CDP Public Cloud Management Console

Control Plane Auditing

Date published: 2019-08-22
Date modified:

https://docs.cloudera.com/

Legal Notice

Contents

Auditing Control Plane activity

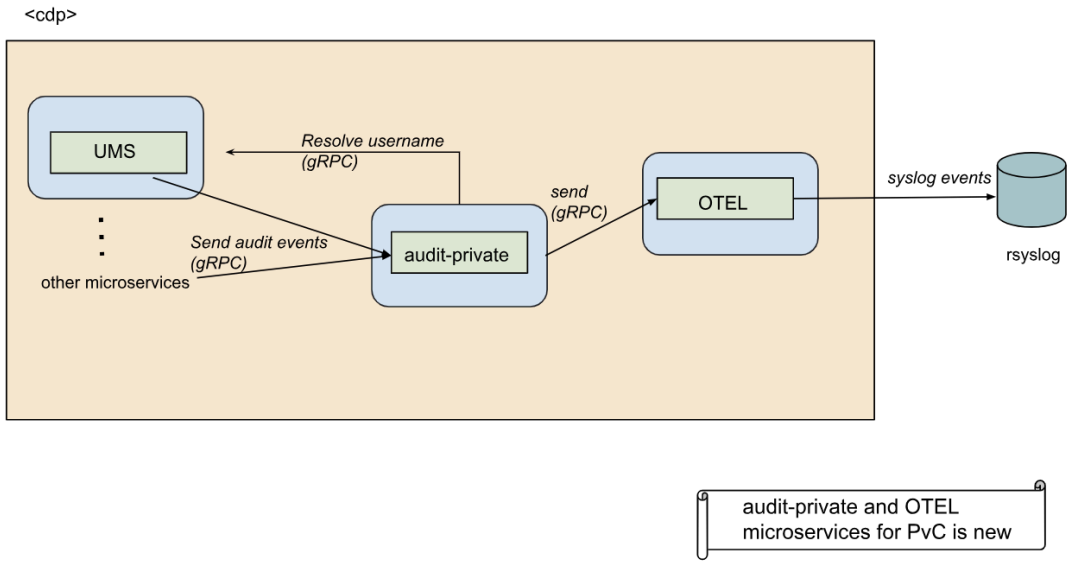
Auditing is used to collect or log evidence of activity in a system that auditors can use to both track and analyze to answer questions such as: Who made a change to the system? When did a change happen? What exactly changed? Why was a change authorized?

Control Plane auditing is based on the concept of an audit event. An audit event is a record of an audited action which is typically a change in the system that is important enough to keep a record of. However, even some read-only actions are audited, because it might be important to know who was able to see information in the system, and not just who could alter it.

Control Plane auditing is scoped to actions that occur within the CDP Control Plane. Audit events are not collected from workload clusters; in fact, many Control Plane audit events are collected without the need for any workload clusters to exist.

In Private Cloud, Control Plane audit data are sent to an OTEL collector. The OTEL collector can be configured to send data to external systems – such IBM Guardian – using the syslog OTEL exporter.

The following image shows the Private Cloud Control Plane auditing architecture:



Control Plane auditing data model

Three are three categories of audit events.

CDP service events

For actions that a service within the CDP Control Plane undertakes. These actions are often as a result of human activity, but can also result from autonomous processes within the Control Plane.

API request events

For calls to public API endpoints. These events are analogous to access logs kept by web servers. Because an API call often leads to actions within the CDP Control Plane, an API request often connects to one or more CDP service events.

Interactive login events

For logins to the CDP Control Plane.

All three categories of audit events share the following common fields:

Field name	Description	Example
version	Version of the audit event model	1.0.0
ID	Unique identifier for the event	a random UUID
source	Control plane service that submitted the event	iam
name	Type of action being audited	createGroupServiceEvent
timestamp	Time when the action occurred	2020-03-18T01:02:03Z
actor identity	Who initiated the action	see below
account ID	Identifier for account within which the action occurred	a UUID
request ID	Identifier for API request that led to the action	a UUID
result code	A string describing the result of the action, whether successful or not	INVALID_ARGUMENT
result message	A short message further describing the result	The group already exists

Each event source (service) defines its own event names. So, two sources may emit events with the same name, but for different actions. Events for an action within one source always use the same name.

An actor is an entity that causes an action to occur. In an audit event, an actor may be specified one of two ways.

Actor CRN

For a human actor, or for the special "internal" actor

Actor service name

For an antonymous process initiated by a control plane service (reserved for future use)

It is possible for the actor service name and the source in an audit event to be different. For example, a high-level service A may kick off an autonomous process that makes calls to another service B to make changes; audit events from that process would have actor service name A but source B.

Every call to the Control Plane public API receives a request ID. The request ID propagates through the Control Plane to services that perform actions, and audit events from those services include the request ID. Therefore, a request ID can be used to tie together multiple audit events under the umbrella of a single API request.

Most audit events include result information, but sometimes that information may be missing. This indicates that the event source experienced a failure such that it could not submit the result information for an event after submitting its initial, known set of information.

For example, consider the action to grant a role to a user. The event source responsible for this action starts by submitting an audit event for role creation, including all the information known before attempting the action: the user CRN, the role CRN, and perhaps more. After role creation either succeeds or fails, the source appends result information to the event. However, if the source crashes, it cannot append the result information. When this happens, at least the initial, known information is recorded in an audit event.

CDP service event

A CDP service event contains additional fields.

Field name	Description	Example
details version	Schema version for the additional details content	2020-03-31
additional details	JSON containing additional event-specific information	{ "groupCrn" : "crn:..." }

resource CRNs	CRNs for an affected resource, if applicable (may be multiple values)	crn:altus:iam:us-west-1:altus:role:PowerUser

Usually there is much more information about an action than can fit into the common audit event fields. The additional details field holds that additional information in a structured way. Each event source defines the details structure for each type of event (by event source / name) it generates.

API request event

An API request event contains additional fields.

Field name	Description	Example
request parameters	JSON format of API request	{ "param1": … }
response parameters	JSON format of API response	{ "param1": … }
mutating	Boolean indicating if this API call changes resource data	true
apiVersion	Version of the API called	2020-03-31
source IP address	IP address from where the request originated	192.168.0.1
user agent	User agent string of the request	Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:74.0) Gecko/20100101 Firefox/74.0

The request ID field can be used to tie an API request event to corresponding CDP service events, because CDP Control Plane services propagate the request ID initially generated for an API request through all resulting service activity.

Interactive login event

There are sufficient additional details about Control Plane login events that they merit their own category. An interactive login event contains the following additional fields. Here, an identity provider is an authentication system outside of the Control Plane that keeps identity information about users, such as Okta.

Field name	Description	Example
identity provider CRN	CRN of the identity provider as known by the control plane	crn:altus:iam:us-west-1:altus:samlProvider:cloudera-sso
identity provider session ID	identifier assigned to the login session by the identity provider	TBD
identity provider user ID	identifier of the user as stored in the identity provider	spongebob@cloudera.com (for Cloudera SSO, email is used)
email	email address of the user logging in	spongebob@cloudera.com
first name	first (given) name of the user logging in	Spongebob
last name	last name (family name / surname) of the user logging in	Squarepants
account admin	a Boolean flag indicating if the login is for an administrative user	true
groups	names of groups to which the user belongs in their account	TBD
source IP address	IP address of the user logging in	192.168.0.1
user CRN	CRN of the user logging in	crn:...

The account admin flag is only available if login succeeds. The user CRN is not known until login is attempted, and the CRN is not always recorded when login fails (for example, if the user's account cannot be determined by the Control Plane). Remaining fields are filled in before the user login attempt, and so should be present in every event.

CDP service event sources and names

CDP defines many service event sources and names.

The defined CDP service event sources and names will expand in the future.

Event source	Event name	Action
iam	AssignRoleServiceEvent	Assignment of a role to a user
	CreateUserServiceEvent	Creation of a new user in the control plane
	CreateGroupServiceEvent	Creation of a new group
	DeleteGroupServiceEvent	Deletion of an existing group
	InteractiveLogoutEvent	Interactive logout by a user in the control plane.
	UnassignRoleServiceEvent	Removal of a role from a user
Env	check-environment-connectivity	Checks connectivity to a Private Cloud environment
	get-environment-setting	Read a configuration setting from the environment service
	list-environments	Lists environments
	create-private-environment	Creates a new Private Cloud environment
	describe-environment	Describes an environment
	delete-environment	Deletes an environment
User	add-machine-user-to-group	Add a machine user to a group
	assign-group-role	Assign a role to a group
	assign-machine-user-role	Assign a role to a machine user
	create-group	Create a group
	create-machine-user	Creates a machine user. A machine user cannot login to the CDP console.
	create-machine-user-access-key	Creates a new access key for a machine user
	delete-group	Delete a group

Event source	Event name	Action
	delete-machine-user	delete-machine-user
	get-account	Retrieves information about the CDP account
	get-account-messages	Get account messages
	get-default-identity-provider	Retrieves the CRN of the default identity provider
	get-user	Gets information on a user
	list-access-keys	Lists access keys
	list-group-assigned-roles	Lists the group's assigned roles
	list-group-members	List the members of a group
	list-groups	Lists groups
	list-groups-for-machine-user	List the groups that the machine user belongs to
	list-machine-user-assigned-roles	Lists the machine user's assigned roles
	list-machine-users	list-machine-users
	list-resource-assignees	List the resource assignees and their respective resource roles for the resource
	list-resource-roles	Lists all the available resource roles
	list-user-assigned-roles	Lists the user's assigned roles
	list-users	Lists users
	remove-machine-user-from-group	Remove a machine user from a group
	remove-user-from-group	Remove a user from a group
	unassign-group-resource-role	Unassign a resource role from a group
	unassign-group-role	Unassign a role from a group
	unassign-machine-user-role	Unassign a role from a machine user
	unassign-user-resource-role	Unassign a resource role from a user
	unassign-user-role	Unassign a role from a user

Event source	Event name	Action
	update-access-key	Updates an access key
	update-group	Update a group
	update-user	Updates a user
drs	create-backup	Creates a backup for the control plane
	delete-backup	Deletes a control plane backup
	describe-backup	Describes the backup
	describe-restore	Restores the backup
	get-logs	get-logs
	list-backups	Lists backups
	list-restores	Lists restores
	restore-backup	Restores backup
compute	getResourcePool	Get Resource Pool Object
	getResourcePoolSubtree	Get Resource Pool Subtree
	getInstantMetricsValue	Get instant value for requested metric queries
	getRangeMetricsValue	Get range value for requested metric query

The schemas for the additional details JSON for each event are defined in the CDP Control Plane Audit Event Details Documentation.

Cloudera Data Warehouse audit events

Event source	Event name	Action
dw	CreateEnvironment	Activation of an environment in CDW
	DeleteEnvironment	Deactivation of an environment in CDW
	UpdateEnvironment	Updating a CDW environment
	CreateDbCatalog	Creation of a Database Catalog
	DeleteDbCatalog	Deleting a Database Catalog
	StartDbCatalog	Starting a Database Catalog
	StopDbCatalog	Stopping a Database Catalog
	UpdateDbCatalog	Updating a Database Catalog
	CloneDbCatalog	Cloning a Database Catalog
	UpgradeDbCatalog	Upgrading a Database Catalog

Event source	Event name	Action
	CreateHiveVirtualWarehouse	Creation of a Hive Virtual Warehouse
	DeleteHiveVirtualWarehouse	Deletion of a Hive Virtual Warehouse
	StartHiveVirtualWarehouse	Starting a Hive Virtual Warehouse
	StopHiveVirtualWarehouse	Stopping a Hive Virtual Warehouse
	UpdateHiveVirtualWarehouse	Updating a Hive Virtual Warehouse
	CloneHiveVirtualWarehouse	Cloning a Hive Virtual Warehouse
	UpgradeHiveVirtualWarehouse	Upgrading a Hive Virtual Warehouse
	CreateImpalaVirtualWarehouse	Creation of an Impala Virtual Warehouse
	DeleteImpalaVirtualWarehouse	Deletion of an Impala Virtual Warehouse
	StartImpalaVirtualWarehouse	Starting an Impala Virtual Warehouse
	StopImpalaVirtualWarehouse	Stopping an Impala Virtual Warehouse
	UpdateImpalaVirtualWarehouse	Updating an Impala Virtual Warehouse
	CloneImpalaVirtualWarehouse	Cloning an Impala Virtual Warehouse
	UpgradeImpalaVirtualWarehouse	Upgrading an Impala Virtual Warehouse
	SetServerSetting	Setting of a server configuration
	CreateDataViz	Creation of a Data Visualization instance
	DeleteDataViz	Deleting a Data Visualization instance
	UpdateDataViz	Updating a Data Visualization instance
	UpgradeDataViz	Upgrading a Data Visualization instance

Retrieving audit events

In Private Cloud, Control Plane audit data can be retrieved by configuring the OpenTelemetry (OTel) collector. The OTel collector can be configured to send data to external systems – such IBM Guardian – using the syslog OTel exporter.

OTel collector configuration

The OTel collector is used to receive the audit events. It supports the following three types of data:

- Traces
- Metrics
- Logs

The audit events are treated as logs in the OTel collector. Currently configuration of an OpenTelemetry exporter is only possible by editing the Kubernetes configmap cdp-release-opentelemetry-collector in the <cdp-project> namespace.

The default config contains only the logging exporter. To collect audit events in an external system such as rsyslog, the appropriate exporter config needs to be added there. To edit the configmap, run the following command:

```
kubectl edit cm cdp-release-opentelemetry-collector -n <cdp-project>
```

The default structure of the configmap is as follows:

```
# Valid values are "daemonset", "deployment", and "statefulset".
mode: "deployment"
```

```
config:
  receivers:
    jaeger: null
    prometheus: null
    zipkin: null
  service:
    pipelines:
      logs:
        exporters:
          - logging
        processors:
          - memory_limiter
          - batch
        receivers:
          - otlp
      metrics: null
      traces: null

ports:
  jaeger-compact:
    enabled: false
  jaeger-thrift:
    enabled: false
  jaeger-grpc:
    enabled: false
  zipkin:
    enabled: false
```

Forwarding to OTel

Forwarding of audit events to the OTel collector is disabled by default. You can enable OTel to receive audit events by configuring the following environment variable:

```
kubectl edit deploy cdp-release-thunderhead-audit-private -n <cdp-project>

# Add the following environment variable
        - name: FORWARDING_ENABLED
          value: "true"
```

Syslog OTel exporter configuration

This section provides an example of how to modify the OTel configmap to send audit events to a rsyslog endpoint using the syslog exporter. An example of adding a syslog exporter is described below. For additional information about the syslog exporter example, see: https://github.com/open-telemetry/opentelemetry-collector-contrib/blob/main/exporter/syslogexporter/examples/config_with_syslog_receiver.yaml

Sample syslog insecure configuration

The following snippet from the cdp-release-opentelemetry-collector configmap shows how to configure a syslog exporter without TLS:

```
apiVersion: v1
data:
  relay: |
    exporters:
      logging:
        verbosity: basic
      syslog:
       network: tcp
       port: 514
       endpoint: adt-demo-1.vpc.cloudera.com
       tls:
```

```
          insecure: true
        protocol: rfc3164
.
.
.
      pipelines:
        logs:
          exporters:
          - logging
          - syslog
```

Additionally syslog needs to be added under the services | logs | pipelines | exporters section.

In this example the rsyslog audit events are logged under: /var/log/messages:

```
Aug 30 22:45:35 ena-3.vpc.cloudera.com - {"action":"setEnvironmentSetting","
actor_crn":"crn:altus:iam:us-west-1:8f5a8f29-7834-4b66-8946-ebd7d2cf8508:use
r:17aa0daf-4f92-45fa-a8c9-6ca0478eec31","agent":"environments","evtTime":169
3435535994,"id":"c1080e42-b0ba-4bd4-b1dd-4bd0f7881f49","reqUser":"admin","re
quest_id":"44c24ab0-34bb-456a-a945-f10a72ad49c7","response_parameters":"{ }"
,"result":"SUCCESS","text":""}
Aug 30 22:46:45 ena-3.vpc.cloudera.com - {"action":"getUser","actor_crn":
"crn:altus:iam:us-west-1:8f5a8f29-7834-4b66-8946-ebd7d2cf8508:user:17aa0daf-
4f92-45fa-a8c9-6ca0478eec31","agent":"iam","api_version":"__API_VERSION__","
cliIP":"10.42.1.7","evtTime":1693435605667,"id":"58724fb9-69d5-4a92-b1f5-541
2809a9e8c","mutating":"false","reqData":"{ \"userId\": null }","reqUser":"ad
min","request_id":"ec66f71d-ce19-4d11-be4d-b7372bd7a23a","user_agent":"Mozil
la/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like G
ecko) Chrome/116.0.0.0 Safari/537.36"}
```

Sample syslog secure configuration

The following example shows how to configure a server CA for TLS. The ca_file must have the value /etc/opt/certs/ca.pem as that is the Private Cloud truststore file.

```
apiVersion: v1
data:
  relay: |
    exporters:
      logging:
        verbosity: basic
      syslog:
       network: tcp
       port: 6514
       endpoint: <rsyslog-hostname>
       tls:
         ca_file: /etc/opt/certs/ca.pem
       protocol: rfc3164
.
.
.
      pipelines:
        logs:
          exporters:
          - logging
          - syslog
```

Please note that this configuration will only work if the rsyslog server has TLS configured. Additional information on rsyslog TLS is available here: https://www.rsyslog.com/doc/master/tutorials/tls.html

We support TLS out of the box. mTLS is not supported – to configure mTLS, see TLS Configuration Settingsfor more information.

For added context, the following steps were done to test rsyslog using TLS. This test was done on a machine running RHEL 8.8.

The following lines were added to /etc/rsyslog.conf

```
module(
load="imtcp"
StreamDriver.Name="gtls"
StreamDriver.Mode="1"
StreamDriver.Authmode="anon"
) # needs to be done just once
input(type="imtcp" port="6514")
#### GLOBAL DIRECTIVES ####
global(
DefaultNetstreamDriver="gtls"
DefaultNetstreamDriverCAFile="/certs/myCA.pem"
DefaultNetstreamDriverCertFile="/certs/rsyslog.crt"
DefaultNetstreamDriverKeyFile="/certs/rsyslog.key"
```

If GnuTLS library is not already present, it must be installed:

yum install rsyslog-gnutls

The certs were created using a self-signed CA. The commands are:

```
# Create the CA private key
openssl genrsa -out myCA.key 2048

# Create the CA public key
openssl req -x509 -new -nodes -key myCA.key -sha256 -days 1825 -out myCA.pem

# Create the server cert private key
openssl genrsa -out rsyslog.key 2048

# Create a certificate signing request using the private key above
openssl req -new -key rsyslog.key -out rsyslog.csr

# Create an ext file rsyslog.ext with the contents below
basicConstraints=CA:FALSE
keyUsage = digitalSignature, nonRepudiation, keyEncipherment, dataEncipherm
ent
subjectAltName = @alt_names
[alt_names]
DNS.1 = <rsyslog-hostname>
# Create the server cert
openssl x509 -req -in rsyslog.csr -CA myCA.pem -CAkey myCA.key -CAcreatese
rial -out rsyslog.crt -days 825 -sha256 -extfile rsyslog.ext
```

Import the CA cert myCA.pem into the miscellaneous section of the CA certificates from the Control Plane UI.

If you are not using a trusted CA cert, the server cert rsyslog.crt must also be imported.