address allocations for the same physical link, a more cost-effective approach is to build your own firewall and create your own internal network behind the firewall. The problem occurs when your Internet IP connection changes as your firewall system needs to change routing tables or perform other related tasks.

1.3 Existing solutions

Dynamic DNS providers such as TZO supply various scripts and programs to allow for automated registration of the changes in your Internet address. However, in changing from a cable-based service to an ADSL-based service, I found the Linux script insufficient for both registering the more frequent changes as well as providing signalling to start other tasks such as re-initialising the Linux IP tables. With this in mind, I set about developing a Java-based solution.

1.4 Requirements

1.4.1 Address change detection

The system will be able to determine the IP address of a network interface. The user should be able to define which interface is to be detected. Where a gateway address is to be used, such as where the Internet broadband connection is performed by an ADSL router with NAT capabilities, the gateway address should be detected. This configuration is not expected to be used very often as it is primarily designed for operating system-based routers and firewalls. The user should also be able to define a hard-coded IP address.

1.4.2 Registration

The system is tuned to operate with the TZO DNS registration service. However, as most dynamic DNS providers perform registration using the HTTP GET method, the system will provide for hard-coded registration URLs. Additionally, it would support the TZO de-registration cycle.

1.4.3 Status information for external programs

The system should provide information that an external script can use to determine on-going actions based on the detection and registration of IP address changes. These should include:

- No IP address change since the last poll
- Failure to register the new IP address
- Failure to determine the IP address of an interface
- Failure to determine the IP address of a gateway
- Successful detection and registration of a change

1.4.4 Constraints

- The system should be as simple as possible
- It should use features of the native operating system
- It should cope with connectivity problems
- It should log relevant actions
- It should be persistent with registration actions to minimise human interaction

1.5 Design

With this in mind, I made some decisions on the architecture.

- The Apache Commons HttpClient library would be used as it provided HTTP connectivity that was robust enough to deal with connection failures, unlike the native Sun Java routines that would lock the program
- The native scheduler for the operating system would be used to run the program, as the target operating systems all have this capability
- The configuration would be read from a file to allow some flexibility for options
- The glue code to allow the operating system to interact with the program would be a shell script
- The Java 2 1.4 logging utilities would be used to manage logging
- A storage file would be used to track the previous IP address

1.6 Operation

A detection cycle consists of the following:

- Check command line parameters
 - Exit if the commands are not understood
- Parse the configuration file
 - Exit with parser messages if the configuration file is not understood
- Read the previously stored IP address
 - If it doesn't exist, define the address as blank
- Check the current IP address
 - If the address is hard-coded in the configuration file, use that
 - If it is a gateway command, retrieve the reported gateway address
 - Use the hard-coded gateway reporting URL if requested
 - Otherwise use the TZO style gateway URL
 - If it is an interface command, retrieve the reported address on the interface
 - If the command is de-register, use the 0.0.0.0 address
 - Exit if no address was able to be retrieved
- Compare the current address with the stored address
 - Exit if the current address matches the stored address
- Register the new address with the dynamic DNS provider
 - Use the hard-coded URL if requested, replacing the %ADDRESS% with the retrieved address
 - Otherwise use the TZO style request
 - Exit if registration did not work
- Store the new IP address

1.7 Installing on Linux

1.7.1 Prerequisites

- Java JRE 2 1.4 or greater
- cron
- Sufficient permission for the cron user to detect network hardware, communicate to Internet systems, and execute any operating system commands required post-detection

1.7.2 Installation

Unpack the distribution to the desired location. This would normally be the `/usr/local` directory but it is not necessary for the running the software. The main distribution directory is called `tzologin`. Located under this directory are the `bin`, `conf` and `docs` sub-directories.

1.7.3 Security

The system doesn't present any major security issues as it operates as a network client. However, to prevent the normal user from executing the script and programs and to prevent the user from viewing the configuration information, it is recommended that permissions are modified on these directories and files accordingly. The log files should similarly be protected.

1.7.4 Detection and registration configuration

The configuration for detecting IP addresses and registering the IP address with the dynamic DNS provider is controlled by the `tzologin.conf` file located in the `conf` directory of the distribution.

None of the configuration parameters should be deleted from the file as regardless of the mode in which you run the program, it expects to be able to read all the parameters.

General users will find the following parameters important:

- Interface
 - This is the name of the interface that the service will scan for an IP address if you are using the interface scan and must match the name exactly, including case
- IPAddress
 - Normally this is set to '?' which forces the program to search the specified interface, otherwise you can force a hard-coded IP address by specifying an IP address for this parameter

The configuration for TZO users replicates the information that TZO expects so not too much detail will be covered in this manual. The important TZO specific information to configure in this area is:

- TZOName
 - This is the domain name that you want to register the IP address against
- Email
 - This is the e-mail address that you provided with TZO when you purchased their service
- TZOKey
 - This is the registration key you have received from TZO to register your domain name
- CType
 - This is the type of connection service and the definitions are provided

Non-TZO users will want to use their own hard-coded registration URLs and these parameters will help with that:

- URL
 - o This is the registration URL and the %ADDRESS% place-marker will be replaced the actual address detected or otherwise hard-coded in IPAddress.
- GatewayURL
 - o This is the URL for the gateway IP address detection and is usually part of the dynamic DNS provider's service. The service detects the IP address from which the request comes and sends this back in the reply. Currently this only works for the TZO gateway reporting service so leave this as set.

```
### ===== ###
##                                     ##
## TZO Login config                    ##
##                                     ##
### ===== ###

### $Id: tzologin.conf,v 1.03 2003/05/18 14:02:03 jbarnett Exp $ ###

#####
# This section is to be configured by the user #
#####

# Interface to scan for the TZO registered IP address
Interface = ppp0

# The machine name to be registered with TZO's DDNS
TZOName = my-domain-name

# The registered email for the TZO account holder
Email = my-email@my-email-domain-name

# The TZO registration key of the TZO account holder
TZOKey = MYTZOKEY

# The type of connection service for the machine
#       1   Dial-up Modem
#       2   Cable Modem
#       3   LAN (Local Area Network)
#       4   ADSL, HDSL, xDSL
#       5   America Online Dial-up
#       6   Other Connection Type
#       7   Static IP Address
CType = 4

# Or use a hardcoded information transfer encoding
URL = http://provider:80/connect.html?D=my-domain-name&K=F12DFE&A=%ADDRESS%
GatewayURL = http://echo.tzo.com:80/ip.shtml

#####
# This section contains standard definitions for TZO #
#####

# You will normally not need to change these

RemoteHost = rh.tzo.com
RemotePort = 80
URI = /webclient/tzoperl.html
GatewayHost = echo.tzo.com
GatewayPort = 80
GatewayURI = /ip.shtml
B1 = Sign+On+to+TZO+DDNS+Servers
IPAddress = ?
```

1.7.5 Environment configuration

The configuration for the shell script wrapper is the only other file necessary to be modified to run the program. The script is located in the *bin* directory of the distribution.

The main areas for change will be the definition for the location of the Java installation, and the location for log files and address storage.

- LOG
 - This defines the location and naming format of log files. The current format will be sufficient as long as the program is run as the *root* user as the specified log directory requires write privileges of *root*. The program uses the Java 2 1.4 logging utilities and will create up to 5 files each containing up to 50 Kb of information.
- ADDRESS
 - This defines the location of the file used to store the IP address detected on the last poll. The program must have read write access to this file.

When running this script as a cron job, you must specifically set the `JAVA_HOME` in the script as the standard security in cron will not allow inheritance of normal environment variables. Note that this is defined for the topmost directory of a JDK 2 1.4.1_01 distribution in this example.

The other parameters that may be used to control the program relate to the type of operation to perform and whether to use of the TZO HTTP communication format or a hard-coded HTTP format. Modify these to suit your needs.

The first parameter in detail is:

- R
 - Register the IP address if a change is detected on the designated interface and save the detected IP address
- U
 - Unregister the IP address by sending an IP address of 0.0.0.0
- G
 - Register the gateway IP address reported by the external reporting service if a change is detected and save the detected IP address
- I
 - Perform the rest of the functions for an interface-based IP address change detection but don't try to register the IP address
- GI
 - Perform the rest of the functions for a gateway-based IP address change detection but don't try to register the IP address
- T
 - Perform the rest of the functions for an interface-based IP address change detection but don't try to register the IP address and don't save the changed IP address
- GT
 - Perform the rest of the functions for a gateway-based IP address change detection but don't try to register the IP address and don't save the changed IP address

The second parameter in detail is:

- N
 - This uses the normal mode of communication, using the TZO parameter definitions for both the gateway detection and the registration of the changed IP address
- H
 - This uses the hard-coded formats defined by GatewayURL and URL in the *tzologin.conf* configuration file

Finally, change the actions taken at the end of the script after the program has executed. In this example, the IP tables of Linux are reconfigured for the change in the IP address. The actual configuration is such that communication through the firewall is blocked when the IP address changes, so the program cannot register the new IP address with TZO. However, the program indicates that the IP address has changed but no communication could be established by returning an exit status of 9. The

program does not store the new address, forcing the next poll to register the address with TZO again. The script processes the return value RETVAL and reconfigures the IP tables. This allows outbound communication to be re-enabled for systems residing behind the firewall.

Based on your requirements, adjust this part of the script accordingly.

```
#!/bin/sh
### ===== ###
##                                     ##
## TZO Login script                   ##
##                                     ##
### ===== ###

### $Id: tzologin.sh,v 1.02 2003/05/18 20:23:19 jbarrett Exp $ ###

# Current directory
DIRNAME=`dirname $0`

# Application fully qualified name
TZO_APP_FQN="com.amity.apps.TZOLogin"

# Command for controlling the system
# The commands are:
#   R - Register using the specified interface's address
#   U - Unregister
#   G - Register using the gateway address
#   I - Ignore registration with TZO
#   GI - Use gateway address but ignore registration with TZO
#   T - Ignore registration with TZO and don't write current address
#   GT - Use gateway address but don't send information or write
TZO_COMMANDS="R"

# URL type for specifying communication
# The commands are:
#   N - Use the TZO information encoding
#   H - Hardcoded information encoding
TZO_COMMANDS="$TZO_COMMANDS N"

# Location of TZO log and address files
LOG="/var/log/tzologin.%g.log"
ADDRESS="/var/log/tzoaddress"

# Location of your Java installation
JAVA_HOME="/usr/local/j2sdk1.4.1_01"

# Local Java options
TZO_JAVA_OPTS=

# Setup TZOLOGIN_HOME
if [ "x$TZOLOGIN_HOME" = "x" ]; then
    # get the full path (without any relative bits)
    TZOLOGIN_HOME=`cd $DIRNAME/..; pwd`
fi

# Setup TZOLOGIN_CONF
if [ "x$TZOLOGIN_CONF" = "x" ]; then
    # get the full path (without any relative bits)
    TZOLOGIN_CONF="$TZOLOGIN_HOME/conf/tzologin.conf"
fi

# Setup the JVM
if [ "x$JAVA" = "x" ]; then
    if [ "x$JAVA_HOME" != "x" ]; then
        JAVA="$JAVA_HOME/bin/java"
    else
        JAVA="java"
    fi
fi

# Setup the path to Apache commons-logging
LOGGINGJAR="$TZOLOGIN_HOME/bin/commons-logging.jar"
if [ ! -f $LOGGINGJAR ]; then
    die "Missing required file: $LOGGINGJAR"
fi
```

```

# Setup the path to Apache HTTP client
APACHEJAR="$TZOLOGIN_HOME/bin/commons-httpclient.jar"
if [ ! -f $APACHEJAR ]; then
    die "Missing required file: $APACHEJAR"
fi

# Setup the path to tzologin
TZOJAR="$TZOLOGIN_HOME/bin/tzologin.jar"
if [ ! -f $TZOJAR ]; then
    die "Missing required file: $TZOJAR"
fi

# Setup TZO_JAVA_OPTS
if [ "x$TZO_JAVA_OPTS" = "x" ]; then
    TZO_JAVA_OPTS="-cp $TZOJAR:$APACHEJAR:$LOGGINGJAR"
else
    TZO_JAVA_OPTS="$TZO_JAVA_OPTS -cp $TZOJAR:$APACHEJAR:$LOGGINGJAR"
fi

# Setup TZO_JAVA_OPTS
if [ "x$@" = "x" ]; then
    TZO_COMMANDS="$TZO_COMMANDS"
else
    TZO_COMMANDS="$@"
fi

# Setup JAVA_OPTS
if [ "x$JAVA_OPTS" = "x" ]; then
    # get the full path (without any relative bits)
    TZO_JAVA_OPTS="$TZO_JAVA_OPTS"
else
    TZO_JAVA_OPTS="$JAVA_OPTS $TZO_JAVA_OPTS"
fi

# Create command line
TZO_EXE="$TZO_JAVA_OPTS $TZO_APP_FQN"

# Execute the TZO address polling
# This will return exit values based on the processing state.
# The values are:
# 0      No change in the IP address since the last poll
# 1      A change in the IP address since the last poll
# 2      The command of [R],[U],[G],[I],[GI],[T] or [GT] was not specified
# 3      The url type of [N] or [H] was not specified
# 4      There was a problem reading the configuration file
# 5      There was a problem querying for a gateway
# 6      There was no IP address for the specified interface
# 7      There was an internal memory problem
# 8      There was a problem forming the TZO connection information
# 9      There was a problem contacting the TZO site
($JAVA $TZO_EXE "$TZOLOGIN_CONF" "$LOG" "$ADDRESS" $TZO_COMMANDS 2>&1) \
    > /dev/null
RETVAL=$?

# Define your custom actions based on the value of $RETVAL returned
# This example executes an iptable reconfigure if the IP address changed
# or if contact could not be established (the forwarding configuration
# is blocking transmission because of the address change)
if [ $RETVAL -eq 1 -o $RETVAL -eq 5 -o $RETVAL -eq 9 ]; then
    (/etc/init.d/iptables restart 2>&1) > /dev/null
    RETVAL=$?
fi

exit $RETVAL

```

1.7.6 Cron job installation

Running the script as a cron job, simply install the following line in the cron file.

```
*/* * * * * /usr/local/tzologin/bin/tzologin.sh
```

This runs the script every 2 minutes and expects the TZOLogin installation to be located at /usr/local/tzologin. Adjust the setting for your requirements.

1.7.7 Standalone operation

There are times that you will want to run the script in test mode or manually. It is designed to allow for this type of operation. You can override the parameter controls from the command line.

For example, to run it in interface-scan test mode using the TZO communication format, at the command line:

```
/usr/local/tzologin/bin/tzologin.sh T N
```

You may also want to operate a de-registration using the hard-coded communication format by running the following at the command-line:

```
/usr/local/tzologin/bin/tzologin.sh U H
```