Logging

This guide covers DS server logs and logging options.

Logs

Understand server logs.

HTTP

Configure HTTP logs.

LDAP

Configure LDAP logs.

Log to Service

Log to local or remote services.

ForgeRock® Identity Platform serves as the basis for our simple and comprehensive Identity and Access Management solution. We help our customers deepen their relationships with their customers, and improve the productivity and connectivity of their employees and partners. For more information about ForgeRock and about the platform, see https://www.forgerock.com.

About Logs

Type	Description
Access	Messages about clients accessing the server. Each message includes a datestamp, information about the connection, and information about the operation. DS servers implement access logs for HTTP and LDAP. It is possible to configure multiple access logs at the same time. Do not enable multiple *unfiltered* file-based access loggers for the same protocol, however. This can put significant write load on the disk subsystem for access log files, because every client request results in at least one new log message.
Audit	Records changes to directory data in LDIF. DS servers implement an audit log as a special type of file-based access log. By default, the server writes messages to `opendj/logs/audit`. For an example, see Enable an Audit Log.
Debug	Messages tracing internal server events, for troubleshooting. By default, this is a file-based log, written to `opendj/logs/debug`. Debug logs can grow large quickly, and therefore no debug logs are enabled by default. For debug logging, you must set a *debug target* to control what gets logged. For details, see Debug Logging.
Error	Messages tracing server events, error conditions, and warnings, categorized and identified by severity. By default, this is a file-based log, written to `opendj/logs/errors`. Messages have the following format: `[datestamp] category=category severity=severity msgID=ID number msg=message string` For lists of severe and fatal error messages by category, see the Log Reference.

Type	Description
Replication repair	Messages to help repair problems in data replication. This is a file-based log, written to `opendj/logs/replication` . Messages have the following format: `[datestamp] category=SYNC severity=severity msgID=ID number msg=message string` The replication log does not trace replication operations. Use the external changelog instead to get notifications about changes to directory data. For details, see Changelog for Notifications.
Server	Messages about server events since startup. This is a file-based log, written to `opendj/logs/server.out` . A `opendj/logs/server.pid` process ID file is also available when the server is running. Messages in this file have the same format as error log messages.

You configure logging using *log publishers*. Log publishers determine which messages to publish, where to publish them, and what output format to use.

DS server logging supports extensibility through the ForgeRock Common Audit event framework. Common Audit deals with any event you can audit, not only the data updates recorded in a directory audit log. The ForgeRock Common Audit event framework provides log handlers for publishing to local files or to remote systems.

Common ForgeRock Access Logs

DS servers support the ForgeRock Common Audit event framework. The log message formats are compatible for all products using the framework. The framework uses transaction IDs to correlate requests as they traverse the platform. This makes it easier to monitor activity and to enrich reports:

- The ForgeRock Common Audit event framework is built on *audit event handlers*. Audit event handlers can encapsulate their own configurations. Audit event handlers are the same in each product in the ForgeRock platform. You can plug in custom handlers that comply with the framework without having to upgrade the server.

- The ForgeRock Common Audit event framework includes handlers for logging to local files and to external services.

  Although the ForgeRock Common Audit event framework supports multiple topics, DS software currently supports handling only access events. DS software divides access events into `ldap-access` events and `http-access` events.

- Common Audit transaction IDs are not recorded by default. To record transaction IDs in the access logs, configure the DS server to trust them.

Common Audit LDAP events have the following format:

```
{
  "eventName": "DJ-LDAP",
  "client": {
    "ip": string,                           // Client IP address
    "port": number                          // Client port number
  },
  "server": {
    "ip": string,                           // Server IP address
    "port": number                          // Server port number
  },
  "request": {                              // LDAP request
    "attrs": [ string ],                    // Requested attributes
    "authType": string,                     // Bind type such as
"SIMPLE"
    "connId": number,                       // Connection ID
    "controls": string,                     // Request controls
    "deleteOldRDN": boolean,                // For a modify DN
request
    "dn": string,                           // Bind DN
    "filter": string,                       // Search filter
    "idToAbandon": number,                  // ID to use to abandon
operation
    "message": string,                      // Localized request
message
    "msgId": number,                        // Message ID
    "name": string,                         // Operation name
    "newRDN": string,                       // For a modify DN
request
    "newSup": string,                       // For a modify DN
request
    "oid": string,                          // Operation name or OID
    "operation": string,                    // Examples: "CONNECT",
"BIND", "SEARCH"
    "opType": "sync",                       // Replication operation
```

```
    "protocol": "LDAP",
    "runAs": string,                        // Authorization ID
    "scope": string,                        // Search scope such as
"sub"
    "version": string                       // Version "2", "3"
  },
  "response": {
    "additionalItems": string               // Additional
information
    "controls": string,                     // Response controls
    "elapsedTime": number,                  // Number of time units
    "elapsedTimeUnits": string,             // Time unit such as
"MILLISECONDS"
    "failureReason": string,                // Human-readable
information
    "maskedMessage": string,                // Real, masked result
message
    "maskedResult": string,                 // Real, masked result
code
    "nentries": number,                     // Number of entries
returned
    "reason": string,                       // Reason for disconnect
    "status": string,                       // "SUCCESSFUL",
"FAILED"
    "statusCode": string                    // For example, "0" for
success
  },
  "timestamp": string,                      // UTC date
  "transactionId": string,                  // Unique ID for the
transaction
  "userId": string,                         // User who requested
the operation
  "_id": string                             // Unique ID for the
operation
}
```

Common Audit HTTP events have the following format:

```
{
  "eventName": "DJ-HTTP",
  "client": {
    "ip": string,                           // Client IP address
    "port": number                          // Client port number
  },
  "server": {
```

```
    "ip": string,                          // Server IP address
    "port": number                         // Server port number
  },
  "http": {                                // HTTP request and
response
    "request": {
      "secure": boolean,                   // HTTP: false; HTTPS:
true
      "method": string,                    // Examples: "GET",
"POST", "PUT"
      "path": string,                      // URL
      "queryParameters": map,              // map: { key-string: [
value-string ] }
      "cookies": map                       // map: { key-string: [
value-string ] }
    },
    "response": {
      "headers": map                       // map: { key-string: [
value-string ] }
    }
  },
  "response": {
    "detail": string,                      // Human-readable
information
    "elapsedTime": number,                 // Number of time units
    "elapsedTimeUnits": string,            // Time unit such as
"MILLISECONDS"
    "status": string,                      // "SUCCESSFUL",
"FAILED"
    "statusCode": string                   // For example, "0" for
success
  },
  "timestamp": string,                     // UTC date
  "transactionId": string,                 // Unique ID for the
transaction
  "trackingIds": [ string ],               // Unique IDs from the
transaction context
  "userId": string,                        // User who requested
the operation
  "_id": string                            // Unique ID for the
operation
}
```

Access Log Filtering

With the default access log configuration (no filtering), for every client application request, the server writes at least one message to its access log. This volume of logging gives you the information to analyze overall access patterns, or to audit access when you do not know in advance what you are looking for.

When you do know what you are looking for, log filtering lets you throttle logging to focus on what you want to see. You specify the criteria for a filtering policy, and apply the policy to a log publisher.

Log filtering policies use the following criteria:

- Client IP address, bind DN, group membership
- Operation type (abandon, add, bind, compare, connect, delete, disconnect, extended operation, modify, rename, search, and unbind)
- Port number
- Protocol used
- Response time
- Result codes (only log error results, for example)
- Search response criteria (number of entries returned, unindexed search, and others)
- Target DN
- User DN and group membership

A log publisher's filtering policy determines whether to include or exclude log messages that match the criteria.

For examples, see Filter Out Administrative Messages and Audit Configuration Changes.

Log HTTP Access to Files

JSON Format

When you install DS using procedures from Installation, the default JSON-based HTTP access log file is `logs/http-access.audit.json`. The name of the access log publisher in the configuration is `Json File-Based HTTP Access Logger`.

The sample DS Docker image logs to standard output instead of files. This makes it easy to read log messages with the `docker logs` command, and is a pattern you should follow when creating your own DS Docker images. The name of the LDAP access log publisher configuration in the sample image is `Console HTTP Access Logger`:

1. Decide whether to trust transaction IDs sent by client applications, used to correlate requests as they traverse multiple servers.

   Client applications using the ForgeRock Common Audit event framework send transaction IDs with their requests. The transaction IDs correlate audit events, tracing the request through multiple applications.

   Transaction IDs are sent over LDAP using an internal DS request control. They are sent over HTTP in an HTTP header.

   By default, DS servers do not trust transaction IDs sent with client application requests.

   When a server trusts transaction IDs from client application requests, outgoing requests reuse the incoming ID. For each outgoing request in the transaction, the request's transaction ID has the form *`original-transaction-id/sequence-number`*, where sequence-number reflects the position of the request in the series of requests for this transaction. For example, if the original-transaction-id is `abc123`, the first outgoing request has the transaction ID `abc123/0`, the second `abc123/1`, the third `abc123/2`, and so on. This lets you distinguish specific requests within a transaction when correlating audit events from multiple services.

   To trust transactions, set the advanced global server property, `trust-transaction-ids:true`:

   ```
   $ dsconfig \
    set-global-configuration-prop \
    --advanced \
    --hostname localhost \
    --port 4444 \
    --bindDN uid=admin \
    --bindPassword password \
    --set trust-transaction-ids:true \
    --usePkcs12TrustStore /path/to/opendj/config/keystore \
    --trustStorePassword:file
   /path/to/opendj/config/keystore.pin \
    --no-prompt
   ```

2. Edit the default HTTP access log publisher as necessary.

   The following example enables the default log publisher for DS installed locally, not in a Docker image:

   ```
   $ dsconfig \
    set-log-publisher-prop \
    --hostname localhost \
   ```

```
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "Json File-Based HTTP Access Logger" \
 --set enabled:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file
/path/to/opendj/config/keystore.pin \
 --no-prompt
```

CSV Format

A CSV handler sends messages to a comma-separated variable (CSV) file.

IMPORTANT

The CSV handler does not sanitize messages when writing to CSV log files.

Do not open CSV logs in spreadsheets and other applications that treat data as code.

The default CSV HTTP access log file is `logs/http-access.csv` :

1. Decide whether to trust transaction IDs sent by client applications, used to correlate requests as they traverse multiple servers.

   Client applications using the ForgeRock Common Audit event framework send transaction IDs with their requests. The transaction IDs correlate audit events, tracing the request through multiple applications.

   Transaction IDs are sent over LDAP using an internal DS request control. They are sent over HTTP in an HTTP header.

   By default, DS servers do not trust transaction IDs sent with client application requests.

   When a server trusts transaction IDs from client application requests, outgoing requests reuse the incoming ID. For each outgoing request in the transaction, the request's transaction ID has the form *`original-transaction-id/sequence-number`* , where sequence-number reflects the position of the request in the series of requests for this transaction. For example, if the original-transaction-id is `abc123` , the first outgoing request has the transaction ID `abc123/0` , the second `abc123/1` , the third `abc123/2` , and so on. This lets you distinguish specific requests within a transaction when correlating audit events from multiple services.

To trust transactions, set the advanced global server property, `trust-transaction-ids:true`:

```
$ dsconfig \
 set-global-configuration-prop \
 --advanced \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --set trust-transaction-ids:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file
/path/to/opendj/config/keystore.pin \
 --no-prompt
```

2. Create an enabled CSV file HTTP access logger with optional rotation and retention policies:

```
$ dsconfig \
 create-log-publisher \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "Common Audit Csv File HTTP Access
Logger" \
 --type csv-file-http-access \
 --set enabled:true \
 --set "rotation-policy:24 Hours Time Limit Rotation
Policy" \
 --set "rotation-policy:Size Limit Rotation Policy" \
 --set "retention-policy:File Count Retention Policy" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file
/path/to/opendj/config/keystore.pin \
 --no-prompt
```

3. For tamper-evident logs, follow these steps.

IMPORTANT

IMPORTANT

Tamper-evident logging relies on digital signatures and regularly flushing messages to the log system. In high-volume directory deployments with heavy access patterns, signing log messages has a severe negative impact on server performance, reducing throughput by orders of magnitude.

Be certain to test the performance impact with realistic access patterns for your deployment before enabling the feature in production.

a. Prepare a keystore.

For details, see Make Tampering Evident.

b. Enable the tamper-evident capability:

```
$ dsconfig \
 set-log-publisher-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "Common Audit Csv File HTTP Access
Logger" \
 --set tamper-evident:true \
 --set key-store-file:config/audit-keystore \
 --set key-store-pin:"&{audit.keystore.pin}" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore
\
 --trustStorePassword:file
/path/to/opendj/config/keystore.pin \
 --no-prompt
```

In this example, `AUDIT_KEYSTORE_PIN` is an environment variable containing the keystore PIN.

Standard HTTP Format

For HTTP requests, you can configure an access logger that uses the Extended Log File Format, a W3C working draft. The default log file is `logs/http-access`:

1. Enable the standard format HTTP access logger:

```
$ dsconfig \
 set-log-publisher-prop \
```

```
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "File-Based HTTP Access Logger" \
 --set enabled:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file
/path/to/opendj/config/keystore.pin \
 --no-prompt
```

The following example shows an excerpt of an HTTP access log with space reformatted:

```
- <client-ip> bjensen  <datestamp> GET  /users/bjensen
HTTP/1.1 200 <user-agent> 3  40
- <client-ip> bjensen  <datestamp> GET  /users/scarter
HTTP/1.1 200 <user-agent> 4   9
- <client-ip> -        <datestamp> GET  /users/missing
HTTP/1.1 401 <user-agent> 5   0
- <client-ip> kvaughan <datestamp> POST /users
HTTP/1.1 200 <user-agent> 6 120
```

Missing values are replaced with `-`. Tabs separate the fields, and if a field contains a tab character, then the field is surrounded with double quotes. DS software repeats double quotes in the field to escape them.

Configure the `log-format` property to set the fields. The default fields are shown here in the order they occur in the log file:

Field	Description
`cs-host`	Client hostname.
`c-ip`	Client IP address.
`cs-username`	Username used to authenticate.
`x-datetime`	Completion timestamp for the HTTP request. Configure with the `log-record-time-format` property.
`cs-method`	HTTP method requested by the client.

Field	Description
`cs-uri`	URI requested by the client.
`cs-uri-stem`	URL-encoded path requested by the client.
`cs-uri-query`	URL-encoded query parameter string requested by the client.
`cs-version`	HTTP version requested by the client.
`sc-status`	HTTP status code for the operation.
`cs(User-Agent)`	User-Agent identifier.
`x-connection-id`	Connection ID used for DS internal operations. When using this field to match HTTP requests with internal operations in the LDAP access log, set the access log advanced property, `suppress-internal-operations:false`. By default, internal operations do not appear in the LDAP access log.
`x-etime`	Execution time in milliseconds needed by DS to service the HTTP request.
`x-transaction-id`	ForgeRock Common Audit event framework transaction ID for the request. This defaults to `0`, unless you configure the server to trust transaction IDs.

The following additional fields are supported:

Field	Description
`c-port`	Client port number.
`s-computername`	Server name writing the access log.

Field	Description
`s-ip`	Server IP address.
`s-port`	Server port number.

Log LDAP Access to Files

JSON Format

When you install DS using procedures from Installation, the primary JSON-based LDAP access log file is `logs/ldap-access.audit.json`. The name of the access log publisher in the configuration is `Json File-Based Access Logger`.

The sample DS Docker image logs to standard output instead of files. This makes it easy to read log messages with the `docker logs` command, and is a pattern you should follow when creating your own DS Docker images. The name of the LDAP access log publisher configuration in the sample image is `Console LDAP Access Logger`.

Primary access logs include messages for each LDAP operation. They can grow quickly, but are particularly useful for analyzing overall client behavior:

1. Decide whether to trust transaction IDs sent by client applications, used to correlate requests as they traverse multiple servers.

   Client applications using the ForgeRock Common Audit event framework send transaction IDs with their requests. The transaction IDs correlate audit events, tracing the request through multiple applications.

   Transaction IDs are sent over LDAP using an internal DS request control. They are sent over HTTP in an HTTP header.

   By default, DS servers do not trust transaction IDs sent with client application requests.

   When a server trusts transaction IDs from client application requests, outgoing requests reuse the incoming ID. For each outgoing request in the transaction, the request's transaction ID has the form *`original-transaction-id/sequence-number`*, where sequence-number reflects the position of the request in the series of requests for this transaction. For example, if the original-transaction-id is `abc123`, the first outgoing request has the transaction ID `abc123/0`, the second `abc123/1`, the third `abc123/2`, and so

on. This lets you distinguish specific requests within a transaction when correlating audit events from multiple services.

To trust transactions, set the advanced global server property, `trust-transaction-ids:true`:

```
$ dsconfig \
 set-global-configuration-prop \
 --advanced \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --set trust-transaction-ids:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file
/path/to/opendj/config/keystore.pin \
 --no-prompt
```

2. Edit the default access log publisher as necessary.

   The following example applies the default settings for DS installed locally, not in a Docker image:

```
$ dsconfig \
 set-log-publisher-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "Json File-Based Access Logger" \
 --set enabled:true \
 --add "rotation-policy:24 Hours Time Limit Rotation
Policy" \
 --add "rotation-policy:Size Limit Rotation Policy" \
 --set "retention-policy:File Count Retention Policy" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file
/path/to/opendj/config/keystore.pin \
 --no-prompt
```

Filtered JSON Format

DS servers write messages to a filtered access log file, `logs/filtered-ldap-access.audit.json`. This log grows more slowly than the primary access log. It includes only messages about the following:

- Administrative requests related to backing up and restoring data, scheduling tasks, and reading and writing configuration settings
- Authentication failures
- Requests from client applications that are misbehaving
- Requests that take longer than one second for the server to process
- Search requests that return more than 1000 entries
- Unindexed searches

Follow these steps to change the configuration:

1. Edit the filtered access log publisher as necessary.

   The following example updates the configuration to include control OIDs in log records:

   ```
   $ dsconfig \
    set-log-publisher-prop \
    --hostname localhost \
    --port 4444 \
    --bindDN uid=admin \
    --bindPassword password \
    --publisher-name "Filtered Json File-Based Access Logger"
   \
    --set log-control-oids:true \
    --usePkcs12TrustStore /path/to/opendj/config/keystore \
    --trustStorePassword:file
   /path/to/opendj/config/keystore.pin \
    --no-prompt
   ```

2. Edit the filtering criteria as necessary.

   The following commands list the relevant default filtering criteria settings for the filtered access log:

   ```
   $ dsconfig \
    get-access-log-filtering-criteria-prop \
    --hostname localhost \
    --port 4444 \
    --bindDN uid=admin \
   ```

```
 --bindPassword password \
 --publisher-name "Filtered Json File-Based Access Logger"
\
 --criteria-name "Administrative Requests" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file
/path/to/opendj/config/keystore.pin \
 --no-prompt

log-record-type                          : add, bind,
compare, delete, extended,
                                         : modify, rename,
search
request-target-dn-equal-to               : "**,cn=config",
"**,cn=tasks",
                                         : cn=config,
cn=tasks

$ dsconfig \
 get-access-log-filtering-criteria-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "Filtered Json File-Based Access Logger"
\
 --criteria-name "Auth Failures" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file
/path/to/opendj/config/keystore.pin \
 --no-prompt

log-record-type                          : add, bind,
compare, delete, extended,
                                         : modify, rename,
search
response-result-code-equal-to            : 7, 8, 13, 48, 49,
50, 123

$ dsconfig \
 get-access-log-filtering-criteria-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
```

```
 --publisher-name "Filtered Json File-Based Access Logger"
\
 --criteria-name "Long Requests" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file
/path/to/opendj/config/keystore.pin \
 --no-prompt

log-record-type                         : add, bind,
compare, delete, extended,
                                        : modify, rename,
search
response-etime-greater-than             : 1000

$ dsconfig \
 get-access-log-filtering-criteria-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "Filtered Json File-Based Access Logger"
\
 --criteria-name "Misbehaving Clients" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file
/path/to/opendj/config/keystore.pin \
 --no-prompt

log-record-type                         : add, bind,
compare, delete, extended,
                                        : modify, rename,
search
response-result-code-equal-to           : 1, 2, 17, 18, 19,
21, 34, 60, 61, 64,
                                        : 65, 66, 67, 69

$ dsconfig \
 get-access-log-filtering-criteria-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "Filtered Json File-Based Access Logger"
\
 --criteria-name "Searches Returning 1000+ Entries" \
```

```
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file
/path/to/opendj/config/keystore.pin \
 --no-prompt

log-record-type                          : search
search-response-nentries-greater-than    : 1000

$ dsconfig \
 get-access-log-filtering-criteria-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "Filtered Json File-Based Access Logger"
\
 --criteria-name "Unindexed Searches" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file
/path/to/opendj/config/keystore.pin \
 --no-prompt

log-record-type                          : search
search-response-is-indexed               : false
```

For details about the LDAP result codes listed in the criteria, see LDAP Result Codes.

For details about how filtering works, see Access Log Filtering.

CSV Format

A CSV handler sends messages to a comma-separated variable (CSV) file.

IMPORTANT

The CSV handler does not sanitize messages when writing to CSV log files.

Do not open CSV logs in spreadsheets and other applications that treat data as code.

The default CSV LDAP access log file is `logs/ldap-access.csv` :

1. Decide whether to trust transaction IDs sent by client applications, used to correlate requests as they traverse multiple servers.

Client applications using the ForgeRock Common Audit event framework send transaction IDs with their requests. The transaction IDs correlate audit events, tracing the request through multiple applications.

Transaction IDs are sent over LDAP using an internal DS request control. They are sent over HTTP in an HTTP header.

By default, DS servers do not trust transaction IDs sent with client application requests.

When a server trusts transaction IDs from client application requests, outgoing requests reuse the incoming ID. For each outgoing request in the transaction, the request's transaction ID has the form *`original-transaction-id/sequence-number`*, where sequence-number reflects the position of the request in the series of requests for this transaction. For example, if the original-transaction-id is `abc123`, the first outgoing request has the transaction ID `abc123/0`, the second `abc123/1`, the third `abc123/2`, and so on. This lets you distinguish specific requests within a transaction when correlating audit events from multiple services.

To trust transactions, set the advanced global server property, `trust-transaction-ids:true`:

```
$ dsconfig \
 set-global-configuration-prop \
 --advanced \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --set trust-transaction-ids:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file
/path/to/opendj/config/keystore.pin \
 --no-prompt
```

2. Create an enabled CSV file access logger with optional rotation and retention policies:

```
$ dsconfig \
 create-log-publisher \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "Common Audit Csv File Access Logger" \
```

```
 --type csv-file-access \
 --set enabled:true \
 --set "rotation-policy:24 Hours Time Limit Rotation
Policy" \
 --set "rotation-policy:Size Limit Rotation Policy" \
 --set "retention-policy:File Count Retention Policy" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file
/path/to/opendj/config/keystore.pin \
 --no-prompt
```

3. For tamper-evident logs, follow these steps.

IMPORTANT

Tamper-evident logging relies on digital signatures and regularly flushing messages to the log system. In high-volume directory deployments with heavy access patterns, signing log messages has a severe negative impact on server performance, reducing throughput by orders of magnitude.

Be certain to test the performance impact with realistic access patterns for your deployment before enabling the feature in production.

a. Prepare a keystore.

For details, see Make Tampering Evident.

b. Enable the tamper-evident capability:

```
$ dsconfig \
 set-log-publisher-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "Common Audit Csv File Access Logger"
\
 --set tamper-evident:true \
 --set key-store-file:config/audit-keystore \
 --set key-store-pin:"&{audit.keystore.pin}" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore
\
 --trustStorePassword:file
/path/to/opendj/config/keystore.pin \
 --no-prompt
```

In this example, `AUDIT_KEYSTORE_PIN` is an environment variable containing the PIN.

Backwards-Compatible Format

This access log format was the default for older DS servers. Use this log format if you already have software configured to consume that format. The default log file is `logs/access`:

1. Enable the LDAP access logger:

```
$ dsconfig \
 set-log-publisher-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "File-Based Access Logger" \
 --set enabled:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file
/path/to/opendj/config/keystore.pin \
 --no-prompt
```

By default, this access log contains a message for each request, and a message for each response. It also includes messages for connection and disconnection.

Write messages only on responses by setting the `log-format:combined` property. The setting is useful when filtering messages based on response criteria. It causes the server to log one message per operation, rather than one for each request and response.

Log to a Service

The Common Audit framework supports logging access events to an external service:

Elasticsearch

An Elasticsearch audit event handler sends messages to an Elasticsearch server.

NOTE

NOTE

This audit handler is deprecated.

Before you enable the Elasticsearch handler, create a mapping in the Elasticsearch server index for the messages. For a sample index definition, see `opendj/config/audit-handlers/elasticsearch-index-setup-example.json`.

To enable the Elasticsearch handler, see Configure a Custom Access Log. The JSON configuration file for the Elasticsearch handler has the following format:

```
{
  "class":
"org.forgerock.audit.handlers.elasticsearch.ElasticsearchAuditEven
tHandler",
  "config": {
    "name": string,                // Handler name, such as
"elasticsearch".
    "topics": [ string, ...],      // LDAP: "ldap-access"; HTTP:
"http-access".
    "enabled": boolean,            // Is the handler enabled?
    "connection": {                // (Optional) Connect using
default settings.
      "host": string,              // Elasticsearch host. Default:
localhost.
      "port": number,              // Elasticsearch host. Default:
9200.
      "useSSL": boolean,           // Connect to Elasticsearch over
HTTPS? Default: false.
      "username": string,          // (Optional) User name for HTTP
Basic auth.
      "password": string           // (Optional) Password for HTTP
Basic auth.
    },
    "indexMapping": {              // (Optional) No mapping
specified.
      "indexName": string          // Name of the Elasticsearch
index.
    },
    "buffering": {                 // (Optional) Default: write
each message separately,
                                   // no buffering.
      "enabled": boolean,          // Buffer messages to be sent?
Default: false.
      "maxSize": number,           // Maximum number of buffered
events.
```

```
      "writeInterval": duration,  // Interval between sending
batch of events.
      "maxBatchedEvents": number  // Number of events to send per
interval.
    }
  }
}
```

For a sample log configuration, see `opendj/config/audit-handlers/elasticsearch-config.json-example`.

The `writeInterval` takes a duration, which is a lapse of time expressed in English, such as `23 hours 59 minutes and 59 seconds`. Durations are not case sensitive. Negative durations are not supported. Durations use these units:

- `indefinite`, `infinity`, `undefined`, `unlimited`: unlimited duration
- `zero`, `disabled`: zero-length duration
- `days`, `day`, `d`: days
- `hours`, `hour`, `h`: hours
- `minutes`, `minute`, `min`, `m`: minutes
- `seconds`, `second`, `sec`, `s`: seconds
- `milliseconds`, `millisecond`, `millisec`, `millis`, `milli`, `ms`: milliseconds
- `microseconds`, `microsecond`, `microsec`, `micros`, `micro`, `us`: microseconds
- `nanoseconds`, `nanosecond`, `nanosec`, `nanos`, `nano`, `ns`: nanoseconds

These steps demonstrate logging HTTP access messages to a local Elasticsearch server:

1. Install and run an Elasticsearch server on `localhost:9200`.
2. Create an `audit` index in the Elasticsearch server for HTTP audit event messages.

   The following command uses the example index configuration file:

   ```
   $ curl \
    --request POST \
    --header "Content-Type: application/json" \
    --data @/path/to/opendj/config/audit-
   handlers/elasticsearch-index-setup-example.json \
    http://localhost:9200/audit

   {"acknowledged":true}
   ```

3. Configure DS servers to enable HTTP access.

   For details, see Configure HTTP User APIs.

4. Add a JSON configuration file for the handler:

```
$ cat /path/to/opendj/config/audit-handlers/elasticsearch-
handler.json

{
  "class":
"org.forgerock.audit.handlers.elasticsearch.ElasticsearchA
uditEventHandler",
  "config": {
    "name": "elasticsearch",
    "topics": ["http-access"],
    "connection": {
      "useSSL": false,
      "host": "localhost",
      "port": 9200
    },
    "indexMapping": {
      "indexName": "audit"
    },
    "buffering": {
      "enabled": true,
      "maxSize": 10000,
      "writeInterval": "100 ms",
      "maxBatchedEvents": 500
    }
  }
}
```

5. Configure the server to use the Elasticsearch audit handler:

```
$ dsconfig \
 create-log-publisher \
 --publisher-name "Elasticsearch HTTP Access Log
Publisher" \
 --type external-http-access \
 --set enabled:true \
 --set config-file:config/audit-handlers/elasticsearch-
handler.json \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
```

```
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file
/path/to/opendj/config/keystore.pin \
 --no-prompt
```

With Elasticsearch and the DS server running, DS sends access event messages to Elasticsearch.

6. Read a resource:

```
$ curl --user bjensen:hifalutin --cacert ca-cert.pem
https://localhost:8443/api/users/bjensen
```

7. Show the resulting audit event content:

```
$ curl 'localhost:9200/audit/_search?q=*&pretty'
```

See the Elasticsearch documentation for details on searching and search results.

JDBC

A JDBC handler sends messages to an appropriately configured relational database table.

Before you enable the JDBC handler, create the necessary schema and tables in the target database. See the following example files:

- `/path/to/opendj/config/audit-handlers/mysql_tables-example.sql`
- `/path/to/opendj/config/audit-handlers/oracle_tables-example.sql`
- `/path/to/opendj/config/audit-handlers/postgres_tables-example.sql`

The JDBC handler depends on the JDBC driver for the database, and on HirakiCP. Copy the JDBC driver .jar file for your database, the HirakiCP .jar file for your Java version, and any other dependent libraries required to the `opendj/extlib/` directory.

To enable the JDBC handler, see Configure a Custom Access Log. The JSON configuration file for the JDBC handler has the following format:

```
{
    "class":
"org.forgerock.audit.handlers.jdbc.JdbcAuditEventHandler",
    "config": {
```

```
        "name": string,                     // Handler name, such
as "jdbc".
        "topics": array,                    // LDAP: "ldap-access";
HTTP: "http-access".
        "databaseType": string,             // Supported by
default: "h2", "mysql",
                                            // "oracle",
"postgres".
        "enabled": boolean,                 // Is the handler
enabled?
        "buffering": {                      // (Optional) Default:
write each message separately,
                                            // no buffering.
            "enabled": boolean,             // Buffer messages to
be sent? Default: false.
            "writeInterval": duration,      // Duration; must be >
0 if buffering is enabled.
            "autoFlush": boolean,           // Flush messages
automatically? Default: true.
            "maxBatchedEvents": number,     // Maximum messages in
prepared statement. Default: 100.
            "maxSize": number,              // Maximum number of
buffered messages. Default: 5000.
            "writerThreads": number         // Threads to write
buffered messages: Default: 1.
        },
        "connectionPool": {
            "dataSourceClassName": string, // Either set this to
the class name of the data source...
            "jdbcUrl": string,              // ...or set this to
the JDBC URL to
                                            // connect to the
database.
            "username": string,             // Username to connect
to the database.
            "password": string,             // Password to connect
to the database.
            "autoCommit": boolean,          // (Optional) Commit
transactions automatically?
                                            // Default: true.
            "connectionTimeout": number,    // (Optional)
Milliseconds to wait before timing out.
                                            // Default: 30,000.
            "idleTimeout": number,          // (Optional)
Milliseconds to wait before timing out.
```

```
                                            // Default: 600,000.
            "maxLifetime": number,          // (Optional)
Milliseconds thread remains in pool.
                                            // Default: 1,800,000.
            "minIdle": number,              // (Optional) Minimum
connections in pool.
                                            // Default: 10.
            "maxPoolSize": number,          // (Optional) Maximum
number of connections in pool.
                                            // Default: 10.
            "poolName": string,             // (Optional) Name of
connection pool.
                                            // Default: audit.
            "driverClassName": string       // (Optional) Class
name of database driver.
                                            // Default: null.
        },
        "tableMappings": [                  // Correspondence of
message fields to database columns.
            {
                "event": string,            // LDAP: "ldap-access";
HTTP: "http-access".
                "table": string,            // LDAP: "ldapaccess";
HTTP: "httpaccess".
                "fieldToColumn": {          // Map of field names
to database column names.
                    "event-field": "database-column" // Event-
field takes JSON pointer.
                }
            }
        ]
    }
}
```

For a sample configuration, see `opendj/config/audit-handlers/jdbc-config.json-example`.

The `writeInterval` takes a duration, which is a lapse of time expressed in English, such as `23 hours 59 minutes and 59 seconds`. Durations are not case sensitive. Negative durations are not supported. Durations use these units:

- `indefinite`, `infinity`, `undefined`, `unlimited`: unlimited duration
- `zero`, `disabled`: zero-length duration
- `days`, `day`, `d`: days

- `hours`, `hour`, `h`: hours
- `minutes`, `minute`, `min`, `m`: minutes
- `seconds`, `second`, `sec`, `s`: seconds
- `milliseconds`, `millisecond`, `millisec`, `millis`, `milli`, `ms`: milliseconds
- `microseconds`, `microsecond`, `microsec`, `micros`, `micro`, `us`: microseconds
- `nanoseconds`, `nanosecond`, `nanosec`, `nanos`, `nano`, `ns`: nanoseconds

JMS

A JMS handler is a JMS producer that publishes messages to an appropriately configured Java Message Service.

To enable the JMS handler, see Configure a Custom Access Log. The JSON configuration file for the JMS handler has the following format:

```
{
    "class":
"org.forgerock.audit.handlers.jms.JmsAuditEventHandler",
    "config": {
        "name": string,                    // Handler name, such
as "jms".
        "enabled": boolean,                // Is the handler
enabled?
        "topics": array,                   // LDAP: "ldap-access";
HTTP: "http-access".
        "deliveryMode": string,            // One of
"NON_PERSISTENT", "PERSISTENT".
        "sessionMode": string,             // One of "AUTO",
"CLIENT", "DUPS_OK".
        "batch": {                         // (Optional) Default:
Use default settings.
            "capacity": number,            // Maximum capacity of
publishing queue. Default: 1.
            "maxBatchedEvents": number,    // Maximum events to
deliver in single publishing call.
                                           // Default: 1.
            "writeInterval": string        // Interval between
transmissions to JMS.
                                           // Default: "10
millis".
        },
        "jndi": {                          // (Optional) Default:
Use default settings.
```

```
            "connectionFactoryName": string, // JNDI name for JMS
connection factory.
                                             // Default:
"ConnectionFactory".
            "topicName": string              // (Optional) Match the
value in the context.
                                             // Default: "audit".
            "contextProperties": {           // JNDI InitialContext
properties.
                // These depend on the JNDI provider. See the
provider documentation for details.
            }
        }
    }
}
```

For a sample configuration, see `opendj/config/audit-handlers/jms-config.json-example`.

Splunk

A Splunk handler sends messages to a Splunk service.

NOTE

This audit handler is deprecated.

To enable the Splunk handler, see Configure a Custom Access Log. The JSON configuration file for the Splunk handler has the following format:

```
{
    "class":
"org.forgerock.audit.handlers.splunk.SplunkAuditEventHandler",
    "config": {
        "name": string,                  // Handler name, such
as "splunk".
        "enabled": boolean,              // Is the handler
enabled?
        "topics": array,                 // LDAP: "ldap-access";
HTTP: "http-access".
        "authzToken": string,            // Splunk authorization
token for HTTP requests.
        "buffering": {                   // Required message
buffering configuration.
            "maxBatchedEvents": number,  // Maximum messages in
```

```
prepared statement.
            "maxSize": number,              // Maximum number of
buffered messages.
            "writeInterval": duration       // Duration as
described below.
        },
        "connection": {                     // (Optional) Default:
Use default settings.
            "host": string,                 // Splunk hostname.
Default: "localhost".
            "port": number,                 // Splunk port number.
Default: "8088".
            "useSSL": boolean               // Use secure
connection to Splunk? Default: false.
        }
    }
}
```

For a sample configuration, see `opendj/config/audit-handlers/splunk-config.json-example`.

The `writeInterval` takes a duration, which is a lapse of time expressed in English, such as `23 hours 59 minutes and 59 seconds`. Durations are not case sensitive. Negative durations are not supported. Durations use these units:

- `indefinite`, `infinity`, `undefined`, `unlimited`: unlimited duration
- `zero`, `disabled`: zero-length duration
- `days`, `day`, `d`: days
- `hours`, `hour`, `h`: hours
- `minutes`, `minute`, `min`, `m`: minutes
- `seconds`, `second`, `sec`, `s`: seconds
- `milliseconds`, `millisecond`, `millisec`, `millis`, `milli`, `ms`: milliseconds
- `microseconds`, `microsecond`, `microsec`, `micros`, `micro`, `us`: microseconds
- `nanoseconds`, `nanosecond`, `nanosec`, `nanos`, `nano`, `ns`: nanoseconds

Syslog

A Syslog handler sends messages to the UNIX system log as governed by RFC 5424, The Syslog Protocol.

NOTE

NOTE

The implementation currently only supports writing *access* messages, not error messages. As a result, this feature is of limited use in most deployments.

To enable a Syslog handler, see Configure a Custom Access Log. The JSON configuration file for the Syslog handler has the following format:

```
{
    "class":
"org.forgerock.audit.handlers.syslog.SyslogAuditEventHandler",
    "config": {
        "name": string,            // Handler name, such as
"syslog".
        "enabled": boolean,        // Default: false.
        "topics": array,           // LDAP: "ldap-access"; HTTP:
"http-access".
        "protocol": string,        // "TCP" or "UDP".
        "host": string,            // Syslog daemon host, such as
localhost;
                                   // must resolve to IP address.
        "port": number,            // Syslog daemon port number,
such as 514; range: 0 to 65535.
        "connectTimeout": number,  // If using TCP, milliseconds
to wait before timing out.
        "facility": string,        // Syslog facility to use for
event messages.
        "buffering": {             // (Optional) Default: write
each message separately, no buffering.
            "enabled": boolean,    // Buffer messages to be sent?
Default: false.
            "maxSize": number      // Maximum number of buffered
messages. Default: 5000.
        }
    }
}
```

For a sample configuration, see `opendj/config/audit-handlers/syslog-config.json-example`.

For additional details, see Syslog Facility Values.

Syslog Facility Values

Value	Description
`kern`	Kernel messages.

Value	Description
`user`	User-level messages.
`mail`	Mail system.
`daemon`	System daemons.
`auth`	Security/authorization messages.
`syslog`	Messages generated internally by `syslogd` .
`lpr`	Line printer subsystem.
`news`	Network news subsystem.
`uucp`	UUCP subsystem.
`cron`	Clock daemon.
`authpriv`	Security/authorization messages.
`ftp`	FTP daemon.
`ntp`	NTP subsystem.
`logaudit`	Log audit.
`logalert`	Log alert.
`clockd`	Clock daemon.
`local0`	Local use 0.
`local1`	Local use 1.
`local2`	Local use 2.
`local3`	Local use 3.
`local4`	Local use 4.
`local5`	Local use 5.
`local6`	Local use 6.
`local7`	Local use 7.

Manage Logs

Configure a Custom Access Log

This procedure applies only to Common Audit logs.

An access logger with a JSON configuration lets you use any Common Audit event handler, including customer handlers. The content of the configuration file depends on the audit event handler:

1. Decide whether to trust transaction IDs sent by client applications, used to correlate requests as they traverse multiple servers.

   Client applications using the ForgeRock Common Audit event framework send transaction IDs with their requests. The transaction IDs correlate audit events, tracing the request through multiple applications.

   Transaction IDs are sent over LDAP using an internal DS request control. They are sent over HTTP in an HTTP header.

   By default, DS servers do not trust transaction IDs sent with client application requests.

   When a server trusts transaction IDs from client application requests, outgoing requests reuse the incoming ID. For each outgoing request in the transaction, the request's transaction ID has the form *`original-transaction-id`*`/`*`sequence-number`*, where sequence-number reflects the position of the request in the series of requests for this transaction. For example, if the original-transaction-id is `abc123`, the first outgoing request has the transaction ID `abc123/0`, the second `abc123/1`, the third `abc123/2`, and so on. This lets you distinguish specific requests within a transaction when correlating audit events from multiple services.

   To trust transactions, set the advanced global server property, `trust-transaction-ids:true`:

   ```
   $ dsconfig \
    set-global-configuration-prop \
    --advanced \
    --hostname localhost \
    --port 4444 \
    --bindDN uid=admin \
    --bindPassword password \
    --set trust-transaction-ids:true \
   ```

```
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file
/path/to/opendj/config/keystore.pin \
 --no-prompt
```

2. Create the external JSON configuration file for the handler.

   Base your work on the appropriate template in the `config/audit-handlers` directory.

3. If this is a custom access logger provided separately, copy the custom handler .jar file to `opendj/lib/extensions` .

4. Create a log publisher configuration for the access log.

   The `type` defines whether the log contains messages about LDAP or HTTP requests:

   a. For LDAP access logging, create an external access log publisher:

```
$ dsconfig \
 create-log-publisher \
 --publisher-name "Custom LDAP Access Logger" \
 --type external-access \
 --set enabled:true \
 --set config-file:config/audit-handlers/handler-
conf.json \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore
\
 --trustStorePassword:file
/path/to/opendj/config/keystore.pin \
 --no-prompt
```

   b. For HTTP access logging, create an external HTTP access log publisher:

```
$ dsconfig \
 create-log-publisher \
 --publisher-name "Custom HTTP Access Logger" \
 --type external-http-access \
 --set enabled:true \
 --set config-file:config/audit-handlers/handler-
conf.json \
 --hostname localhost \
```

```
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore
\
 --trustStorePassword:file
/path/to/opendj/config/keystore.pin \
 --no-prompt
```

Log Access to Standard Output

This procedure applies only to Common Audit file-based logs, and when you install DS using procedures from Installation.

The sample DS Docker image creates these console access loggers by default:

- `Console LDAP Access Logger`
- `Console HTTP Access Logger`

A JSON stdout handler sends messages to standard output.

IMPORTANT

Only use this logger when running the server with **`start-ds --noDetach`**.

When running as a daemon without the `--noDetach` option, the server also logs the messages to the file, `/path/to/logs/server.out`. The server has no mechanism for rotating or removing the `server.out` log file, which is only cleared when the server starts.

As a result, using the JSON stdout handler when running the server without the `--noDetach` option can cause the server to eventually run out of disk space.

1. Enable the JSON stdout handler.

   For details, see Configure a Custom Access Log.

   The JSON configuration file for the JSON stdout handler has the following format:

```
{
  "class":
"org.forgerock.audit.handlers.json.stdout.JsonStdoutAuditE
ventHandler",
  "config": {
```

```
    "enabled": boolean,                    // Is the
handler enabled?
    "name": string,                        // Handler name,
such as "json.stdout".
    "elasticsearchCompatible" : boolean,   // If true, the
message ID field is named _eventId.
                                           // (Default:
_id)
    "topics": array,                       // LDAP: "ldap-
access"; HTTP: "http-access".
  }
}
```

For a sample configuration, see `opendj/config/audit-handlers/json-stdout-config.json-example`.

Log Errors to Standard Output

A `console-error` logger sends messages to standard output.

This procedure applies only when you install DS using procedures from Installation. The sample DS Docker image creates a `Console Error Logger` by default.

IMPORTANT

Only use this logger when running the server with **`start-ds --noDetach`**.

When running as a daemon without the `--noDetach` option, the server also logs the messages to the file, `/path/to/logs/server.out`. The server has no mechanism for rotating or removing the `server.out` log file, which is only cleared when the server starts.

As a result, using the JSON stdout handler when running the server without the `--noDetach` option can cause the server to eventually run out of disk space.

1. Switch to a `console-error` logger while the server is offline:

```
$ stop-ds

$ dsconfig \
 delete-log-publisher \
 --publisher-name "File-Based Error Logger" \
 --offline \
 --configFile /path/to/opendj/config/config.ldif \
 --no-prompt
```

```
$ dsconfig \
 create-log-publisher \
 --type console-error \
 --publisher-name "Console Error Logger" \
 --set enabled:true \
 --set default-severity:error \
 --set default-severity:warning \
 --set default-severity:notice \
 --offline \
 --configFile /path/to/opendj/config/config.ldif \
 --no-prompt

$ start-ds --noDetach
```

Rotate and Retain Logs

Each file-based log has a *rotation policy*, and a *retention policy*.

The rotation policy specifies when to rotate a log file based on a time, log file age, or log file size. Rotated logs have a rotation timestamp appended to their name.

The retention policy specifies whether to retain logs based on the number of logs, their size, or how much free space should be left on the disk.

1. List log rotation policies:

```
$ dsconfig \
 list-log-rotation-policies \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file
/path/to/opendj/config/keystore.pin \
 --no-prompt

Log Rotation Policy                 : Type       : file-
size-limit : rotation-interval : time-of-day
------------------------------------:------------:--------
---------:-------------------:------------
24 Hours Time Limit Rotation Policy : time-limit : -
```

```
: 1 d                : -
7 Days Time Limit Rotation Policy    : time-limit : -
: 1 w                : -
Fixed Time Rotation Policy           : fixed-time : -
: -                  : 2359
Size Limit Rotation Policy           : size-limit : 100 mb
: -                  : -
```

2. List log retention policies:

```
$ dsconfig \
 list-log-retention-policies \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file
/path/to/opendj/config/keystore.pin \
 --no-prompt

Log Retention Policy              : Type            : disk-
space-used : free-disk-space : number-of-files
----------------------------------:-----------------:------
-----------:-----------------:----------------
File Count Retention Policy       : file-count      : -
: -                : 10
Free Disk Space Retention Policy : free-disk-space : -
: 500 mb           : -
Size Limit Retention Policy       : size-limit      : 500
mb           : -                : -
```

3. View the policies that apply to a given log with the **`dsconfig get-log-publisher-prop`** command.

   The following example shows that the server keeps 10 access log files, rotating either each day or when the log size reaches 100 MB:

```
$ dsconfig \
 get-log-publisher-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "Json File-Based Access Logger" \
```

```
 --property retention-policy \
 --property rotation-policy \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file
/path/to/opendj/config/keystore.pin \
 --no-prompt

Property         : Value(s)
-----------------:-----------------------------------------
---------------------
retention-policy : File Count Retention Policy
rotation-policy  : 24 Hours Time Limit Rotation Policy,
Size Limit Rotation
                 : Policy
```

4. Use the **dsconfig** command to create, update, delete, and assign log rotation and retention policies. Set the policy that applies to a logger with the **dsconfig set-log-publisher-prop** command.

NOTE

When using access logs based on the ForgeRock Common Audit event framework, you can only configure one of each type of retention or rotation policy.

This means you can configure only one file count, free disk space, and size limit log retention policy. You can configure only one fixed time, size limit, and time limit log rotation policy.

Enable an Audit Log

1. Enable a file-based audit logger:

```
$ dsconfig \
 set-log-publisher-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "File-Based Audit Logger" \
 --set enabled:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file
/path/to/opendj/config/keystore.pin \
 --no-prompt
```

2. Wait for, or make a change to directory data.

   The following example changes a description:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file
/path/to/opendj/config/keystore.pin \
 --bindDN "uid=bjensen,ou=People,dc=example,dc=com" \
 --bindPassword hifalutin << EOF
dn: uid=bjensen,ou=People,dc=example,dc=com
changetype: modify
replace: description
description: New description
EOF
```

   The audit log records the changes as shown in the following excerpt:

```
# <datestamp>; conn=<number>; op=<number>
dn: cn=File-Based Audit Logger,cn=Loggers,cn=config
changetype: modify
replace: ds-cfg-enabled
ds-cfg-enabled: true
-

# <datestamp>; conn=<number>; op=<number>
dn: uid=bjensen,ou=people,dc=example,dc=com
changetype: modify
add: description
description: New description
-
```

   Audit logs record changes in LDIF format. This means that when an LDAP entry is deleted, the audit log records only its DN.

Filter Out Administrative Messages

A common development troubleshooting technique consists of sending client requests while tailing the access log:

```
$ tail -f /path/to/opendj/logs/ldap-access.audit.json
```

When the **dsconfig** command accesses the configuration, the access log records this. Such messages can prevent you from seeing the messages of interest from client applications.

You can filter access log messages for administrative connections to the administration port:

1. Configure access log filtering criteria:

```
$ dsconfig \
 create-access-log-filtering-criteria \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "Json File-Based Access Logger" \
 --criteria-name "Exclude LDAPS on 4444" \
 --type generic \
 --set connection-port-equal-to:4444 \
 --set connection-protocol-equal-to:ldaps \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file
/path/to/opendj/config/keystore.pin \
 --no-prompt
```

2. Activate filtering to exclude administrative messages:

```
$ dsconfig \
 set-log-publisher-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "Json File-Based Access Logger" \
 --set filtering-policy:exclusive \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file
/path/to/opendj/config/keystore.pin \
 --no-prompt
```

The publisher filters messages about administrative requests to the administration port.

Audit Configuration Changes

This example demonstrates how to set up an audit log file to track changes to the server configuration.

Audit log change records have timestamped comments with connection and operation IDs. You can use these to correlate the changes with messages in access logs:

1. Create an audit log publisher:

```
$ dsconfig \
 create-log-publisher \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "File-Based Server Configuration Audit
Log" \
 --type file-based-audit \
 --set enabled:true \
 --set filtering-policy:inclusive \
 --set log-file:logs/config-audit \
 --set rotation-policy:"24 Hours Time Limit Rotation
Policy" \
 --set rotation-policy:"Size Limit Rotation Policy" \
 --set retention-policy:"File Count Retention Policy" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file
/path/to/opendj/config/keystore.pin \
 --no-prompt
```

2. Create log filtering criteria for the logger that matches operations targeting `cn=config`:

```
$ dsconfig \
 create-access-log-filtering-criteria \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "File-Based Server Configuration Audit
```

```
Log" \
 --criteria-name "Record changes to cn=config" \
 --set request-target-dn-equal-to:"**,cn=config" \
 --set request-target-dn-equal-to:"cn=config" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file
/path/to/opendj/config/keystore.pin \
 --no-prompt
```

The server now writes to the audit log file, `/path/to/opendj/logs/config-audit` , whenever an administrator changes the server configuration. The following example output shows the resulting LDIF that defines the log filtering criteria:

```
# <datestamp>; conn=<id>; op=<id>
dn: cn=Record changes to cn=config,cn=Filtering
Criteria,cn=File-Based Server Configuration Audit
Log,cn=Loggers,cn=config
changetype: add
objectClass: top
objectClass: ds-cfg-access-log-filtering-criteria
cn: Record changes to cn=config
ds-cfg-request-target-dn-equal-to: **,cn=config
ds-cfg-request-target-dn-equal-to: cn=config
createTimestamp: <timestamp>
creatorsName: uid=admin
entryUUID: <uuid>
```

Allow Log Message Fields

1. When an object is passed in a Common Audit event, it might contain information that should not be logged. By default, the Common Audit implementation uses a whitelist to specify which fields of the event appear:

   a. For Common Audit HTTP access log publishers, edit the `log-field-whitelist` property.

   The following fields appear by default, with each field listed by its JSON path. You cannot change the default whitelist.

   If a whitelisted field contains an object, then listing the field means the whole object is whitelisted:

   - `/_id`

- `/timestamp`
- `/eventName`
- `/transactionId`
- `/trackingIds`
- `/userId`
- `/client`
- `/server`
- `/http/request/secure`
- `/http/request/method`
- `/http/request/path`
- `/http/request/headers/accept`
- `/http/request/headers/accept-api-version`
- `/http/request/headers/content-type`
- `/http/request/headers/host`
- `/http/request/headers/user-agent`
- `/http/request/headers/x-forwarded-for`
- `/http/request/headers/x-forwarded-host`
- `/http/request/headers/x-forwarded-port`
- `/http/request/headers/x-forwarded-proto`
- `/http/request/headers/x-original-uri`
- `/http/request/headers/x-real-ip`
- `/http/request/headers/x-request-id`
- `/http/request/headers/x-requested-with`
- `/http/request/headers/x-scheme`
- `/request`
- `/response`

For CSV logs, the values map to the column headers. The terms are separated by dots (`.`) rather than by slashes (`/`).

a. LDAP access loggers do not support whitelisting.

By default, all fields are whitelisted.

Deny Log Message Fields

When an object is passed in a Common Audit event, it might contain information that should not be logged. Loggers allow all fields that are safe to log by default. The whitelist is processed before the blacklist, so blacklist settings overwrite the whitelist defaults:

1. Blacklist individual fields in common audit access logs to prevent the fields from appearing in messages.

   The following example prevents all request headers from appearing in JSON HTTP access logs:

   ```
   $ dsconfig \
    set-log-publisher-prop \
    --hostname localhost \
    --port 4444 \
    --bindDN uid=admin \
    --bindPassword password \
    --publisher-name "Json File-Based HTTP Access Logger" \
    --set log-field-blacklist:/http/response/headers \
    --usePkcs12TrustStore /path/to/opendj/config/keystore \
    --trustStorePassword:file
   /path/to/opendj/config/keystore.pin \
    --no-prompt
   ```

   The blacklist values are JSON paths to the fields in log messages.

   For CSV logs, the blacklist values map to the column headers. The terms are separated by dots (`.`) rather than by slashes (`/`).

Make Tampering Evident

This procedure applies only to Common Audit-based logs.

Tamper-evident logging depends on a public key/private key pair and a secret key. The Common Audit framework accesses the keys in a JCEKS-type keystore. Follow these steps to prepare the keystore:

1. Create a password for the keystore.

   The examples below use an `AUDIT_KEYSTORE_PIN` environment variable that contains the password.

2. Generate a key pair in the keystore.

The keystore holds a signing key with the alias `Signature`. Generate the key with the `RSA` key algorithm, and the `SHA256withRSA` signature algorithm.

The following example uses the default file name:

```
$ keytool \
 -genkeypair \
 -keyalg RSA \
 -sigalg SHA256withRSA \
 -alias "Signature" \
 -dname "CN=ds.example.com,O=Example Corp,C=FR" \
 -keystore /path/to/opendj/config/audit-keystore \
 -storetype JCEKS \
 -storepass:env AUDIT_KEYSTORE_PIN \
 -keypass:env AUDIT_KEYSTORE_PIN
```

You can configure the file name with the log publisher `key-store-file` property.

3. Generate a secret key in the keystore.

   The keystore holds a symmetric key with the alias `Password`. Generate the key with the `HmacSHA256` key algorithm, and 256-bit key size.

   The following example uses the default file name:

```
$ keytool \
 -genseckey \
 -keyalg HmacSHA256 \
 -keysize 256 \
 -alias "Password" \
 -keystore /path/to/opendj/config/audit-keystore \
 -storetype JCEKS \
 -storepass:env AUDIT_KEYSTORE_PIN \
 -keypass:env AUDIT_KEYSTORE_PIN
```

   You can configure the file name with the log publisher `key-store-file` property.

4. Verify that the keystore contains signature and password keys:

```
$ keytool \
 -list \
 -keystore /path/to/opendj/config/audit-keystore \
 -storetype JCEKS \
 -storepass:env AUDIT_KEYSTORE_PIN
```

```
signature, <date>, PrivateKeyEntry,
<fingerprint>
password, <date>, SecretKeyEntry,
<fingerprint>
```

Was this helpful?