FORGEROCK

Admin Guide

/ Autonomous Identity 2020.10.2

Latest update: 2020.10.2

,
, ,

Abstract

This guide is targeted to administrators who must set up and maintain the Autonomous Identity system.

Table of Contents

Overview

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 the Apache Cassandra data.	Creating and Removing Users Learn how to creating and remove users.
Customize Domain 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.	Prepare Spark Learn how to prepare Apache Spark.	Data Preparation Learn how to prepare the data.

Run Analytics Learn how to run the Analytics pipeline.	Appendix A: analytics_init_config.yml Learn about the configuration file.	Appendix B: ui-config.json Learn about the UI configuration file.

For installation instructions, see the Installation Guide.

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

Chapter 1
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 scales linearly as the load increases.
- **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.
- **Automated Workflows**. Autonomous Identity reduces the burden on managers who must manually approve new entitlements, for example, assigning access for new hires, by auto-approving high confidence, low-risk access requests and automate the re-certification of entitlements. Predictive recommendations lends itself well to automation, which saves time and cost.
- **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.
- **Powerful Explainable AI Algorithms**. The Analytics Engine provides transparent and explainable results that lets business users get insight into why the end-user has the access they have, or what access they should have.
- **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.

Chapter 2
Stopping and Starting

The following commands are for Linux distributions.

Stopping Docker

- 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 http://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-a
utoid-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/l
ogs/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.

Chapter 3

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.SizeTieredCompactionStrategy',
 '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.SizeTieredCompactionStrategy',
 '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.

Chapter 4

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
```

Chapter 5
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.

Chapter 6
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
```

Chapter 7
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.

Chapter 8 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/AllowedAttributesForFiltering
{
  "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/AllowedAttributesForFiltering
```

   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/AllowedAttributesForFiltering
configuration item updated
```

Chapter 9
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
```

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
   ```

Chapter 10
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
```

Chapter 11
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
```

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%3Dautoid
    ```

Chapter 12
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>>
   ```

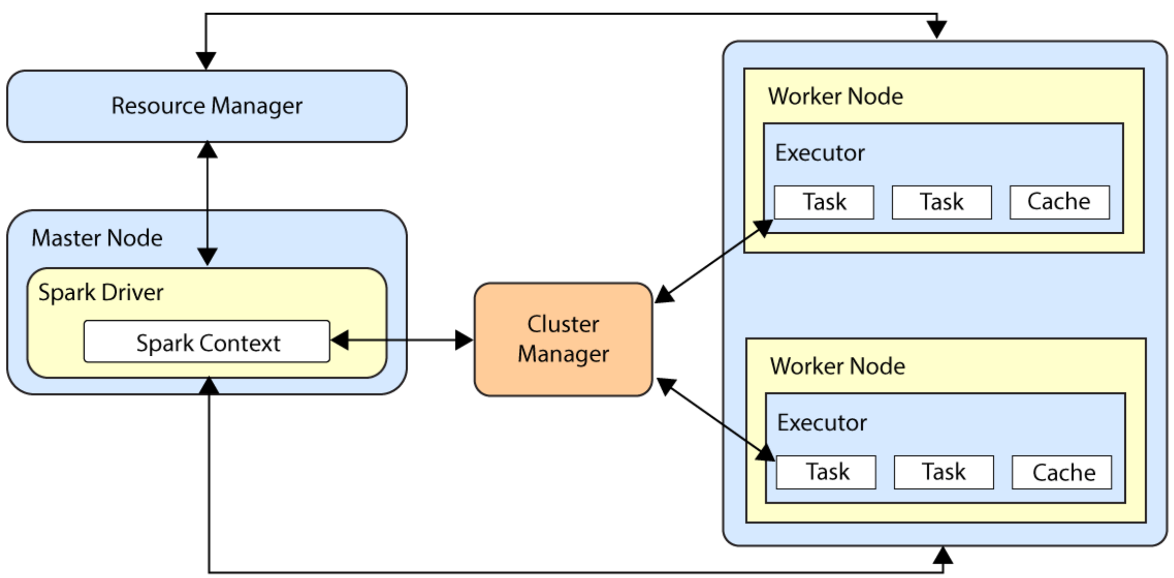
   The Docker manager node re-initiates the worker node. Repeat this step on any other worker node.

Chapter 13
Prepare Spark Environment

Apache Spark requires configuration for optimal performance. The key task is to properly tune the Spark server's memory and executors per core. You must also enable the Spark History logs and configure the SSL connections to the Spark server.

Determine the Optimal Core and Memory Configuration

The analytics engine requires tuning prior to processing the entitlements data. Spark distributes its dataset into *partitions* to allow parallel processing. Each partition runs a executor, which is a JVM process that is launched in a worker node and processes a unit of work called a *task* on a portion of the dataset.



There are three main tuning parameters that you must configure in the `analytics_init_config.yml` file for optimal performance:

- **Number of Executor Cores**. Indicates the maximum number of tasks that an executor can run at a time. In Spark, this property is set using the `--executor-cores` flag. On the target node, specify this property using the **spark.executor.cores** parameter.
- **Number of Executors**. Indicates the number of executors the application allocates in the cluster. In Spark, this property is set using the `--num-executors` flag. On the Analytics container, you specify this property using the **spark.total.cores** parameter.
- **Amount of Memory Allocated to Each Executor**. Indicates the amount of memory allocated to the JVM heap memory for each executor. In Spark, this property is set using the `--executor-memory`. On the target node, you specify this property using the **spark.executor.memory** parameter.

Before configuring these parameters, consider these points:

- **Running Too Much Memory or Too Few Executors**. In general, running executors with too much memory (for example, +64GB/executor) often results in excessive garbage collection delays. Running with too few executors (for example, 1 executor/core) does not benefit from caching as it cannot run multiple tasks in a single JVM. This will not leave enough memory for the resource manager.
- **Maximum Overhead Memory**. The total amount of memory requested by a resource manager per Spark executor is the sum of the executor memory plus memory overhead. The memory overhead is needed for JVM heap and resource manager processes.
- **One Executor for the App Manager**. One executor should be assigned to the application manager and the rest for task processing.
- **Number of Cores**. One core should be dedicated to the driver and resource manager application. For example, if the node has 8 cores, 7 of them are available for the executors; one for the driver and resource manager.

  The choice is relative to the size of the cluster. The number of executors is then a ratio between the number of cores available for executors and the number of cores per executor times the number of nodes on the cluster.

  For optimal performance, run a maximum of 5 cores per executor for good I/O throughput. Minimum number of cores should be 3 cores per executor.
- **Executor Memory**. Allocate 7 to 10% of memory to the application manager and executor overhead. For example, if you have a cluster of 96 GB of RAM, you should distribute up to 86GB to the executors.

Let's look at an example. If you have the following cluster configuration:

- 6 nodes
- 16 cores per node
- 64GB RAM per node

If we have 16 cores per node, leave one core to the Spark application, then we will have 15 cores available per node. The total number of available cores in the cluster will be 15 x 6= 90. The number of available executors is the total cores divided by the number of cores per executor (90/5 = 18). We leave one executor to the application manager; thus, 18 - 1 = 17 executors. Number of executors per node = 17/6 ~ 3. The memory per executor = 64GB/3 = 21 GB. 7 to 10% of the memory must be allocated to heap overhead. Let's use 7%. Then, executor memory will be 21 - 3 (that is, 7% x 21) = 18GB.

- Number of Executor Cores = 5
- Number of Executors = 17
- Executor Memory = 18GB RAM

The table below show example Spark executor memory and core combinations for different node configurations:

Number of Nodes	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Num cores/node	8	8	8	8	8	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16
Memory size/node	32	32	32	32	32	32	32	32	32	32	64	64	64	64	64	64	64	64	64	64
spark.total.cores	3	3	3	3	3	3	3	3	3	3	4	4	4	4	4	5	5	5	5	5
spark.executor.cores	2	4	6	9	11	4	9	13	18	23	3	6	10	13	17	2	5	8	11	13
spark.executor.memory	14	14	14	9	9	7	5	5	5	5	18	18	14	14	14	28	18	18	18	18

These Spark values are set in the `analytics_init_config.yml` file. For more information, see "Create Initial Analytics Template".

Configure Spark History

For proper maintenance of the analytics machine, the Spark history server must be enabled to record executor logs. You can enable the logging mechanism and history by add the following lines to the Spark configuration file, located at `$SPARK_HOME/conf/spark-defaults.conf`.

1. Open the `$SPARK_HOME/conf/spark-defaults.conf` file. If the file does not exist, careate one from the sample file provided in the `conf` directory.

2. Add the following lines to the file with a text editor.

```
spark.eventLog.enabled              true
spark.evenLog.dir                   file:///tmp/spark-events  # the dir is arbitrary
spark.history.fs.logDirectory       file:///tmp/spark-events
```

3. Start the Spark History server if installed. The Spark History server is accessible on port 18080 on the master server.

```
$SPARK_HOME/sbin/start-history-server.sh
```

Note

You can also access the Spark History server using the REST API at `http://<server-url>:18080/api/v1/applications`.

Configure Spark SSL

You can configure SSL connection to all Spark interfaces by setting the configuration file, `analytics_init_config.yml` file. For additional information, see the Spark documentation.

Run the following commands to configure SSL;

1. Generate SSL keys. The keys can be self-signed or public CA-signed certificates.
2. Write an Authentication filter class in Java that implements the Java Filter interface. The interface determines how the credentials are handled. Compile the class to a jar file.
3. Copy the artifact to `$SPARK_HOME/jars` directory on all nodes.
4. Update the Spark configuration to enable SSL and set up SSL namespace parameters. Update the Spark configurations on all nodes to enable SSL, indicating the keys and authentication filter to use. Make the changes to the configuration file as follows:

```
# Enable SSL
spark.ssl.enabled                  true
spark.ui.https.enabled             true
spark.ssl.fs.enabled               true

# Add public/private SSL keys
spark.ssl.keyPassword              Acc#1234
spark.ssl.keyStore                 /opt/autoid/certs/zoran-cassandra-client-keystore.jks
spark.ssl.keyStorePassword         Acc#1234
spark.ssl.trustStore               /opt/autoid/certs/zoran-cassandra-server-trustStore.jks
spark.ssl.trustStorePassword       Acc#1234
spark.ssl.protocol                 TLSv1.2

# 'com.autoid.spark' is the package in the jar file and
# 'BasicAuthFilter' is the class that contains the filter class
# Advanced filter implementat would not have parameters to reduce the blast radius
spark.ui.filters                               com.autoid.spark.BasicAuthFilter
spark.com.autoid.spark.BasicAuthFilter.params  user=user,password=password
```

5. Restart the Spark services for the new configurations to take effect.

Chapter 14
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.

Clients can transfer the data to ForgeRock via some portable media, like USB, or through a connector from the client systems. The analysts review the data to ensure that it is in its proper formatted form.

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"
- "CSV Files and Schema"

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 various file formats: .csv, .xlsx, .txt. The data can be in a single file, or multiple files. Data includes user attributes, entitlements descriptions, and entitlement assignments.
- Duplicate values should be removed.
- Add optional columns for additional training attributes, for example, `MANAGERS_MANAGER` and `MANAGER_FLAG`.
- Merge user attribute information and entitlement metadata into the entitlement assignments. This creates one large dataframe that should have an individual row for each assignment. Each row should contain the relevant user attribute profile information and entitlement metadata for the assignment.

- Rename any columns that Autonomous Identity uses to the appropriate names, for example, `employeeid` to `USR_KEY`, `entitlement_name` to `ENT`.
- Build seven dataframes needed for Autonomous Identity, for example, features, labels, HRName, etc. This step may also include adding some additional columns to each dataframe, for example, `labels['IS_ASSIGNED'] = 'Y'`.
- Write out the nine dataframes to nine csv files and store them in the `/data/input` directory.

CSV Files and Schema

ForgeRock provides a transformation script that takes in raw data and converts them to acceptable .csv formatted files.

You can access a Python script template to transform your client files to correct the .csv files. Run the following steps:

1. On the target machine, go to the `/data/conf/`.
2. Open a text editor, and view the `zoran_client_transformation.py` template. You can edit this script for your company's dataset.

The script outputs files with the following contents:

CSV Files Outputs

Files	Description
features.csv	Contains one row for each employee with all of their user attributes.
labels.csv	Contains the user-to-entitlement mappings. Also, includes usage data if provided.
HRName.csv	Maps user ID’s to their names. This file is needed for the UI.
EntName.csv	Maps entitlement ID’s to their names. This file is needed for the UI.
RoleOwner.csv	Maps entitlements ID’s to the employees who "owns" these entitlements, the people responsible for approving or revoking accesses to this entitlement.
JobAndDeptDesc.csv	Maps user ID’s to the department in which they work, and also includes a description of their job within the company.
AppToEnt.csv	Maps entitlements to the applications they belong to. This file is needed for the UI.
app_atrributes.csv	Maps attributes to applications. This file is needed for attribute filtering on the applications page.
ent_attributes.csv	Maps attributes to entitlements. This file is needed for entitlement attribute filtering on the applications page.

The schema for the input files are as follows:

CSV Files Schema

Files	Schema
features.csv	This file depends on the attributes that the client wants to include. These are some required columns: • **USR_KEY**. Specifies the user's unique ID. • **USR_DISPLAY_NAME**. Specifies the user's name. If not provided, use the user ID, but the name works best for the UI. • **USR_MANAGER_KEY**. Specifies the ID of the user's manager. • **USR_EMP_TYPE**. Specifies the employment status of the user, for example, PERMANENT, CONTRACT, EMPLOYEE, NON-EMPLOYEE, VENDOR, etc. • **IS_ACTIVE**. Specifies whether this user is an active employee. Sometimes companies submit inactive accounts, which is not included in the analysis.
label.csv	• **USR_KEY**. Specifies the user's unique ID. • **ENT**. Specifies the unique entitlement identifier. • **HIGH_RISK**. Determines whether an access is considered HIGH, MEDIUM, or LOW risk. • **IS_ASSIGNED**. Determines whether an access is assigned (used internally). • **LAST_USAGE**. Specifies the last time an entitlement was accessed.
HRName.csv	• **USR_KEY**. Specifies the user's unique ID. • **USR_NAME**. Specifies a human readable username. For example, `John Smith`.
EntName.csv	• **ENT_NAME**. Specifies the entitlement name. • **ENT**. Specifies the entitlement. For example, `Outlook_LotusNotes_Lync_Access_Suite`.
RoleOwner.csv	• **ROLE**. Specifies the unique user ID of the entitlement owner. • **ENT**. Specifies the unique entitlement identifier.
JobAndDeptDesc.csv	• **USR_KEY**. Specifies the user's unique ID. • **DEPARTMENT**. Specifies the human readable department. • **JOB_DESCRIPTION**. Specifies the human readable job description.
AppToEnt.csv	• **ENT**. Specifies the unique entitlement identifier. • **APP_NAME**. Specifies a human readable application name. • **APP_ID**. Specifies the unique application identifier.
app_atrributes.csv	• **APP_ID**. Specifies the application identifier.

Files	Schema
	• **APP_OWNER**. Specifies application owner. • **APP_NAME**. Specifies the application name. • **CRITICALITY**. Specifies the criticality (Essential, Non-Essential). • **RISK_LEVEL**. Specifies the risk level (Low, Medium, High).
ent_atrributes.csv	• **ENT**. Specifies the entitlement. • **APP_ID**. Specifies the application identifier. • **CRITICALITY**. Specifies the criticality (Essential, Non-Essential). • **RISK_LEVEL**. Specifies the risk level (Low, Medium, High).

Chapter 15
Data Ingestion

After running the **analytics create-template** command, you can configure the analytics pipeline run by editing the `/data/conf/analytics_init_config.yml` file.

Autonomous Identity supports data ingestion for three types of Identity Governance systems: general `csv` files, `iiq` for Sailpoint IdentityIQ files, and `oim` for Oracle Identity Manager files. You can configure the `/data/conf/analytics_init_config.yml` file for `iiq` or `oim` files.

Data Ingestion

Field	Sample	Description
ingestion:		
drop_if_create:	True/False	Not used by these connectors
catalog_step	True/False	Creates a csv file with possible attributes from the client system. If set to `True`, the data pull does not occur. This property only needs to be set true in the first ingestion run to identify possible attributes.
staging:	True/False	Determines whether to pull data to `autoid_staging` or `autoid_base` keyspaces. If `staging` is set to `True`, the system pulls to `autoid_staging` keyspace; while set to `False`, it pulls to `autoid_base` keyspace. The `autoid_staging` keyspace lets us pull data without on-the-fly transformations, while also supporting pulling from multiple sources.
connector:		
type:	`csv`, `iiq`, `oim`	Connector for type of source data. Currently only supporting three options.
batchsize:	1000	Integer. Specifies the number of rows or pages pulled as part of each batch.

timeout:	10	Integer representing seconds. Specifies the amount of time to wait before timeout.
change_reconciliation:		
enabled:	True/False	Determines whether to pull data that filters only results that been modified after a certain date, specified by the `time` property.
time:	'2013-05-17T00:00:00Z','2020-05-17 00:00:00'	Specifies the data-time after which the data should be pulled. Note IIQ requires a time zone (first value).
features_mappings:		
chief_yes_no	'User:CHIEF_YES_NO'	Value take from the csv generated in the catalog step. Note that `chief_yes_no` is not required, but is included here as an example. For reference, see "*Appendix A: The analytics_init_config.yml File*".

Chapter 16
Run the Analytics Pipeline

The Analytics pipeline is the heart of Autonomous Identity. It 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 some 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.

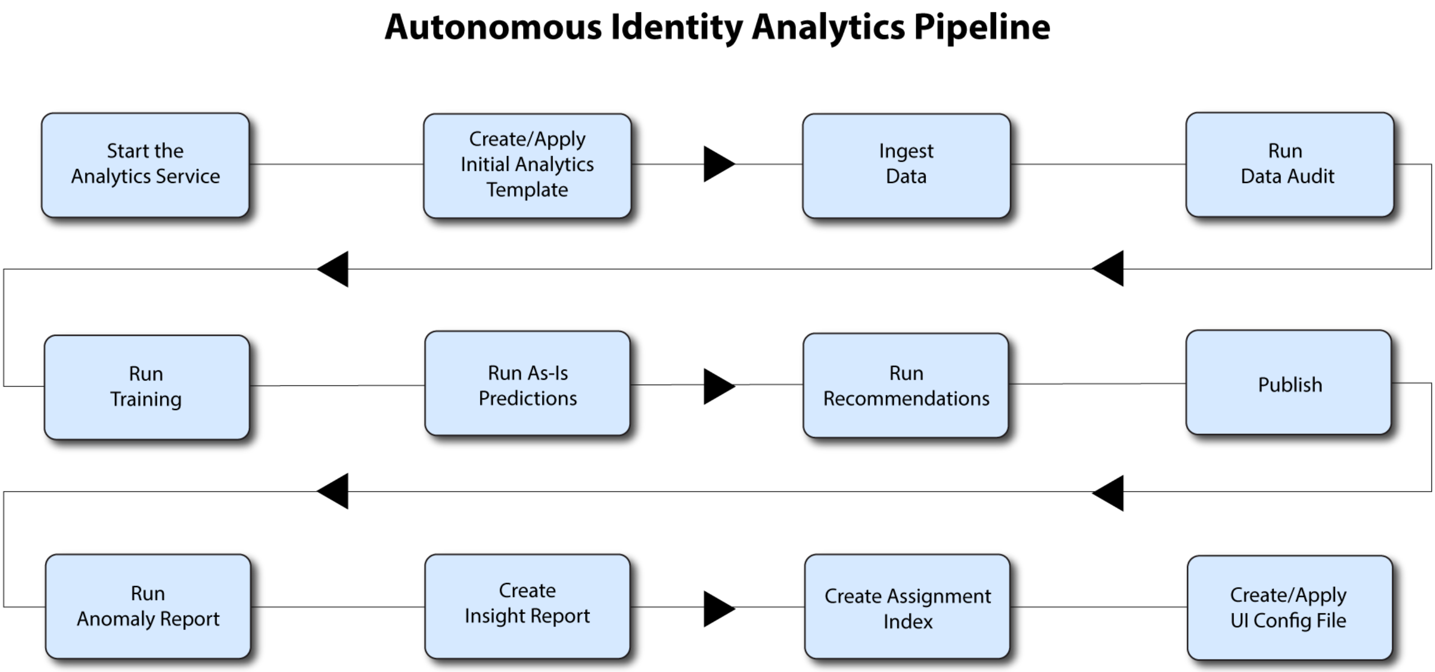
The initial pipeline step is to create, edit, and apply the `analytics_init_config.yml` configuration file. The `analytics_init_config.yml` file configures the key properties for the analytics pipeline. In general, you will not need to change this file too much, except for the Spark configuration options. For more information, see "*Prepare Spark Environment*".

Next, run a job to validate the data, then, when acceptable, ingest the data into the database. After that, run a final audit of the data to ensure accuracy. If everything passes, 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 this, run a recommendations job that looks at all users who do not have a specific entitlement but should, based on their user attribute data. Once the predictions and recommendations are completed, run an insight report to get a summary of the analytics pipeline run, and an anomaly report that reports any anomalous entitlement assignments.

The final steps are to push the data to the backend Cassandra or MongoDB database, and then configure and apply any UI configuration changes to the system.

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.

Analytic Actions

The Deployer-based installation of the analytics services provides an "analytics" alias (alias analytics='docker exec -it analytics bash analytics') on the server, with which you can perform a number of actions for configuration or to run the pipeline on the target machine.

A Summary of the Analytics Services Commands

Command	Description
analytics create-template	Run this command to create the `analytics_init_config.yml` configuration file.

Command	Description
analytics apply-template	Apply the changes to `analytics_init_config.yml` file and create the `analytics_config.yml` file.
analytics ingest	Ingest data into Autonomous Identity.
analytics audit	Run a data audit to ensure if meets the specifications.
analytics train	Runs an analytics training run.
analytics predict-as-is	Run as-is predictions.
analytics predict-recommendation	Run recommendations.
analytics publish	Push the data to the Apache Cassandra/MongoDB backend.
analytics anomaly	Create the Anomaly report.
analytics insight	Create the Insights report.
analytics create-assignment-index	Generate the Elasticsearch index.
analytics create-ui-config	Create the `ui_config.json` file.
analytics apply-ui-config	Apply the `ui_config.json` file.
analytics run-pipeline	Run all of the pipeline commands at once in the following order: validate, ingest, audit, train, predict-as-is, predict-recommendation, publish, create-ui-config, apply-ui-config.
analytics upgrade	Run an analytics upgrade when updating from one Autonomous Identity version to the latest version. For more information, see "*Upgrade Autonomous Identity*" in the *Installation Guide*.

Create Initial Analytics Template

The main configuration file for the Analytics service is `analytics_init_config.yml`. You generate this file by running the **analytics create-template** command.

1. On the deployer node, SSH to the target node:

```
$ ssh autoid@<Target-IP-Address>
```

2. Create the initial configuration template. The command generates the `analytics_init_config.yml` in the `/data/conf/` directory.

```
$ analytics create-template
```

   You should see the following output if the job completed successfully:

```
analytics template config file created at CONF_DIR/analytic_init_config.yml. Please edit it and run
 apply-template
```

3. Edit the `analytics_init_config.yml` for any specific configurations for your deployment.
   - For information on data preparation, see "*Data Preparation*".
   - For information on Spark tuning, see "*Prepare Spark Environment*".
   - For information on data ingestion, see "*Data Ingestion*".
4. Copy the .csv files to the `/data/input` folder. Note if you are using the sample dataset, it is located at the `/data/conf/demo-data` directory.

   ```
   $ cp *.csv /data/input/
   ```

5. Apply the template to the analytics service. The command generates the `analytics_config.yml` file in the `/data/conf/` directory. Autonomous Identity uses this configuration for other analytic jobs.

   > **Note**
   >
   > Note that you do not directly edit the `analytics_config.yml` file. If you want to make any additional configuration changes, edit the `analytics_init_config.yml` file again, and then re-apply the new changes using the **analytics apply-template** command.

   ```
   $ analytics apply-template
   ```

   You should see the following output if the job completed successfully:

   ```
   analytics template config file created at CONF_DIR/analytic_init_config.yml. Please edit it and run
    apply-template
   ```

6. You have the option now to run the analytics pipeline individually in a specific order, or run the full pipeline all at once.

   If this is your first time running the pipeline, we recommend running each step individually in the order shown in the procedure, starting with "Ingest the Data Files".

   If you are familiar with the analytics pipeline process, run the full pipeline as presented in "Run Full Pipeline".

Ingest the Data Files

By this point, you should have prepared and validated the data files for ingestion into Autonomous Identity. This process imports the seven .csv files into the Cassandra or MongoDB database.

Ingest the data into the Cassandra or MongoDB database:

1. Make sure Cassandra or MongoDB is up-and-running.
2. Make sure you have determined your Spark configuration in terms of the number of executors and memory.

3. Run the data ingestion command.

```
$ analytics ingest
```

You should see the following output if the job completed successfully:

```
Script : /home/analytics/autoid-analytics/ai_ingest.py is successful
```

Run Data Audit

Before running the analytics training run, we need to do one final audit of the data. The audit runs through the seven .csv files as loaded into the database and generates initial metrics for your company.

Run the Data Audit:

1. Verify that the .csv files are in the `/data/input/` directory.

2. Run the audit command.

```
$ analytics audit
```

You should see the following output if the job completed successfully:

```
Script : Script : /home/analytics/autoid-analytics/ai_test.py is successful
```

You can access the audit report (`audit_report.txt` in the `/data/input/spark_runs/reports` directory on the target server.

The script provides the following metrics:

CSV File Audit

File	Description
features.csv	• Number of columns/categories (number of features). • Number of rows (number of users). • Number of unique values in each column. • Number of null values in each column. • 10 most common values for specific attributes in each column.
labels.csv	• Number of columns/categories (number of labels). • Number of rows (number of entitlement assignments). • Number of unique values and category names. • Number of unique values in the high risk column.

File	Description
	• Number of entitlements with single mappings (that is, only one user has been assigned this entitlement), 2, 3-5, 6-10, 11-100, 100-1000, 1000+ mappings. Number of multiple numbers of mappings. • Entitlement and count of assigned users. • Assigned user and count.
apptoent.csv	• Number of columns/categories. • Number of rows. • Number of unique entitlements. • Number of unique applications. • Number of users in entitlement assignments. • Application name and associated number of entitlements with the application
roleowner.csv	• Number of columns/categories. • Number of rows. • Number of unique entitlements. • Number of unique role owners. • Top 10 role owners with most assigned entitlements.
Other Insights	• Number of user keys in labels file but not in features file • Number of manager keys in features that do not exist as Users

Run Training

Now that you have ingested the data into Autonomous Identity. Start the training run.

Training involves two steps: the first step is an initial machine learning run where Autonomous Identity analyzes the data and produces the association rules. In a typical deployment, you can have several million generated rules. 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 Cassandra database.

Start the training process:

- Run the training command.

  ```
  $ analytics train
  ```

You should see the following output if the job completed successfully:

```
Script : /home/analytics//autoid-analytics/ai_train.py is successful
```

Run Predictions and Recommendations

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. During a pre-processing phases, the `labels.csv` and `features.csv` are combined in a way that appends them to only the access rights that each user has. 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 Cassandra database.

- **recommendation**. During the Recommendations process, confidence scores are assigned to all entitlements. This allows Autonomous Identity to recommend entitlements to users who do not have them. The lowest confidence entitlement is bound by the confidence threshold used in the initial training step. During a pre-processing phase, the `labels.csv` and `features.csv` are combined in a way that appends them to all access rights. The script 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 Cassandra database.

Run as-is Predictions:

1. In most cases, there is no need to make any changes to the configuration file. However, if you want to modify the analytics, make changes to your `analytics_init_config.yml` file.

   For example, check that you have set the correct parameters for the association rule analysis (for example, minimum confidence score) and for deciding the rules for each employee (for example, the confidence window range over which to consider rules equivalent).

2. Run the as-is predictions command.

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

   You should see the following output if the job completed successfully:

   ```
   Script : /home/analytics/autoid-analytics/ai_predict_asis.py is successful
   ```

Run Recommendations:

1. Make any changes to the configuration file, `analytics_init_config.yml`, to ensure that you have set the correct parameters (for example, minimum confidence score).

2. Run the recommendations command.

```
$ analytics predict-recommendation
```

You should see the following output if the job completed successfully:

```
Script : /home/analytics/autoid-analytics/ai_predict_recommend.py is successful
```

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.

Publish the data to the backend:

- ```
  $ analytics publish
  ```

  You should see the following output if the job completed successfully:

  ```
  Script : /home/analytics/autoid-analytics/ai_load.py is successful
  ```

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 script writes the anomaly report to a Cassandra or MongoDB database. The report is written to a `report_anomaly` table in the `autoid_analytics` keyspace in Cassandra and MongoDB.

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 unlikely an anomaly. The entitlement is either missing data or the assignment is poorly managed.

- Run the anomaly report:

  ```
  $ analytics anomaly
  ```

  You should see the following output if the job completed successfully:

  ```
  Script : /home/analytics/autoid-analytics/ai_report_anomaly.py is successful
  ```
  ```

Run the 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.

Run the Insight Report:

- ```
  $ analytics insight
  ```

  You should see the following output if the job completed successfully:

  ```
  Script : /home/analytics//autoid-analytics/ai_report_insight.py is successful
  ```

Create Assignment Index

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

Create the index:

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

You should see the following output if the job completed successfully:

```
Script : CreateElasticIndex is successful
```

Create the Analytics UI Config File

Once the analytics pipeline has completed, you can configure the UI using the **analytics create-ui-config** command if desired.

1. Run the **analytics create-ui-config** to generate the `ui_config.json` file in the `/data/conf/` directory. The file sets what is displayed in the Autonomous Identity UI.

   ```
   $ analytics create-ui-config
   ```

   You should see the following output if the job completed successfully:

   ```
   Script : init.py is successful
   ```

2. In most cases, you can run the file as-is. If you want to make changes, make edits to the `ui_config.json` file and save it to the `/data/conf/` directory.

3. Apply the file.

   ```
   $ analytics apply-ui-config
   ```

   You should see the following output if the job completed successfully:

   ```
   Script : init.py is successful
   ```

   If every pipeline process has ended successfully, you have successfully run the full analytics pipeline.

Run Full Pipeline

You can run the full analytics pipeline with a single command using the **run-pipeline** command. Make sure your data is in the correct directory, `/data/input`, and that any UI configuration changes are set in the `ui_config.json` file in the `/data/conf/` directory.

The **run-pipeline** command runs the following jobs in order:

1. analytics ingest
2. analytics audit
3. analytics train
4. analytics predict-as-is

5. analytics predict-recommendation
6. analytics publish
7. analytics anomaly
8. analytics insight
9. analytics create-assignment-index
10. analytics create-ui-config
11. analytics apply-ui-config

Run the full pipeline:

- ```
  $ analytics run-pipeline
  ```

  You should see the following output if the job completed successfully:

  ```
  Script : init.py is successful
  #analytics-run-pipeline-output

  Pipe Line Ends
  ```
  ```

Appendix A. Appendix A: The analytics_init_config.yml File

The `analytics_init_config.yml` is an important configuration file in Autonomous Identity. For each deployment, you customize the parameters to the environment. Deployers should configure this file before ingesting the input data into Cassandra.

The process to use the `analytics_init_config.yml` is as follows:

- On the target node, use the **analytics create-template** command to generate the `analytics_init_config.yml` file.
- Make changes to the `analytics_init_config.yml` tailored to your deployment and production environment.
- Run the **analytics apply-template** command to apply your changes. The output file is `analytics_config.yml` file that is used for the other analytics jobs.

> **Note**
>
> Do not make changes to the `analytics_config.yml`. If you want to make changes to the configuration, update the `analytics_init_config.yml` file and then re-run the **analytics create-template** command.

The file is used to do the following:

- Sets the input and output paths.
- Configures the association rule options.
- Sets the user attributes to be used for training.

- Sets up connection to the Cassandra database.
- Configures column names and mappings for the UI dataload.

The following `analytics_init_config.yml` file version is v0.32.0. The file is not expected to change much for each release, but some revision will occur intermittently.

```
################################
# Common configuration options #
################################
common:
  base_path:                /data/              # Base directory for analytics I/O. Configurable.

  ######################################
  # Data-related configuration options #
  # (Input & Output of files/rules)    #
  ######################################
data:
  # input data
  input:
    input_path:             input               # Input file directory under base_path. Configurable.
    features_file:          features.csv        # Contains user attribute data
    labels_file:            labels.csv          # Contains user-to-entitlement mappings.
    application_file:       AppToEnt.csv        # Contains entitlements-to-applications mappings.
    role_owner_file:        RoleOwner.csv       # Contains entitlement IDs to employees who "own the
                                                #   entitlements"
    account_name_file:      HRName.csv          # Contains user ID mappings to names.
    entitlement_name_file:  EntName.csv         # Contains entitlement IDs to their names.
    job_dept_desc_file:     JobAndDeptDesc.csv  # Contains user ID mappings to the departments where
                                                #   they work plus job descriptions

#########################################
# Extract Transform Load to Database    #
# (Database Technologies i.e. Cassandra #
#########################################
etl:
  med_conf:                 0.35                # Confidence threshold for medium confidence assignments
  high_conf:                0.75                # Confidence threshold for high confidence assignments
  edf_thresh:               0.75                # Confidence threshold for driving factor rules
  org_column_value:         test                # Use client organization identifier
  app_source_column:        test                # Use client organization identifier
  filtering_columns:        CITY                # Specifies any filtering columns

############################################
# Association Rules configuration options  #
# (Training & As-Is/Recommend Predictions) #
############################################
assoc_rules:
  # base config
  features_filter:        USR_KEY,CITY,USR_DEPARTMENT_NAME,       # update with columns you want to be
                          COST_CENTER,JOBCODE_NAME,               #   used in training (must contain
                          LINE_OF_BUSINESS,                       #   USR_KEY or equivalent)
                          LINE_OF_BUSINESS_SUBGROUP,
                          CHIEF_YES_NO,USR_EMP_TYPE,
                          USR_DISPLAY_NAME,MANAGER_NAME,
                          USR_MANAGER_KEY,IS_ACTIVE
  features_table_columns: USR_KEY,CITY,USR_DEPARTMENT_NAME,       # update with list of all columns in
                          COST_CENTER,JOBCODE_NAME,               #   features
```

```
                          LINE_OF_BUSINESS,
                          LINE_OF_BUSINESS_SUBGROUP,
                          CHIEF_YES_NO,USR_EMP_TYPE,
                          USR_DISPLAY_NAME,MANAGER_NAME,
                          USR_MANAGER_KEY,IS_ACTIVE

############################################
# User Column Description for the Feature CSV Headers
# ( user_name : User Name , user_manager :Manager Name)
############################################
ui_config:                                                   # Configurable column descriptions in the UI.
  user_column_descriptions: USR_KEY:User Key,CITY:City Location Building,USR_DEPARTMENT_NAME:User
 Department Name,
  COST_CENTER:User Cost Center,JOBCODE_NAME:Job Code Name,LINE_OF_BUSINESS:LOB,CHIEF_YES_NO:Manager Flag,
  USR_EMP_TYPE:Employee Type,USER_DISPLAY_NAME:User name,MANAGER_NAME:Manager Name,USR_MANAGER_KEY:Manager
 Key,
  IS_ACTIVE:Active

#########################################
# Spark-related configuration options   #
#########################################
spark:
  logging_level:  WARN                # Logging level
  config:
    spark.executor.memory:            6G        # Recommended value >= 6
    spark.driver.memory:              4G        # Memory allocation to job on master.
    spark.driver.maxResultSize:       2G        # Maximum size of results storage capacity.
    spark.executor.cores:             3         # Number of executor cores
    spark.total.cores:                3         # Modify this value based on the cluster size. Number of
                                                #  executors will be calculated automatically.
    spark.scheduler.mode:             FAIR      # Set the scheduler for resources.
    spark.sql.shuffle.partitions:     6
    spark.task.maxFailures:           200
    spark.driver.blockManager.port:   39999
    spark.blockManager.port:          40016
ingestion:
  drop_if_create: true
  catalog_step: false
  staging: false
  connector:
    type: iiq
    batchsize: 100
    timeout: 30
    change_reconciliation:
      enabled: false
      time: '2013-05-17T00:00:00Z'
    features_mappings:
      chief_yes_no: User:CHIEF_YES_NO
      city: User:CITY
      jobcode_name: User:JOBCODE_NAME
      line_of_business: User:LINE_OF_BUSINESS
      line_of_business_subgroup: User:LINE_OF_BUSINESS_SUBGROUP
      usr_manager_name: User:MANAGER_NAME
      usr_dep_name: User:USR_DEPARTMENT_NAME
      usr_display_name: User:USR_DISPLAY_NAME
      usr_emp_type: User:USR_EMP_TYPE
      costcenter: User:COST_CENTER
```

Appendix B. Appendix B: The ui-config.json File

The `ui-config.json` file contains configuration properties for the Autonomous Identity user interface and API services that are loaded as key-value pairs into Hashicorp Consul. The values are loaded through the Configuration Service that are shared across Autonomous Identity's microservices.

The process to use the `ui-config.json` is as follows:

- On the target node, run the **analytics create-ui-config** command to generate the `ui_config.json` file in the `/data/conf/` directory.
- Make changes to the `ui_config.json` tailored to your deployment and production environment.
- Run the **analytics apply-ui-config** command to apply your changes.

The file is used to do the following:

- Define DB schemas used by the API to interface with the data layer. These schemas all contain a `table_name` property.
- Defines the organization name property, `OrgNameConf`, used for data displaoyed on the Company Overview Dashboard page.
- Defines various confidence score or count criteria, such as high, medium, and low score threshold levels and a minimum score threshold level to allow auto-certify or auto-request actions. The key properties are:
  - `ConfidenceScorethresholdsConf`
  - `ConfigThresholdsConf`

  - `MostAssigned`
  - `HighVolume`
  - `HighRiskConf`
- Defines the permissions that are applied to each Autonomous Identity user group. The key property is `PermissionsConf`.
- Maps database columns to user-friendly titles for fields, labels, and headings in the UI. The key properties are:
  - `UIConfig`
  - `UIHRData`
  - `UIJustifications`
- Defines the user and entitlement attributes to be used as filters in the UI. The key properties are:
  - `FilteringOptionsConf`
  - `AllowedAttributesForFiltering`
- Defines the character delimiter used in the raw justification string data to separate the justification key from its value. The key property is: `JustificationDelimeter`.

The following `ui-config.json` file version is v0.32.0. The file is not expected to change much for each release, but some revision may occur intermittently.

```
{
    "RecommendPredictionsConf": {
        "name": "RecommendPredictions",
        "modelDefinition": {
            "fields": {
                "usr_key": "text",
                "ent": "text",
                "conf": "decimal",
                "freq": "decimal",
                "frequnion": "decimal",
                "rule": {
                    "type": "list",
                    "typeDef": "<text>"
                },
                "last_usage": "text"
            },
            "key": [
                "usr_key",
                "ent"
            ],
            "table_name": "recommend_predictions"
        }
    },
    "CompanyViewOverviewConf": {
```

```
        "name": "CompanyViewOverview",
        "modelDefinition": {
            "fields": {
                "key": "text",
                "total_employees": "int",
                "employees_wo_manager": "int",
                "employees_w_manager": "int",
                "entitlements_without_roleowners": "int",
                "entitlements_with_roleowners": "int",
                "total_entitlements": "int",
                "entitlements_covered_by_model": "int",
                "entitlements_not_covered": "int",
                "entitlements_w_no_users": "int",
                "entitlements_w_one_user": "int",
                "entitlements_w_zero_to_five_users": "int",
                "entitlements_w_five_to_ten_users": "int",
                "entitlements_w_ten_to_hundred_users": "int",
                "entitlements_w_hundred_to_onek_user": "int",
                "entitlements_w_onek_to_tenk_users": "int",
                "entitlements_w_tenk_users": "int",
                "entitlements_w_hundredk_users": "int"
            },
            "key": [
                "key"
            ],
            "table_name": "company_view_overview"
        }
    },
    "CompanyViewEmployeeTypeConf": {
        "name": "CompanyViewEmployeeType",

    "EntitlementUserManagerScoresConf": {
        "name": "EntitlementUserManagerScores",
        "modelDefinition": {
            "fields": {
                "entitlement": "text",
                "entitlement_name": "text",
                "user": "text",
                "user_name": "text",
                "manager": "text",
                "score": "float",
                "justification": {
                    "type": "list",
                    "typeDef": "<text>"
                },
                "app_id": "text",
                "app_name": "text"
            },
            "key": [
                "entitlement",
                "score",
                "user",
                "manager"
            ],
            "table_name": "entitlement_user_manager_scores"
        }
    },
    "GraphByRoleConf": {
        "name": "GraphByRole",
```

```
        "modelDefinition": {
            "fields": {
                "role": "text",
                "entitlement": "text",
                "entitlement_name": "text",
                "app_id": "text",
                "app_name": "text",
                "high_risk": "text"
            },
            "key": [
                "entitlement",
                "app_id",
                "role"
            ],
            "indexes": [
                "entitlement_name"
            ],
            "table_name": "graph_by_role"
        }
    },
    "GraphConf": {
        "name": "Graph",
        "modelDefinition": {
            "fields": {
                "manager": "text",
                "user": "text",
                "manager_name": "text",

    "RoleOwnerConf": {
        "name": "RoleOwner",
        "modelDefinition": {
            "fields": {
                "role": "text",
                "role_name": "text",
                "entitlement": "text",
                "entitlement_name": "text",
                "user": "text",
                "user_name": "text",
                "score": "decimal",
                "justification": {
                    "type": "list",
                    "typeDef": "<text>"
                },
                "app_id": "text",
                "app_name": "text",
                "high_risk": "text"
            },
            "key": [
                "role",
                "entitlement",
                "score",
                "app_id",
                "user"
            ],
            "table_name": "usr_scores_by_role"
        }
    },
    "RuleAccessDecisionConf": {
        "name": "RuleAccessDecision",
```

```
        "modelDefinition": {
            "fields": {
                "entitlement": "text",
                "justification": {
                    "type": "frozen",
                    "typeDef": "<list<text>>"
                },
                "is_autocertify": "boolean",
                "date_autocertified": "timestamp",
                "is_autorequest": "boolean",
                "date_autorequested": "timestamp",
                "author": "text",
                "author_name": "text",
                "author_type": "text",
                "reason": "text"
            },
            "key": [
                "entitlement",
                "justification"
            ],
            "table_name": "rule_access_decisions"
        }
    },
    "RuleAccessDecisionHistoryConf": {
        "name": "RuleAccessDecisionHistory",
        "modelDefinition": {
            "fields": {
                "entitlement": "text",
                "justification": {
                    "type": "frozen",
                    "typeDef": "<list<text>>"
                },
                "is_autocertify": "boolean",
                "date_autocertified": "timestamp",
                "is_autorequest": "boolean",
                "date_autorequested": "timestamp",
                "author": "text",
                "author_name": "text",
                "author_type": "text",
                "reason": "text",
                "date_created": "timestamp"
            },
            "key": [
                "entitlement",
                "justification",
                "date_created"
            ],
            "table_name": "rule_access_decisions_history"
        }
    },
    "UserAccessDecisionConf": {
        "name": "UserAccessDecision",
        "modelDefinition": {
            "fields": {
                "user": "text",
                "entitlement": "text",
                "is_certified": "boolean",
                "date_certified": "timestamp",
                "is_revoked": "boolean",
```

```
                "date_revoked": "timestamp",
                "is_requested": "boolean",
                "date_requested": "timestamp",
                "author": "text",
                "author_name": "text",
                "author_type": "text",
                "reason": "text",
                "is_processed": "boolean",
                "is_archived": "boolean"
            },
            "key": [
                "user",
                "entitlement"
            ],
            "table_name": "user_access_decisions"
        }
    },
    "UserAccessDecisionHistoryConf": {
        "name": "UserAccessDecisionHistory",
        "modelDefinition": {
            "fields": {
                "user": "text",
                "entitlement": "text",
                "is_certified": "boolean",
                "date_certified": "timestamp",
                "is_revoked": "boolean",
                "date_revoked": "timestamp",
                "is_requested": "boolean",
                "date_requested": "timestamp",
                "author": "text",
                "author_name": "text",
                "author_type": "text",
                "reason": "text",
                "is_processed": "boolean",
                "is_archived": "boolean",
                "date_created": "timestamp"
            },
            "key": [
                "user",
                "entitlement",
                "date_created"
            ],
            "table_name": "user_access_decisions_history"
        }
    },
    "UserConf": {
        "name": "User",
        "modelDefinition": {
            "fields": {
                "user": "text",
                "chiefyesno": "text",
                "city": "text",
                "costcenter": "text",
                "jobcodename": "text",
                "lineofbusiness": "text",
                "managername": "text",
                "usrdepartmentname": "text",
                "userdisplayname": "text",
                "usremptype": "text",
```

```
                "usrmanagerkey": "text",
                "isactive": "text"
            },
            "key": [
                "user"
            ],
            "table_name": "user"
        }
    },
    "UserScoreConf": {
        "name": "UserScore",
        "modelDefinition": {
            "fields": {
                "manager": "text",
                "user": "text",
                "manager_name": "text",
                "user_name": "text",
                "score": "decimal",
                "entitlement": "text",
                "entitlement_name": "text",
                "justification": {
                    "type": "list",
                    "typeDef": "<text>"
                },
                "app_id": "text",
                "app_name": "text",
                "high_risk": "text"
            },
            "key": [
                "app_id",
                "manager",
                "user",
                "entitlement"
            ],
            "table_name": "usr_scores_by_manager"
        }
    },
    "UserEntitlementMappingsConf": {
        "name": "UserEntitlementMappings",
        "modelDefinition": {
            "fields": {
                "user": "text",
                "ent": "text",
                "high_risk": "text",
                "is_assigned": "text",
                "last_usage": "timestamp"
            },
            "key": [
                "user",
                "ent"
            ],
            "table_name": "user_entitlement_mappings"
        }
    },
    "SupervisorAppEnttConf": {
        "name": "SupervisorAppEntt",
        "modelDefinition": {
            "fields": {
                "manager": "text",
```

```
                "entitlement": "text",
                "app_id": "text"
            },
            "key": [
                "manager",
                "app_id"
            ],
            "table_name": "app_entitlement_by_manager"
        }
    },
    "FilteringOptionsModelConf": {
        "name": "FilteringOptions",
        "modelDefinition": {
            "fields": {
                "role": "text",
                "type": "text",
                "owner_id": "text",
                "group": "text",
                "id": "text",
                "name": "text",
                "objects": {
                    "type": "list",
                    "typeDef": "<text>"
                }
            },
            "key": [
                "id"
            ],
            "table_name": "filtering_options"
        }
    },
    "CompanyViewMostCriticalEnttConf": {
        "name": "CompanyViewMostCriticalEntt",
        "modelDefinition": {
            "fields": {
                "org": "text",
                "entt_id": "text",
                "entt_name": "text",
                "high": "int",
                "medium": "int",
                "seq": "int",
                "low": "int",
                "total_employees": "int",
                "avg_conf_score": "float"
            },
            "key": [
                "org",
                "entt_id"
            ],
            "table_name": "company_view_most_critical_entt"
        }
    },
    "AutoprovisionEntitlementResults": {
        "name": "AutoprovisionEntitlementResults",
        "modelDefinition": {
            "fields": {
                "decision": "int",
                "score": "float",
                "threshold": "float",
```

```
                "app_id": "text",
                "app_name": "text",
                "entitlement_name": "text",
                "entitlement": "text",
                "user": "text",
                "user_name": "text",
                "created": "timestamp",
                "updated": "timestamp"
            },
            "key": [
                "decision",
                "entitlement",
                "user"
            ],
            "table_name": "autoprovision_entitlement_results"
        }
    },
    "EntitlementsCounts": {
        "name": "EntitlementsCounts",
        "modelDefinition": {
            "fields": {
                "type": "text",
                "count": "int",
                "entitlement": "text"
            },
            "key": [
                "type",
                "count",
                "entitlement"
            ],
            "clustering_order": {
                "count": "desc",
                "entitlement": "desc"
            },
            "table_name": "entitlements_counts"
        }
    },
    "MaxScoreEntitlementsUserCount": {
        "name": "MaxScoreEntitlementsUserCount",
        "modelDefinition": {
            "fields": {
                "max_score": "float",
                "users_count": "int",
                "entitlement": "text"
            },
            "key": [
                "max_score",
                "users_count",
                "entitlement"
            ],
            "table_name": "max_score_entitlements_user_count"
        }
    },
    "MinScoreEntitlementsUserCount": {
        "name": "MinScoreEntitlementsUserCount",
        "modelDefinition": {
            "fields": {
                "min_score": "float",
                "users_count": "int",
```

```
                "entitlement": "text"
            },
            "key": [
                "min_score",
                "users_count",
                "entitlement"
            ],
            "table_name": "min_score_entitlements_user_count"
        }
    },
    "EntitlementDrivingFactorConf": {
        "name": "EntitlementDrivingFactor",
        "modelDefinition": {
            "fields": {
                "ent": "text",
                "attribute": "text",
                "count": "int"
            },
            "key": [
                "ent",
                "attribute"
            ],
            "table_name": "entitlement_driving_factor"
        }
    },
    "ReportsConf": {
        "RoleMining": "RoleMining",
        "Automatic Re-certification Feed": "AutomaticRecertificationFeed",
        "Full Output (IDM) Feed": "FullOutputFeed",
        "Anomaly Report": "AnomalyReport",
        "Recommend Predictions": "RecommendPredictions",
        "Event Based Certification": "EventBasedCertification"
    },
    "RoleMining": {
        "name": "RoleMining",
        "modelDefinition": {
            "fields": {
                "policy": {
                    "type": "frozen",
                    "typeDef": "<list<text>>"
                },
                "total_employees": "int",
                "entt_id": {
                    "type": "list",
                    "typeDef": "<text>"
                },
                "entt_name": {
                    "type": "list",
                    "typeDef": "<text>"
                },
                "total_entts": "int",
                "role": "int"
            },
            "key": [
                "policy"
            ],
            "table_name": "role_mining"
        }
    },
```

```
    "AutomaticRecertificationFeed": {
        "name": "AutomaticRecertificationFeed",
        "modelDefinition": {
            "fields": {
                "user": "text",
                "entitlement": "text",
                "app_id": "text",
                "app_name": "text",
                "auto_recert": "text",
                "chiefyesno": "text",
                "city": "text",
                "costcenter": "text",
                "ent_size": "decimal",
                "entitlement_name": "text",
                "event_recert": "text",
                "freq": "decimal",
                "frequnion": "decimal",
                "jobcodename": "text",
                "justification": {
                    "type": "list",
                    "typeDef": "<text>"
                },
                "lineofbusiness": "text",
                "lineofbusinesssubgroup": "text",
                "managername": "text",
                "score": "decimal",
                "user_name": "text",
                "userdepartmentname": "text",
                "userdisplayname": "text",
                "usremptype": "text",
                "usrmanagerkey": "text"
            },
            "key": [
                "user",
                "entitlement"
            ],
            "table_name": "master_feed"
        }
    },
    "FullOutputFeed": {
        "name": "FullOutputFeed",
        "modelDefinition": {
            "fields": {
                "user": "text",
                "entitlement": "text",
                "app_id": "text",
                "app_name": "text",
                "auto_recert": "text",
                "chiefyesno": "text",
                "city": "text",
                "costcenter": "text",
                "ent_size": "decimal",
                "entitlement_name": "text",
                "event_recert": "text",
                "freq": "decimal",
                "frequnion": "decimal",
                "jobcodename": "text",
                "justification": {
                    "type": "list",
```

```
                    "typeDef": "<text>"
                },
                "lineofbusiness": "text",
                "lineofbusinesssubgroup": "text",
                "managername": "text",
                "score": "decimal",
                "user_name": "text",
                "userdepartmentname": "text",
                "userdisplayname": "text",
                "usremptype": "text",
                "usrmanagerkey": "text"
            },
            "key": [
                "user",
                "entitlement"
            ],
            "table_name": "master_feed"
        }
    },
    "RecommendPredictions": {
        "name": "RecommendPredictions",
        "modelDefinition": {
            "fields": {
                "usr_key": "text",
                "ent": "text",
                "conf": "decimal",
                "freq": "decimal",
                "frequnion": "decimal",
                "rule": {
                    "type": "list",
                    "typeDef": "<text>"
                },
                "last_usage": "text"
            },
            "key": [
                "usr_key",
                "ent"
            ],
            "table_name": "recommend_predictions"
        }
    },
    "AnomalyReport": {
        "name": "AnomalyReport",
        "modelDefinition": {
            "fields": {
                "user": "text",
                "user_name": "text",
                "manager_name": "text",
                "entitlement": "text",
                "entitlement_name": "text",
                "justification": {
                    "type": "list",
                    "typeDef": "<text>"
                },
                "app_name": "text",
                "confidence": "float",
                "avg_conf_score": "float",
                "total_assignees": "int",
                "num_below_conf_threshold": "int",
```

```
                "percent_below_threshold": "float",
                "freq": "decimal",
                "frequnion": "decimal",
                "median": "float",
                "last_usage": "timestamp"
            },
            "key": [
                "user_name",
                "avg_conf_score"
            ],
            "table_name": "anomaly_report"
        }
    },
  "EventBasedCertification": {
        "name": "EventBasedCertification",
        "modelDefinition": {
            "fields": {
                "id": "text",
                "type": "text",
                "batch_id": "int",
                "original": "text",
                "update": "text"
            },
            "key": [
                "id",
                "type",
                "batch_id"
            ],
            "table_name": "event_based_certification"
        }
    },
    "FilteringOptionsConf": {
        "filteringOptions": [
            {
                "groupName": "CITY",
                "title": "CITY",
                "optionTextField": "id"
            }
        ]
    },
    "JobStatus": {
        "name": "JobStatus",
        "modelDefinition": {
            "fields": {
                "job_name": "text",
                "start_time": "text",
                "batch_id": "int",
                "end_time": "text",
                "flag": "text"
            },
            "key": [
                "job_name",
                "start_time",
                "batch_id"
            ],
            "table_name": "job_status"
        }
    },
    "EntitlementAssignmentConfSummary": {
```

```
        "name": "EntitlementAssignmentConfSummary",
        "modelDefinition": {
            "fields": {
                "timestamp": "timestamp",
                "num_high_conf_assignments": "int",
                "num_low_conf_assignments": "int",
                "num_med_conf_assignments": "int"
            },
            "key": [
                "timestamp"
            ],
            "table_name": "entitlement_assignment_conf_summary"
        }
    },
    "OrgNameConf": {
        "orgName": "test"
    },
    "ConfidenceScoreThresholdsConf": {
        "thresholds": {
            "top": 1.01,
            "high": 0.75,
            "medium": 0.35,
            "low": 0,
            "autoAccess": 0.5
        }
    },
    "ConfigThresholdsConf": {
        "volumeThresholds": {
            "high": 90,
            "low": 20
        }
    },
    "MostAssigned": {
        "count": 100
    },
    "HighVolume": {
        "high": {
            "minScore": 0.9,
            "minUsersCount": 100
        },
        "low": {
            "maxScore": 0.2,
            "minUsersCount": 100
        }
    },
    "UIConfig": {
        "userDisplayNameKey": "userdisplayname"
    },
    "UIHRData": {
        "user": "User",
        "chiefyesno": "Chief",
        "city": "City",
        "costcenter": "Cost Center",
        "jobcodename": "Job Code Name",
        "lineofbusiness": "Line of Business",
        "managername": "Manager",
        "usrdepartmentname": "Department",
        "userdisplayname": "User Display Name",
        "usremptype": "Employee Type",
```

```
    "UIJustifications": {
        "USR_KEY": "User",
        "CITY": "City",
        "USR_DEPARTMENT_NAME": "Department",
        "COST_CENTER": "Cost Center",
        "JOBCODE_NAME": "Job Code Name",
        "LINE_OF_BUSINESS": "Line of Business",
        "LINE_OF_BUSINESS_SUBGROUP": "Line of Business Subgroup",
        "CHIEF_YES_NO": "Chief",
        "USR_EMP_TYPE": "Employee Type",
        "MANAGER_NAME": "Manager",
        "USR_DISPLAY_NAME": "User Display Name",
        "USR_MANAGER_KEY": "Manager Key"
    },
    "HighRiskConf": {
        "filterValue": "1"
    },
    "JustificationDelimeter": {
        "justificationDelimeter": "_"
    },
    "SearchConf": {
        "name": "Search",
        "modelDefinition": {
            "fields": {
                "userdisplayname": "text",
                "user": "text",
                "isentitlementowner": "boolean",
                "issupervisor": "boolean",
                "isapplicationowner": "boolean"
            },
            "indexes": [
                "userdisplayname"
            ],
            "key": [
                "user"
            ],
            "table_name": "search_user"
        }
    },
    "EntitlementsConf": {
        "name": "Entitlements",
        "modelDefinition": {
            "fields": {
                "id": "text",
                "app_id": "text",
                "app_name": "text",
                "entt_id_at_app": "text",
                "entt_name": "text"
            },
            "indexes": [
                "entt_name"
            ],
            "key": [
                "id"
            ],
            "table_name": "entitlements"
        }
    },
   "PermissionsConf": {
```

```
        "permissions": {
            "Zoran Admin": {
                "title": "Admin",
                "can": "*"
            },
            "Zoran Application Owner": {
                "title": "Application Owner",
                "can": [
                    "SHOW__APPLICATION_VIEW",
                    "FILTER__ENTITLEMENTS",
                    "SEARCH__USER_ENTITLEMENTS",
                    "SHOW_OVERVIEW_PAGE",
                    "SHOW__ENTITLEMENT",
                    "SHOW__ENTITLEMENT_USERS",
                    "SHOW__FILTER_OPTIONS",
                    "SHOW__ROLEOWNER_UNSCORED_ENTITLEMENTS",
                    "SHOW__ROLE_OWNER_PAGE",
                    "SHOW__ROLE_OWNER_USER_PAGE",
                    "SHOW__ROLE_OWNER_ENT_PAGE",
                    "SHOW__ROLE_OWNER_AUTO_DATA",
                    "SHOW__USER_ENTITLEMENTS",
                    "SHOW__UNSCORED_ENTITLEMENTS",
                    "SHOW__RULES_BY_APP_OWNER",
                    "CERTIFY__ENTITLEMENTS_TO_USERS",
                    "CERTIFY__USERS_TO_ENTITLEMENTS",
                    "REVOKE__CERTIFY_ACCESS"
                ]
            },
            "Zoran Entitlement Owner": {
                "title": "Entitlement Owner",
                "can": [
                    "FILTER__ENTITLEMENTS",
                    "SEARCH__USER_ENTITLEMENTS",
                    "SHOW_OVERVIEW_PAGE",
                    "SHOW__ENTITLEMENT",
                    "SHOW__ENTITLEMENT_USERS",
                    "SHOW__FILTER_OPTIONS",
                    "SHOW__ROLEOWNER_UNSCORED_ENTITLEMENTS",
                    "SHOW__ROLE_OWNER_PAGE",
                    "SHOW__ROLE_OWNER_USER_PAGE",
                    "SHOW__ROLE_OWNER_ENT_PAGE",
                    "SHOW__ROLE_OWNER_AUTO_DATA",
                    "SHOW__USER_ENTITLEMENTS",
                    "SHOW__UNSCORED_ENTITLEMENTS",
                    "SHOW__RULES_BY_ENTT_OWNER",
                    "CERTIFY__ENTITLEMENTS_TO_USERS",
                    "CERTIFY__USERS_TO_ENTITLEMENTS",
                    "REVOKE__CERTIFY_ACCESS"
                ]
            },
            "Zoran Executive": {
                "title": "Executive",
                "can": [
                    "SEARCH__USER",
                    "SHOW__ASSIGNMENTS_STATS",
                    "SHOW__COMPANY_PAGE",
                    "SHOW__COMPANY_COVERAGE_PAGE",
                    "SHOW__COMPANY_ENTITLEMENTS_PAGE",
                    "SHOW__COMPANY_ENTITLEMENTS_DATA",
```

```
                "SEARCH__USER",
                "FILTER__ENTITLEMENTS",
                "SHOW_OVERVIEW_PAGE",
                "SHOW__EMPLOYEE",
                "SHOW__FILTER_OPTIONS",
                "SHOW__SUPERVISOR_PAGE",
                "SHOW__SUPERVISOR_DETAILS_PAGE",
                "SHOW__SUPERVISOR_ENT_DATA",
                "SHOW__SUPERVISOR_USER_DATA",
                "SHOW__SUPERVISOR_ENTITLEMENT_USERS",
                "SHOW__SUPERVISOR_USER_ENTITLEMENTS",
                "SEARCH__SUPERVISOR_USER_ENTITLEMENTS",
                "SHOW__SUPERVISOR_UNSCORED_ENTITLEMENTS",
                "CERTIFY__ENTITLEMENTS_TO_USERS",
                "CERTIFY__USERS_TO_ENTITLEMENTS",
                "REVOKE__CERTIFY_ACCESS"
            ]
        },
        "Zoran User": {
            "title": "User",
            "can": [
                "SHOW__ENTITLEMENT",
                "SHOW__USER",
                "SHOW__CERTIFICATIONS"
            ]
        }
    }
},
"AllowedAttributesForFiltering": {
    "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"
    ]
},
"UIJustification": {
    "USR_KEY": "User",
    "CHIEF_YES_NO": "Chief",
    "CITY": "City",
    "COST_CENTER": "Cost Center",
    "JOBCODE_NAME": "Job Code Name",
    "LINE_OF_BUSINESS": "Line of Business",
    "MANAGER_NAME": "Manager",
    "USR_DEPARTMENT_NAME": "Department",
    "USR_DISPLAY_NAME": "User Display Name",
    "USR_EMP_TYPE": "Employee Type",
    "USR_MANAGER_KEY": "Manager Key",
```

```
        "IS_ACTIVE": "Active"
    }
}
```

Glossary

anomaly report	A report that identifies potential anomalous assignments.
as-is predictions	A process where confidence scores are assigned to the entitlements that users have.
auto-certify	An action that an entitlement owner can do to approve a justification. Auto-certify indicates that anyone who has the justification is automatically approved for the entitlement.
auto-request	An action that an entitlement owner can do to approve a justification. Auto-request indicates that anyone who matches these justification attributes but may not already have access should automatically get provisioned for this entitlement.
confidence score	A score from a scale from 0 to 100% that indicates the strength of correlation between an assigned entitlement and a user's data profile.
data audit	A pre-analytics process that audits the seven data files to ensure data validity with the client.
data ingestion	A pre-analytics process that pushes the seven .csv files into the Cassandra database. This allows the entire training process to be performed from the database.
data sparsity	A reference to data that has null values. Autonomous Identity requires dense, high quality data with very few null values in the user attributes to get accurate analysis scores.
data validation	A pre-analytics process that tests the data to ensure that the content is correct and complete prior to the training process.

driving factor	An association rule that is a key factor in a high entitlement confidence score. Any rule that exceeds a confidence threshold level (e.g., 75%) is considered a driving factor.
entitlement	An entitlement is a specialized type of `assignment`. A user or device with an entitlement gets access rights to specified resources.
insight report	A report that provides metrics on the rules and predictions generated in the analytics run.
recommendation	A process run after the as-is predictions that assigns confidence scores to all entitlements and recommends entitlements that users do not currently have. If the confidence score meets a threshold, set by the `conf_thresh` property in the configuration file, the entitlement will be recommended to the user in the UI console.
resource	An external system, database, directory server, or other source of identity data to be managed and audited by an identity management system.
REST	Representational State Transfer. A software architecture style for exposing resources, using the technologies and protocols of the World Wide Web. REST describes how distributed data objects, or resources, can be defined and addressed.
stemming	A process that occurs after training that removes similar association rules that exist in a parent-child relationship. If the child meets three criteria, then it will be removed by the system. The criteria are: 1) the child must match the parent; 2) the child (e.g., [San Jose, Finance]) is a superset of the parent rule. (e.g., [Finance]); 3) the child and parent's confidence scores are within a +/- range of each other. The range is set in the configuration file.
training	A multi-step process that generates the association rules with confidence scores for each entitlement. First, Autonomous Identity models the frequent itemsets that appear in the user attributes for each user. Next, Autonomous Identity merges the user attributes with the entitlements that were assigned to the user. It then applies association rules to model the sets of user attributes that result in an entitlement access and calculates confidence scores, based on their frequency of appearances in the dataset.