Administration

This guide is written for administrators who must manage and maintain Autonomous Identity.

ForgeRock® Autonomous Identity is an entitlements analytics system that lets you fully manage your company's access to your data.

An entitlement refers to the rights or privileges assigned to a user or thing for access to specific resources. A company can have millions of entitlements without a clear picture of what they are, what they do, and who they are assigned to. Autonomous Identity solves this problem by using advanced artificial intelligence (AI) and automation technology to determine the full entitlements landscape for your company. The system also detects potential risks arising from incorrect or over-provisioned entitlements that lead to policy violations. Autonomous Identity eliminates the manual re-certification of entitlements and provides a centralized, transparent, and contextual view of all access points within your company.

Quick Start

Stopping and Starting

Learn how to stop and start Autonomous Identity.

Backup and Restore

Learn how to back up and restore data.

Exporting and Importing Data

Learn how to export and import data.

Creating Users

Learn how to creating and remove users.

Configuring LDAP

Learn how to configure the LDAP repository.

Customize Domain

Learn how to change the domain name and target environment.

Configuring Filters Learn how to change the domain name and target environment.	**Change Vault Passwords** Learn how to change the vault passwords.	**Access Logs** Learn how to monitor Autonomous Identity using the logs.
Set Up Single Sign-On Learn how to set up single sign-on using OpenID Connect.	**Session Duration** Learn how to set the session duration.	**UI Tasks** Learn how to run admin tasks on the UI.
Data Preparation Learn how to prep your data for ingestion.	**Set Entity Definitions** Learn how to use the UI.	**Set Data Sources** Learn how to set data sources for ingestion.
Set Attribute Mappings Learn how to set attribute mappings.	**Set Analytics Thresholds** Learn how to set the analytics thresholds for machine learning.	**Run Analytics** Learn how to run the Analytics pipeline.

For installation instructions, see the Autonomous Identity Installation Guide.

For a description of the Autonomous Identity UI console, see the Autonomous Identity Users Guide.

Features

Autonomous Identity provides the following features:

- **Broad Support for Major Identity Governance and Administration (IGA) Providers**. Autonomous Identity supports a wide variety of Identity as a Service (IDaaS) and Identity Management (IDM) data including but not limited to comma-

separated values (CSV), Lightweight Directory Access Protocol (LDAP), human resources (HR), database, and IGA solutions.

- **Highly-Scalable Architecture**. Autonomous Identity deploys using a microservices architecture, either on-prem, cloud, or hybrid-cloud environments. Autonomous Identity's architecture supports scalable reads and writes for efficient processing.
- **Powerful UI dashboard**. Autonomous Identity displays your company's entitlements graphically on its UI console. You can immediately investigate those entitlement outliers as possible security risks. The UI also lets you quickly identify those entitlements that are good candidates for automated low-risk approvals or re-certifications. Users can also view a trend-line indicating how well they are managing their entitlements. The UI also provides an application-centric view and a single-page rules view for a different look at your entitlements.
- **Powerful Analytics Engine**. Autonomous Identity's analytics engine is capable of processing millions of access points within a short period of time. Autonomous Identity lets you configure the machine learning process and prune less productive rules. Customers can run analyses, predictions, and recommendations frequently to improve the machine learning process.
- **UI-Driven Schema Extension**. Autonomous Identity lets administrators discover and extend the schema, and set up attribute mappings using the UI.
- **UI-Driven Data Ingestion and Mappings**. Autonomous Identity provides improved data ingestion tools to define multiple csv input files needed for analysis and their attribute mappings to the schema using the UI.
- **UI-Driven Data Ingestion and Mappings**. Autonomous Identity provides improved data ingestion tools to define multiple csv input files needed for analysis and their attribute mappings to the schema using the UI.
- **Broad Database Support**. Autonomous Identity supports both Apache Cassandra and MongoDB databases. Both are highly distributed databases with wide usage throughout the industry.
- **Improved Search Support**. Autonomous Identity now incorporates Open Distro for Elasticsearch, a distributed, open-source search engine based on Lucene, to improve database search results and performance.

Stopping and Starting

The following commands are for Linux distributions.

Stopping Docker

> 1. Stop docker. This will shutdown all of the containers.

```
$ sudo systemctl stop docker
```

Restarting Docker

1. To restart docker, first set the docker to start on boot using the **enable** command.

```
$ sudo systemctl enable docker
```

2. To start docker, run the **start** command.

```
$ sudo systemctl start docker
```

Shutting Down Cassandra

1. On the deployer node, SSH to the target node.
2. Check Cassandra status.

```
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving —  Address
Load       Tokens       Owns (effective)  Host ID
Rack
UN  10.128.0.38  1.17 MiB   256          100.0%
d134e7f6-408e-43e5-bf8a-7adff055637a  rack1
```

3. To stop Cassandra, find the process ID and run the kill command.

```
$ pgrep -u autoid -f cassandra | xargs kill -9
```

4. Check the status again.

```
nodetool: Failed to connect to '127.0.0.1:7199' -
ConnectException: 'Connection refused (Connection
refused)'.
```

Re-Starting Cassandra

1. On the deployer node, SSH to the target node.
2. Restart Cassandra. When you see the `No gossip backlog; proceeding` message, hit **`Enter`** to continue.

```
$ cassandra

…
INFO  [main] 2020-11-10 17:22:49,306 Gossiper.java:1670 -
Waiting for gossip to settle…
INFO  [main] 2020-11-10 17:22:57,307 Gossiper.java:1701 -
No gossip backlog; proceeding
```

3. Check the status of Cassandra. You should see that it is in `UN` status ("Up" and "Normal").

```
$ nodetool status
```

Shutting Down MongoDB

1. Check the status of the MongDB

```
$ ps -ef | grep mongod
```

2. Connect to the Mongo shell.

```
$ mongo --tls --tlsCAFile
/opt/autoid/mongo/certs/rootCA.pem --tlsCertificateKeyFile
/opt/autoid/mongo/certs/mongodb.pem
    --tlsAllowInvalidHostnames --host <ip-address>

MongoDB shell version v4.2.9
connecting to: mongodb://<ip-address>:27017/?
compressors=disabled&gssapiServiceName=mongodb
2020-10-08T18:46:23.285+0000 W  NETWORK  [js] The server
certificate does not match the host name. Hostname: <ip-
address> does not match CN: mongonode
Implicit session: session { "id" : UUID("22c0123-30e3-
```

```
4dc9-9d16-5ec310e1ew7b") }
MongoDB server version: 4.2.9
```

3. Switch the admin table.

```
> use admin

switched to db admin
```

4. Authenticate using the password set in `vault.yml` file.

```
> db.auth("root", "Welcome123")

1
```

5. Start the shutdown process.

```
> db.shutdownServer()

2020-10-08T18:47:06.396+0000 I  NETWORK  [js]
DBClientConnection failed to receive message from <ip-
address>:27017 - SocketException: short read
server should be down…
2020-10-08T18:47:06.399+0000 I  NETWORK  [js] trying
reconnect to <ip-address>:27017 failed
2020-10-08T18:47:06.399+0000 I  NETWORK  [js] reconnect
<ip-address>:27017 failed
```

6. Exit the mongo shell.

```
$ quit()
or <Ctrl-C>
```

7. Check the status of the MongDB

```
$ ps -ef | grep mongod

no instance of mongod found
```

Re-Starting MongoDB

1. Re-start the MongoDB service.

```
$ /usr/bin/mongod --config /opt/autoid/mongo/mongo.conf

about to fork child process, waiting until server is ready
for connections.
forked process: 31227
child process started successfully, parent exiting
```

2. Check the status of the MongDB

```
$ ps -ef | grep mongod

autoid    9245     1  0 18:48 ?        00:00:45
/usr/bin/mongod --config /opt/autoid/mongo/mongo.conf
autoid   22003  6037  0 21:12 pts/1    00:00:00 grep --
color=auto mongod
```

Shutting Down Spark

1. On the deployer node, SSH to the target node.
2. Check Spark status. You should see that it is up-and-running.

```
$ elinks http://localhost:8080
```

3. Stop the Spark Master and workers.

```
$ /opt/autoid/spark/spark-2.4.4-bin-hadoop2.7/sbin/stop-
all.sh

localhost: stopping org.apache.spark.deploy.worker.Worker
stopping org.apache.spark.deploy.master.Master
```

4. Check the Spark status again. You should see: `Unable to retrieve htp://localhost:8080: Connection refused.`

Re-Starting Spark

1. On the deployer node, SSH to the target node.
2. Start the Spark Master and workers. Enter the user password on the target node when prompted.

```
$ /opt/autoid/spark/spark-2.4.4-bin-hadoop2.7/sbin/start-all.sh

starting org.apache.spark.deploy.master.Master, logging to /opt/autoid/spark/spark-2.4.4-bin-hadoop2.7/logs/spark-autoid-org.apache.spark.deploy.master.Master-1.out
autoid-2 password:
localhost: starting org.apache.spark.deploy.worker.Worker, logging to /opt/autoid/spark/spark-2.4.4-bin-hadoop2.7/logs/spark-autoid-org.apache.spark.deploy.worker.Worker-1.out
```

3. Check the Spark status again. You should see that it is up-and-running.

Backing Up and Restoring

Autonomous Identity stores its entitlement analytics results, association rules, predictions, and confidence scores in the Apache Cassandra, MongoDB, and Open Distro for Elasticsearch databases. Cassandra is an open-source, NoSQL database system where data is distributed across multiple nodes in a master-less cluster. MongoDB is a popular schema-free database that uses JSON-like documents. Open Distro for Elasticsearch is a distributed search engine based on Apache Lucene.

For single-node deployments, however, you need to back up Cassandra or MongoDB on a regular basis. If the machine goes down for any reason, you need to restore the database as required.

To simplify the backup process, ForgeRock provides backup and restore scripts in the target directory.

Backing Up Cassandra

1. On the ForgeRock Google Cloud Registry (gcr.io), download the `cassandra-backup.sh` script.
2. Move the script to the Cassandra home directory on your deployment.
3. Run the backup.

```
$ ./cassandra-backup.sh \
    -d <Cassandra Database path> \
    -b <Backup folder path> \
    -u <Cassandra Username> \
```

```
    -p <Cassandra Password> \
    -s <SSL enable true/false> \
    -k <Keyspace (optional) default value: zoran>
```

Restore Cassandra

1. On the ForgeRock Google Cloud Registry (gcr.io), download the `cassandra-restore.sh` script.
2. Move the script to the Cassandra home directory on your deployment.
3. Run the restore.

```
$ ./cassandra-restore.sh \
    -d <Cassandra Database path> \
    -b <Snapshot Backup tar file> \
    -f <Schema file> \
    -u <Cassandra Username> \
    -p <Cassandra Password> \
    -c <Cassandra commitlog path> \
    -i <Cassandra install path> \
    -s <SSL enable true/false> \
    -k <Keyspace (optional) default value: zoran>
```

Backing Up Assignment Index Data in Elasticsearch

1. From the deployer node, SSH to the target node.
2. Change to the `/opt/autoid/elastic` directory. The directory was configured during the `./deployer.sh run` .

```
$ cd /opt/autoid/elastic
```

3. Run the backup.

```
$ ./assignment-index-backup.sh

Elastic Host: 10.128.0.52
Elastic Server Status : 200
Elastic server is up and running …
assignment index exists status : 200
```

```
 registerSnapshotStatus 200
backup snapshot name with time stamp :
assignment_snapshot_2020_10_07__19_31_53
 entitlement-assignment backup status : 200
* entitlement-assignment backup successful *
```

4. Make note of the snapshot name. For example, `assignment_snapshot_2020_10_07__19_31_53` .

Restoring Assignment Index Data in Elasticsearch

1. From the deployer node, SSH to the target node.
2. Change to the `/opt/autoid/elastic` directory.

```
$ cd /opt/autoid/elastic
```

3. Run the restore using the snapshot taken from the previous procedure. When prompted if you want to close the existing index, enter `Y` . When prompted for the snapshot name, enter the name of the snapshot.

```
$ ./assignment-index-restore.sh

[Elastic Host: 10.128.0.55
 Elastic Server Status : 200
 Elastic server is up and running …
 assignment index exists status : 200
 index with alias name -→ entitlement-assignment exists
and is in open state…
 Do you want to close the existing index -→ entitlement-
assignment .(Required for restoring from snapshot ) (Y/N)
?
  y
 Restore snapshot ? true
  registerSnapshotStatus 200
 registering assignment_index_backup successful…
 proceeding with index restore…
 Enter the snapshot name to restore [snapshot_01]:
assignment_snapshot_2020_10_07 19_31_53
 snapshot to restore -→
assignment_snapshot_2020_10_07 19_31_53
 entitlement-assignment index restore status -→ 200
 * entitlement-assignment restore successful *
```

Accessing Elasticsearch Index Data using Kibana

During the Autonomous Identity deployment, Open Distro for Elasticsearch (ODFE) is installed to facilitate the efficient searching of entitlement data within the system. A typical deployment may have millions of different entitlements and assignments that require fast search processing. ODFE provides that performance.

ODFE comes bundled with its visualization console, Kibana, that lets you monitor and manage your Elasticsearch data. Once you run the **`analytics create-assignment-index`** command that populates the Elasticsearch index, you can configure an SSL tunnel to access Kibana. This is particularly useful when you want to retrieve a list of your backup snapshots.

1. Open a local terminal, and set up an SSL tunnel to your target node. The syntax is as follows:

   ```
   $ ssh -L < local-port >:<private-ip-remote>:<remote-port>
   -i <private-key> <user@public-ip-remote>
   ```

   For example:

   ```
   $ ssh -L 5601:10.128.0.71:5601 -i ~/.ssh/id_rsa
   autoid@34.70.190.144

   Last login: Fri Oct  9 20:10:59 2020
   ```

2. Open a browser and point it to `localhost:5601` Login in as `elasticadmin`. Enter your password that you set in the `~/autoid-config/vault.yml` file on the deployer node during install.
3. On the Elasticsearch page, click Explore on my own.
4. On the Elasticsearch Home page, click the menu in the top left corner, and click Dev Tools.
5. On the Dev Tools page, get a total count of indices.

   ```
   $ GET /entitlement-assignment/_count
   ```

6. On the Dev Tools page, search the indices.

   ```
   $ GET /entitlement-assignment/_search
   ```

7. On the Dev Tools page, get the list of snapshot backups.

```
$ GET /_cat/snapshots/assignment_index_backup
```

Exporting and Importing Data

Export Your Data

If you are migrating data, for example, from a development server to a QA server, then follow this section to export your data from your current deployment. Autonomous Identity provides a python script to export your data to .csv files and stores them to a folder in your home directory.

1. On the target machine, change to the `dbutils` directory.

   ```
   $ cd /opt/autoid/dbutils
   ```

2. Export the database.

   ```
   $ python dbutils.py export ~/backup
   ```

Import the Data into the Autonomous Identity Keyspace

If you are moving your data from another server, import your data to the target environment using the following steps.

1. First, create a `zoran_user.cql` file. This file is used to drop and re-create the Autonomous Identity `user` and `user_history` tables. The file should go to the same directory as the other .csv files. Make sure to create this file from the source node, for example, the development server, from where we exported the data.

   Start cqlsh in the source environment, and use the output of these commands to create the `zoran_user.cql` file:

   ```
   $ describe zoran.user;
   ```

   ```
   $ describe zoran.user_history;
   ```

Make sure the **DROP TABLE** cql commands precedes the **CREATE TABLE** commands as shown in the `zoran_user.cql` example file below:

```
USE zoran ;

DROP TABLE IF EXISTS  zoran.user_history ;

DROP TABLE IF EXISTS zoran.user ;

CREATE TABLE zoran.user (
    user text PRIMARY KEY,
    chiefyesno text,
    city text,
    costcenter text,
    isactive text,
    jobcodename text,
    lineofbusiness text,
    lineofbusinesssubgroup text,
    managername text,
    usrdepartmentname text,
    userdisplayname text,
    usremptype text,
    usrmanagerkey text
) WITH bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition':
'NONE'}
    AND comment = ''
    AND compaction = {'class':
'org.apache.cassandra.db.compaction.SizeTieredCompactionSt
rategy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '64',
'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';

CREATE TABLE zoran.user_history (
    user text,
    batch_id int,
```

```
    chiefyesno text,
    city text,
    costcenter text,
    isactive text,
    jobcodename text,
    lineofbusiness text,
    lineofbusinesssubgroup text,
    managername text,
    usrdepartmentname text,
    userdisplayname text,
    usremptype text,
    usrmanagerkey text,
    PRIMARY KEY (user, batch_id)
) WITH CLUSTERING ORDER BY (batch_id ASC)
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition':
'NONE'}
    AND comment = ''
    AND compaction = {'class':
'org.apache.cassandra.db.compaction.SizeTieredCompactionSt
rategy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '64',
'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';
```

2. Copy the `ui-config.json` from the source environment where you ran an analytics pipeline, usually under `/data/config` , to the same folder where you have your .csv files.
3. On the target machine, change to the `dbutils` directory.

```
$ cd /opt/autoid/dbutils
```

4. Use the **`dbutils.py import`** command to populate the Autonomous Identity keyspace with the .csv files, generated from the **`export`** command from the source environment using the previous steps. Note that before importing the data, the script truncates the existing tables to remove duplicates. Again, make sure the `zoran_user.cql` and the `ui-config.json` are in the `/import-dir` .

```
$ python dbutils.py import /import-dir
```

For example:

```
$ python dbutils.py import ~/import/AutoID-data
```

5. Verify that the data is imported in the directory on your server.

Creating and Removing Users

You can set up users within Autonomous Identity using the **`phpldapadmin`** command.

Log in to **`phpldapadmin`**

1. Make sure you have Autonomous Identity successfully installed and deployed in your environment.
2. Add the phpldapadmin URL to your `/etc/hosts` file. Add your specific IP address to the file.

```
<IP-Address>    autoid-openldap.forgerock.com
```

3. Access the phpldapadmin tool via your browser. Enter the following URL:

```
https://autoid-openldap.forgerock.com
```

4. On the phpldapadmin page, click login in the navigation bar on the left side.
5. On the Authenticate to server openldap page, enter `cn=admin`, `dc=zoran`, `dc=com`, and then enter your admin password. Click Authenticate to proceed.
6. On the left-hand navigation bar, expand the menu, and then click `ou=People`.
7. Under `ou=People`, select any user to see their profile, and then click Copy or move this entry.
8. On the Destination DN, change the name of the user to the user you want to add, and then click Copy. For example, let's create a new user: Mary Smith

```
cn=mary.smith@forgerock.com,ou=People,dc=zoran,dc=com
```

9. On the Create Object page, change the following fields, and then click Create Object.
    - displayName. `Mary Smith`
    - givenName. `Smith`
    - homeDirectory. `/home/users/mary.smith`
    - Password. Enter a password for this user.
    - sn. `Mary`
    - title. Enter a title: admin, supervisor, entitlement owner, or user.
    - uidNumber. Enter a unique `uid` number.
    - User Name. Enter `mary.smith`.
10. On the Create LDAP Entry page, review the entry, and click Commit.

Add User to a Group

The user that you created must be assigned to one of six groups: User, Supervisor, Executive, Entitlement Owner, Application Owner, and Admin.

1. On the phpldapadmin screen, click a user group. For this example, click `cn=Zoran User`.
2. Under uniqueMember, click add value, and then enter the user DN. For this example, enter `cn=mary.smith@forgerock.com,ou=People,dc=zoran,dc=com`.
3. Under uniqueMember, click Update Object.
4. Verify that you want to add the user under the New Value column, and then click Update Object.

Delete a User

1. On the phpldapadmin screen, click `ou=People` to expand it, and then click the user who you want to delete.
2. At the top, click Delete this entry.
3. Under uniqueMember, click Update Object.
4. Verify that you want to delete the user. Click Delete. The user will be removed from the branch and from the `ou=Groups` branch.

Configuring LDAP

Autonomous Identity installs an OpenLDAP Docker image on the target server to hold user data. Administrators can add or remove users or change their group privileges using the **`phpldapadmin`** command (see Creating and Removing Users).

You can configure the OpenLDAP repository specific to your environment using the `~/autoid-config/vars.yml` file.

1. Determine the LDAP domain, base DN, URL, group search base DN, and phpldapadmin port for your OpenLDAP repository.
2. On the deployer node, add the OpenLDAP configuration settings specific to your system to the `~/autoid-config/vars.yml` file:

```
openldap:
    ldap_domain: zoran.com
    ldap_base_dn: dc=zoran,dc=com
    ldap_url: ldap://openldap
    ldap_groupsearchbase: ou=Groups,dc=zoran,dc=com
    ldap: true
    phpldapadmin_port: 80
```

Customize the Domain and Namespace

By default, the Autonomous Identity URL and domain for the UI console is set to `autoid-ui.forgerock.com` , and the URL and domain for the self-service feature is `autoid-selfservice.forgerock.com` .

1. Customize the domain name and target environment by editing the `/autoid-config/vars.xml` file. By default, the domain name is set to `forgerock.com` and the target environment is set to `autoid`. The default Autonomous Identity URL will be: `https://autoid-ui.forgerock.com` . For example, we set the domain name to `abc.com` and the target environment to `myid`:

```
domain_name: forgerock.com
target_environment: autoid
```

2. If you set up your domain name and target environment in the previous step, you need to change the certificates to reflect the changes. Autonomous Identity generates

self-signed certificates for its default configuration. You must generate new certificates as follows:

a. Generate the private key (that is, `privatekey.pem`).

```
$ openssl genrsa 2048 > privatekey.pem
```

b. Generate the certificate signing request.

```
$ openssl req -new -key privatekey.pem -out csr.pem
```

c. Generate the Diffie-Hellman (DH) parameters file (dhparam4096.pem).

```
$ openssl dhparam -out dhparam4096.pem 4096
```

d. Create a self-signing certificate.

```
$ openssl x509 -req -days 365 -in csr.pem -signkey
privatekey.pem -out server.crt
```

e. Use your Certificate Authority (CA) to sign the certificate. The certificate must be `server.crt` .

f. Copy the files to the `/autoid-config/certs` directory.

g. Make the domain changes on your DNS server or update your `/etc/hosts` file locally on your machine.

Configuring Your Filters

The filters on the Applications pages let you focus your searches based on entitlement and user attributes. In most cases, the default filters should suffice for most environments. However, if you need to customize the filters, you can do so by accessing the configuration service API endpoint as show below.

The default filters for an entitlement are the following:

- Risk Level
- Criticality

The default filters for an user attributes are the following:

- User Department Name
- Line of Business Subgroup
- City

- Jobcode Name
- User Employee Type
- Chief Yes No
- Manager Name
- Line of Business
- Cost Center

Configure the Filters:

1. From the deployer node, SSH to the target node.
2. Run the **`curl`** command to retrieve the current filters configuration.

```
$ curl -i -k -u configadmin:<configadmin-password> --
header "Content-Type: application/json" --request GET
  https://autoid-configuration-
service.forgerock.com/api/configuration/AllowedAttributesF
orFiltering

{
  "entitlement": [
    "risk_level",
    "criticality",
    "owner"
  ],
  "user": [
    "usr_department_name",
    "line_of_business_subgroup",
    "city",
    "jobcode_name",
    "usr_emp_type",
    "chief_yes_no",
    "manager_name",
    "line_of_business",
    "cost_center"
  ]
}
```

3. Update the filters configuration. The syntax is as follows:

```
$ curl -i -k -u configadmin:<configadmin-password> \
      --request PUT \
      --header "Content-Type: application/json" \
      --data '{<UPDATED_FILTERING_JSON_DATA>}' \
```

```
    https://autoid-configuration-
service.forgerock.com/api/configuration/AllowedAttributesF
orFiltering
```

For example, update the filters list with fewer attributes:

```
$ curl -i -k -u configadmin:<configadmin-password> \
      --request PUT \
      --header "Content-Type: application/json"
      --data '{ "entitlement":
["risk_level","criticality","owner"], \
        "user":
["usr_department_name","line_of_business_subgroup","city",
"jobcode_name"]}' \
    https://autoid-configuration-
service.forgerock.com/api/configuration/AllowedAttributesF
orFiltering

configuration item updated
```

Change the Vault Passwords

Autonomous Identity uses the ansible vault to store passwords in encrypted files, rather than in plaintext. Autonomous Identity stores the vault file at `/autoid-config/vault.yml` saves the encrypted passwords to `/config/.autoid_vault_password` . The `/config/` mount is internal to the deployer container. The default encryption algorithm used is AES256.

By default, the `/autoid-config/vault.yml` file uses the following parameters:

```
configuration_service_vault:
  basic_auth_password: Welcome123

openldap_vault:
  openldap_password: Welcome123

cassandra_vault:
  cassandra_password: Welcome123
  cassandra_admin_password: Welcome123

mongo_vault:
  mongo_admin_password: Welcome123
```

```
  mongo_root_password: Welcome123

elastic_vault:
  elastic_admin_password: Welcome123
  elasticsearch_password: Welcome123
```

Assume that the vault file is encrypted during the installation. To edit the file:

1. Change to the `/autoid-config/` directory.

   ```
   $ cd ~/autoid-config/
   ```

2. First, decrypt the vault file.

   ```
   $ ./deployer.sh decrypt-vault
   ```

3. Open a text editor and edit the `vault.yml` file.
4. Encrypt the file again.

   ```
   $ ./deployer.sh encrypt-vault
   ```

Accessing Log Files

Autonomous Identity provides different log files to monitor or troubleshoot your system.

Getting Docker Container Information

1. On the target node, get system wide information about the Docker deployment. The information shows the number of containers running, paused, and stopped containers as well as other information about the deployment.

   ```
   $ docker info
   ```

2. If you want to get debug information, use the `-D` option. The option specifies that all docker commands will output additional debug information.

   ```
   $ docker -D info
   ```

3. Get information on all of your containers on your system.

   ```
   $ docker ps -a
   ```

4. Get information on the docker images on your system.

```
$ docker images
```

5. Get docker service information on your system.

```
$ docker service ls
```

6. Get docker the logs for a service.

```
$ docker service logs <service-name>
```

   For example, to see the nginx service:

```
$ docker service logs nginx_nginx
```

   Other useful arguments:

   - `--details`. Show extra details.
   - `--follow, -f`. Follow log output. The command will stream new output from STDOUT and STDERR.
   - `--no-trunc`. Do not truncate output.
   - `--tail {n|all}`. Show the number of lines from the end of log files, where `n` is the number of lines or `all` for all lines.
   - `--timestamps, -t`. Show timestamps.

Getting Cassandra Logs

The Apache Cassandra output log is kicked off at startup. Autonomous Identity pipes the output to a log file in the directory, `/opt/autoid/`.

1. On the target node, get the log file for the Cassandra install.

```
$ cat /opt/autoid/cassandra/installcassandra.log
```

2. Get startup information. Cassandra writes to `cassandra.out` at startup.

```
$ cat /opt/autoid/cassandra.out
```

3. Get the general Cassandra log file.

```
$ cat /opt/autoid/apache-cassandra-3.11.2/logs/system.log
```

By default, the log level is set to `INFO`. You can change the log level by editing the `/opt/autoid/apache-cassandra-3.11.2/conf/logback.xml` file. After any edits, the change will take effect immediately. No restart is necessary. The log levels from most to least verbose are as follows:

- TRACE
- DEBUG
- INFO
- WARN
- ERROR
- FATAL

4. Get the JVM garbage collector logs.

```
$ cat /opt/autoid/apache-cassandra-3.11.2/logs/gc.log.
<number>.current
```

For example:

```
$ cat /opt/autoid/apache-cassandra-
3.11.2/logs/gc.log.0.current
```

The output is configured in the `/opt/autoid/apache-cassandra-3.11.2/conf/cassandra-env.sh` file. Add the following JVM properties to enable them:

- `JVM_OPTS="$JVM_OPTS -XX:+PrintGCDetails"`
- `JVM_OPTS="$JVM_OPTS -XX:+PrintGCDateStamps"`
- `JVM_OPTS="$JVM_OPTS -XX:+PrintHeapAtGC"`
- `JVM_OPTS="$JVM_OPTS -XX:+PrintGCApplicationStoppedTime"`

5. Get the debug log.

```
$ cat /opt/autoid/apache-cassandra-3.11.2/logs/debug.log
```

Other Useful Cassandra Monitoring Tools and Files

Apache Cassandra has other useful monitoring tools that you can use to observe or diagnose and issue. To see the complete list of options, see the Apache Cassandra documentation.

1. View statistics for a cluster, such as IP address, load, number of tokens,

```
$ /opt/autoid/apache-cassandra-3.11.2/bin/nodetool status
```

2. View statistics for a node, such as uptime, load, key cache hit, rate, and other information.

```
$ /opt/autoid/apache-cassandra-3.11.2/bin/nodetool info
```

3. View the Cassandra configuration file to determine how properties are pre-set.

```
$ cat /opt/autoid/apache-cassandra-3.11.2/conf/cassandra.yaml
```

Apache Spark Logs

Apache Spark provides several ways to monitor the server after an analytics run.

1. To get an overall status of the Spark server, point your browser to `http://<spark-master-ip>:8080` .
2. Print the logging message sent to the output file during an analytics run.

```
$ cat /opt/autoid/spark/spark-2.4.4-bin-hadoop2.7/logs/<file-name>
```

   For example:

```
$ cat /opt/autoid/spark/spark-2.4.4-bin-hadoop2.7/logs/spark-org.apache.spark.deploy.master.Master-1-autonomous-id-test.out
```

3. Print the data logs that were written during an analytics run.

```
$ cat /data/log/files/<filename>
```

   For example:

```
$ cat /data/log/files/f6c0870e-5782-441e-b145-b0e662f05f79.log
```

Set Up Single Sign-On

Autonomous Identity supports single sign-on (SSO) using OpenID Connect (OIDC) JWT tokens. SSO lets you log in once and access multiple applications without the need to re-authenticate yourself. You can use any third-party identity provider (IdP) to connect to

Autonomous Identity. In this example, we use ForgeRock Access Management (AM) as an OpenID Connect (OIDC) IdP for Autonomous Identity.

NOTE

If you set up SSO-only, be aware that the following microservices are not deployed with this setting:

- openldap
- phpldapadmin
- self-service

If you want to use these microservices and SSO, set up the authentication as `"LdapAndSSO"`.

Set Up SSO Using ForgeRock AM

The following procedures requires a running instance of ForgeRock AM. For more information, see ForgeRock Access Management Quick Start Guide.

1. First, set up your hostnames locally in `/etc/hosts`.

```
35.189.75.99  autoid-ui.forgerock.com autoid-
selfservice.forgerock.com
35.246.65.234 openam.example.com
```

2. Open a browser and point to `http://openam.example.com:8080/openam`. Log in with username: `amadmin`, password: `cangetinam`.
3. On AM, go to Identities > Groups, and add the following groups:
    - AutoIdAdmin
    - AutoIdEntitlementOwner
    - AutoIdExecutive
    - AutoIdSupervisor
    - AutoIdUser
4. Add the `demo` user to each group.
5. Go back to the main AM Admin UI page. Click **`Configure OAuth Provider`**.
6. Click **`Configure OpenID Connect`**, and then **`Create`**.
7. Go to Applications > OAuth 2.0, and then click **`Add Client`**. Enter the following properties, specific to your deployment:

```
Client ID:          <autoid>
Client secret:      <password>
```

```
Redirection URIs:  https://<autoi-ui>.<domain>/api/sso/finish
Scope(s):          openid profile
```

For example:

```
Client ID:         autoid
Client secret:     Welcome123
Redirection URIs:  https://autoid-
ui.forgerock.com/api/sso/finish
Scope(s):          openid profile
```

8. On the New Client page, go to to the Advanced tab, and enable **`Implied Consent`**. Next, change the `Token Endpoint Authentication Method` to `client_secret_post`.
9. Edit the OIDC claims script to return `roles (groups)`, so that AM can match the Autonomous Identity groups.

```
"groups": { claim, identity -> [ "groups" :
identity.getMemberships(IdType.GROUP).collect { group ->
group.name }]}
```

For more information about the OIDC claims script, see the ForgeRock Knowledge Base.

10. The `id_token` returns the content that includes the group names.

```
{
  "at_hash": "QJRGiQgr1c1sOE4Q8BNyyg",
  "sub": "demo",
  "auditTrackingId": "59b6524d-8971-46da-9102-704694cae9bc-
48738",
  "iss": "http://openam.example.com:8080/openam/oauth2",
  "tokenName": "id_token",
  "groups": [
    "AutoIdAdmin",
    "AutoIdSupervisor",
    "AutoIdUser",
    "AutoIdExecutive",
    "AutoIdEntitlementOwner"
  ],
  "given_name": "demo",
  "aud": "autoid",
  "c_hash": "SoLsfc3zjGq9xF5mJG_C9w",
  "acr": "0",
```

```
  "org.forgerock.openidconnect.ops":
"B15A_wXm581fO8INtYHHcwSQtJI",
  "s_hash": "bOhtX8F73IMjSPeVAqxyTQ",
  "azp": "autoid",
  "auth_time": 1592390726,
  "name": "demo",
  "realm": "/",
  "exp": 1592394729,
  "tokenType": "JWTToken",
  "family_name": "demo",
  "iat": 1592391129,
  "email": "demo@example.com"
}
```

11. You have successfully configured AM as an OIDC provider. Next, we set up Autonomous Identity.
12. Change to the Autonomous Identity install directory on the deployer machine.

```
$ cd ~/autoid-config/
```

13. Open a text editor, and set the SSO parameters in the `/autoid-config/vars.yml` file. Make sure to change `LDAP` to `SSO`.

```
authentication_option: "SSO"

oidc_issuer: "http://openam.example.com:8080/openam/oauth2"
oidc_auth_url:
"http://openam.example.com:8080/openam/oauth2/authorize"
oidc_token_url:
"http://openam.example.com:8080/openam/oauth2/access_token"
oidc_user_info_url:
"http://openam.example.com:8080/openam/oauth2/userinfo"
oidc_jwks_url:
"http://openam.example.com:8080/openam/oauth2/connect/jwk_uri"
oidc_callback_url: "https://autoid-
ui.forgerock.com/api/sso/finish"
oidc_client_scope: 'openid profile'
oidc_groups_attribute: groups
oidc_uid_attribute: sub
oidc_client_id: autoid
oidc_client_secret: Welcome1
admin_object_id: AutoIdAdmin
entitlement_owner_object_id: AutoIdEntitlementOwner
executive_object_id: AutoIdExecutive
```

```
supervisor_object_id: AutoIdSupervisor
user_object_id: AutoIdUser
application_owner_object_id: AutoIDAppOwner
oidc_end_session_endpoint:
"http://openam.example.com:8080/openam/oauth2/logout"
oidc_logout_redirect_url:
"http://openam.example.com:8088/openman/logout"
```

14. On the Target machine, edit the `/etc/hosts` file, and add an entry for `openam.example.com` .

```
35.134.60.234  openam.example.com
```

15. On the Deployer machine, run **`deployer.sh`** to push the new configuration.

```
$ deployer.sh run
```

16. Test the connection now. Access `https://autoid-ui/forgerock.com` . The redirect should occur with the following:

```
http://openam.example.com:8080/openam/XUI/?
realm=%2F&goto=http%3A%2F%2Fopenam.example.com%3A8080%2Fopenam
%2Foauth2%2Fauthorize%3Fresponse_type%3Dcode%26client_id%3Daut
oid
```

Setting the Session Duration

By default, the session duration is set to 30 minutes. You can change this value at installation by setting the `JWT_EXPIRY` property in the `/autoid-config/vars.yml` file.

If you did not set the value at installation, you can make the change after installation by setting the `JWT_EXPIRY` property using the API service.

To set the session duration:

1. Log in to the Docker manager node.
2. Verify the `JWT_EXPIRY` property.

```
$ docker inspect api_zoran-api
```

3. Go to the API folder.

```
$ cd /opt/autoid/res/api
```

4. Edit the `docker-compose.yml` file and update the `JWT_EXPIRY` property. The `JWT_EXPIRY` property is set to minutes.
5. Redeploy the Docker stack API.

```
$ docker stack deploy --with-registry-auth --compose-file
docker-compose.yml api
```

If the command returns any errors, such as "image could not be accessed by the registry," then try the following command:

```
$ docker stack deploy --with-registry-auth --resolve-image
changed \
    --compose-file /opt/autoid/res/api/docker-compose.yml api
```

6. Verify the new `JWT_EXPIRY` property.

```
$ docker inspect api_zoran-api
```

7. Log in to the Docker worker node.
8. Stop the worker node.

```
$ docker stop [container ID]
```

The Docker manager node re-initiates the worker node. Repeat this step on any other worker node.

Admin User Tasks

The Admin user functionality is similar to that of a system administration *superuser*. Admin users have the access rights to company-wide entitlement data on the Autonomous Identity console. Admin users can approve or revoke a user's entitlement.

Performing Admin Tasks

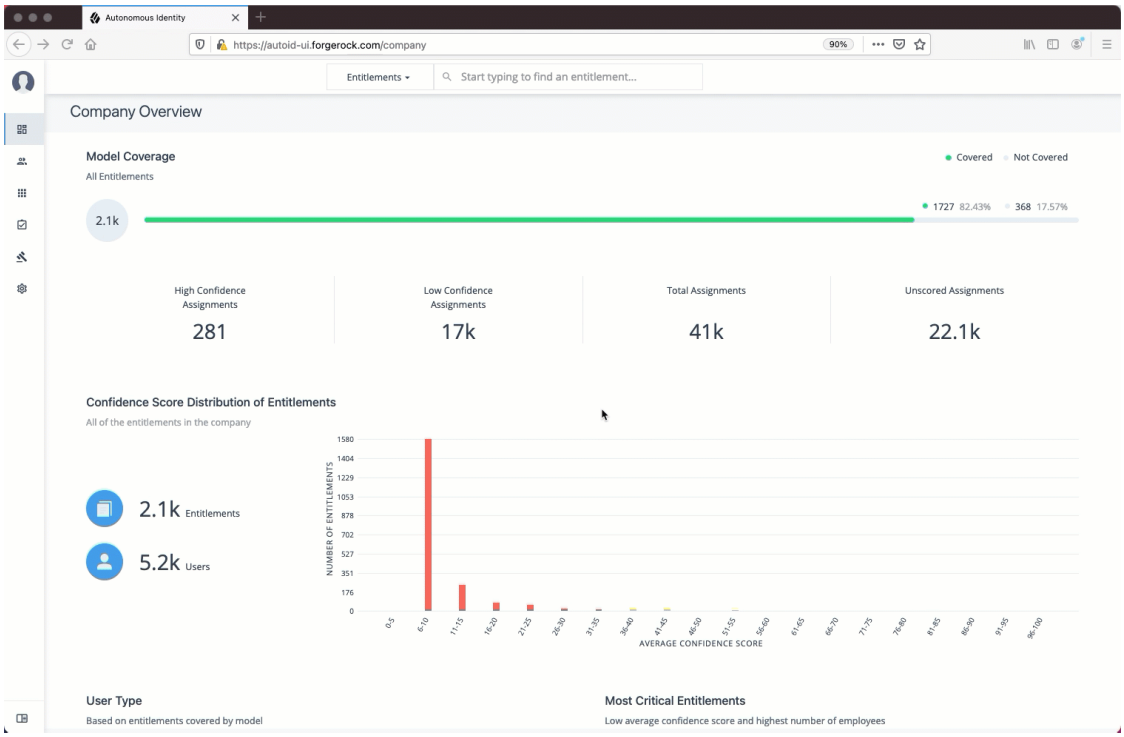
▼ Investigate Most Critical Entitlements

One important task that an administrator must perform is to examine all critical entitlements. Critical entitlements are assigned entitlements that have are highly-assigned but have a low confidence score associated with it. The Autonomous Identity console provides a means to examine these entitlements.

Follow these steps to evaluate the most critical entitlements list:

1. On the Dashboard, scroll down to the Most Critical Entitlements section. This section displays the entitlements that have low confidence scores and a high number of employees who have this entitlement.
2. Click an entitlement to view its details.
3. On the Entitlements detail page, review the key metrics.
4. Click the right arrow in one of the category ranges to view the users, and then click one of the users in the list.
5. On the User's Entitlements page, scroll down to review the Confidence Score Comparison table to see the differences between the user's attribute and the driving factor attributes.
6. Click Employees associated with this entitlement to review other uses who have this entitlement.
7. Click Actions, and then click Approve or Revoke for this entitlement. You can also bulk approve more than one entitlement. You can only revoke one entitlement at a time.

▼ *See it in action*



▼ Approve or Revoke Access an Entitlement for a User

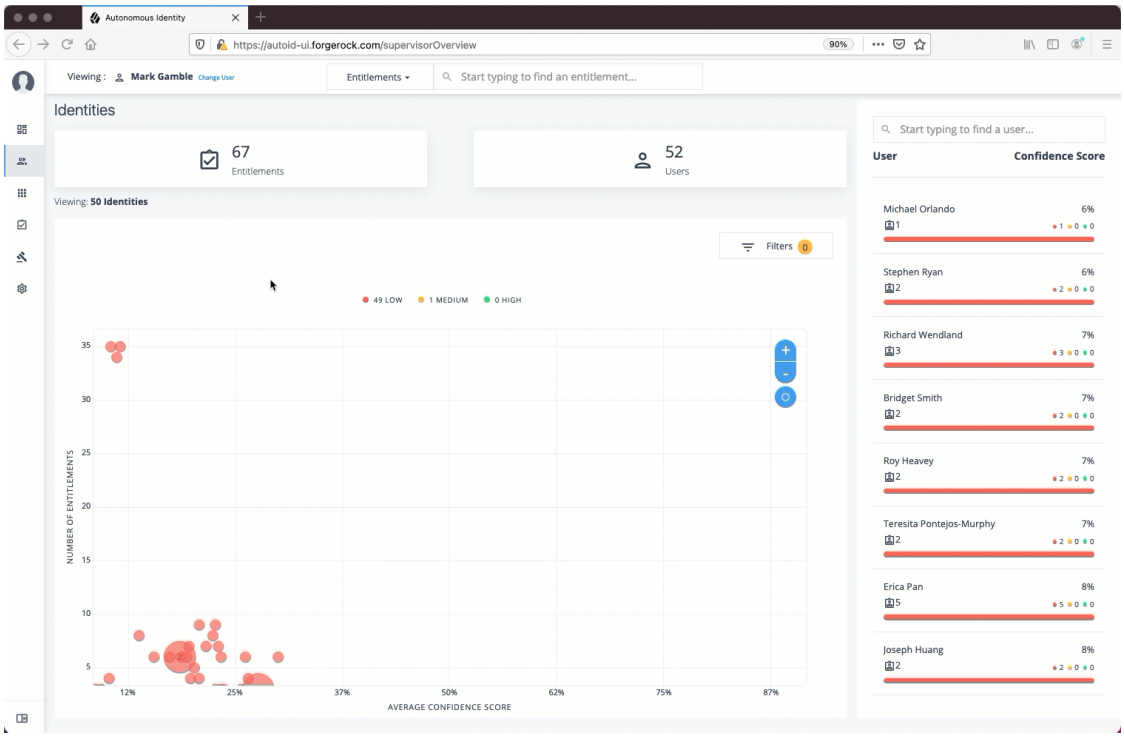
Follow these steps to investigate a confidence score and approve or revoke access an entitlement assigned to a specific user:

1. On Autonomous Identity console, click Identities, and enter a name of a supervisor. The only way to access a user's entitlements is through the Most

Critical Entitlements section or the Identities page.

2. On the Identities page, click a circle, and then click the user in the list on the right.
3. On the User Entitlement page, click a confidence circle on the graph to highlight the entitlement below.
4. For the selected entitlement, click the down arrow on the right to view the Driving Factor Comparison.
5. Click Employees associated with this entitlement to view the justifications for those users with high confidence scores with this entitlement.
6. Click Actions, and then click **Approve Access** or **Revoke access**. If you have more than one entitlement that you want to approve, select them all and do a bulk Approval. You can only do one Revoke Access at a time.

▼ *See it in action*



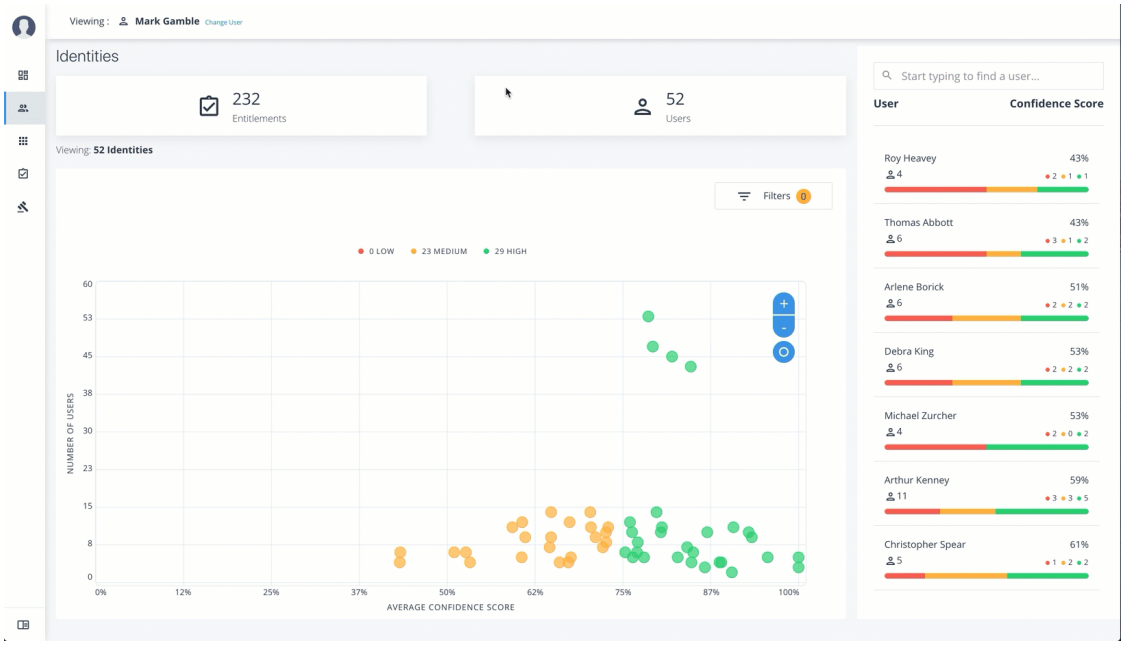
▼ Check Non-Scored Users

Follow these steps to check Not Scored entitlements. Not scored indicates that it does not have a justification associated with the entitlement:

1. On Autonomous Identity console, click Identities, and enter a name of a supervisor. The only way to access a user's entitlements is through the Most Critical Entitlements section or the Identities page.
2. On the Identities page, click a circle, and then click the user in the list on the right.

3. On the User Entitlement page, click Not Scored.
4. On the Not Scored Entitlements page, click the down arrow to view the driving factors comparison.
5. Click Employees associated with this entitlement to view the justifications for those users with high confidence scores with this entitlement.
6. Click Actions, and then click **`Approve Access`** or **`Revoke access`**. At a later date, you can re-click the Approve or Revoke button to cancel the operation.

▼ *See it in action*



▼ Apply Filters

Follow these steps to apply filters to your confidence score graphs on the Identities and Entitlements pages:

NOTE

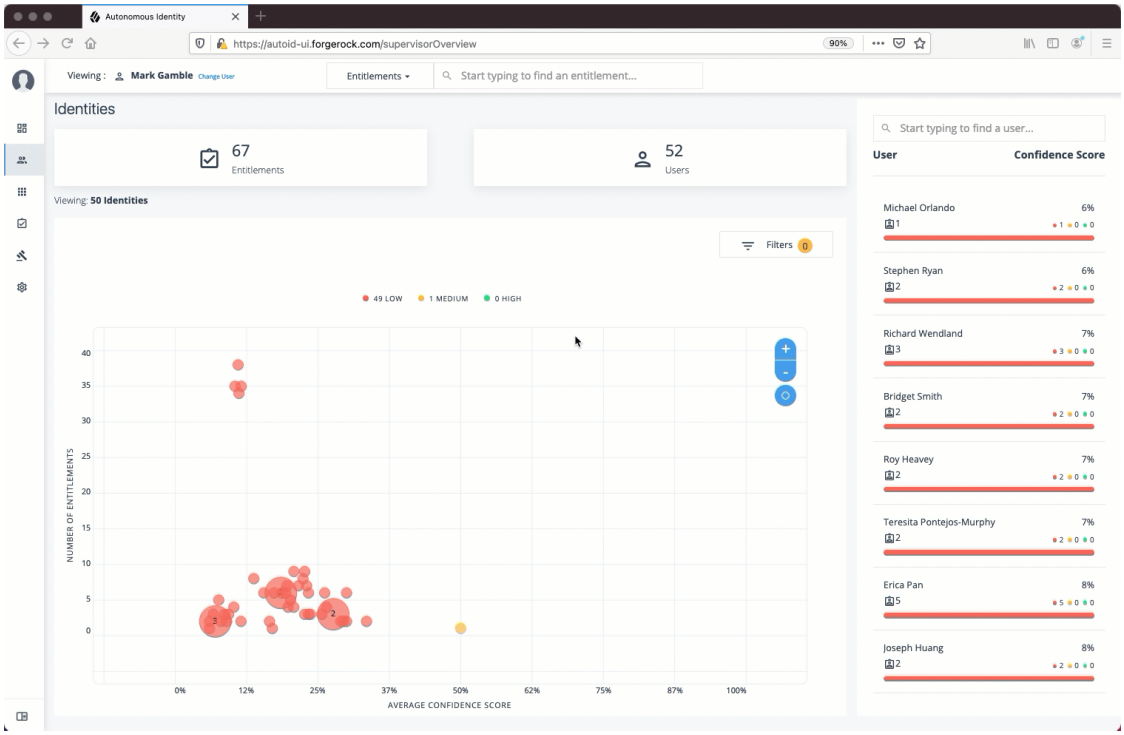
The Filters for the Identities and Entitlements are similar. The filters for the Applications and Rules pages offer different options to filter your searches.

1. On the Identities or Entitlements page, view the average confidence score graph.
2. On the right, click Filters.
3. Under filters, do one or all of the following:
    - Click **`Remove High Scores from Average`** or enable any filter in the Application Filters section.
    - Under Applications, click one or more applications to see the identities or entitlements asssociated with the selected application.

- Click Add Filters to further see only those identities or entitlements based on a user attribute, such as `city`. When ready, click Apply Filters.

4. Click Clear Filters to remove your filters.

▼ *See it in action*



Data Preparation

Once you have deployed Autonomous Identity, you can prepare your dataset into a format that meets the schema.

The initial step is to obtain the data as agreed upon between ForgeRock and your company. The files contain a subset of user attributes from the HR database and entitlement metadata required for the analysis. Only the attributes necessary for analysis are used.

There are a number of steps that must be carried out before your production entitlement data is input into Autonomous Identity. The summary of these steps are outlined below:

Data Collection

Typically, the raw client data is not in a form that meets the Autonomous Identity schema. For example, a unique user identifier can have multiple names, such as `user_id`, `account_id`, `user_key`, or `key`. Similarly, entitlement columns can have several names, such as `access_point`, `privilege_name`, or `entitlement`.

To get the correct format, here are some general rules:

- Submit the raw client data in `.csv` file format. The data can be in a single file or multiple files. Data includes application attributes, entitlement assignments, entitlements decriptions, and identities data.
- Duplicate values should be removed.
- Add optional columns for additional training attributes, for example, `MANAGERS_MANAGER` and `MANAGER_FLAG`. You can add these additional attributes to the schema using the Autonomous Identity UI. For more information, see Set Entity Definitions.
- Make a note of those attributes that differ from the Autonomous Identity schema, which is presented below. This is crucial for setting up your attribute mappings. For more information, see Set Attribute Mappings.

CSV Files and Schema

The required attributes for the schema are as follows:

CSV Files Schema

Files	Schema
applications.csv	This file depends on the attributes that the client wants to include. Here are some required columns: * **app_id**. Specifies the applications's unique ID. * **app_name**. Specifies the applications's name. * **app_owner_id**. Specifies the ID of the application's owner.
assignments.csv	* **user_id**. Specifies the unique user ID to which the entitlement is assigned. * **ent_id**. Specifies the entitlements's unique ID.
entitlements.csv	* **ent_id**. Specifies the entitlements's unique ID. * **ent_name**. Specifies the entitlement name. * **ent_owner_id**. Specifies the entitlement's owner. * **app_id**. Specifies the applications's unique ID.
identities.csv	* **usr_id**. Specifies the user's unique ID. * **user_name**. Specifies a human readable username. For example, `John Smith`. * **usr_manager_id**. Specifies the user's manager ID.

Set Entity Definitions

Before you run analytics, you can add new attributes to the schema using the Autonomous Identity UI. Only administrators have access to this functionality.

1. Open a browser. If you set up your own url, use it for your login.

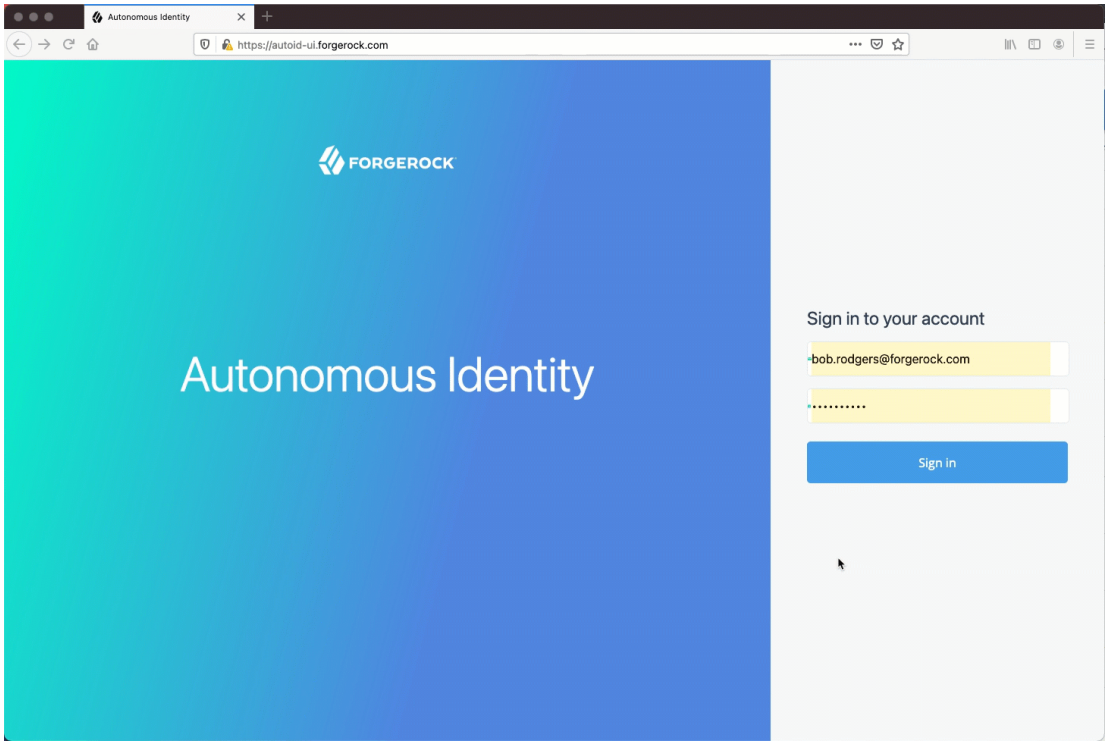
```
$ https://autoid-ui.forgerock.com/
```

2. Log in as an admin user, specific to your system. For example:

```
test user: bob.rodgers@forgerock.com
password: Welcome123
```

3. Click the Administration icon, indicated by the sprocket. Then, click **Entity Definitions**.
4. On the Entity Definitions page, click **Applications** to add any new attributes to the schema.
    a. Click the **Add attribute** button.
    b. In the Attribute Name field, enter the name of the new attribute. For example, `app_owner_id`.
    c. In the Display Name field, enter a human-readable name of the attribute.
    d. Select the attribute type. The options are: `Text`, `Boolean`, `Integer`, `Float`, `Date`, and `Number`.
    e. Click Searchable if you want the attribute to be indexed.
    f. Click Save.
    g. Click Save again to apply your attribute.
    h. If you need to edit the attribute, click the **Edit** icon. If you need to remove the attribute, click the **Remove** icon. Note that attributes marked as `Required` cannot be removed.
5. Click **Assignments**, and repeat the previous steps.
6. Click **Entitlements**, and repeat the previous steps.
7. Click **Identities**, and repeat the previous steps.

▼ *See it in action*



7. Next, you must set the data sources. See Set Data Sources.

Set Data Sources

After defining any new attributes, you must set your data sources for your CSV files, so that $Autonomous Identity can import your data. Currently, you can only specify CSV flat files. Future releases will introduce additional file types. This functionality is only available to administrators.

1. Log in to the Autonomous Identity UI as an administrator.
2. On the Autonomous Identity UI, click the Administration icon.
3. Click Data Sources.
    a. Click the **Add data sources** button.
    b. In the Add Data Source dialog box, click **CSV**, and then click Next.
    c. In the CSV Details dialog box, enter a human-readable name for your CSV file.
    d. In the **Applications** field, enter the path to the `application.csv` file. For example, `/data/input/applications.csv`.
    e. In the **Assignments** field, enter the path to the `assignments.csv` file. For example, `/data/input/assignments.csv`.

f. In the **Entitlements** field, enter the path to the `entitlements.csv` file. For example, `/data/input/entitlements.csv`.

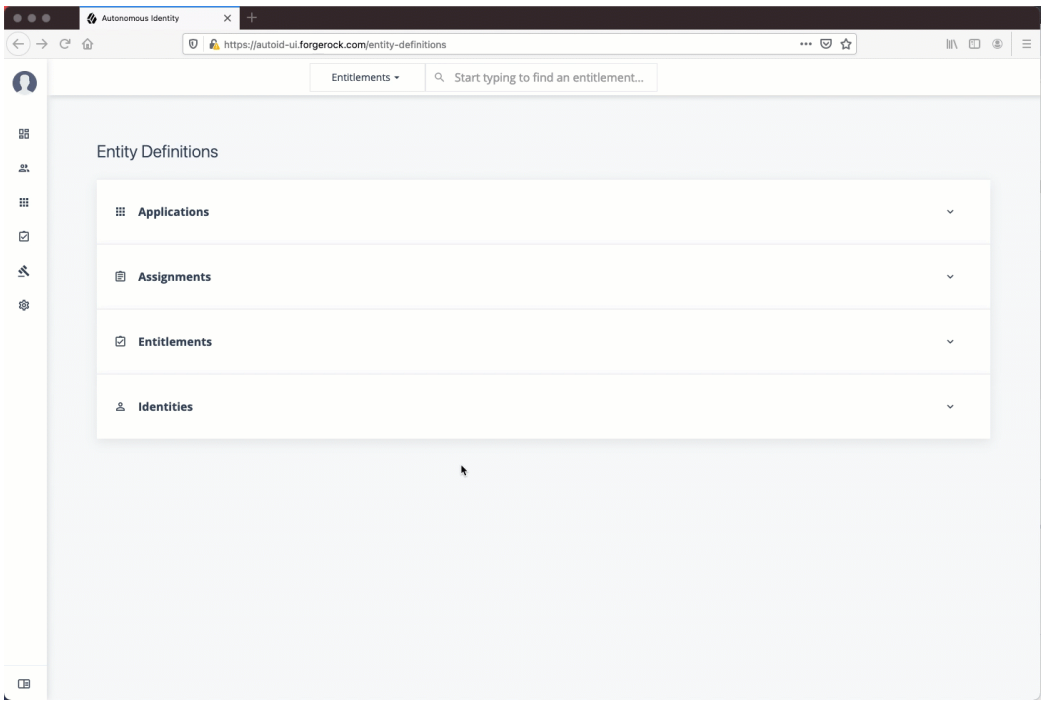
g. In the **Identities** field, enter the path to the `identities.csv` file. For example, `/data/input/identities.csv`.

h. Click Save.

i. Click **Activate** to apply your changes.

j. Repeat the previous steps to add more CSV data source files.

▼ *See it in action*



4. Next, you must set the attribute mappings. This is a critical step to ensure a successful analytics run. See Set Attribute Mappings.

Set Attribute Mappings

After setting your data sources for your CSV files, you must map any attributes specific to each of your data files to the Autonomous Identity schema.

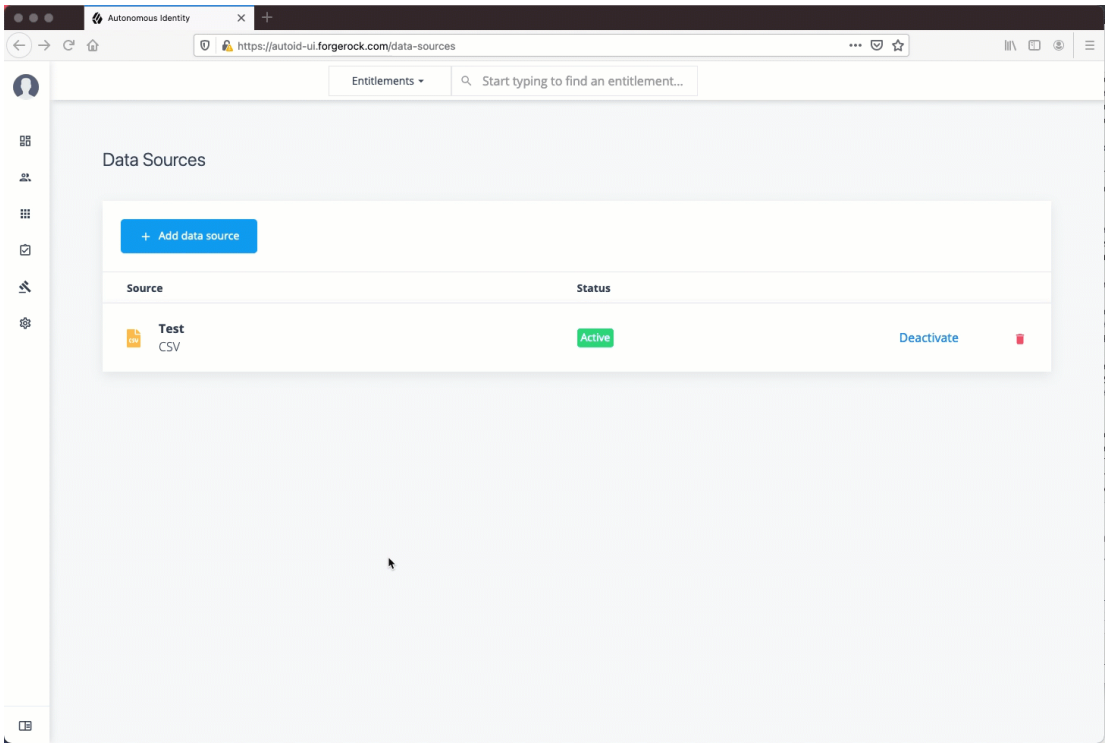
1. On the Autonomous Identity UI, click the Administration icon.
2. Click a specific csv file. The data files appear below on the page.
3. Click **Applications** to set up its attribute mappings.
    a. Click Discover Schema to view the current attributes in the schema, and then click Save.

b. Click Edit mapping to set up attribute mappings. On the Choose an attribute menu, select the corresponding attribute to map to the required attributes. Repeat for each attribute.

c. Click Save.

4. Click **Assignments** and repeat the previous steps.
5. Click **Entitlements** and repeat the previous steps.
6. Click **Identities** and repeat the previous steps.
7. Repeat the procedures for each data source file.

▼ *See it in action*



8. Optional. Next, adjust the analytics thresholds. See Set Analytic Thresholds.

Set Analytics Thresholds

The Autonomous Identity UI now supports the configuration of the analytics threshold values to calculate confidence scores, predications, and recommendations.

WARNING

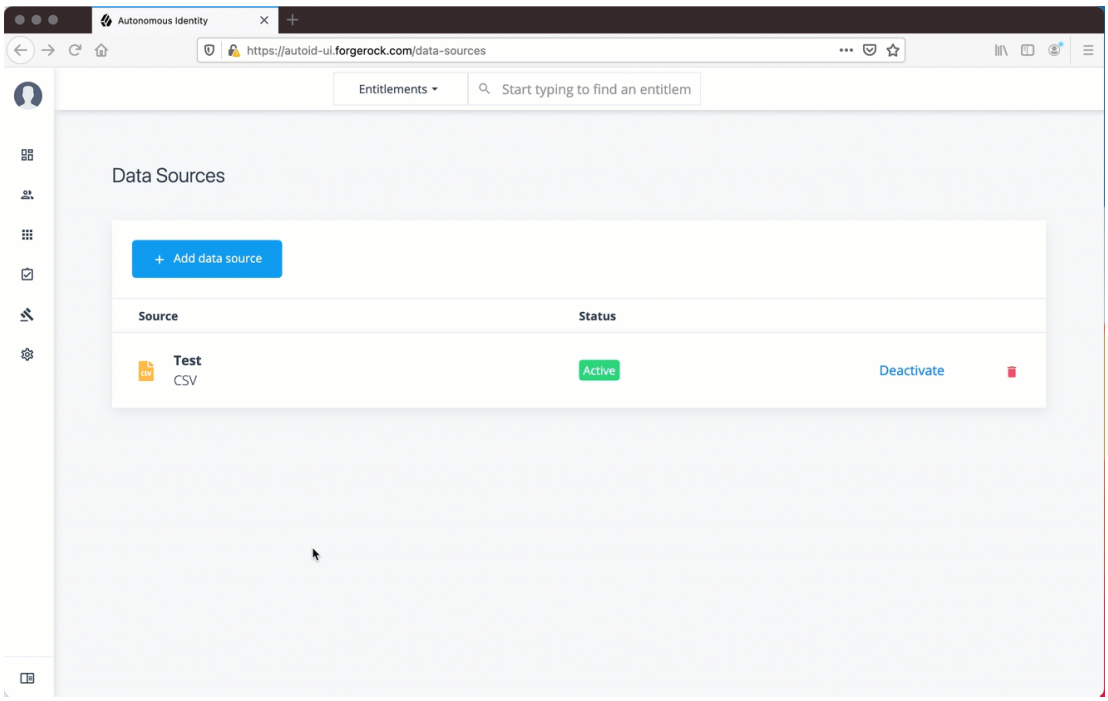
In general, there is little reason to change the default threshold values. If you do edit these values, be aware that incorrect threshold values could ruin your analytics.

There are three types of threshold values that you can edit:

- **Confidence Score Threshold**. These thresholds are used to identify High, Medium, and Low confidence access. Autonomous Identity computes a confidence score for each access assignment based on its machine learning algorithm. An administrator specifies the thresholds in the UI and API to categorize an assignment as having high, medium, or low confidence.
- **Automation Threshold**. Administrators can review access patterns and make a decision on which access pattern to automate for provisioning, approvals, and certifications. The automation threshold controls which access pattern are available for automation.
- **Recommendation Threshold**. Approvers and certifiers look to Autonomous Identity to provide a recommendation on whether access should be approved or certified. This recommendation relies on a confidence score threshold. Confidence scores above the threshold result is a positive recommendation, while those below that threshold result are not recommended.

1. Log in to the Autonomous Identity UI as an administrator.
2. On the Autonomous Identity UI, click Administration.
3. Click Analytics Settings.
4. Under Confidence Score Thresholds, click Edit next to the High threshold value, and then enter a new value. Click Save. Repeat for the Medium threshold value.
5. Under Automation Score Threshold, click Edit next to the threshold value, and then enter a new value.
6. Under Recommendation Threshold, click Edit next to the threshold value, and then enter a new value.

▼ *See it in action*



> 7. Next, you can run the analytics. See Run Analytics.

Run the Analytics Pipeline

The Analytics pipeline is the heart of Autonomous Identity. The pipeline analyzes, calculates, and determines the association rules, confidence scores, predictions, and recommendations for assigning entitlements to the users.

The analytics pipeline is an intensive processing operation that can take time depending on your dataset and configuration. To ensure an accurate analysis, the data needs to be as complete as possible with little or no null values. Once you have prepared the data, you must run a series of analytics jobs to ensure an accurate rendering of the entitlements and confidence scores.

Before running the analytics, you must run the following pre-analytics steps to set up your datasets and schema using the Autonomous Identity UI:

- Add attributes to the schema. For more information, see Set Entity Definitions.
- Define your CSV datasources. You can enter more than one data source, specifying the dataset location on your target machine. For more information, see Set Data Sources.
- Define attribute mappings between your data and the schema. For more information, see Set Attribute Mappings.
- Configure your analytics threshold values. For more information, see Set Analytics Thresholds.

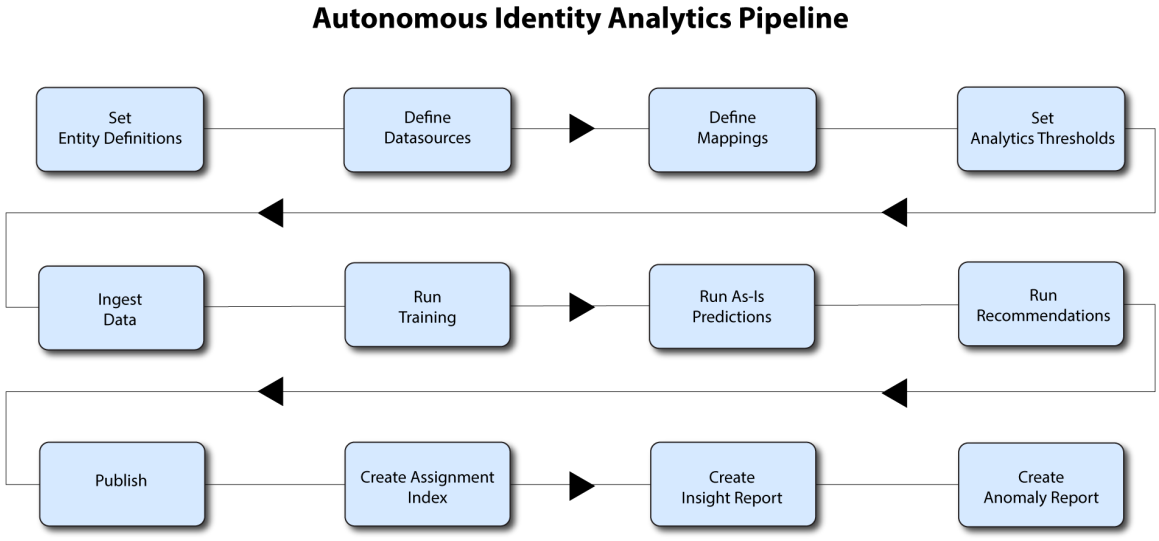
Once you have finished the pre-analytics steps, you can start the analytics. First, ingest the data into the database. After that, run the data through its initial training process to create the association rules for each user-assigned entitlement. This is a somewhat intensive operation as the analytics generates a million or more association rules. Once the association rules have been determined, they are applied to user-assigned entitlements.

After the training run, run predictions to determine the current confidence scores for all assigned entitlements. After predictions, run a recommendations job that looks at all users who do not have a specific entitlement but should, based on their user attribute data.

After this, publish the data to the backend Cassandra or MongoDB database, and then create the Autonomous Identity index.

The final steps is to run an insight report to get a summary of the analytics pipeline run, and an anomaly report that reports any anomalous entitlement assignments.

The general analytics process is outlined as follows:



NOTE

The analytics pipeline requires that DNS properly resolve the hostname before its start. Make sure to set it on your DNS server or locally in your `/etc/hosts` file.

Ingest the Data Files

At this point, you should have set your data sources and configured your attribute mappings. You can now run the initial analytics job to import the data into the Cassandra or MongoDB database.

1. SSH to the target machine.
2. Change to the analytics directory:

```
$ cd /opt/autoid/apache-livy/analytics
```

3. View the default Spark memory settings and available analytics commands.

```
$ analytics --help
driverMemory=2g
driverCores=3
executorCores=6
executorMemory=3G
livy_url=http://<IP-Address>:8998

Usage: analytics <command>
```

```
Commands:
  audit
  ingest
  train
  predict-as-is
  predict-recommendation
  publish
  create-assignment-index
  create-assignment-index-report
  anomaly
  insight
```

4. If you want to make changes to your Spark memory settings, edit the `analytics.cfg` file, and then save it.
5. Run the data ingestion command.

```
$ analytics ingest
```

The analytics job displays the JSON output log during the process and refreshes every 15 seconds. When the job completes, you should see the following JSON output at the end of the log if the job completed successfully:

```
      ...
      "+ Executing Load for Base Dataframes",
      "  - identities: Success",
      "  - entitlements: Success",
      "  - assignments: Success",
      "  - applications: Success",
      "",
      "Time to Complete: 48.7060821056366s",
      "
      "
stderr: "
   ],
   "total": 50
}
```

Run Training

After you have ingested the data into Autonomous Identity, start the training run.

Training involves two steps:

- Autonomous Identity starts an initial machine learning run where it analyzes the data and produces association rules, which are relationships discovered within your large set of data. In a typical deployment, you can have several million generated rules. The training process can take time depending on the size of your data set.
- Each of these rules are mapped from the user attributes to the entitlements and assigned a confidence score.

The initial training run may take time as it goes through the analysis process. Once it completes, it saves the results directly to the database.

1. Run the training command.

```
$ analytics train
```

The analytics job displays the JSON output log during the process and refreshes every 15 seconds. When the job completes, you should see the following JSON output at the end of the log if the job completed successfully:

```
        ...
        "+ Training Chunk Range: {'min': 3, 'max': 160,
'min_group': 3}",
        "  - Train Batch: 1-1 (15000 entitlements per
batch)...",
        "+ Training Chunk Range: {'min': 161, 'max': 300,
'min_group': 4}",
        "  - Train Batch: 2-1 (15000 entitlements per
batch)...",
        "+ Training Chunk Range: {'min': 301, 'max': 500,
'min_group': 5}",
        "  - Train Batch: 3-1 (15000 entitlements per
batch)...",
        "",
        "Time to Complete: 157.52734184265137s",
        "
stderr: "
    ],
    "total": 52
}
```

Run As-Is Predictions

After your initial training run, the association rules are saved to disk. The next phase is to use these rules as a basis for the predictions module.

The predictions module is comprised of two different processes:

- **as-is**. During the As-Is Prediction process, confidence scores are assigned to the entitlements that users do not have. The as-is process maps the highest confidence score to the highest `freqUnion` rule for each user-entitlement access. These rules will then be displayed in the UI and saved directly to the database.
- **Recommendations**. See Run Recommendations.

1. Run the as-is predictions command.

```
$ analytics predict-as-is
```

The analytics job displays the JSON output log during the process and refreshes every 15 seconds. When the job completes, you should see the following JSON output at the end of the log if the job completed successfully:

```
        ...
        "- As-Is Batch: 1 (15000 entitlements per
batch)...",
        "",
        "Time to Complete: 86.35014176368713s",
        "
stderr: "
    ],
    "total": 47
}
```

Run Recommendations

During the second phase of the predictions process, the recommendations process analyzes each employee who may not have a particular entitlement and predicts the access rights that they should have according to their high confidence score justifications. These rules will then be displayed in the UI and saved directly to the database.

1. Run the recommendations command.

```
$ analytics predict-recommendation
```

The analytics job displays the JSON output log during the process and refreshes every 15 seconds. When the job completes, you should see the following JSON output at the end of the log if the job completed successfully:

```
        ...
        "- Recommend Batch: 1 (1000 features per
batch)...",
        "- Recommend Batch: 2 (1000 features per
batch)...",
        "- Recommend Batch: 3 (1000 features per
batch)...",
        "- Recommend Batch: 4 (1000 features per
batch)...",
        "- Recommend Batch: 5 (1000 features per
batch)...",
        "- Recommend Batch: 6 (1000 features per
batch)...",
        "",
        "Time to Complete: 677.2908701896667s",
        "
stderr: "
    ],
    "total": 52
}
```

Publish the Analytics Data

Populate the output of the training, predictions, and recommendation runs to a large table with all assignments and justifications for each assignment. The table data is then pushed to the Cassandra or MongoDB backend.

1. Run the publish command.

```
$ analytics publish
```

The analytics job displays the JSON output log during the process and refreshes every 15 seconds. When the job completes, you should see the following JSON output at the end of the log if the job completed successfully:

```
    …
    "+ Creating backups of existing table data",
       "  - backup: api.entitlement_average_conf_score",
```

```
        "  - backup:
api.entitlement_assignment_conf_summary",
        "  - backup: api.entitlement_driving_factor",
        "  - backup: api.search_user",
        "",
        "+ Backup Results",
        "                                        Status",
        "Tables                                        ",
        "entitlement_average_conf_score       Success",
        "entitlement_assignment_conf_summary  Success",
        "entitlement_driving_factor           Success",
        "search_user                          Success",
        "+ Building dataframes for table load",
        "  - Building dataframe:
api.entitlement_average_conf_score",
        "    - write: api.entitlement_average_conf_score",
        "  - Building dataframe:
api.entitlement_assignment_conf_summary",
        "    - write:
api.entitlement_assignment_conf_summary",
        "  - Building dataframe:
api.entitlement_driving_factor",
        "    - write: api.entitlement_driving_factor",
        "  - Building dataframe: api.search_user",
        "    - write: api.search_user",
        "",
        "+ Load Results",
        "                                        Status",
        "Tables                                        ",
        "entitlement_average_conf_score       Success",
        "entitlement_assignment_conf_summary  Success",
        "entitlement_driving_factor           Success",
        "search_user                          Success",
        "",
        "Time to Complete: 71.09975361824036s",
        "
stderr: "
    ],
    "total": 75
}
```


Create Assignment Index

Next, generate the Elasticsearch index using the **`analytics create-assignment-index`** command.

1. Run the create-assignment-index command.

   ```
   $ analytics create-assignment-index
   ```

   The analytics job displays the JSON output log during the process and refreshes every 15 seconds. When the job completes, ou should see the following JSON output near the end of the log if the job completed successfully:

   ```
   …
   "21/03/10 06:48:18 INFO EntitlementAssignmentIndex: ------
   -----JOB COMPLETED-----------",
   …
   ```

2. Optional. Run the create-assignment-index-report command. The command writes the entire index data to a file at `/data/report/assignment-index`. Note that the file can get quite large but is useful to back up your csv data.

   ```
   $ analytics create-assignment-index-report
   ```

Run Insight Report

Next, run an insight report on the generated rules and predictions that were generated during the training and predictions runs. The analytics command generates `insight_report.txt` and `insight_report.xlsx` and writes them to the `/data/input/spark_runs/reports` directory.

The report provides the following insights:

- Number of assignments received, scored, and unscored.
- Number of entitlements received, scored, and unscored.
- Number of assignments scored >80% and <5%.
- Distribution of assignment confidence scores.
- List of the high volume, high average confidence entitlements.
- List of the high volume, low average confidence entitlements.
- Top 25 users with more than 10 entitlements.
- Top 25 users with more than 10 entitlements and confidence scores greater than 80%.

- Top 25 users with more than 10 entitlements and confidence scores less than 5%.
- Breakdown of all applications and confidence scores of their assignments.
- Supervisors with most employees and confidence scores of their assignments.
- Top 50 role owners by number of assignments.
- List of the "Golden Rules", high confidence justifications that apply to a large volume of people.

1. Run the insight command.

```
$ analytics insight
```

The analytics job displays the JSON output log during the process and refreshes every 15 seconds. When the job completes, you should see the following JSON output at the end of the log if the job completed successfully:

```
        …
        "Time to Complete: 47.645989656448364s",
        "
stderr: "
    ],
    "total": 53
}
```

2. Access the insight report. The report is available at `/data/output/reports` in `.xlsx` format.

Run Anomaly Report

Autonomous Identity provides a report on any anomalous entitlement *assignments* that have a low confidence score but are for entitlements that have a high average confidence score. The report's purpose is to identify true anomalies rather than poorly managed entitlements.

The report generates the following points:

- Identifies potential anomalous assignments.
- Identifies the number of users who fall below a low confidence score threshold. For example, if 100 people all have low confidence score assignments to the same entitlement, then it is likely not an anomaly. The entitlement is either missing data or the assignment is poorly managed.

1. Run the anomaly command to generate the report.

```
$ analytics anomaly
```

The analytics job displays the JSON output log during the process and refreshes every 15 seconds. When the job completes, you should see the following JSON output at the end of the log if the job completed successfully:

```
        "",
        "Time to Complete: 97.65438652038574s",
        "
stderr: "
    ],
    "total": 48
}
```

2. Access the anomaly report. The report is available at `/data/output/reports` in `.csv` format.

Appendix A: Default User Permissions

Table: Summary of Default Autonomous Identity Users Permissions

Permission	App Owner	Entitlement Owner	Executive	Supervisor	User
SHOW__APPLICATION_PAGE	✔				
SHOW__USER	✔	✔		✔	✔
SEARCH__USER	✔		✔	✔	
SEARCH__USER_ENTITLEMENTS_BY_APP_OWNER	✔				
SEARCH__USER_ENTITLEMENTS_BY_ENTT_OWNER		✔			
SHOW__OVERVIEW_PAGE	✔	✔		✔	
SHOW__ASSIGNMENTS_STATS			✔		
SHOW__COMPANY_PAGE			✔		

Permission	App Owner	Entitlement Owner	Executive	Supervisor	User
SHOW__COMPANY_ENTITLEMENTS_DATA			✔		
SHOW__ENTITLEMENTS	✔	✔		✔	✔
SHOW__ENTITLEMENT_USERS	✔	✔			
SHOW__CRITICAL_ENTITLEMENTS			✔		
SHOW__ENTITLEMENT_AVG_GROUPS			✔		
SHOW__APP_OWNER_FILTER_OPTIONS	✔				
SHOW__ENTT_OWNER_FILTER_OPTIONS		✔			
SHOW__SUPERVISOR_FILTER_OPTIONS				✔	
SHOW__ENTT_OWNER_UNSCORED_ENTITLEMENTS	✔	✔			
SHOW__ENTT_OWNER_PAGE	✔	✔			
SHOW__ENTT_OWNER_ENT_PAGE	✔	✔			
SHOW__ENTT_OWNER_USER_PAGE	✔	✔			
SHOW__SUPERVISOR_PAGE				✔	
SHOW__SUPERVISOR_ENTITLEMENT_USERS				✔	
SHOW__SUPERVISOR_USER_ENTITLEMENTS				✔	
SEARCH__SUPERVISOR_USER_ENTITLEMENTS				✔	

Permission	App Owner	Entitlement Owner	Executive	Supervisor	User
SHOW__SUPERVISOR_UNSCORED_ENTITLEMENTS				✔	
SHOW__USER_ENTITLEMENTS	✔	✔	✔		
SHOW__RULES_BY_APP_OWNER	✔				
SHOW__RULES_BY_ENTT_OWNER		✔			
SHOW__CERTIFICATIONS	✔	✔		✔	✔
REVOKE__CERTIFY_ACCESS	✔	✔		✔	

Was this helpful?