Security Guideline: Audit Log for IRF1000 / IRF3000

Audit Log Events and Usage

Firmware Version	2.2.6
Document Version	1.0
Autor	ADS-TEC IIT GmbH
Datum	2025-12-11
Generated	2025-12-11 10:26

© 2025 ADS-TEC Industrial IT GmbH

Table of Contents

1 Introduction 4
1.1 Audit Log . . . 4
1.1.1 Introduction . . . 4
1.1.2 What is the audit log on IRF1000/IRF3000? . . . 4
1.1.3 Where to view the audit log . . . 4
1.1.4 How auditd works on the device (short version) . . . 4
1.1.5 Best practices . . . 5
1.1.6 Notes on event coverage and differences from generic Linux . . . 5
1.1.7 Audit Log to Syslog . . . 5
1.1.8 Audit log views in the web interface . . . 6
2 Audit Events 6
Anomaly . . . 6
Network Anomaly . . . 6
Introduction . . . 6
Webinterface View . . . 6
Explanation of the raw audit.log entry . . . 7
When the event is generated . . . 8
Sample audit.log entry . . . 8
NTP/NTS-KE - Integrity . . . 9
Introduction . . . 9
Webinterface View . . . 9
Explanation of the raw audit.log entry . . . 10
When the event is generated . . . 11
Sample audit.log entry . . . 11
Configuration . . . 11
Introduction . . . 11
Webinterface View . . . 12
Explanation of the raw audit.log entry . . . 12
When the event is generated . . . 13
Sample audit.log entry . . . 14
Device . . . 14
Introduction . . . 14
Webinterface View . . . 14
Columns in the Device audit table . . . 15
Explanation of the raw audit.log entry . . . 15
USER_DEVICE fields in audit.log . . . 15
When the event is generated . . . 16
Sample audit.log entry . . . 17
Packet Filter . . . 17
Introduction . . . 17

© 2025 ADS-TEC Industrial IT GmbH

Webinterface View . . . 18
Explanation of the raw audit.log entry . . . 18
When the event is generated . . . 19
Sample audit.log entry . . . 19
Remote Service . . . 19
Introduction . . . 19
Webinterface view . . . 19
Explanation of the raw audit.log entries . . . 20
When the event is generated . . . 21
Sample audit.log entry . . . 22
System . . . 22
Introduction . . . 22
Webinterface View . . . 22
Fields in the System audit table . . . 23
Explanation of the raw audit.log entry . . . 23
Example structure . . . 23
Relevant fields for CUSTOM_TIME_JUMP . . . 23
When the event is generated . . . 25
Sample audit.log entry . . . 25
Authentication . . . 26
Authentication - Login . . . 26
Introduction . . . 26
Webinterface View . . . 26
Explanation of the raw audit.log entry . . . 27
When the event is generated . . . 28
Sample audit.log entry . . . 28
Authentication - Session Timeouts . . . 28
Introduction . . . 28
Webinterface View . . . 29
Explanation of the raw audit.log entry . . . 29
When the event is generated . . . 30
Sample audit.log entry . . . 30
Authentication - Logout . . . 31
Introduction . . . 31
Webinterface View . . . 31
Explanation of the raw audit.log entry . . . 31
When the event is generated . . . 32
Sample audit.log entry . . . 33

© 2025 ADS-TEC Industrial IT GmbH

1 Introduction

1.1 Audit Log

1.1.1 Introduction

The ADS-TEC Industrial IT IRF1000 and IRF3000 are industrial VPN router firewalls designed for secure remote access, network segmentation, and reliable operation in demanding environments. A key security capability of these devices is the audit log. It provides a trustworthy, tamper-resistant record of relevant security and system events so that administrators and auditors can reconstruct who did what, when, and on which part of the system.

This chapter explains how audit logging works on the devices, what you can see in the web interface, how to download the raw log file, and what kinds of events are recorded. It is written for end users and administrators and avoids Linux internals where possible—while still providing enough context to use the feature effectively.

1.1.2 What is the audit log on IRF1000/IRF3000?

- The audit log on the IRF series is built on top of the Linux standard service "auditd" (the Linux Auditing System).
- ADS-TEC adds device-specific records and consistent formatting so that events are meaningful for appliance administration.
- The device web interface provides an "Audit Log" view where you can read selected events without leaving the browser.
- For in-depth analysis or compliance archiving, you can download the raw audit.log file exactly as produced by auditd on the device.

This will include much more events including standard internal auditid startup and configuration change events or operating system events which are not visible in the web interface.

Why this matters:

- Provides traceability for configuration changes and administrative actions.
- Helps detect suspicious activity (for example, repeated failed logins).
- Supports compliance processes and incident response by offering an authoritative event history.

1.1.3 Where to view the audit log

You can access the audit log via the device web interface:

1. Sign in to the IRF1000/IRF3000 web UI. All web UI accounts have permissions to view logs.
2. Open: Diagnostics → Audit. Select one of the tabs to view a specific category.
3. Open the Download tab to export the raw audit.log file as a .tar.gz archive.

1.1.4 How auditd works on the device (short version)

-
  - auditd is the Linux Auditing System service responsible for reliably recording security-relevant events.
  - The IRF firmware configures auditd with rules that cover common administrative and security actions on the appliance.

© 2025 ADS-TEC Industrial IT GmbH

- Some events come directly from standard Linux audit facilities (for example, authentication), while other events are added or enriched by ADS-TEC components to better represent device-specific operations (for example, VPN state or packet filter activity).

You do not need to manage auditd rules manually. The device provides a ready-to-use configuration and a simple web interface for reading and exporting logs.

1.1.5 Best practices

- Regularly review failed authentication attempts and unexpected configuration changes.
- Use the raw audit.log download for incident reviews and to retain evidence according to your organization's policy.
- Keep device time synchronized. If you integrate with external logging or SIEM, align time zones and formats. You have to use NTP as the devices do not offer a battery buffered RTC.

1.1.6 Notes on event coverage and differences from generic Linux

- On IRF devices, audit logging focuses on security-relevant appliance actions rather than every low-level kernel event.
- Some entries are enriched with human-readable fields (for example, configuration object names) to make them easier to interpret in the web UI.
- When comparing with standard Linux audit logs, expect naming and field additions that reflect ADS-TEC appliance context.

Important rate limits and retention:

- The audit.log is a 50 MB ring buffer shared by all audit message types; oldest entries are overwritten once full.
- The ring buffer uses 5 files with 10 MB each, thus the minimum storage is 40 MB and the maximum storage is 50 MB.
- Packet filter (NETFILTER_PKT): one entry per minute per audited rule.
- Device interface (USER_DEVICE): LINK max 20 messages/day; IO max 20 messages/hour; USB not rate-limited.
- Remote service (CUSTOM_REMOTE_SERVICE): max 5 messages/hour.

1.1.7 Audit Log to Syslog

Starting with firmware version 2.2.5, the IRF1000/IRF3000 can forward audit log entries to a syslog server. This is useful for centralized logging, long-term retention, and compliance reporting.

To enable this:

1. Sign in to the device web interface.
2. Open: **Diagnostics** → **Eventlog** → **Configuration**.
3. Enter the **Syslog Server IP address**.
4. Enable the **Include Audit Log** option.
5. Apply or save the settings.

From this point on, selected audit log entries are sent to the configured syslog server in addition to being written to `audit.log` on the device.

© 2025 ADS-TEC Industrial IT GmbH

> **Important:** Standard syslog transport is unencrypted and unauthenticated. To protect the log data in transit, place the syslog server in a trusted network segment and use IPsec or OpenVPN to encrypt the traffic between the device and the syslog server.

1.1.8 Audit log views in the web interface

The device web interface currently supports the following audit log views. The chapters in this document explain them in detail.

- AUDITVIEW — Authentication audit (covers USER_LOGIN, USER_LOGOUT, USER_END)
- AUDITCONF — Configuration audit (USYS_CONFIG)
- AUDITSYSTEM — System audit
- AUDITNETFILTER — Packet filter audit (NETFILTER_PKT)
- AUDITACCESS — Access audit
- AUDITANOMALY — Anomaly audit (no events yet as of firmware version 2.2.4)
- AUDITDEVICE — Device audit (USER_DEVICE)
- AUDITREMOTESERVICE — Remote service audit (CUSTOM_REMOTE_SERVICE)

The given constants represent the path on the web server of the device. i.e. https://192.168.0.254/priv/priv.php?id=AUDITNETFILTER will open the AUDITNETFILTER view.

2 Audit Events

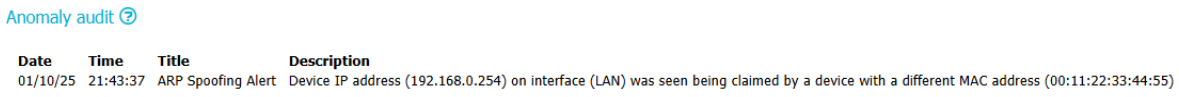
Anomaly

Network Anomaly

Introduction

RESP_ANOMALY events report network anomaly detections on ADS-TEC IRF1000 and IRF3000 devices. The appliance raises this event when the integrated monitor detects threats such as ARP spoofing or IP conflicts that may affect the OT network. The device limits event creation to five messages per hour to prevent audit.log overflow. When you see such events, verify the switch port and connected device, check for duplicated IP addresses, and ensure that only trusted devices are present on the affected network segment. Where possible, apply port security or DHCP snooping on switches and review your remote access policies to reduce the risk of successful spoofing attacks.

Webinterface View



In the VIEW web interface, RESP_ANOMALY entries are shown in the **Anomaly audit** section as human-readable alerts. The system parses the raw audit.log data and displays one line per detected anomaly. This view displays alerts from the integrated network monitoring system and reports potential security threats such as ARP spoofing attacks or duplicate IP addresses that conflict with the device's own IP address.

The table in the web interface contains the following columns:

© 2025 ADS-TEC Industrial IT GmbH

Column	Description	Example
Date	Local date when the anomaly was logged.	`01/10/25`
Time	Local time when the anomaly was logged.	`21:43:37`
Title	Short title of the anomaly, derived from the audit field `title`.	`ARP Spoofing Alert`
Description	Human-readable description taken from the audit field `description`. It includes the device IP address, interface, and the conflicting MAC address.	`Device IP address (192.168.0.254) on interface (LAN) was seen being claimed by a device with a different MAC address (00:11:22:33:44:55)`

Use this view for quick operational checks. You can immediately see when the device detects suspicious ARP activity and which IP address and interface are affected. For deeper analysis, correlate the same timestamp and title with the full RESP_ANOMALY entry in the audit.log file that is ingested into your SIEM or SOC tools.

Explanation of the raw audit.log entry

The device writes RESP_ANOMALY records into `audit.log`. Generic auditd fields follow standard auditd behavior; refer to the official auditd documentation for full semantics. The anomaly-specific details are contained inside the `msg='...'` payload.

The following table summarizes the relevant fields for RESP_ANOMALY entries:

Field	Description	Example
type	Audit record type.	`RESP_ANOMALY`
msg=audit(ts:serial)	Audit header with Unix timestamp and serial number.	`audit(1736541817.640:20)`
pid	Process ID of the program that emitted the message.	`6493`
uid	Effective UID of the process.	`65534`
auid	Login audit user ID.	`4294967295`
ses	Audit session ID.	`4294967295`
msg (string)	Start of the user message payload that contains anomaly details.	`msg='...'`
interface	Ethernet interface on which the network anomaly occurred.	`LAN`
ip_address	IP address involved in the anomaly.	`192.168.0.254`
mac_address	MAC address that was seen claiming the IP address.	`00:11:22:33:44:55`
title	Short title of the anomaly for display in the web UI.	`ARP Spoofing Alert`

© 2025 ADS-TEC Industrial IT GmbH

Field	Description	Example
description	Human-readable description of the network anomaly.	`Device IP address (192.168.0.254) on interface (LAN) was seen being claimed by a device with a different MAC address (00:11:22:33:44:55)`
exe	Path of the executable that wrote the audit-log message.	`/sbin/arp-spoof-detector`
hostname	Auditd hostname field; not filled for this event.	?
addr	Auditd address field; not filled for this event.	?
terminal	Auditd terminal field; not filled for this event.	?
res	Result flag from auditd. For RESP_ANOMALY this is always set to `failed`, because no context-dependent success evaluation is possible.	`failed`
UID	Text rendering of `uid` appended by auditd.	`UID="nobody"`
AUID	Text rendering of `auid` appended by auditd.	`AUID="unset"`

When integrating into a SIEM or SOC, use `type=RESP_ANOMALY` as the main filter and parse the key–value pairs inside `msg='...'` to extract interface, IP address, MAC address, title, and description for correlation and alerting.

When the event is generated

The RESP_ANOMALY event is generated by the **arp-spoof-detector** program when it observes that one IP address is claimed by more than one MAC address on a monitored interface. This typically indicates ARP spoofing activity or a duplicate IP address situation that may also conflict with the firewall's own IP configuration.

The detector monitors ARP traffic on the configured interfaces. When it detects such an anomaly, it calls `audit_log_user_message()` from *libaudit* and writes a RESP_ANOMALY entry into the audit.log file. The detector sends a user message when its detection state changes, for example when it detects a new anomaly or when an existing anomaly is cleared. To protect log storage, the system limits RESP_ANOMALY event creation to five messages per hour.

Sample audit.log entry

```
type=RESP_ANOMALY msg=audit(1736541817.640:20): pid=6493 uid=65534 auid=4294967295
                  ses=4294967295 msg='interface="LAN" ip_address="192.168.0.254"
                  mac_address="00:11:22:33:44:55" title="ARP Spoofing Alert"
                  description="Device IP address (192.168.0.254) on interface (LAN) was
                  seen being claimed by a device with a different MAC address
                  (00:11:22:33:44:55)"  exe="/sbin/arp-spoof-detector" hostname=?
                  addr=? terminal=? res=failed'UID="nobody" AUID="unset"
```


© 2025 ADS-TEC Industrial IT GmbH

NTP/NTS-KE - Integrity

Introduction

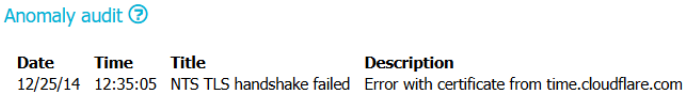
The `CUSTOM_NTS_INTEGRITY` audit event documents problems during the secure Network Time Security (NTS) TLS handshake between the firewall appliance and its configured NTS time server.

If this handshake fails, the device may not be able to synchronize time securely. In industrial and OT environments this can impact log integrity, time-based access rules, and correlation of events in a SIEM or SOC.

This event is created by `chronyd` and written into the Linux `audit.log`. It is rate-limited to 4 messages per hour to avoid filling the audit log with repeated handshake errors.

From a security perspective you should investigate these events, because they can indicate invalid or untrusted certificates, misconfigured TLS parameters, or network issues that prevent secure time synchronization. Until the issue is fixed, treat timestamps from the device with care when performing forensic analysis or compliance reporting.

Webinterface View



In the VIEW web interface this event appears in the **Anomaly audit** section.

The `CUSTOM_NTS_INTEGRITY` records are shown together with other anomaly alerts from the integrated monitoring system. An example row looks like in the screenshot:

- Title: `NTS TLS handshake failed`
- Description: `Error with certificate from time.cloudflare.com`

The table columns in the Anomaly audit view are:

Web field	Description	Example value
Date	Local date when the audit event was recorded, derived from the audit time.	`12/25/14`
Time	Local time when the event was recorded.	`12:35:05`
Title	Short summary taken from the `title` field of the audit record.	`NTS TLS handshake failed`
Description	Detailed text from the `description` field of the audit record.	`Error with certificate from time.cloudflare.com`

Usage hints for operators and SOC analysts:

-
  - Use the Anomaly audit view to quickly see recent NTS integrity problems without parsing the raw `audit.log`.
  - Correlate the **Date** and **Time** with other events in your SIEM (for example VPN changes or certificate deployments).
  - If `NTS TLS handshake failed` appears:

© 2025 ADS-TEC Industrial IT GmbH

- Check the NTS server configuration on the device.
- Verify the server certificate and CA trust on both sides.
- Verify that intermediate firewalls or TLS-inspection devices are not breaking the TLS handshake.

All entries visible in the Anomaly audit table are backed by entries in `/var/log/audit/audit.log`, so they can be forwarded and parsed centrally for SIEM/SOC integration.

Explanation of the raw audit.log entry

The `CUSTOM_NTS_INTEGRITY` event is written into `/var/log/audit/audit.log` by `auditd`. A sample line is:

```
type=CUSTOM_NTS_INTEGRITY msg=audit(1419507305.264:11): pid=7121 uid=323 auid=4294967295 ses=42949
```

Only the event-specific fields are explained below. For the generic audit fields such as `pid`, `uid`, `auid`, and `ses` you can refer to the official `auditd` documentation.

Field	Description	Example value
`type`	Audit record type. Identifies this message as an NTS integrity event.	`CUSTOM_NTS_INTEGRITY`
`msg` (hdr)	Standard audit header with timestamp and event ID.	`audit(1419507305.264:11)`
`pid`	Process ID of the process that triggered the event (`chronyd`).	`7121`
`uid`	User ID under which the process runs.	`323`
`auid`	Audit user ID.	`4294967295`
`ses`	Audit session ID.	`4294967295`
`msg` (body)	Free-form key-value block with human readable details.	`title="..." description="..." ...`
`title`	Short summary of the problem. This text is shown as **Title** in the Anomaly audit table.	`NTS TLS handshake failed`
`description`	Detailed description, usually naming the affected NTS server or certificate problem. Shown as **Description** in the web UI.	`Error with certificate from time.cloudflare.com`
`exe`	Full path to the executable that generated the audit record.	`/usr/sbin/chronyd`
`hostname`	Not used for this event. Always ?.	?
`addr`	Not used for this event. Always ?.	?
`terminal`	Not used for this event. Always ?.	?
`res`	Result flag from `auditd`. For this custom event it is always set to `success` for technical reasons.	`success`
`UID`	Textual user name corresponding to `uid`.	`chrony`

© 2025 ADS-TEC Industrial IT GmbH

Field	Description	Example value
AUID	Textual representation of auid.	unset

For SIEM/SOC integration you typically:

- Filter on type=CUSTOM_NTS_INTEGRITY to capture all NTS integrity events.
- Extract title, description, and exe from the msg body for use as normalized fields (event.action, event.reason, process.name, and similar).
- Use the audit header timestamp audit(…) as the primary event time and convert it to your SIEM time zone.

Repeated occurrences of this event for the same NTS server indicate a persistent misconfiguration or connectivity problem that should be resolved to maintain correct and trustworthy system time.

When the event is generated

A CUSTOM_NTS_INTEGRITY event is generated whenever an NTS-TLS handshake between the device (NTS client) and a configured NTS server fails.

The event is produced as follows:

- The NTP daemon chronyd detects that it cannot complete an NTS-TLS handshake with an NTS time server.
- Typical technical reasons are:
  - Invalid, expired, or mismatching server certificate.
  - The server certificate chain cannot be validated to a trusted CA.
  - Incompatible TLS parameters or cipher suites.
  - Network interruptions or middleboxes interfering with the TLS handshake.
- chronyd calls audit_log_user_message() from libaudit.
- auditd writes an entry of type CUSTOM_NTS_INTEGRITY into /var/log/audit/audit.log with a short title and a more detailed description.

To protect the audit log against flooding, the device limits the creation of this audit event to 4 messages per hour. Additional handshake failures during that hour will not create further CUSTOM_NTS_INTEGRITY entries.

Note: the res field inside the message is always set to success for technical reasons, even though the semantic meaning of the event is a handshake *failure*. The failure is expressed in the title and description fields.

Sample audit.log entry

```
type=CUSTOM_NTS_INTEGRITY msg=audit(1419507305.264:11): pid=7121 uid=323
                    auid=4294967295 ses=4294967295 msg='title="NTS TLS handshake
                    failed" description="Error with certificate from
                    time.cloudflare.com" exe="/usr/sbin/chronyd" hostname=? addr=?
                    terminal=? res=success'UID="chrony" AUID="unset"
```

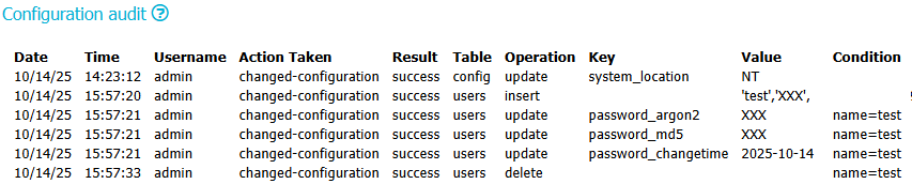
Configuration

Introduction

© 2025 ADS-TEC Industrial IT GmbH

USYS_CONFIG events record every configuration change on ADS-TEC IRF1000/IRF3000 devices. The event payload contains the affected configuration table, the changed key, the new value, and the condition that selected the row. The device logs sensitive or deprecated values as XXX to avoid credential leakage and to indicate cleared legacy fields such as `password_md5`. Use these events to understand who changed what and when in OT environments. From a security view you should look for unexpected user insert or delete operations, password updates outside approved maintenance windows, bulk edits to the `config` table, or repeated failed changes. Investigate events where the acting user is unexpected, where changes originate from maintenance-disabled accounts, or where a configuration change is immediately followed by VPN or session creation from a new source address.

Webinterface View


Configuration audit

Date	Time	Username	Action Taken	Result	Table	Operation	Key	Value	Condition
10/14/25	14:23:12	admin	changed-configuration	success	config	update	system_location	NT	
10/14/25	15:57:20	admin	changed-configuration	success	users	insert		'test','XXX',	
10/14/25	15:57:21	admin	changed-configuration	success	users	update	password_argon2	XXX	name=test
10/14/25	15:57:21	admin	changed-configuration	success	users	update	password_md5	XXX	name=test
10/14/25	15:57:21	admin	changed-configuration	success	users	update	password_changetime	2025-10-14	name=test
10/14/25	15:57:33	admin	changed-configuration	success	users	delete			name=test

Configuration changes are shown in the VIEW web interface on the **Configuration audit** page. Each row in the table represents one USYS_CONFIG operation and is displayed in device local time. For these events the **Action Taken** column always shows the normalized label `changed-configuration`.

Field	Description	Example
Date	Event date in device local time	10/14/25
Time	Event time with seconds	15:57:21
Username	User that performed the change (mapped from `terminal`)	admin
Action Taken	Normalized action label for configuration writes	changed-configuration
Result	Outcome of the operation	success
Table	Target configuration table	users
Operation	Operation type (`insert`, `update`, `delete`)	update
Key	Column or configuration key that changed	password_md5
Value	New value. Sensitive or legacy values are shown as XXX.	XXX
Condition	Selector used to identify the affected row; empty for inserts	name=test

Insert operations show an empty **Condition** and a **Value** list that contains the new row values, with sensitive fields masked as XXX.

Updates of `password_*` keys are masked in **Value**. In the example, the `password_md5` key is a legacy column that is no longer used; it is cleared and therefore appears as XXX.

Explanation of the raw audit.log entry

Each USYS_CONFIG line in `audit.log` uses the standard Linux audit record format. ADS-TEC specific semantics are contained in the quoted `msg` payload. Auditd-internal fields such as `pid`, `uid`, `auid`, `ses`, `UID`, and `AUID` follow the normal auditd behaviour; see the official auditd documentation for details. The device does not populate `hostname` or `addr` for these events.

© 2025 ADS-TEC Industrial IT GmbH

Field	Description	Example
type	Audit record type	USYS_CONFIG
msg header	`audit(epoch:serial)` identifier of the record	audit(1760450241.912:100)
pid	Process ID (auditd standard field)	5465
uid	Effective UID (auditd standard field)	0
auid	Login UID (auditd standard field)	4294967295
ses	Audit session ID (auditd standard field)	4294967295
msg payload	Quoted key-value list carrying change details	'table="users" ...'
table	Target table of the configuration change	users
operation	Operation applied to the table (`insert`, `update`, `delete`)	update
key	Column or configuration key whose value is changed	password_argon2
value	New value written to the key. Secrets are logged as XXX.	XXX
condition	WHERE-like selector that identifies the affected row; can be empty	name=test
exe	Executable that performs the change	/usr/sbin/systemd
hostname	Not populated for these events	?
addr	Not populated for these events	?
terminal	User name that initiated the change, as seen by the system	admin
res	Result of the operation (`success` or `failure`)	success
UID	Human-readable form of `uid`	root
AUID	Human-readable form of `auid`	unset

SIEM and SOC integration hints:

- Alert on `operation=insert` or `operation=delete` on `table=users`, and on `key=password_*` updates outside approved change windows.
- Correlate the `terminal` field (shown as Username in the web interface) with authentication logs to confirm the real user identity.
- Track bursts of updates on `table=config` to detect bulk or scripted unauthorized changes.
- Investigate sequences with `res=failure` followed by `res=success` for the same user or table, because they can indicate guessing or trial-and-error changes.

When the event is generated

© 2025 ADS-TEC Industrial IT GmbH

A USYS_CONFIG audit event is generated whenever the device writes to its configuration database through any supported path: the VIEW web interface, JSON-RPC, or the Adsdp API. The change is executed by `systemd` / `adsdpd` using the ADS-TEC `libnvram` implementation. `libnvram` calls the libaudit function `audit_log_user_message()`, which writes the event into `audit.log`. The event does not distinguish which API path triggered the change; you have to use the recorded user in the `terminal` field (shown as Username in the web interface) together with other logs to attribute the action. Logged operations include `insert`, `update`, and `delete` on configuration tables such as `config` and `users`. The `res` field records whether the operation ended with `success` or `failure`.

Sample audit.log entry

```
(1) type=USYS_CONFIG msg=audit(1760444592.644:86): pid=5465 uid=0 auid=4294967295
    ses=4294967295 msg='table="config" operation="update" key="system_location"
    value="NT" condition="" exe="/usr/sbin/systemd" hostname=? addr=? terminal=admin
    res=success'UID="root" AUID="unset"
(2) type=USYS_CONFIG msg=audit(1760450240.952:98): pid=5465 uid=0 auid=4294967295
    ses=4294967295 msg='table="users" operation="insert" key=""
    value="'test','XXX','XXX','1','XXX','1970-01-01','1970-01-01'" condition=""
    exe="/usr/sbin/systemd" hostname=? addr=? terminal=admin res=success'UID="root"
    AUID="unset"
(3) type=USYS_CONFIG msg=audit(1760450241.912:100): pid=5465 uid=0 auid=4294967295
    ses=4294967295 msg='table="users" operation="update" key="password_argon2"
    value="XXX" condition="name=test" exe="/usr/sbin/systemd" hostname=? addr=?
    terminal=admin res=success'UID="root" AUID="unset"
(4) type=USYS_CONFIG msg=audit(1760450241.912:101): pid=5465 uid=0 auid=4294967295
    ses=4294967295 msg='table="users" operation="update" key="password_md5" value="XXX"
    condition="name=test" exe="/usr/sbin/systemd" hostname=? addr=? terminal=admin
    res=success'UID="root" AUID="unset"
(5) type=USYS_CONFIG msg=audit(1760450241.912:102): pid=5465 uid=0 auid=4294967295
    ses=4294967295 msg='table="users" operation="update" key="password_changetime"
    value="2025-10-14" condition="name=test" exe="/usr/sbin/systemd" hostname=? addr=?
    terminal=admin res=success'UID="root" AUID="unset"
(6) type=USYS_CONFIG msg=audit(1760450253.616:105): pid=5465 uid=0 auid=4294967295
    ses=4294967295 msg='table="users" operation="delete" key="" value=""
    condition="name=test" exe="/usr/sbin/systemd" hostname=? addr=? terminal=admin
    res=success'UID="root" AUID="unset"
```

Device

Introduction

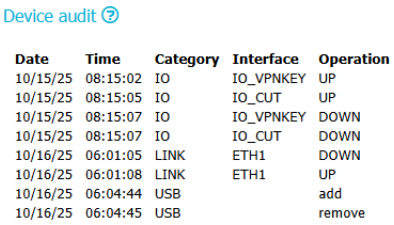
USER_DEVICE events show changes on external interfaces of the ADS-TEC IRF1000 and IRF3000. They record USB device add/remove, Ethernet link UP/DOWN, and VPN-Key/CUT input transitions. Use these entries to detect unauthorized USB usage, suspicious link flapping, or unexpected VPN-Key/CUT state changes.

If a cabinet should be sealed, any USB add operation or IO state change to UP outside a defined maintenance window is a likely policy violation. Configure SIEM rules for repetitive link toggles and for USB activity on production systems.

Note: the IRF1000 has no external USB ports. Therefore IRF1000 devices do not generate USB USER_DEVICE events.

Webinterface View

© 2025 ADS-TEC Industrial IT GmbH



Device audit

Date	Time	Category	Interface	Operation
10/15/25	08:15:02	IO	IO_VPNKEY	UP
10/15/25	08:15:05	IO	IO_CUT	UP
10/15/25	08:15:07	IO	IO_VPNKEY	DOWN
10/15/25	08:15:07	IO	IO_CUT	DOWN
10/16/25	06:01:05	LINK	ETH1	DOWN
10/16/25	06:01:08	LINK	ETH1	UP
10/16/25	06:04:44	USB		add
10/16/25	06:04:45	USB		remove

The VIEW web interface lists USER_DEVICE entries in the **Device audit** table. The table shows one line per event, ordered by time. The screenshot in this section shows typical IO VPN-Key/CUT UP/DOWN events, LINK ETH1 DOWN/UP events, and USB add/remove events.

For category **USB**, the vendor ID and product ID are not displayed in the web interface at the current time. The **Interface** column is empty for USB events.

Columns in the Device audit table

Field	Description	Example
Date	Local device date of the event	10/16/25
Time	Local device time in HH:MM:SS	06:01:05
Category	Event category. Possible values: `USB`, `LINK`, `IO`	LINK
Interface	Affected interface. LINK: `ETHx`. IO: `IO_VPNKEY` or `IO_CUT`. USB: empty in the VIEW UI	ETH1
Operation	State change of the interface. LINK/IO: `UP` or `DOWN`. USB: `add` or `remove`	UP

The VIEW web interface only reformats the audit.log data and labels the columns. It does not add extra interpretation. All detailed information, including mapping to physical interfaces, comes from the audit.log fields described below.

Explanation of the raw audit.log entry

The raw `audit.log` contains `type=USER_DEVICE` records written via `audit_log_user_message()`. Only USER_DEVICE specific fields are described here. Standard auditd fields are filled by auditd itself; for details refer to the official auditd documentation.

USER_DEVICE fields in audit.log

Field	Description	Example
type	Audit event type	USER_DEVICE
audit(…)	Audit timestamp and sequence in the form `audit(epoch.sec:seq)`	1760587…:119
pid	PID of the process that wrote the event	7496
uid	Effective UID of the process	0
auid	Audit login UID	4294967295
ses	Audit session ID	4294967295
msg.category	Category of the interface event	LINK / USB / IO
msg.operation	Operation or state change	UP / DOWN / add / remove

© 2025 ADS-TEC Industrial IT GmbH

Field	Description	Example
msg.interface	Interface identifier. LINK: `ETHx`. IO: `IO_VPNKEY` or `IO_CUT`. Not present for USB	ETH1
msg.vendor	USB vendor ID (only for category USB)	23a9
msg.product	USB product ID (only for category USB)	ef18
msg.exe	Executable that logged the event (`audit-hw-log`, `audit-log-netlink`, or `ubus-gpiocutalarmd`)	/usr/bin/audit-…
msg.hostname	Not filled on this platform	?
msg.addr	Not filled on this platform	?
msg.terminal	Not filled on this platform	?
msg.res	Result string. Set to `success` by default because no context dependent check is done	success
UID	Text form of `uid`	root
AUID	Text form of `auid`	unset

For SOC or SIEM integration parse the key/value pairs inside `msg='…'`. Treat a missing `interface` field for USB events as expected behaviour. A common normalisation is to combine `category`, `operation`, and `interface` into standard fields, for example `device.interface=ETH1` and `action=link_up` for a LINK/UP event.

When the event is generated

The device generates a USER_DEVICE event when a monitored external interface changes state. The same behaviour applies to IRF1000 and IRF3000, except that IRF1000 has no USB USER_DEVICE events.

Categories:

- USB: a USB device with vendor and product IDs is added or removed on the accessible USB port.
- LINK: an Ethernet port link goes UP or DOWN. This usually happens when a cable is plugged in or unplugged, but current firmware also logs software link changes from configuration.
- IO: the VPN-Key or CUT input signal goes UP or DOWN. During device boot a transition can be logged even if no external change occurred.

Built-in rate limits:

- LINK: at most 20 messages per day. This protects the audit log from excessive link flapping.
- IO (VPN-Key/CUT): at most 20 messages per hour.
- USB: no rate limit at this time. Very frequent USB plug/unplug therefore appears fully in the audit log.

Event producers on the device:

-

© 2025 ADS-TEC Industrial IT GmbH

- USB events are generated by `audit-hw-log`, which is triggered by a kernel uevent via a hotplug shell script.
- LINK events are generated by `audit-log-netlink`, which listens to link changes via a netlink socket.
- IO events are generated by `ubus-gpiocutalarmd`, which uses the Linux epoll subsystem to monitor GPIO inputs.

These user space programs write USER_DEVICE messages through the libaudit function `audit_log_user_message()`.

Sample audit.log entry

```
(1) type=USER_DEVICE msg=audit(1760508902.053:8): pid=5480 uid=0 auid=4294967295
    ses=4294967295 msg='category="IO" operation="UP" interface="IO_VPNKEY"
    exe="/usr/sbin/ubus-gpiocutalarmd" hostname=? addr=? terminal=?
    res=success'UID="root" AUID="unset"
(2) type=USER_DEVICE msg=audit(1760508905.215:9): pid=5480 uid=0 auid=4294967295
    ses=4294967295 msg='category="IO" operation="UP" interface="IO_CUT"
    exe="/usr/sbin/ubus-gpiocutalarmd" hostname=? addr=? terminal=?
    res=success'UID="root" AUID="unset"
(3) type=USER_DEVICE msg=audit(1760508907.624:8): pid=5480 uid=0 auid=4294967295
    ses=4294967295 msg='category="IO" operation="DOWN" interface="IO_VPNKEY"
    exe="/usr/sbin/ubus-gpiocutalarmd" hostname=? addr=? terminal=?
    res=success'UID="root" AUID="unset"
(4) type=USER_DEVICE msg=audit(1760508907.624:9): pid=5480 uid=0 auid=4294967295
    ses=4294967295 msg='category="IO" operation="DOWN" interface="IO_CUT"
    exe="/usr/sbin/ubus-gpiocutalarmd" hostname=? addr=? terminal=?
    res=success'UID="root" AUID="unset"
(5) type=USER_DEVICE msg=audit(1760587265.684:118): pid=7496 uid=0 auid=4294967295
    ses=4294967295 msg='category="LINK" operation="DOWN" interface="ETH1"
    exe="/usr/bin/audit-log-netlink" hostname=? addr=? terminal=?
    res=success'UID="root" AUID="unset"
(6) type=USER_DEVICE msg=audit(1760587268.032:119): pid=7496 uid=0 auid=4294967295
    ses=4294967295 msg='category="LINK" operation="UP" interface="ETH1"
    exe="/usr/bin/audit-log-netlink" hostname=? addr=? terminal=?
    res=success'UID="root" AUID="unset"
(7) type=USER_DEVICE msg=audit(1760587484.444:120): pid=13569 uid=0 auid=4294967295
    ses=4294967295 msg='category="USB" operation="add" vendor="23a9" product="ef18"
    exe="/usr/sbin/audit-hw-log" hostname=? addr=? terminal=? res=success'UID="root"
    AUID="unset"
(8) type=USER_DEVICE msg=audit(1760587485.148:121): pid=13643 uid=0 auid=4294967295
    ses=4294967295 msg='category="USB" operation="remove" vendor="23a9" product="ef18"
    exe="/usr/sbin/audit-hw-log" hostname=? addr=? terminal=? res=success'UID="root"
    AUID="unset"
```

Packet Filter

Introduction

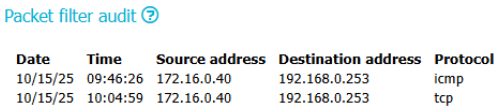
Packet filter audit events on IRF1000 and IRF3000 indicate that forwarded Layer-3 traffic matched a packet filter rule with the **Audit** option enabled. The event records the IPv4 source address, destination address, and protocol of the forwarded packet and makes this visible both in the web interface and in `audit.log`.

These events help you confirm expected communication paths in OT networks and detect unexpected peers or protocols that reach critical assets. Do not use this event type for full traffic accounting, because

© 2025 ADS-TEC Industrial IT GmbH

logging is rate-limited to one message per minute per audited rule. For security monitoring, correlate these sampled events with your firewall rules and with dedicated flow or packet monitoring where required.

Webinterface View



The VIEW web interface shows these events in the **Packet filter audit** section. The web view reads `NETFILTER_PKT` entries from `audit.log`, extracts the relevant fields, and converts the numeric protocol value into a protocol name, for example `1` to `icmp` or `6` to `tcp`. Only forward-path hits are displayed. Rule names, ports, interfaces, and actions are not shown.

The table columns in the web interface have the following meaning:

Field	Description	Example
Date	Device local calendar date when the audit event was recorded.	`10/15/25`
Time	Device local time when the audit event was recorded.	`09:46:26`
Source address	IPv4 source address of the forwarded packet.	`172.16.0.40`
Destination address	IPv4 destination address of the forwarded packet.	`192.168.0.253`
Protocol	IP protocol name derived from the numeric `proto` field in the audit record.	`icmp` / `tcp`

Entries in this view are sampled because of the rate limit. If a large number of packets match the same audited rule, you still see at most one row per minute for that rule. For SIEM or SOC integration, use the raw `audit.log` file as the primary data source and treat the web view as a human-readable summary.

Explanation of the raw audit.log entry

Lines with `type=NETFILTER_PKT` in `audit.log` are created by the kernel netfilter audit integration via the iptables `audit` extension. Each line contains basic packet metadata. Rule names and explicit actions are not part of this record. Generic auditd fields behave as documented in the standard auditd manuals and are not described in detail here.

A typical `NETFILTER_PKT` record looks like this and contains the following components:

Field	Description	Example
type	Audit record type. For packet filter audit events this is `NETFILTER_PKT`.	`type=NETFILTER_PKT`
msg	Audit metadata with timestamp and record ID in auditd format.	`msg=audit(1760514386.800:94)`
mark	Netfilter mark value if present. The IRF does not change this by default.	`mark=0x0`
saddr	Source IPv4 address of the forwarded packet.	`saddr=172.16.0.40`

© 2025 ADS-TEC Industrial IT GmbH

Field	Description	Example
daddr	Destination IPv4 address of the forwarded packet.	`daddr=192.168.0.253`
proto	IP protocol number as assigned by IANA (for example 1 = ICMP, 6 = TCP, 17 = UDP).	`proto=1` / `proto=6`

For SIEM or SOC use, filter for `type=NETFILTER_PKT`, extract `saddr`, `daddr`, and `proto`, and map the protocol numbers to protocol names using the official IANA protocol list. Remember that these records are rate-limited to one message per minute per audited rule, so they represent sampled matches and not every packet that traverses the rule.

When the event is generated

A `NETFILTER_PKT` audit event is generated only under specific technical conditions on the IRF device.

1. A packet traverses the Linux netfilter **FORWARD** path on the IRF. This means the device forwards the packet between two networks; it is not traffic that terminates on the IRF itself.
2. The packet matches a configured Layer-3 packet filter rule where the **Audit** checkbox is enabled in the rule definition.
3. The packet is IPv4 Layer-3 traffic. The current implementation covers only forwarded IPv4 packets.

For such a rule the IRF creates an iptables chain and inserts a jump to the `AUDIT` target. When a packet enters this chain, the iptables `audit` extension sends packet metadata to the audit subsystem, which then writes a `NETFILTER_PKT` record to `audit.log`. Rule names are not included in the record. To protect the device and log storage from overload, the logging is rate-limited to at most one message per minute per audited rule, even if many packets match during that minute.

Sample audit.log entry

```
(1) type=NETFILTER_PKT msg=audit(1760514386.800:94): mark=0x0 saddr=172.16.0.40
    daddr=192.168.0.253 proto=1
(2) type=NETFILTER_PKT msg=audit(1760515499.024:96): mark=0x0 saddr=172.16.0.40
    daddr=192.168.0.253 proto=6
```

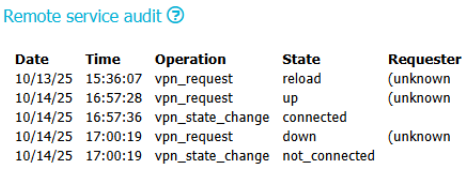
Remote Service

Introduction

CUSTOM_REMOTE_SERVICE audit events report Big-LinX VPN requests and VPN state changes on ADS-TEC IRF1000 and IRF3000 devices. They document when the device requests to bring the VPN tunnel up or down and when the tunnel actually becomes connected or not connected. This supports change tracking, incident analysis, and SOC correlation for remote service activity in OT networks. Watch for unexpected **connected** or **up** events outside maintenance windows, rapid up/down toggling, or frequent **reload** states, because these patterns can indicate misuse, misconfiguration, or instability. Event emission is rate-limited to five messages per hour, so always correlate with other logs when VPN activity is very frequent.

Webinterface view

© 2025 ADS-TEC Industrial IT GmbH


Remote service audit

Date	Time	Operation	State	Requester
10/13/25	15:36:07	vpn_request	reload	(unknown
10/14/25	16:57:28	vpn_request	up	(unknown
10/14/25	16:57:36	vpn_state_change	connected	
10/14/25	17:00:19	vpn_request	down	(unknown
10/14/25	17:00:19	vpn_state_change	not_connected	

The VIEW web interface shows these entries under **Remote service audit** in the **Remote Service - VPN** section. It lists the parsed content of the CUSTOM_REMOTE_SERVICE events in column form and hides internal auditd metadata.

Field	Description	Example
Date	Local device date derived from the audit timestamp.	10/14/25
Time	Local device time derived from the audit timestamp.	16:57:36
Operation	Operation from the audit message. Values: `vpn_request`, `vpn_state_change`.	vpn_state_change
State	Requested or resulting VPN state. Values: `reload`, `up`, `down`, `connected`, `not_connected`.	connected
Requester	Source of the request. Currently not set and displayed as `(unknown)`.	(unknown)

The list matches the example in the screenshot: `vpn_request` with states `reload`, `up`, and `down`, and `vpn_state_change` with states `connected` and `not_connected`, with the requester column always showing `(unknown)` at the moment. The web view focuses on readability for operators and does not display internal auditd fields such as `pid`, `uid`, `auid`, or `ses`.

Explanation of the raw audit.log entries

Each record in `/var/log/audit/audit.log` is a standard auditd line with type `CUSTOM_REMOTE_SERVICE`. The key data for SIEM or SOC correlation is inside the `msg='…'` part of the record. Generic auditd fields are only summarized here; for full semantics please refer to the official auditd documentation.

Field	Description	Example
type	Audit record type. Identifies this as a remote service VPN event.	CUSTOM_REMOTE_SERVICE
msg=audit(ts:serial)	Audit header with seconds.microseconds since epoch and a serial ID.	audit(1760453856.061:164)
pid	Process ID of the emitter (`statusd`).	6964
uid	Effective UID of the emitting process.	0
auid	Audit user ID (login UID).	4294967295
ses	Audit session ID.	4294967295

© 2025 ADS-TEC Industrial IT GmbH

Field	Description	Example
msg.operation	Operation field. Maps to VIEW " Operation". Values: `vpn_request`, `vpn_state_change`.	vpn_request
msg.state	State field. Maps to VIEW "State". Values: `reload`, `up`, `down`, `connected`, `not_connected`.	connected
msg.requester	Source of the request. Currently not set; appears as `(unknown)`. Maps to VIEW "Requester".	(unknown)
msg.exe	Executable that emitted the event.	/usr/sbin/statusd
hostname	Hostname field from auditd. Not used here.	?
addr	Address field from auditd. Not used here.	-
terminal	Terminal field from auditd. Not used here.	?
res	Result flag from the emitter. For these events usually `success`.	success
UID	Textual UID resolved by auditd.	root
AUID	Textual AUID resolved by auditd.	unset

SOC guidance:

- Alert on `vpn_state_change` with state `connected` outside approved maintenance or service windows.
- Alert when a `vpn_request` with state `up` is not followed by a `vpn_state_change` with state `connected` within the expected time window, as this can indicate connection problems.
- Watch for repeated `reload` states or fast up/down cycles, which can indicate configuration issues or unstable links.
- Correlate VPN audit events with administrator logins, change tickets, and other system logs. Consider the five-events-per-hour throttle when building correlation rules and counters.

When the event is generated

The daemon `statusd` generates CUSTOM_REMOTE_SERVICE events by using the libaudit function `audit_log_user_message()`. It controls Big-LinX VPN connections and writes an audit record whenever the device requests a VPN action or observes a VPN state change.

Operations:

- **vpn_request**: the device requested a change of the VPN state.
- **vpn_state_change**: the observed VPN state changed.

States:

-
  - **reload**: the device fetches current configuration and task data from the Big-LinX servers.
  - **up**: the device requests to establish the VPN tunnel.

© 2025 ADS-TEC Industrial IT GmbH

- **down**: the device requests to tear down the VPN tunnel.
- **connected**: the VPN tunnel is established and connected to Big-LinX.
- **not_connected**: the VPN tunnel is down and no longer connected.

The `requester` field is currently not filled and is shown as `(unknown)` in all outputs. Emission is throttled to five CUSTOM_REMOTE_SERVICE messages per hour to protect the audit.log file. Very bursty VPN activity may therefore not produce a one-to-one event stream in the audit log and must be correlated with other sources such as system logs or VPN diagnostics.

Sample audit.log entry

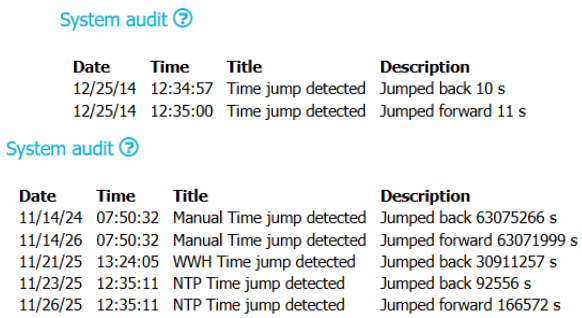
(1) type=CUSTOM_REMOTE_SERVICE msg=audit(1760362567.867:32): pid=7843 uid=0 auid=4294967295 ses=4294967295 msg='operation="vpn_request" state="reload" requester=(unknown) exe="/usr/sbin/statusd" hostname=? addr=- terminal=? res=success'UID="root" AUID="unset" (2) type=CUSTOM_REMOTE_SERVICE msg=audit(1760453848.569:163): pid=6964 uid=0 auid=4294967295 ses=4294967295 msg='operation="vpn_request" state="up" requester=(unknown) exe="/usr/sbin/statusd" hostname=? addr=- terminal=? res=success'UID="root" AUID="unset" (3) type=CUSTOM_REMOTE_SERVICE msg=audit(1760453856.061:164): pid=6964 uid=0 auid=4294967295 ses=4294967295 msg='operation="vpn_state_change" state="connected" exe="/usr/sbin/statusd" hostname=? addr=- terminal=? res=success'UID="root" AUID="unset" (4) type=CUSTOM_REMOTE_SERVICE msg=audit(1760454019.474:169): pid=6964 uid=0 auid=4294967295 ses=4294967295 msg='operation="vpn_request" state="down" requester=(unknown) exe="/usr/sbin/statusd" hostname=? addr=- terminal=? res=success'UID="root" AUID="unset" (5) type=CUSTOM_REMOTE_SERVICE msg=audit(1760454019.674:170): pid=6964 uid=0 auid=4294967295 ses=4294967295 msg='operation="vpn_state_change" state="not_connected" exe="/usr/sbin/statusd" hostname=? addr=- terminal=? res=success'UID="root" AUID="unset"

System

Introduction

A `CUSTOM_TIME_JUMP` event reports that the system time of the IRF1000 or IRF3000 has changed abruptly. These events make it possible to detect manual clock manipulation, incorrect NTP synchronisation, or malfunction of the time management services. For OT and industrial automation environments, stable and trusted timestamps are critical, because they influence log correlation, incident reconstruction, certificate validation and scheduled tasks. If your SIEM or SOC sees unexpected or frequent time jump events, you should investigate for configuration errors, failing NTP servers or potential attempts to hide activities by changing the device time. The device limits the creation of `CUSTOM_TIME_JUMP` audit entries to 4 messages per service per hour to avoid uncontrolled growth of the audit log file, so extremely noisy time changes may not be fully visible in the log.

Webinterface View



© 2025 ADS-TEC Industrial IT GmbH

In the VIEW web interface, `CUSTOM_TIME_JUMP` audit entries appear in the **System audit** section. The events are shown as simple rows in a table. The screenshot example shows entries such as:

- `Title`: `Time jump detected`, `Manual Time jump detected`, `WWH Time jump detected`, `NTP Time jump detected`
- `Description`: `Jumped back 10 s`, `Jumped forward 11 s`, or larger values like `Jumped back 63075266 s`

The web interface does not show all raw `audit.log` fields. It presents only the most important information for operators.

Fields in the System audit table

Web field	Description	Example value
Date	Local calendar date when the time jump was logged.	`11/23/25`
Time	Local time of the event, derived from the audit timestamp at log time.	`12:35:11`
Title	Short text from the `title` field in the audit message. It identifies the source of the time jump (manual, WWH, NTP).	`NTP Time jump detected`
Description	Detailed text from the `description` field in the audit message. It states the direction of the change and how many seconds the clock moved.	`Jumped back 92556 s`

Mapping between web view and raw audit data:

- **Title** in the table is the value of `title="..."` inside the `msg` payload of the audit record.
- **Description** in the table is the value of `description="..."` inside the `msg` payload.
- **Date** and **Time** are derived from the numeric audit timestamp that appears after `audit(` in the raw log line.

The web view does not display fields such as `pid`, `uid`, `exe` or `res`. To analyse those, you must use the raw `audit.log` file or your SIEM, which can parse the full message.

Explanation of the raw audit.log entry

The raw `audit.log` records for time jumps use the custom type `CUSTOM_TIME_JUMP`. Each record is written by `auditd` when one of the time-management services reports a detected jump.

Example structure

A typical entry looks like this (line breaks added only for readability):

```
type=CUSTOM_TIME_JUMP msg=audit(1763897711.524:66): pid=17180 uid=323 auid=4294967295 \
  ses=4294967295 msg='title="NTP Time jump detected" description="Jumped back 92556 s" \
  exe="/usr/sbin/chronyd" hostname=? addr=? terminal=? res=success'UID="chrony" AUID="unset"
```

Relevant fields for `CUSTOM_TIME_JUMP`

© 2025 ADS-TEC Industrial IT GmbH

Field name	Description	Example value
`type`	Audit record type. For these events it is always `CUSTOM_TIME_JUMP`.	`CUSTOM_TIME_JUMP`
`msg` (header)	Standard audit header containing the event timestamp and a sequence id.	`audit(1763897711.524:66)`
`pid`	Process ID of the service that reported the time jump.	`17180`
`uid`	Effective Unix user ID of that process.	`323`
`auid`	Audit user ID set by auditd. For system daemons it is usually the unset value.	`4294967295`
`ses`	Audit session ID. Usually unset for system daemons.	`4294967295`
`msg` (payload)	Free-form text payload with key–value pairs like `title=`, `description=`, `exe=`, `hostname=`, etc.	`msg='title="..." description="..." ...'`
`title`	Summary of the event. Indicates whether the jump was manual, WWH or NTP-based.	`NTP Time jump detected`
`description`	Detailed description with direction and magnitude of the jump in seconds.	`Jumped back 92556 s`
`exe`	Absolute path of the executable that detected and reported the time jump.	`/usr/sbin/chronyd`
`hostname`	Hostname field from auditd for this event. On the appliance it is not filled and appears as `?`.	?
`addr`	Remote address associated with the event. Not used for time jumps and appears as `?`.	?
`terminal`	Terminal identifier. For daemon-generated events this is not set and appears as `?`.	?
`res`	Result flag from auditd. For these events it is always set to `success`, because the detection itself succeeded.	`success`
`UID`	Text representation of `uid` as a name, provided by auditd.	`chrony`
`AUID`	Text representation of `auid`. For system daemons this is typically `unset`.	`unset`

© 2025 ADS-TEC Industrial IT GmbH

For SIEM or SOC integration, you usually parse on `type=CUSTOM_TIME_JUMP`, then extract `title`, `description`, `exe`, and the numeric timestamp from the header `msg=audit(...)`. Use the magnitude and direction of the jump to create alerts for suspicious manual changes or abnormal NTP or WWH corrections.

When the event is generated

A `CUSTOM_TIME_JUMP` event is generated whenever the system detects that the current system time has jumped forward or backward by a defined threshold.

The following services can generate this event by calling `audit_log_user_message()` from `libaudit`:

- `systemd` for all manual time changes
- `chronyd` (NTP client)
- `wwhd` (WWH service)
- `wwhngd` (WWH next-generation service)

Generation rules:

- Manual time changes via `systemd` are always logged, independent of the time difference.
- `chronyd` logs a `CUSTOM_TIME_JUMP` when NTP synchronisation changes the time by **10 seconds or more**.
- `wwhd` and `wwhngd` log a `CUSTOM_TIME_JUMP` when WWH time adjustment changes the time by **60 seconds or more**.
- Both forward and backward corrections are logged, with the direction and number of seconds shown in the `description` field.
- Each involved service is limited to **4** `CUSTOM_TIME_JUMP` **events per hour** to protect the size of `audit.log`.

Use in security monitoring:

- Investigate any manual time jump on productive systems.
- Correlate time jump events with configuration changes, software updates or security incidents.
- Check for unstable NTP or WWH time sources when many automatic time jumps appear.

Sample audit.log entry

```
(1) type=CUSTOM_TIME_JUMP msg=audit(1731567032.156:14): pid=6122 uid=0 auid=4294967295
    ses=4294967295 msg='title="Manual Time jump detected" description="Jumped back
    63075266 s" exe="/usr/sbin/systemd" hostname=? addr=? terminal=?
    res=success'UID="root" AUID="unset"
(2) type=CUSTOM_TIME_JUMP msg=audit(1794639032.140:16): pid=6122 uid=0 auid=4294967295
    ses=4294967295 msg='title="Manual Time jump detected" description="Jumped forward
    63071999 s" exe="/usr/sbin/systemd" hostname=? addr=? terminal=?
    res=success'UID="root" AUID="unset"
(3) type=CUSTOM_TIME_JUMP msg=audit(1763727845.000:18): pid=8340 uid=0 auid=4294967295
    ses=4294967295 msg='title="WWH Time jump detected" description="Jumped back
    30911257 s" exe="/usr/sbin/wwhd" hostname=? addr=? terminal=?
    res=success'UID="root" AUID="unset"
(4) type=CUSTOM_TIME_JUMP msg=audit(1763897711.524:66): pid=17180 uid=323
    auid=4294967295 ses=4294967295 msg='title="NTP Time jump detected"
    description="Jumped back 92556 s" exe="/usr/sbin/chronyd" hostname=? addr=?
```


© 2025 ADS-TEC Industrial IT GmbH

```
    terminal=? res=success'UID="chrony" AUID="unset"
(5) type=CUSTOM_TIME_JUMP msg=audit(1764156911.372:67): pid=22722 uid=323
    auid=4294967295 ses=4294967295 msg='title="NTP Time jump detected"
    description="Jumped forward 166572 s" exe="/usr/sbin/chronyd" hostname=? addr=?
    terminal=? res=success'UID="chrony" AUID="unset"
```

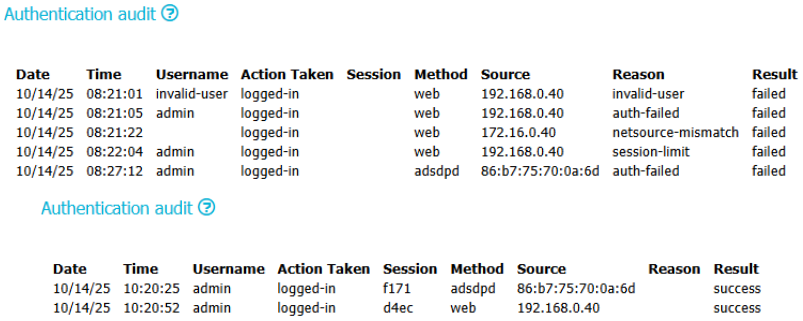
Authentication

Authentication - Login

Introduction

Authentication - Login events record every attempt to access an IRF1000/IRF3000 device via the web UI, JSON-RPC, or Adsdpd. The event helps you demonstrate who tried to log in, from where, and with which result. It also covers explicit logouts and sessions that ended because of an idle timeout. Watch for repeated `auth-failed`, `invalid-user`, `netsource-mismatch`, and `session-limit` reasons. Investigate source changes for the same session hash and restrict access to trusted IP or MAC addresses. Use strong passwords and keep the login-ban feature active to slow down brute-force attempts.

Webinterface View



The VIEW web interface shows Authentication audit entries in a single table. This table combines `USER_LOGIN`, `USER_LOGOUT`, and `USER_END` events into one view. The **Session** column displays the first four characters of the session hash. **Method** shows `web` (including JSON-RPC) or `adsdpd`. During an active login-ban period, additional failed login attempts are not added to this table.

For each row, the columns are mapped as follows:

Field	Description	Example
Date	Local date when the event was written	10/14/25
Time	Local time when the event was written	08:21:05
Username	Account name from the attempt; for failed logins this is the attempted name	admin
Action Taken	High-level action derived from the audit `type` and `op` fields	logged-in; logged-out; idle-timeout
Session	First four characters of the session hash (maps to `terminal` in the log)	d4ec
Method	Login interface that created the event	web; adsdpd

© 2025 ADS-TEC Industrial IT GmbH

Field	Description	Example
Source	Origin address of the request. IP address for web, MAC address for adsdpd	192.168.0.40; 86:b7:75:70:0a:6d
Reason	Reason string. For failures it shows the cause from the log (for example `invalid-user`); for successful logins the UI displays `success`	invalid-user; auth-failed; netsource-mismatch; session-limit; success
Result	Outcome of the action	success; failed

Explanation of the raw audit.log entry

Each Authentication - Login related line in `audit.log` is a Linux audit record written by `rpcd`. For SIEM integration, focus mainly on the structured fields inside `msg='…'`. Generic auditd-internal fields should be interpreted according to the official auditd documentation.

Field	Description	Example
type	Audit record type. For this view: `USER_LOGIN`, `USER_LOGOUT`, `USER_END`	USER_LOGIN
msg=audit(ts:seq)	Auditd timestamp and sequence number	audit(1760430052.348:23)
pid	Process ID that emitted the event (auditd-internal)	5447
uid	Real UID (auditd-internal)	0
auid	Audit UID (auditd-internal)	4294967295
ses	Audit session ID (auditd-internal)	4294967295
msg='op=…'	Structured payload from `rpcd`, containing the fields below	op=user-login …
op	Action keyword. Examples: `user-login`, `user-session-limit`, `session-access-mismatch-net_src`	user-login
acct	Username used in the attempt	admin; invalid-user
exe	Source component of the request	web; adsdpd
hostname	Not used by this product	?
addr	Source address. IP address for web, MAC address for adsdpd	192.168.0.40; 86:b7:…
terminal	First four characters of the SHA-256 hash of the session ID	d4ec
id / ID	Session identifier if present. May be `unset` in some records	4294967295; unset
reason	Reason string on failures	invalid-user; auth-failed; netsource-mismatch; session-limit
res	Result of the action	success; failed

SIEM tips:

© 2025 ADS-TEC Industrial IT GmbH

- Parse at least `type`, `op`, `acct`, `exe`, `addr`, `terminal`, `reason`, and `res`.
- Alert on many `auth-failed` or `invalid-user` results from the same source, on any `netsource-mismatch`, and on surges of `session-limit` events per user.
- Remember that during a login-ban, additional failed attempts are not logged; correlate with other lockout or ban telemetry if available.

When the event is generated

The device generates Authentication - Login related audit records whenever the rpcd process authenticates a user or ends a session.

- **type=USER_LOGIN** is emitted on each authentication attempt via web (including JSON-RPC) or Adsdpd. On success `res=success` and the `reason` field in the raw log is empty. On failure `res=failed` and `reason` contains the cause, for example `invalid-user`, `auth-failed`, `netsource-mismatch`, or `session-limit`. During an active login-ban period, additional failed attempts are not logged.
- **type=USER_LOGOUT** is emitted when a user explicitly logs out from a session.
- **type=USER_END** is emitted when the device terminates a session because of an idle timeout.

The rpcd process creates and tracks sessions and calls `audit_log_acct_message()` from libaudit to write these records into `audit.log`. For web and JSON-RPC access the source address is an IP address. For Adsdpd the source address is a MAC address because it is a Layer-2 request. The `terminal` value is the first four characters of the SHA-256 hash of the session ID.

Sample audit.log entry

(1) type=USER_LOGIN msg=audit(1760430025.988:20): pid=5447 uid=0 auid=4294967295 ses=4294967295 msg='op=user-login acct="admin" exe="adsdpd" hostname=? addr=86:b7:75:70:0a:6d terminal=f171 reason= res=success'UID="root" AUID="unset" (2) type=USER_LOGIN msg=audit(1760430052.348:23): pid=5447 uid=0 auid=4294967295 ses=4294967295 msg='op=user-login acct="admin" exe="web" hostname=? addr=192.168.0.40 terminal=d4ec reason= res=success'UID="root" AUID="unset"

(1) type=USER_LOGIN msg=audit(1760422861.352:32): pid=5446 uid=0 auid=4294967295 ses=4294967295 msg='op=user-login acct="invalid-user" exe="web" hostname=? addr=192.168.0.40 terminal=? reason="invalid-user" res=failed'^]UID="root" AUID="unset" (2) type=USER_LOGIN msg=audit(1760422865.716:33): pid=5446 uid=0 auid=4294967295 ses=4294967295 msg='op=user-login acct="admin" exe="web" hostname=? addr=192.168.0.40 terminal=? reason="auth-failed" res=failed'^]UID="root" AUID="unset" (3) type=USER_LOGIN msg=audit(1760422882.944:35): pid=5446 uid=0 auid=4294967295 ses=4294967295 msg='op=session-access-mismatch-net_src id=4294967295 exe="web" hostname=? addr=172.16.0.40 terminal=? reason="netsource-mismatch" res=failed'^]UID="root" AUID="unset" ID="unset" (4) type=USER_LOGIN msg=audit(1760422924.180:55): pid=5446 uid=0 auid=4294967295 ses=4294967295 msg='op=user-session-limit acct="admin" exe="web" hostname=? addr=192.168.0.40 terminal=? reason="session-limit" res=failed'^]UID="root" AUID="unset" (5) type=USER_LOGIN msg=audit(1760423232.724:79): pid=5446 uid=0 auid=4294967295 ses=4294967295 msg='op=user-login acct="admin" exe="adsdpd" hostname=? addr=86:b7:75:70:0a:6d terminal=? reason="auth-failed" res=failed'UID="root" AUID="unset"

Authentication - Session Timeouts

Introduction

© 2025 ADS-TEC Industrial IT GmbH

USER_END records that an authenticated session ended because the configured idle timeout expired. It applies to Web UI and JSON-RPC API sessions on ADS-TEC Industrial IT IRF1000 and IRF3000. The default timeout is 300 seconds. Any interaction with the Web UI resets the timer. Any valid JSON-RPC request resets the timer.

Use this event to verify that unattended sessions are closed automatically. Investigate frequent USER_END events for privileged accounts, because they can indicate open operator consoles that are not logged out. Check the Source value for unknown IP addresses (web) or MAC addresses (adsdpd). For OT networks, configure a strict timeout on `/priv/priv.php?id=USERS_SECURITYSETTINGS`.

Webinterface View

The Authentication audit view in the current VIEW web interface displays authentication-related audit entries in one table. It includes successful and failed logins (USER_LOGIN), explicit logouts (USER_LOGOUT), and idle timeouts that end a session (USER_END).

During an active login-ban period, failed login attempts are not logged, so you may see time gaps between visible failed logins. The Session column displays the first four characters of the session hash. Method shows `web` (including JSON-RPC access) or `adsdpd`. Source is the origin of the request: an IP address for web access or a MAC address for adsdpd service access.

The columns are mapped as follows:

Field	Description	Example
Date	Local device date of the event	10/14/25
Time	Local device time of the event	10:26:31
Username	Account name that owned the session	admin
Action Taken	Normalized action text. For a timeout it shows `ended-session`	ended-session
Session	First four characters of the session hash	d4ec
Method	Login method. `web` includes JSON-RPC; `adsdpd` is the device service	web
Source	Origin address. IP for web, MAC for adsdpd. May be empty if unknown	192.0.2.15
Reason	Error reason text if available. Empty on a normal timeout	
Result	Outcome of the operation (`success` or `failure`)	success

Explanation of the raw audit.log entry

Each idle timeout writes a USER_END record into `audit.log`. Standard auditd fields such as `pid`, `uid`, `auid`, `ses`, `UID`, and `AUID` are populated by auditd itself and follow the upstream auditd documentation.

The example entry looks like this in the log and is wrapped here for readability:

© 2025 ADS-TEC Industrial IT GmbH

```
type=USER_END msg=audit(1760430391.184:24): pid=5447 uid=0 auid=4294967295
ses=4294967295 msg='op=user-session-timeout acct="admin" exe="web" hostname=?
addr=? terminal=d4ec reason= res=success'^]UID="root" AUID="unset"
```

The fields are interpreted as follows:

Field	Description	Example
type	Audit message type	USER_END
msg	Audit timestamp and record id (`audit(…:…)`)	audit(1760430391.184:24)
pid	Process ID of the emitter (rpcd)	5447
uid	Effective user ID of the emitter	0
auid	Audit user ID (unset here)	4294967295
ses	Numeric audit session ID	4294967295
op	Operation string inside `msg='…'`	user-session-timeout
acct	Account name inside `msg='…'`	admin
exe	Source component inside `msg='…'`. `web` includes JSON-RPC; `adsdpd` is service access	web
hostname	Not used; remains ?	?
addr	Source address. IP for web, MAC for adsdpd	?
terminal	First four characters of the session SHA-256 hash	d4ec
reason	Reason text on errors. Empty on a normal timeout	
res	Result of the operation (`success` or `failure`)	success
UID	Text copy of `uid`	root
AUID	Text copy of `auid`	unset

For SIEM or SOC integration, parse the key–value pairs inside the quoted `msg='…'` section to extract `op`, `acct`, `exe`, `addr`, `terminal`, `reason`, and `res`. To correlate authentication activity, combine `acct` (username), `terminal` (session hash prefix), and `exe` (method). A normal idle timeout has `res=success` and an empty `reason`.

When the event is generated

USER_END is generated when a logged-in session reaches the configured idle timeout and the rpcd service terminates the session.

The rpcd process calls its `destroy()` function, which uses `audit_log_acct_message()` from libaudit to write a USER_END record to `audit.log`. This applies to both Web UI sessions and JSON-RPC API sessions; JSON-RPC sessions appear as method `web` in the logs.

USER_END is different from USER_LOGOUT, which records a user-initiated logout. A normal timeout has `op=user-session-timeout`, a valid `acct` username, `exe=web` or `exe=adsdpd`, and `res=success`.

Sample audit.log entry

© 2025 ADS-TEC Industrial IT GmbH

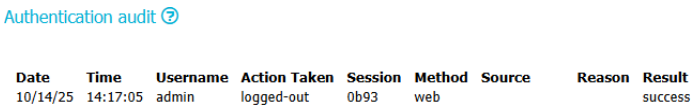
```
type=USER_END msg=audit(1760430391.184:24): pid=5447 uid=0 auid=4294967295
              ses=4294967295 msg='op=user-session-timeout acct="admin" exe="web"
              hostname=? addr=? terminal=d4ec reason= res=success'^]UID="root"
              AUID="unset"
```

Authentication - Logout

Introduction

A USER_LOGOUT event records that a signed-in user ended a session on the ADS-TEC IRF1000/IRF3000 device. It confirms that the appliance removed the session token and that the web interface or service acknowledged the logout request. Monitor these events to detect unusual behavior, for example rapid logouts across many accounts, administrator logouts outside maintenance windows, or logouts coming from unexpected sources. If you see logouts without a preceding successful login for the same user, investigate possible session hijacking, credential sharing, or misuse of shared accounts.

Webinterface View



The current VIEW page lists authentication events in the **Authentication audit** table. It shows USER_LOGIN, USER_LOGOUT, and USER_END entries together. During an active login-ban period, failed login attempts are not logged and therefore do not appear in this view. For a USER_LOGOUT the row is displayed as follows.

Web column	Description	Example
Date	Local device date of the event	10/14/25
Time	Local device time of the event	14:17:05
Username	Account name whose session was ended	admin
Action Taken	Human-readable action text	logged-out
Session	First four characters of the session hash (SHA-256 of session ID)	0b93
Method	Origin of the request: `web` (incl. JSON-RPC) or `adsdpd`	web
Source	Request source. IP address for `web`, MAC address for `adsdpd`. Can be empty if unknown	—
Reason	Error text if the logout failed. Empty on success	—
Result	Outcome of the operation: `success` or `failure`	success

Use the **Session** column to correlate the logout with the matching login that has the same hash prefix. Use **Method** and **Source** to verify that the logout came from the expected client and access path.

Explanation of the raw audit.log entry

© 2025 ADS-TEC Industrial IT GmbH

A `USER_LOGOUT` entry in `audit.log` contains standard auditd header fields and an embedded message payload with logout details. The device does not change auditd's default behavior; for generic field semantics refer to the official auditd documentation.

Field	Description	Example
type	Audit message type	USER_LOGOUT
msg (header)	Timestamp and record ID from auditd	audit(1760444225.960:84)
pid	Process ID writing the record (`rpcd`)	5447
uid	Effective UID of the process	0
auid	Audit UID (set by auditd)	4294967295
ses	Audit session ID (set by auditd)	4294967295
msg.op	Operation label	user-logout
msg.acct	Username of the account	admin
msg.exe	Source component: `web` or `adsdpd`	web
msg.hostname	Hostname. Not used on this device	?
msg.addr	Source address. IP (web) or MAC (adsdpd). May be unknown	?
msg.terminal	First four characters of the SHA-256 hash of the session ID	0b93
msg.reason	Error reason on failure; empty on success	
msg.res	Result of the operation: `success` or `failure`	success
UID	Human-readable UID label from auditd	root
AUID	Human-readable AUID label from auditd	unset

The `msg.terminal` value matches the **Session** column in the web view. The `msg.exe` and `msg.addr` fields correspond to the **Method** and **Source** columns and can be used in a SIEM or SOC to verify which access path and client performed the logout.

When the event is generated

The `USER_LOGOUT` event is generated when the `rpcd` process handles a Logout action and calls the `audit_log_acct_message()` function from libaudit. This creates an audit record that is written to `audit.log`.

On this device it means that the user actively ended the session, usually by pressing the **Logout** button in the web interface, or that a component requested active removal of a session. The `exe` field shows the method: `web` (includes JSON-RPC calls made by the GUI) or `adsdpd`. The `terminal` field contains the first four characters of the SHA-256 hash of the session ID. The `addr` field is the request source: for `web` this is an IP address, for `adsdpd` this can be a MAC address. The `res` field reports `success` or `failure`. The `reason` field contains an error reason on failure and is empty on success. The `hostname` field is not populated on this device. Standard auditd fields (`pid`, `uid`, `auid`, `ses`, `UID`, `AUID`) are set by auditd according to its defaults.

© 2025 ADS-TEC Industrial IT GmbH

Sample audit.log entry

```
type=USER_LOGOUT msg=audit(1760444225.960:84): pid=5447 uid=0 auid=4294967295
                 ses=4294967295 msg='op=user-logout acct="admin" exe="web" hostname=?
                 addr=? terminal=0b93 reason= res=success'UID="root" AUID="unset"
```


© 2025 ADS-TEC Industrial IT GmbH