IPlogger □□

> **IPlogger, Debian (+Derivate)**
> ***Wiki-Stand: 12.05.2026***
> ***Script-Stand: 12.05.2026***
>
> ***Download***

IPLogger DB – IP- und Verfügbarkeitsüberwachung

Überblick

Der `iplogger-db.sh` ist ein datenbankgestützter IP-Logger zur dauerhaften Überwachung der eigenen Internetverbindung und der öffentlichen WAN-IP-Adresse.
Das Script wurde dafür entwickelt, langfristig und nachvollziehbar festzuhalten:

- welche öffentliche IP-Adresse zu welchem Zeitpunkt aktiv war
- ob eine Internetverbindung vorhanden war
- ob lediglich die IP-Ermittlung fehlgeschlagen ist
- welche Prüfdienste funktioniert oder versagt haben
- wie lange einzelne IPs aktiv waren
- wann Provider oder Verbindungen ausgefallen sind

Der Fokus liegt nicht auf einer simplen „Wie ist meine IP“-Abfrage, sondern auf einer historisch nachvollziehbaren Langzeitaufzeichnung mit Auswertung und Zeiträumen.

Das System arbeitet vollständig datenbankbasiert und speichert jede einzelne Messung dauerhaft ab.

Ziel des Systems

Der Logger beantwortet unter anderem folgende Fragen:

- Welche öffentliche IP war am 12.03.2024 um 14:00 Uhr aktiv?
- Wie lange blieb eine bestimmte IP bestehen?
- Wann erfolgte ein DSL-/WAN-Reconnect?
- War die Leitung tatsächlich offline oder nur ein Prüfdienst gestört?
- Welche Provider lieferten fehlerhafte Antworten?
- Welche DNS-/Connectivity-Ziele waren erreichbar?
- Wie oft kam es zu Unterbrechungen?
- Wie stabil arbeitet die Internetanbindung?

Gerade bei:

- DSL-Zwangstrennungen
- CGNAT-/Providerwechseln
- VPN-/Tunnelproblemen
- Routingfehlern
- sporadischen Ausfällen
- Providerstörungen

liefert die Historie eine klare technische Nachvollziehbarkeit.

Architektur

Das System besteht aus mehreren Komponenten:

Komponente	Aufgabe
`providers`	Öffentliche IP-Abfragedienste
`connectivity_checks`	Prüft, ob Internet grundsätzlich erreichbar ist
`state`	Speichert Rotationszustände
`iplog`	Historische Hauptdatenbank
`summary_*`	Auswertung und Zusammenfassung
PDF-/CSV-Export	Archivierung und Dokumentation

Grundlogik

Der Ablauf einer Messung sieht vereinfacht so aus:

1. Connectivity prüfen
2. Wenn Internet vorhanden:
3. Öffentliche IP ermitteln
4. Ergebnis speichern
5. Statistiken aktualisieren

Wichtig dabei:

Das Script unterscheidet sauber zwischen:

Zustand	Bedeutung
`OFFLINE`	Keine Internetverbindung
`CHECK_FAILED`	Internet vorhanden, aber IP nicht ermittelbar
`OK`	IP erfolgreich ermittelt
`INVALID_RESPONSE`	Provider lieferte ungültige Antwort
`NO_RUN`	Reservierter Status

Dadurch wird verhindert, dass ein einfacher Fehler eines einzelnen IP-Dienstes fälschlich als Internetausfall interpretiert wird.

Initialisierung

Einstiegspunkt: `init`

```
./iplogger-db.sh init
```

Dieser Befehl erstellt die komplette Datenbankstruktur.

Dabei werden automatisch angelegt:

- Tabellen
- Standardprovider
- Connectivity-Ziele
- Rotationsstatus

Datenbanktabellen

Es muss vorab ein Benutzer und eine Datenbank erstellt sein. Die init-Datenbankinitialisierung erstellt dann die Tabellen.

providers

Enthält alle IP-Ermittlungsdienste.

Standardmäßig:

- icanhazip.com
- ifconfig.me
- api.ipify.org
- checkip.amazonaws.com

Gespeichert werden zusätzlich:

- Erfolgszähler
- Fehlerzähler
- letzter Erfolg
- letzter Fehler
- Timeoutwerte

Beispiel:

Name	URL
icanhazip.com	https://icanhazip.com

Name	URL
api.ipify.org	https://api.ipify.org

---

connectivity_checks

Prüft die generelle Internetverfügbarkeit.

Standardmäßig:

- Google DNS (`8.8.8.8`)
- Quad9 (`9.9.9.9`)
- Cloudflare (`1.1.1.1`)

Die Prüfung erfolgt per Ping.

---

state

Speichert interne Rotationsinformationen.

Dadurch wird nicht immer derselbe Provider zuerst verwendet.

Beispiele:

Name	Wert
next_provider_index	2
next_connectivity_index	1

---

iplog

Die zentrale Historientabelle.

Hier wird jede Messung dauerhaft gespeichert.

Gespeichert werden unter anderem:

- Messzeitpunkt
- öffentliche IP
- Status
- verwendeter Provider
- Anzahl Versuche
- Fehlermeldungen
- Rohantworten
- Connectivity-Status

Rotationssystem

Warum Rotation?

Würde immer derselbe Provider zuerst verwendet werden:

- wäre dieser stärker belastet
- Fehler würden schwerer auffallen
- andere Provider würden kaum genutzt

Deshalb arbeitet das System rotierend.

Beispiel

Durchlauf 1

1. icanhazip.com
2. ifconfig.me
3. ipify

Durchlauf 2

1. ifconfig.me
2. ipify
3. icanhazip.com

Durchlauf 3

1. ipify
2. icanhazip.com
3. ifconfig.me

Dadurch verteilt sich die Last gleichmäßig.

Connectivity-Prüfung

Einstiegspunkt: `check` oder `run`

```
./iplogger-db.sh check
```

oder:

```
./iplogger-db.sh run
```

Beides startet dieselbe Messlogik.

Ablauf der Connectivity-Prüfung

Vor der eigentlichen IP-Ermittlung wird geprüft:

- besteht überhaupt Internet?
- ist Routing vorhanden?
- funktionieren externe Ziele?

Dazu werden mehrere bekannte Ziele angepingt.

Erst wenn mindestens eines erreichbar ist, wird die IP-Abfrage gestartet.

Provider-Prüfung

Nach erfolgreicher Connectivity-Prüfung:

```
curl -> Provider -> Antwort validieren
```

Dabei wird geprüft:

- ist die Antwort leer?
- ist die Antwort eine gültige IPv4?
- ist die Antwort eine gültige IPv6?

Nur dann gilt der Durchlauf als erfolgreich.

Beispielabläufe

Erfolgreicher Lauf

Connectivity OK

Provider erreichbar

IP gültig

→ Status OK

Gespeichert wird beispielsweise:

2026-05-12 22:00:00

IP: 93.192.117.44

Status: OK

Internet vorhanden, aber Provider kaputt

Connectivity OK

Provider liefert Müll

→ CHECK_FAILED

Beispiel:

Provider https://api.example liefert:

ERROR 500

Das Internet funktioniert trotzdem.

Komplett offline

```
Google DNS nicht erreichbar
Cloudflare nicht erreichbar
Quad9 nicht erreichbar
→ OFFLINE
```

Summary-System

Das eigentliche Kernstück ist die Verlaufsanalyse.

Einzelne Messungen werden zu Zeiträumen zusammengefasst.

Einstiegspunkt: `summary`

Tabellenansicht

```
./iplogger-db.sh summary tabelle
```

Beispiel:

```
BEGIN_AT         END_AT           IP              SAMPLES OFFLINE FAILS
2026-05-01 10:00  2026-05-03 02:00   93.192.117.44       91      0     0
```

Textansicht

```
./iplogger-db.sh summary text
```

Beispiel:

```
01.05.2026 10:00 Uhr - 03.05.2026 02:00 Uhr:
IP 93.192.117.44 (91x online)
```

Wie die Periodenerkennung funktioniert

Die Funktion `summary_rows()` analysiert die komplette Historie.

Sobald sich die IP ändert:

```
alte Periode schließen
neue Periode beginnen
```

Zusätzlich werden innerhalb der Periode gezählt:

- Offline-Ereignisse

- Fehler
- erfolgreiche Samples

Detailmodus

Einstiegspunkt

```
./iplogger-db.sh summary export pdf details
```

Der Detailmodus erzeugt eine vollständige Zeitauflistung aller erfolgreichen Messungen innerhalb einer Periode.

Beispiel

```
16.02.2020 05:00 Uhr - 16.02.2020 11:00 Uhr:
IP 93.192.117.44 (24x online)

- 16.02.2020 05:00 06:00 07:00 08:00
- 16.02.2020 09:00 10:00 11:00
```

Oder bei Tageswechsel:

```
07.02.2021 04:00 05:00 06:00
08.02.2021 00:00 01:00 02:00
```

Dadurch werden lange Zeiträume kompakt lesbar dargestellt.

PDF-Export

Einstiegspunkt

```
./iplogger-db.sh summary export pdf
```

oder:

```
./iplogger-db.sh summary export pdf details
```

Der Export erzeugt automatisch:

- TXT-Zwischendatei
- PDF-Datei
- Überschrift
- Datum
- Seitenzahlen

Verwendet werden bevorzugt:

- `enscript`
- `ps2pdf`

Alternativ:

- `pandoc`

CSV-Export

Einstiegspunkt

```
./iplogger-db.sh summary export csv
```

Exportiert die zusammengefassten Perioden in maschinenlesbarer Form.

Beispiel:

```
begin_at;end_at;ip;samples;offline;fails
2026-05-01 10:00;2026-05-03 02:00;93.192.117.44;91;0;0
```

Historischer CSV-Import

Einstiegspunkt

```
./iplogger-db.sh import ipdb.csv ";"
```

Der CSV-Import dient primär zur Übernahme historischer Daten aus der ersten Generation des IPLoggers.

Dabei werden ältere Aufzeichnungen in die neue Datenbankstruktur übernommen.

Unterstützt werden:

- erfolgreiche IPs
- Offline-Einträge
- Fehlerzustände

Typische Einsatzszenarien

WAN-Reconnects nachvollziehen

IP 1:
01:00 - 12:00

IP 2:
12:05 - aktuell

→ DSL-Zwangstrennung klar sichtbar.

Providerprobleme erkennen

15x CHECK_FAILED
aber keine OFFLINE-Zustände

→ Internet vorhanden, aber IP-Dienst gestört.

VPN-/Tunnelprobleme analysieren

Wenn:

- Connectivity vorhanden
- aber bestimmte Ziele nicht erreichbar

kann nachvollzogen werden:

- wann Routingprobleme begannen
- wie lange sie bestanden
- ob WAN-Wechsel beteiligt waren

Besonderheiten

IPv4 und IPv6

Das System akzeptiert beide Varianten automatisch.

Zeitbasierte Rotation

Provider und Connectivity-Checks werden nicht statisch verwendet.

Das verhindert:

- Single-Point-Abhängigkeiten
- einseitige Last
- verfälschte Statistiken

Dauerhafte Historie

Die Datenbank arbeitet append-only.

Neue Einträge werden ausschließlich ergänzt.

Dadurch bleibt die komplette Historie nachvollziehbar erhalten.

Beispiel für Cronjob

```
0 * * * * /srv/scripte/iplogger-db.sh check >/dev/null 2>&1
```

Dadurch erfolgt stündlich eine neue Messung.

Fazit

`iplogger-db.sh` ist kein einfacher „Wie ist meine IP“-Checker, sondern ein vollständiges Langzeit-Überwachungs- und Auswertungssystem für:

- öffentliche WAN-IPs
- Internetverfügbarkeit
- Providerstabilität
- Fehleranalysen
- historische Zeiträume
- dokumentierbare Verbindungsverläufe

Besonders wertvoll wird das System durch:

- die periodische Zusammenfassung
- die saubere Statusunterscheidung
- die rotierenden Prüfmechanismen
- die nachvollziehbare Langzeithistorie
- die Exportfunktionen für Archivierung und Dokumentation

□□ Download

Das **Programm** kann hier in immer der neuesten Version heruntergeladen werden:

IPlogger Script (via mariobeh.de)

Revision #4
Created 12 May 2026 21:46:57 by Admin
Updated 13 May 2026 11:01:43 by Admin