

# Status setzen durch WLAN-Verbindung

Dieses Bash-Script durchsucht das angeschlossene Netzwerk nach dem unter *HOSTNAME* angegebenen Gerät. Bei Verbinden/Trennen wird automatisch der passende Status gesetzt. Dafür müssen die Variablen *STATUSURL\_CONNECTED/STATUSURL\_DISCONNECTED* an die im Statusgeber zu findende URL angepasst werden.

## Abhängigkeiten installieren

Das Paket *arp-scan* wird zum Scannen des Netzwerks genutzt und muss falls nötig zuvor installiert werden.

## Raspberry Pi/Debian/Ubuntu

```
sudo apt-get install arp-scan
```

## macOS

Falls der Paketmanager *brew* noch nicht installiert ist, kann er über folgenden Befehl installiert werden:

```
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

Anschließend kann *arp-scan* geladen werden:

```
brew install arp-scan
```

## Programm installieren

Das Bash-Script kann bspw. über *nano* als *divera247-wlan-scanner.sh* im Home-Verzeichnis erstellt werden.

```
nano ~/divera247-wlan-scanner.sh
```

Anschließend folgenden Inhalt einfügen und die Datei über Strg+O speichern und über Strg+X den Editor schließen.

```
#!/bin/sh

DEBUG=false # Enable debug output
STATUSURL_CONNECTED="https://www.divera247.com/statusgeber.html?status=1&accesskey=..." # Replace with your
personal status url if connected
STATUSURL_DISCONNECTED="https://www.divera247.com/statusgeber.html?status=2&accesskey=..." # Replace with your
personal status url if disconnected

HOSTNAME="divera247-phone" # Use `arp -a` to get your devicename
NETWORK_INTERFACE="en0" # Use `ifconfig` to get you connected network adapter

ACK_THRESHOLD=3 # Minimum times device must not be reachable to be marked as disconnected
ACK_COUNTER=0 # Number of times device did not acknowledge
DELAY=15 # Delay between network scans
CONNECTED=false # Initial state

function connectionStateToggled {
    if [ "$CONNECTED" = true ]; then
        echo "$HOSTNAME connected"

        # Set connected status
        CURL_OUTPUT=$(curl --silent -w "\n\nStatus Code: %{http_code}\n\n" "$STATUSURL_CONNECTED" 2>&1)
        STATUS_CODE=$(echo "$CURL_OUTPUT" | grep "Status Code")

        # Print server output in error case
        if [ $DEBUG = true ] || [[ "$STATUS_CODE" != *"200"* ]]; then
            echo "$CURL_OUTPUT"
        fi
    else
        echo "$HOSTNAME disconnected"
    fi
}
```

```

# Set disconnected status
CURL_OUTPUT=$(curl --silent -w "\n\nStatus Code: %{http_code}\n\n" "$STATUSURL_DISCONNECTED" 2>&1)
STATUS_CODE=$(echo "$CURL_OUTPUT" | grep "Status Code")

# Print server output in error case
if [ $DEBUG = true ] || [[ "$STATUS_CODE" != *"200"* ]]; then
    echo "$CURL_OUTPUT"
fi
fi
}

while true; do
    HOSTNAMES_FROM_ROUTER=$(arp -a)

    # Check if device is in cached list and get IP address from it
    IP_OF_DEVICE=$(echo "$HOSTNAMES_FROM_ROUTER" | grep $HOSTNAME | awk -F '[](){}' '{print $(NF-1)}')

    if [[ ! -z $IP_OF_DEVICE ]]; then
        # Scan network to check if device is really still connected
        NETWORKSCAN=$(arp-scan --interface=$NETWORK_INTERFACE --localnet)

        if [[ "$NETWORKSCAN" == *"$IP_OF_DEVICE"* ]]; then
            $DEBUG && echo "Found $HOSTNAME as $IP_OF_DEVICE in network scan"

            # Device is reachable by network scanner
            # If previously have been disconnected, set status at home
            if [ "$CONNECTED" = false ]; then
                CONNECTED=true
                ACK_COUNTER=0
                connectionStateToggled
            fi
        else
            $DEBUG && echo "$HOSTNAME did not respond to network scan"

            # Device is not reachable by network scanner
            ACK_COUNTER=$((ACK_COUNTER + 1))

            # If device didn't respond ACK_THRESHOLD times: set status or ignore
            if (( ACK_COUNTER >= ACK_THRESHOLD )) && [ "$CONNECTED" = true ]; then
                CONNECTED=false
                connectionStateToggled
            fi
        fi
    else
        $DEBUG && echo "$HOSTNAME wasn't found in routers list of known devices"

        # Device is not in cached list, so it has been disconnected a while ago
        ACK_COUNTER=$((ACK_COUNTER + 1))

        if (( ACK_COUNTER >= ACK_THRESHOLD )) && [ "$CONNECTED" = true ]; then
            CONNECTED=false
            connectionStateToggled
        fi
    fi

    # Wait some time before next iteration, otherwise devices could not respond to scan
    sleep $DELAY
done

```

## Als Administrator ausführen

Das Programm muss nun als *root* ausgeführt werden, also über:

```
sudo divera247-wlan-sniffer.sh
```

Wenn die Statusgeber-URL, der Hostname und der Netzwerkadapter des Geräts korrekt angepasst wurden, sollte nun automatisch der passende Status gesetzt werden und eine entsprechende Nachricht vom Programm ausgegeben werden.



#### Verzögerung bei der Erkennung

Das Skript scannt nur alle 15 Sekunden das Netzwerk, bei einer höheren Frequenz antworteten die Geräte nur noch sehr unzuverlässig. Um *false positives* zu vermeiden kann über *ACK\_THRESHOLD* bestimmt werden, wie oft das Gerät nicht erreichbar sein muss, bis der Status gesetzt wird.

## Tipps

### Hostname herausfinden

Der Hostname des Gerätes kann vom eingestellten abweichen. Der Befehl *arp -a* zeigt einem die vom Router festgelegten Hostnames an. Darüber kann der eigene zur Identifizierung herausgefunden werden (bspw. *divera247-phone.fritz.box*) und in der Variablen *HOSTNAME* eingetragen werden.

### Netzwerkadapter anpassen

Standardmäßig wird *en0* als Netzwerkadapter zur Überprüfung genutzt. Der Befehl *ifconfig* zeigt einem die Namen der eigenen Netzwerkadapter an, falls nötig kann die Variable *NETWORK\_INTERFACE* entsprechend angepasst werden.

### Debugging/Testing

Zum Testen wird empfohlen die Variable *DEBUG* auf *true* zu setzen und *DELAY* herabzusetzen.