

Subscription Editor Guide

The Subscription Editor is a built-in web UI for managing log subscriptions without editing JSON directly. It is available on essential tier and above at: <https://<dashboards-domain>/editor>

The editor is protected by the same Cognito authentication as OpenSearch Dashboards.

The screenshot displays the Subscription Editor interface. On the left, a sidebar titled 'Subscriptions' lists 13 subscriptions, each with a status (enabled/disabled) and a stream count. The main area shows 'Subscription 1' with fields for Log Group Name (regex), Retention (days), Subscribe (enabled/managed), and Account ID. Below this, the 'Streams' section shows 'Stream 1' with fields for Log Stream Name (regex), Index, Pipeline, Targets (opensearch, datalake), Pattern Mode, and Metadata (service, vendor, purpose). The interface includes a top navigation bar with buttons like Reload, Undo, Versions, Preview, Diff, Save, Sync, and a user profile icon. The bottom of the sidebar has links for Pattern Rules, Reports, Accounts, and FAQ.

Subscriptions

- 1. /%stackname%/nginx/access enabled · 1 stream
- 2. /%stackname%/nginx/error enabled · 1 stream
- 3. /%stackname%/nginx/cloud-init enabled · 1 stream
- 4. /%stackname%/nginx/system enabled · 1 stream
- 5. /aws/lambda/%stackname%-log-sampl... disabled · 1 stream
- 6. /aws/lambda/%stackname%-log-sampl... disabled · 1 stream
- 7. /aws/lambda/loggenerator enabled · 1 stream
- 8. /%stackname%/sample/lambda enabled · 1 stream
- 9. /%stackname%/sample/json enabled · 1 stream
- 10. /%stackname%/sample/nginx enabled · 1 stream
- 11. /%stackname%/sample/syslog enabled · 1 stream
- 12. /%stackname%/sample/tomcat enabled · 1 stream
- 13. /%stackname%/sample/spring

Subscription 1

Log Group Name (regex): /%stackname%/nginx/access Matches: 14 Retention (days): 14

Subscribe: enabled — manage CloudWatch filters Account ID (enterprise only): * (all accounts)

Blank = local account. * = all. Select one or more accounts.

Streams

Stream 1

Log Stream Name (regex): . Index: app Pipeline: none (raw message)

Targets: ☒ opensearch ☒ datalake

Pattern Mode: off

Metadata:

service	log processor	×
vendor	perfware.cloud	×
purpose	https access logs	×

+ Add Metadata

+ Add Stream

Accessing the Editor

1. Open your Dashboards URL and log in with your [Cognito](#) credentials
2. Navigate to [/editor](#) (or replace [/_dashboards](#) with [/editor](#) in the URL)
3. The editor loads your current subscriptions from S3 automatically

Note: A landing page has been provided with links to the editor and OpenSearch (<https://<dashboards-domain>>).

Toolbar Buttons

The toolbar is fixed at the top of the page and contains:

- **Reload** – Reload subscriptions from S3, discarding any unsaved changes in the editor
- **Undo** – Revert all unsaved changes to the last saved version
- **Versions** – View S3 version history. Preview any version or rollback with one click
- **Preview** – Show the raw JSON that will be saved
- **Diff** – Show a diff of your changes against the last saved version (additions in green, removals in red)
- **Save** – Validate and save subscriptions to S3. The Lambda picks up changes on its next invocation (within 15 minutes), or click Sync Now.
- **Sync** – Trigger immediate subscription processing. Invokes the log processor Lambda to apply your saved changes right away instead of waiting for the next scheduled run (1–15 minutes).

Jump To Navigation

Below the toolbar is a dropdown that lists all subscriptions by log group name, stream count, and status. Select one and click **Go** to scroll directly to it.

Managing Subscriptions

Adding a Subscription

Click **+ Add Subscription**. A new empty subscription card appears with one default stream.

Subscription Fields

- **Log Group Name** – CloudWatch log group name or regex pattern (e.g. [/aws/lambda/sample.*](#)). Patterns are evaluated in order; first match wins.
- **Retention (days)** – CloudWatch Logs retention policy. Leave empty or set to [-1](#) to leave unchanged. Snaps to the nearest valid CloudWatch value.
- **Subscribe** – Controls CloudWatch subscription filter management:
 - [enabled](#) – Create and maintain a subscription filter local and cross-account - enterprise
 - [disabled](#) – Remove the subscription filter, stop ingesting events (local groups)
 - [unmanaged](#) – Does not manage filters; match events only. Used for cross-account log ingestion

where the remote account creates its own subscription filter.

- **Account ID** (enterprise only) – Restrict matching to events from specific AWS accounts in the same region. Options:

[Empty](#) or [\(local\)](#) – local account only

* – match any account

[Check Boxes](#) – select specific account

Subscription Actions

Each subscription card has buttons in the header:

- **Clone** – Duplicate this subscription (log group name is cleared) below. Useful for creating similar entries.
- **↑ / ↓** – Move the subscription up or down in the list. Order matters – first match wins.
- **Remove** – Delete this subscription (with confirmation)

Click the subscription in the list to manage:

The screenshot shows the 'Subscription 1' management interface. At the top, there are buttons for 'Clone', up/down arrows, and 'Remove'. Below these, the 'Log Group Name (regex)' field contains '/%stackname%/nginx/access'. To its right is a 'Matches' button and a 'Retention (days)' field set to '14'. A note below the retention field states '-1 or empty = no change'. The 'Subscribe' section shows a dropdown menu set to 'enabled — manage CloudWatch filters'. The 'Account ID (enterprise only)' field is set to '* (all accounts)'. A footer note reads: 'Blank = local account. * = all. Select one or more accounts.'

For enabled subscriptions you can retrieve the matching log groups by clicking [Matches](#), click [x](#) to close. For the example below all logs emitted by current and future lambda functions will be forwarded to the specified targets (OpenSearch and/or Athena Datalake). Right click a log in the list to Copy or Use as Log Group Name.

The screenshot shows the 'Subscription 24' management interface. It has the same top controls as Subscription 1. The 'Log Group Name (regex)' field contains '/aws/lambda/*'. The 'Matches' button is highlighted, and a list of 34 matches is displayed below it. The first match is '/aws/lambda/AIMonitor-billing', which is highlighted in blue. Other matches include '/aws/lambda/AIMonitor-LowerCaseFn-bQObzs5ceBMv', '/aws/lambda/AIMonitor-collector', and '/aws/lambda/AIMonitor-correlator'. The 'Retention (days)' field is set to '-1 = skip', with a green box around it and a note '-1 or empty = no change'. The 'Subscribe' section is set to 'enabled — manage CloudWatch filters'. The 'Account ID (enterprise only)' field is set to '(local account)'. The same footer note is present.

Managing Streams

Each subscription has one or more stream routing rules. Streams control how matched log events are processed.

Stream Fields

- **Log Stream Name** – Regex pattern to match CloudWatch log stream names. Default `.*` matches all streams. First match wins.
- **Index** – OpenSearch index type: `app` (application logs) or `audit` (audit/compliance logs). Each has independent retention policies.
- **Pipeline** – Each subscription stream can specify a pipeline name (e.g. "lambda", "nginx", "json"). When events are indexed into OpenSearch, the pipeline runs server-side and extracts structured fields from the raw message — request IDs, log levels, IP addresses, HTTP status codes, query times, etc. The original message is always preserved alongside the parsed fields (refer to [IngestPipelinesGuide.rtf](#) for more information).
- **Targets** – Where to send events:
 - `opensearch` – Index into OpenSearch for search and dashboards
 - `datalake` – Write to S3 datalake for Athena queries
- **Pattern Mode** – PII/pattern detection mode:
 - `off` – No pattern scanning (default)
 - `redact` – Replace matched patterns with a placeholder (e.g. `[SSN:REDACTED]`)
 - `filter` – Drop events that contain matched patterns
 - `tag` – Add metadata tags to events with detected patterns
- **Pattern Types** – Which patterns to scan for (shown when mode is not `off`). Built-in types: `ssn`, `credit_card`, `email`, `phone`, `aws_key`, `sql`. Custom patterns also appear here.
- **Metadata** – Key/value pairs added to every event in this stream. Searchable in OpenSearch and Athena. Use for tagging (e.g. `service=auth`, `environment=prod`).

Click **+ Add Stream** to add another routing rule. Click **Remove** on a stream to delete it.

Stream 1

Remove

Log Stream Name (regex)

Index

Pipeline

.*

app

none (raw message)

Targets

☒ opensearch
 ☒ datalake

Pattern Mode

off

Metadata

+ Add Metadata

+ Add Stream

Patterns Mode varies by tier, add metadata name value pairs to decorate your logs:

Log Stream Name (regex)
Index
Pipeline

.*
app
lambda

Targets
☒ opensearch ☒ datalake

Pattern Mode
tag

▼ Pattern Types (8 of 52 selected)

All
None

☐ api_key_generic ☒ aws_access_key ☐ aws_secret_key ☐ bearer_token ☐ cookie_header ☒ credit_card ☐ dob ☐ dob_refined ☐ drivers_license ☐ drivers_license_refined ☒ email ☐ generic_secret ☐ github_token ☒ iban ☐ iban_refined ☐ icd10 ☒ ipv4 ☐ ipv6 ☐ jwt_token ☐ mac_address ☐ mac_address_refined ☐ medicare_mbi ☐ mrn_generic ☐ nhs_number ☐ npa ☐ passport_us ☐ passport_us_refined ☐ phone_intl ☐ phone_intl_refined ☐ phone_us ☐ phone_us_refined ☐ private_key ☐ slack_token ☐ sql_alter ☐ sql_comment ☐ sql_delete ☒ sql_drop ☐ sql_exec ☒ sql_injection ☐ sql_insert ☐ sql_select ☐ sql_truncate ☐ sql_union ☐ sql_update ☒ ssn ☐ ssn_nondash ☐ swift_bic ☐ swift_bic_refined ☐ tomcat_config ☐ tomcat_debug ☐ tomcat_severe ☐ tomcat_warning

Metadata

service

log processor

×

vendor

pervware.cloud

×

purpose

testing lambda logs with lambda pipeline

×

+ Add Metadata

Custom Pattern Rules

The Custom Pattern Rules section (collapsed by default) lets you define your own regex patterns for detection. We've provided some examples – feel free to add your own.

Pattern 14: iban
↑ Top
Remove

Name	Category	Severity	Replacement
iban	FIN	High	[PII:IBAN]
	Optional grouping label for reports	Optional severity for reports	Text that replaces matches in redact mode. Use * to leave text intact (detect only).
			☐ AI Monitor Correlate
			Include in AI Monitor correlation analysis

Regex

(?m)\b[A-Z]{2}\d{2}[A-Z0-9]{4}\d{7}(?:[A-Z0-9]{0,16})\b

Test string...

Pattern rules are the most reusable part of the config, use Export|Import — share detection libraries (e.g. "HIPAA patterns", "PCI patterns", "custom SQL injection rules", "Errors", "Limit Violations", "Application Errors") with others on your team or with the community. Import allows you to 'replace' or 'merge':

Importing 49 pattern rule(s).

Type "replace" to replace all existing rules, or "merge" to merge
 (duplicates by name will be skipped):

merge

OK Cancel

Merged: 1 added, 48 skipped (duplicate names)

Pattern Rule Fields

- **Name** – Unique identifier for the pattern. Appears in the Pattern Types checkboxes on streams. If the name matches a built-in type, it overrides it.
- **Category** – Detection category (e.g. PHI, PII, FIN, ERRORS, EXCEPTIONS).
- **Severity** – Detection severity (e.g. -, Low, Medium, High, Critical).
- **Regex** – Regular expression to match. Validated in real-time.
- **AI Monitor Correlate** – When enabled these patterns are available to AI Monitor, its correlation engine searches the Log Processor's OpenSearch indices for log entries when an Anomaly is detected.
- **Replacement** – Text that replaces matches in redact mode (e.g. [PII:CUSTOM], * leave intact on redact).
- **Test** – Enter a test string to verify your regex matches. Use the Presets dropdown for common test values (SSN, credit card, email, SQL injection, etc.).

You can also use pattern detection to track, and alert on application behavior.

For example you may want to track significant errors coming from tomcat, e.g. error storms:

Pattern 57: tomcat_severe
↑ Top
Remove

Name	Category	Severity	Replacement
tomcat_severe	APP	High	*
Optional grouping label for reports		Optional severity for reports	Text that replaces matches in redact mode. Use * to leave text intact (detect only).
			☒ AI Monitor Correlate
			Include in AI Monitor correlation analysis

Regex

`^(\d{2}-[A-Za-z]{3}-\d{4})\s\d{2}:\d{2}:\d{3})\sSEVERE\s.*`

Test string...

Ditto for warnings:

Pattern 58: tomcat_warning
↑ Top
Remove

Name	Category	Severity	Replacement
tomcat_warning	APP	Medium	*
Optional grouping label for reports		Optional severity for reports	Text that replaces matches in redact mode. Use * to leave text intact (detect only).
			☒ AI Monitor Correlate
			Include in AI Monitor correlation analysis

Regex

`^(\d{2}-[A-Za-z]{3}-\d{4})\s\d{2}:\d{2}:\d{3})\sWARNING\s.*`

Test string...

Or if you happen to leave debugging on:

Pattern 56: tomcat_debug

↑ Top

Remove

Name

tomcat_debug

Category

APP

Optional grouping label for reports

Severity

High

Optional severity for reports

Replacement

*

Text that replaces matches in redact mode. Use * to leave text intact (detect only).

☐ AI Monitor Correlate

Include in AI Monitor correlation analysis

Regex

`^(\d{2}-[A-Za-z]{3}-\d{4})\s\d{2}:\d{2}:\d{3})\s(FINE|FINER|FINEST)\s.*`

Test string...

Or possibly emitting sensitive internal configuration information:

Pattern 55: tomcat_config

↑ Top

Remove

Name

tomcat_config

Category

APP

Optional grouping label for reports

Severity

Low

Optional severity for reports

Replacement

*

Text that replaces matches in redact mode. Use * to leave text intact (detect only).

☐ AI Monitor Correlate

Include in AI Monitor correlation analysis

Regex

`^(\d{2}-[A-Za-z]{3}-\d{4})\s\d{2}:\d{2}:\d{3})\sCONFIG\s.*`

Test string...

Version History

Click **Versions** in the toolbar to view the S3 version history of your subscriptions file. S3 versioning tracks every save.

- **Preview** – View the raw JSON of any previous version
- **Diff** – Compare selected to current version
- **Rollback** – Restore a previous version (saves it as the new current version)

The current version is highlighted with a blue badge. A maximum number of versions is supported, after which old versions are automatically deleted (Basic: 3 versions/30 days; Essential: 15/120 days; Advanced: 15/365 days; Enterprise: 25/365 days).

Version History

Close

S3 versioning tracks every save. Select a version to preview, diff, or rollback.

5/29/2026, 7:57:21 AM current InCdt6etizTT... · 23.8 KB	Preview	Diff
5/28/2026, 11:16:58 AM NlItqIveaAf_... · 23.8 KB	Preview	Diff Rollback
5/28/2026, 11:09:17 AM hG3TdkNlJs3... · 23.8 KB	Preview	Diff Rollback
5/28/2026, 8:18:37 AM yZ5W7mwutiz3... · 24.0 KB	Preview	Diff Rollback
5/27/2026, 12:12:15 PM DB.RZBUmNRDe... · 22.2 KB	Preview	Diff Rollback
5/27/2026, 8:52:35 AM Vf.nI.n9pm5B... · 20.2 KB	Preview	Diff Rollback
5/27/2026, 8:52:19 AM jn3myUrYRLa8... · 19.5 KB	Preview	Diff Rollback

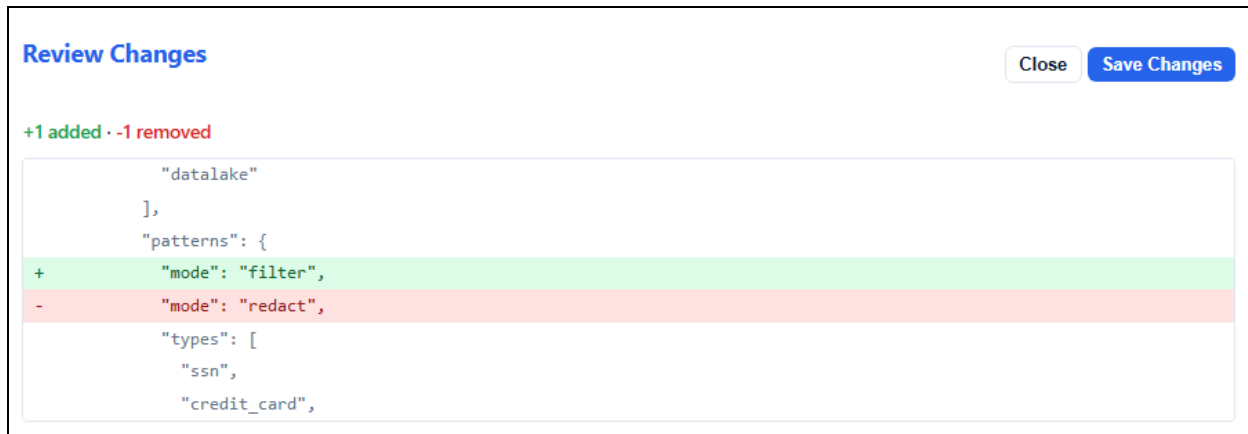
Note: If your session expires with pending edits, you may be presented with a prompt to load the last saved draft. You can choose to cancel, where the draft is lost, or load where you can inspect the changes, undo or save.

Diff (Review Changes)

Click **Diff** before saving to see exactly what changed:

- Lines in **green** are additions
- Lines in **red** are removals
- Unchanged context lines are shown around each change
- A summary shows total additions and removals

Click **Save Changes** from the diff view to save, or **Close** to continue editing.



Preview (JSON Preview)

Click **Preview** to view current:



Saving and Validation

When you click **Save**, the editor:

1. Collects all fields from the form
2. Sends the data to the server for validation
3. The server validates:
 - Log group names and stream names are valid regex
 - Subscribe values are [enabled](#), [disabled](#), or [unmanaged](#) (a remote team manages their own filters)
 - Account IDs are 12 digits, comma-separated, blank, or *
 - Each stream has an index and at least one target
 - Pattern regex is valid
4. If validation passes, the file is saved to S3
5. The Lambda picks up the new file on its next invocation (within 15 minutes), or click Sync Now to apply in real-time

If validation fails, error messages are displayed at the top of the page. Fix the issues and save again. If others have saved before you have completed your edits a message will request you reload.

Tips

- **Order matters** – Subscriptions are evaluated top to bottom. Place specific log group names before broad regex patterns. To assist you, metadata contains a key named [_sub](#) specifying the editor subscription #, e.g. [_sub: 1](#).
- **Use Clone** – When adding similar subscriptions, clone an existing one and change the log group name.
- **Review before saving** – Always use Review Changes to verify your edits. A mistake in the subscriptions file can break log ingestion.
- **Use Versions for safety** – If something goes wrong, rollback to a previous version immediately.
- **Metadata for filtering** – Add metadata like service, team, or environment to make logs easier to filter in Athena and OpenSearch Dashboards.
- **Test patterns** – Always test custom pattern regex using the built-in test field before saving.

Reporting

Users on the 'advanced' tier and above can click on the [Reports](#) to access to Automated and Dynamic Compliance Reporting:

Compliance Reports

Generate Now

Auto-generate schedule

Select days to automatically generate and email a compliance report. Save to apply.

☐ Mon
 ☐ Tue
 ☐ Wed
 ☐ Thu
 ☐ Fri
 ☐ Sat
 ☒ Sun
 Lookback days (1-30)

Recent Reports

5

compliance-20260607-134030.html (7 days)

Generated: 2026-06-07 13:40 UTC - 3.7 KB

compliance-20260607-133803.html (7 days)

Generated: 2026-06-07 13:38 UTC - 3.7 KB

compliance-20260607-133746.html (7 days)

Generated: 2026-06-07 13:37 UTC - 3.7 KB

compliance-20260607-133547.html (7 days)

Generated: 2026-06-07 13:35 UTC - 3.7 KB

compliance-20260607-133520.html (7 days)

Generated: 2026-06-07 13:35 UTC - 3.7 KB

Prev

Page 1 of 7

Next

Report Preview

Close

Compliance Report - LogProcessor

Period: 2026-06-02 to 2026-06-07 (5 days) • Generated: 2026-06-07 13:40 UTC

Event Volume

Metric	Count
S3 Objects Processed	3,698
Log Events Processed	24,992
Processing Errors	0
Active Subscriptions	30

Accounts

Users on the 'enterprise' tier can click on the [Accounts](#) to manage remote accounts:

Remote Accounts 3 account(s)

12-digit account ID **Add** **Save** **Close**

111111111111	✓ 8m ago
222222222222	✓ 8m ago
000000000000 (local)	✓ 8m ago

Validate **Remove** Double-click to validate

Cross-Account Log Ingestion

Centralize logs from multiple AWS accounts using the 'enterprise' tier.

Note: Leveraging provided (.cmd/.sh) scripts.

Step 1 - When deploying the CloudFormation enterprise stack specify your external accounts in the [CrossAccountIds](#) (comma separated):

CrossAccountIds
Comma-separated AWS account IDs allowed to send logs and replicate S3 access logs to this stack (e.g. 111111111111,222222222222). Leave empty for single-account.

111111111111

And/or [OrganizationId](#):

OrganizationId
AWS Organization ID to allow all accounts in the organization to send logs (e.g. o-abc123xyz0). Leave empty to use explicit account IDs only.

o-aaabbbcccc

Step 2 – Run *setup-remote-access* script

```
setup-remote-access.cmd setup <central-account-id> [region] [--org <org-id>]
setup-remote-access.cmd setup 000000000000 us-east-1 --org o-aaabbbcccc
```

Step 3 – Add and validate the new remote account in the Editor, via [Accounts](#) form. We have provided an *add-account* script for those that wish to script it.

We continue to provide the legacy scripts, especially if you want to manage your own filters:

Step 1 – Run `get-destination.cmd` [<stack>](#) on the target account to get the destination ARN:

get-destination.cmd LogProcessor
arn:aws:logs:us-east-1:000000000000:destination:log-dest-LogProcessor-000000000000-us-east-1

Step 2 – Run `setup-cross-account.cmd` [<log-group>](#) [<destination-arn-from-step-1>](#) on the source account:

setup-cross-account.cmd setup /aws/lambda/loggenerator arn:aws:logs:us-east-1:000000000000:destination:log-dest-LogProcessor-000000000000-us-east-1
Setting up cross-account subscription ... Log Group: /aws/lambda/loggenerator Destination: arn:aws:logs:us-east-1:000000000000:destination:log-dest-LogProcessor-000000000000-us-east-1 Filter: "" Region: us-east-1 Done. Logs from /aws/lambda/loggenerator will now flow to the LogProcessor destination. They will appear in OpenSearch and Athena within a few minutes.

You will see the subscription applied to the source log group:

/aws/lambda/loggenerator

Actions

View in Logs Insights

Start tailing

Search log group

▼ Log group details

Log class

Standard

Info

ARN

arn:aws:logs:us-east-1:111111111111:log-group:/aws/lambda/loggenerator:*

Creation time

19 hours ago

Retention

5 days

Stored bytes

366.21 KB

Metric filters

0

Subscription filters

1

Contributor Insights rules

-

KMS key ID

-

Deletion protection

Off

Data protection

-

Sensitive data count

-

Custom field indexes

Configure

Transformer

Configure

Anomaly detection

Configure

Bearer token authentication

Off

< 1

Anomaly detection

Metric filters

Subscription filters

Contributor Insights

Field indexes

Transformer

>

Subscription filters (1) Info

Refresh

Delete

Create

We support up to 1 account-level and up to 2 log group-level subscription filters.

↓

☐	Filter name	Filter pattern	Destination ARN	Subscription filter type	Applied on transfo
☐	LogProcessor-CrossAccount	-	arn:aws:logs:us-east-1:000.	Log group-level	-

Step 3 – Add a new subscription using the Accounts form in the Editor

- 1 – Add the log group (literal or using regular expression)
- 2 – You can select the external account e.g. [111111111111](#) or wild card *
- 3 – Select [unmanaged](#) for self-managed filters

Subscription 7

Clone
↑
↓
Remove

Log Group Name (regex)
Matches

Supports regex, e.g. /aws/lambda/*
-1 or empty = no change

Subscribe

Account ID (enterprise only)

Empty = local account. Use * for any account.

Streams

Stream 1
Remove

Log Stream Name (regex)
Index
Pipeline

Targets
☒ opensearch ☒ datalake

Pattern Mode

Metadata

service

log processor

vendor

perfware.cloud

purpose

testing cross account lambda log subscription with log genera

+ Add Metadata

+ Add Stream

You will find these logs in Athena and/or OpenSearch (depending on your targeting) e.g.

Time	@timestamp	accountId	logGroup	logStream	message
> May 7, 2026 @ 07:59:03.429	May 7, 2026 @ 07:59:03.429	111111111111	/aws/lambda/loggenerator	2026/05/07/[\$LATEST]b2ce79e1ff974f68965a4ab561a50be9	END RequestId: efbe2946-9f19-4e60-99e7-2249fda9a518
> May 7, 2026 @ 07:59:03.429	May 7, 2026 @ 07:59:03.429	111111111111	/aws/lambda/loggenerator	2026/05/07/[\$LATEST]b2ce79e1ff974f68965a4ab561a50be9	REPORT RequestId: efbe2946-9f19-4e60-99e7-2249fda9a518 Duration: 17.34 ms BMB Init Duration: 117.90 ms
> May 7, 2026 @ 07:59:03.395	May 7, 2026 @ 07:59:03.395	111111111111	/aws/lambda/loggenerator	2026/05/07/[\$LATEST]b2ce79e1ff974f68965a4ab561a50be9	[INFO] 2026-05-07T11:59:03.395Z efbe2946-9f19-4e60-99e7-2249fda9a518 Period


```
api_key="abcdef1234567890"
```

```
access_token: ghp_XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

github_token

ghp_ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz

gho_ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz

iban

GB29NWBK60161331926819

DE89370400440532013000

FR7630006000011234567890189

iban_refined

iban: GB29NWBK60161331926819

account: DE89370400440532013000

icd10

A01.0

J18.9

M54.5

E11.65

ipv4

192.168.1.1

10.0.2.100

172.16.0.12

ipv6

2001:0db8:85a3:0000:0000:8a2e:0370:7334

```
fe80:0000:0000:0000:0000:0000:0000:0001
```

jwt_token

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MzE1ODQyfQ.

SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c

mac_address

00:1A:2B:3C:4D:5E

AA-BB-CC-DD-EE-FF

mac_address_refined

mac: 00:1A:2B:3C:4D:5E

hwaddr: AA-BB-CC-DD-EE-FF

ether: 11:22:33:44:55:66

medicare_mbi

1A00B00CD01

2B12C34DE56

mrn_generic

MRN: PAT-123456

MRN:ABC789012

nhs_number

943 476 5919

123-456-7890

npi

1234567890

1987654321

passport_us

A12345678
C98765432
passport_us_refined
passport: A12345678
ppn: C98765432
phone_intl
+1-555-123-4567
+44 7911 123456
+61 412345678
phone_intl_refined
phone: +1-555-123-4567
tel: +44 7911 123456
int: +61 412345678
phone_us
(555) 123-4567
555-123-4567
+1 555.123.4567
phone_us_refined
phone: (555) 123-4567
tel: 555-123-4567
cell: 555.123.4567
mobile: (800) 555-0199
private_key
-----BEGIN RSA PRIVATE KEY-----
-----BEGIN OPENSSH PRIVATE KEY-----
-----BEGIN EC PRIVATE KEY-----
slack_token
xoxb-1234567890-1234567890123-AbCdEfGhIjKlMnOpQrStUvWx
xoxp-1234567890-1234567890123-1234567890123-abcdef1234
sql_alter
ALTER TABLE users ADD COLUMN email VARCHAR(255)
sql_comment
-- DROP TABLE users
/* DELETE FROM orders WHERE 1=1 */
sql_delete
DELETE FROM sessions WHERE expired = true
sql_drop
DROP TABLE customers
DROP DATABASE production
DROP INDEX idx_users_email
sql_exec
EXEC sp_executesql @sql
EXECUTE dbo.UpdateInventory
sql_injection
' OR '1'='1
; DROP TABLE users
sql_insert

```
INSERT INTO users (name, email) VALUES ('test', 'test@example.com')
```

sql_select

```
SELECT * FROM users WHERE id = 1  
SELECT name, email FROM customers ORDER BY created_at DESC
```

sql_truncate

```
TRUNCATE TABLE temp_data
```

sql_union

```
UNION SELECT username, password FROM admin_users  
UNION ALL SELECT * FROM secrets
```

sql_update

```
UPDATE users SET password = 'newpass' WHERE id = 42
```

ssn

```
123-45-6789  
987-65-4321
```

ssn_nondash

```
123456789  
987654321
```

swift_bic

```
DEUTDEFF  
BNPAFRPP  
CHASUS33XXX
```

swift_bic_refined

```
swift: DEUTDEFF  
bic: BNPAFRPP
```

tomcat_config

```
28-May-2026 14:44:29.461 CONFIG [main] org.apache.catalina.startup.VersionLoggerListener.log  
Server version
```

tomcat_debug

```
28-May-2026 14:44:29.461 FINE [http-nio-8080-exec-1] com.example.app.CacheManager.lookup  
Cache miss  
28-May-2026 14:44:29.461 FINER [main] org.apache.catalina.core.StandardContext.startInternal  
Starting
```

tomcat_severe

```
28-May-2026 14:44:29.461 SEVERE [http-nio-8080-exec-5]  
com.example.app.PaymentGateway.processRequest Error
```

tomcat_warning

```
28-May-2026 14:44:29.461 WARNING [http-nio-8080-exec-3]  
org.apache.catalina.connector.CoyoteAdapter.service Session expired
```

Notes

- Patterns use Python regex with flags like (?im) for case-insensitive multiline matching
- The credit_card pattern includes Luhn algorithm validation
- Patterns with _refined variants require a keyword prefix for higher precision
- In tag mode, matches are annotated but not modified
- In redact mode, matched text is replaced with the replacement string
- In filter mode, the entire log event is dropped