



# Admin Guide

/ Autonomous Identity 2020.6.4

Latest update: 2020.6.4

---

Copyright © 2020 ForgeRock AS.

## Abstract

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



This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License.

To view a copy of this license, visit <https://creativecommons.org/licenses/by-nc-nd/3.0/> or send a letter to Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.

© Copyright 2010-2020 ForgeRock, Inc. All rights reserved. ForgeRock is a registered trademark of ForgeRock, Inc. Other marks appearing herein may be trademarks of their respective owners.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, and distribution. No part of this product or document may be reproduced in any form by any means without prior written authorization of ForgeRock and its licensors, if any.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESSED OR IMPLIED CONDITIONS, REPRESENTATIONS, AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

DejaVu Fonts

Bitstream Vera Fonts Copyright

Copyright (c) 2003 by Bitstream, Inc. All Rights Reserved. Bitstream Vera is a trademark of Bitstream, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy of the fonts accompanying this license ("Fonts") and associated documentation files (the "Font Software"), to reproduce and distribute the Font Software, including without limitation the rights to use, copy, merge, publish, distribute, and/or sell copies of the Font Software, and to permit persons to whom the Font Software is furnished to do so, subject to the following conditions:

The above copyright and trademark notices and this permission notice shall be included in all copies of one or more of the Font Software typefaces.

The Font Software may be modified, altered, or added to, and in particular the designs of glyphs or characters in the Fonts may be modified and additional glyphs or characters may be added to the Fonts, only if the fonts are renamed to names not containing either the words "Bitstream" or the word "Vera".

This License becomes null and void to the extent applicable to Fonts or Font Software that has been modified and is distributed under the "Bitstream Vera" names.

The Font Software may be sold as part of a larger software package but no copy of one or more of the Font Software typefaces may be sold by itself.

THE FONT SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OF COPYRIGHT, PATENT, TRADEMARK, OR OTHER RIGHT. IN NO EVENT SHALL BITSTREAM OR THE GNOME FOUNDATION BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, INCLUDING ANY GENERAL, SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF THE USE OR INABILITY TO USE THE FONT SOFTWARE OR FROM OTHER DEALINGS IN THE FONT SOFTWARE.

Except as contained in this notice, the names of GNOME, the GNOME Foundation, and Bitstream Inc., shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Font Software without prior written authorization from the GNOME Foundation or Bitstream Inc., respectively. For further information, contact: [fonts@gnome dot org](mailto:fonts@gnome dot org).

Arev Fonts Copyright

Copyright (c) 2006 by Tavmjong Bah. All Rights Reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of the fonts accompanying this license ("Fonts") and associated documentation files (the "Font Software"), to reproduce and distribute the modifications to the Bitstream Vera Font Software, including without limitation the rights to use, copy, merge, publish, distribute, and/or sell copies of the Font Software, and to permit persons to whom the Font Software is furnished to do so, subject to the following conditions:

The above copyright and trademark notices and this permission notice shall be included in all copies of one or more of the Font Software typefaces.

The Font Software may be modified, altered, or added to, and in particular the designs of glyphs or characters in the Fonts may be modified and additional glyphs or characters may be added to the Fonts, only if the fonts are renamed to names not containing either the words "Tavmjong Bah" or the word "Arev".

This License becomes null and void to the extent applicable to Fonts or Font Software that has been modified and is distributed under the "Tavmjong Bah Arev" names.

The Font Software may be sold as part of a larger software package but no copy of one or more of the Font Software typefaces may be sold by itself.

THE FONT SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OF COPYRIGHT, PATENT, TRADEMARK, OR OTHER RIGHT. IN NO EVENT SHALL TAVMJONG BAH BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, INCLUDING ANY GENERAL, SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF THE USE OR INABILITY TO USE THE FONT SOFTWARE OR FROM OTHER DEALINGS IN THE FONT SOFTWARE.

Except as contained in this notice, the name of Tavmjong Bah shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Font Software without prior written authorization from Tavmjong Bah. For further information, contact: [tavmjong@free.fr](mailto:tavmjong@free.fr).

FontAwesome Copyright

Copyright (c) 2017 by Dave Gandy, <https://fontawesome.com/>.

This Font Software is licensed under the SIL Open Font License, Version 1.1. See <https://opensource.org/licenses/OFL-1.1>.

---

# Table of Contents

Overview .....	v
1. Features .....	1
2. Stopping and Starting .....	2
Stopping Docker .....	2
Re-Starting Docker .....	2
Shutting Down Cassandra .....	2
Re-Starting Cassandra .....	3
Shutting Down Spark .....	3
Re-Starting Spark .....	3
3. Exporting and Importing Data .....	4
4. Backing Up and Restoring .....	7
Backing Up Cassandra .....	7
Restore Cassandra .....	7
5. Creating and Removing Users .....	9
Log in to <b>phpldapadmin</b> .....	9
Add User to a Group .....	10
Delete a User .....	10
6. Customize the Domain and Namespace .....	11
7. Change the Vault Passwords .....	12
8. Accessing Log Files .....	13
Getting Docker Container Information .....	13
Getting Cassandra Logs .....	14
Other Useful Cassandra Monitoring Tools and Files .....	15
Configuring JMX Authentication for Cassandra .....	15
Apache Spark Logs .....	16
9. Set Up Single Sign-On .....	18
Set Up SSO Using ForgeRock AM .....	18
10. Setting the Session Duration .....	22
11. Prepare Spark Environment .....	23
Determine the Optimal Core and Memory Configuration .....	23
Configure Spark History .....	25
Configure Spark SSL .....	26
12. Data Preparation .....	27
Data Collection .....	27
CSV Files and Schema .....	28
13. Run the Analytics Pipeline .....	30
Analytic Actions .....	31
Create Initial Analytics Template .....	32
Ingest the Data Files .....	33
Run Data Validation .....	33
Run Data Audit .....	34
Run Training .....	35
Run Predictions and Recommendations .....	36
Run the Insight Report .....	37

Run Anomaly Report .....	38
Publish the Analytics Data .....	38
Create the Analytics UI Config File .....	39
Run Full Pipeline .....	39
A. Appendix A: The analytics_init_config.yml File .....	41
B. Appendix B: The ui-config.json File .....	44
Glossary .....	58













# 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

 <p>Stopping and Starting</p> <p>Learn how to stop and start Autonomous Identity.</p>	 <p>Backup and Restore</p> <p>Learn how to back up and restore the Apache Cassandra data.</p>	 <p>Creating and Removing Users</p> <p>Learn how to creating and remove users.</p>
 <p>Customize Domain</p> <p>Learn how to change the domain name and target environment.</p>	 <p>Change Vault Passwords</p> <p>Learn how to change the vault passwords.</p>	 <p>Access Logs</p> <p>Learn how to monitor Autonomous Identity using the logs.</p>
 <p>Set Up Single Sign-On</p> <p>Learn how to set up single sign-on using OpenID Connect.</p>	 <p>Prepare Spark</p> <p>Learn how to prepare Apache Spark.</p>	 <p>Data Preparation</p> <p>Learn how to prepare the data.</p>
		

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

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 that could potentially be a security risk. The UI also lets you quickly identify the 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.
- **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.

## 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
```

## Re-Starting 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.

```
Datcenter: 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 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

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.

### Exporting Data

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

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

2. Export the database.

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

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

### Importing Data

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,
managename text,
usrdepartmentname text,
userdisplayname text,
usremptytype 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 dlocal_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,
  managename text,
  usrdepartmentname text,
  userdisplayname text,
  usremptytype 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 dlocal_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';
```

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

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

- 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
```

- Verify that the data is present in the new 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 database. Cassandra is an open-source, NoSQL database system where data is distributed across multiple nodes in a master-less cluster.

For single-node deployments, however, you need to back up Cassandra 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, available on the ForgeRock Google Cloud Registry ([gcr.io](https://gcr.io)).

## Backing Up Cassandra

1. On the ForgeRock Google Cloud Registry ([gcr.io](https://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](https://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>
```

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

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

3. On the phpldapadmin page, click login in the navigation bar on the left side.
4. On the Authenticate to server openldap page, enter `cn=admin,dc=zoran,dc=com`, and then enter your admin password. Click Authenticate to proceed.
5. On the left-hand navigation bar, expand the menu, and then click `ou=People`.
6. Under `ou=People`, select any user to see their profile, and then click Copy or move this entry.
7. 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
```

8. 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`.
9. 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 four groups: User, Supervisor, Executive, Entitlement 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

# 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: abc.com
target_environment: myid
```

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 7

# 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 8

# Accessing Log Files

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

## Getting Docker Container Information

- 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
```

- 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
```

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

```
$ docker ps -a
```

- Get information on the docker images on your system.

```
$ docker images
```

- Get docker service information on your system.

```
$ docker service ls
```

- 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/`.

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

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

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

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

- 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
- 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"`
- 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.

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

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

- 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
```

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

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

## Configuring JMX Authentication for Cassandra

By default, Cassandra makes JMX accessible only from localhost. Autonomous Identity provides a configuration script to enable remote JMX authentication for Cassandra on the target node. Once enabled, you cannot use the **nodetool** command, for example, **nodetool status** without authentication parameters.

You can obtain a tar file from ForgeRock with the following files that you run on the deployer node:

- `enable_jmxauth_cassandra.sh`
- `jmxremote.access`
- `jmxremote.password`

The script has the following syntax:

```
Usage:
./enable_jmxauth_cassandra.sh
-u <remote ssh username for example autoid>
-i <cassandra server IPs list where first ip is seed node>
-c <cassandra home path for example /opt/autoid/apache-cassandra-3.11.2>
-p <cassandra dba password>
-s [Optional: SSL enabled true/false. Default is true]
-h <usage>
```

## Enable JMX Authentication for Cassandra

1. Obtain the tar file from ForgeRock.
2. On the deployer node, unpack the tar file and copy the files to the home directory, for example, `/home/autoid`.
3. Update the `jmxremote.access` and `jmxremote.password` files.

- a. Using a text editor, add the username and privilege in the `jmxremote.access` file. For example:

```
$ autoid readwrite
```

- b. Using a text editor, add the username and password in the `jmxremote.password` file. For example:

```
$ autoid <password>
```

4. Run the `enable_jmxauth_cassandra.sh` script to enable JMX authentication with the required properties.

```
$ Usage:
./enable_jmxauth_cassandra.sh
-u autoid
-i <server IPs list where first IP is seed node>
-c /opt/autoid/apache-cassandra-3.11.2
-p <cassandra dba password>
```

5. SSH to the target node.
6. Run `nodetool` with the Cassandra user and password.

```
$ nodetool -u autoid -pw <password> status
```

### Note

After enabling JMX remote authentication, you cannot use `nodetool status` alone without the `--username` and `--password` parameters.

## Apache Spark Logs

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

- To get an overall status of the Spark server, point your browser to [elinks http://<spark-master-ip>:8080](http://<spark-master-ip>:8080).

- 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
```

- 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 9

# 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": "QJRGiQgr1c1s0E4Q8BNyyg",
  "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_wXm581f08IntYHHcwSQtJI",
  "s_hash": "b0htX8F73IMjSPeVAqxyTQ",
  "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 10

# 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>>
```

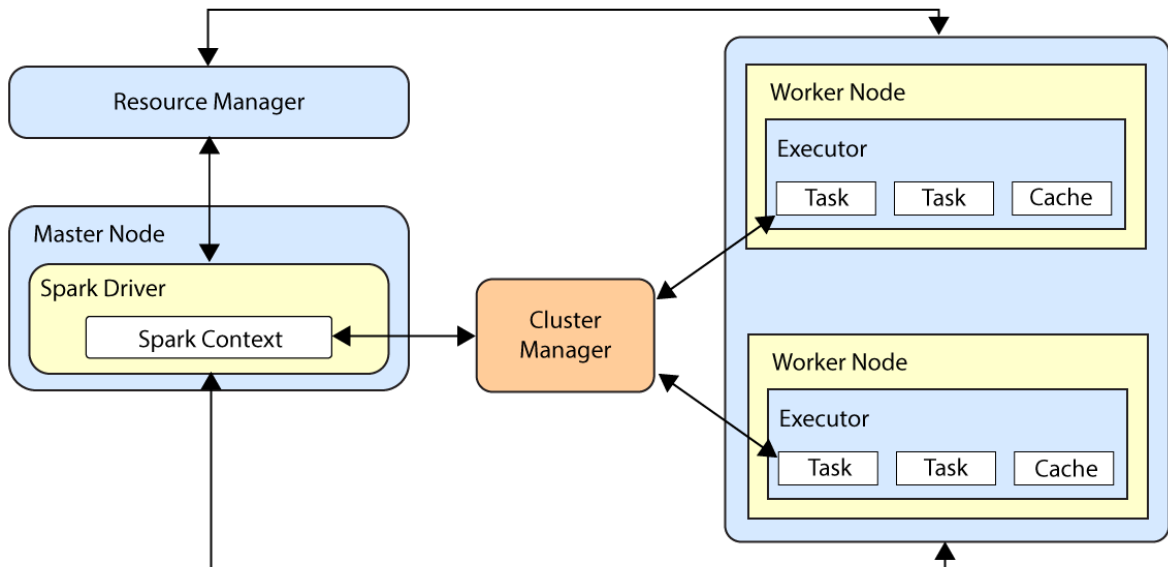
The Docker manager node re-initiates the worker node. Repeat this step on any other worker node.

## Chapter 11 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  $15 \times 6 = 90$ . 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 =  $17/6 \sim 3$ . 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,  $7\% \times 21$ ) = 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:

<b>Number of Nodes</b>	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
<b>Num cores/node</b>	8	8	8	8	8	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16
<b>Memory size/node</b>	32	32	32	32	32	32	32	32	32	32	64	64	64	64	64	64	64	64	64	64
<b>spark.total.cores</b>	3	3	3	3	3	3	3	3	3	3	4	4	4	4	4	5	5	5	5	5
<b>spark.executor.cores</b>	2	4	6	9	11	4	9	13	18	23	3	6	10	13	17	2	5	8	11	13
<b>spark.executor.memory</b>	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, create 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.eventLog.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 12

# 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 seven dataframes to seven csv files.

## 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 seven 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.

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:

Files	Schema
	<ul style="list-style-type: none"> <li>• <b>USR_KEY.</b> Specifies the user's unique ID.</li> <li>• <b>USR_DISPLAY_NAME.</b> Specifies the user's name. If not provided, use the user ID, but the name works best for the UI.</li> <li>• <b>USR_MANAGER_KEY.</b> Specifies the ID of the user's manager.</li> <li>• <b>USR_EMP_TYPE.</b> Specifies the employment status of the user, for example, PERMANENT, CONTRACT, EMPLOYEE, NON-EMPLOYEE, VENDOR, etc.</li> <li>• <b>IS_ACTIVE.</b> Specifies whether this user is an active employee. Sometimes companies submit inactive accounts, which is not included in the analysis.</li> </ul>
label.csv	<ul style="list-style-type: none"> <li>• <b>USR_KEY.</b> Specifies the user's unique ID.</li> <li>• <b>ENT.</b> Specifies the unique entitlement identifier.</li> <li>• <b>HIGH_RISK.</b> Determines whether an access is considered HIGH, MEDIUM, or LOW risk.</li> <li>• <b>IS_ASSIGNED.</b> Determines whether an access is assigned (used internally).</li> <li>• <b>LAST_USAGE.</b> Specifies the last time an entitlement was accessed.</li> </ul>
HRName.csv	<ul style="list-style-type: none"> <li>• <b>USR_KEY.</b> Specifies the user's unique ID.</li> <li>• <b>USR_NAME.</b> Specifies a human readable username. For example, <b>John Smith</b>.</li> </ul>
AppToEnt.csv	<ul style="list-style-type: none"> <li>• <b>ENT.</b> Specifies the unique entitlement identifier.</li> <li>• <b>APP_NAME.</b> Specifies a human readable application name..</li> <li>• <b>APP_ID.</b> Specifies the unique application identifier.</li> </ul>
RoleOwner.csv	<ul style="list-style-type: none"> <li>• <b>ROLE.</b> Specifies the unique user ID of the entitlement owner.</li> <li>• <b>ENT.</b> Specifies the unique entitlement identifier.</li> </ul>
JobAndDeptDesc.csv	<ul style="list-style-type: none"> <li>• <b>USR_KEY.</b> Specifies the user's unique ID.</li> <li>• <b>DEPARTMENT.</b> Specifies the human readable department.</li> <li>• <b>JOB_DESCRIPTION.</b> Specifies the human readable job description.</li> </ul>

## Chapter 13

# 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*".

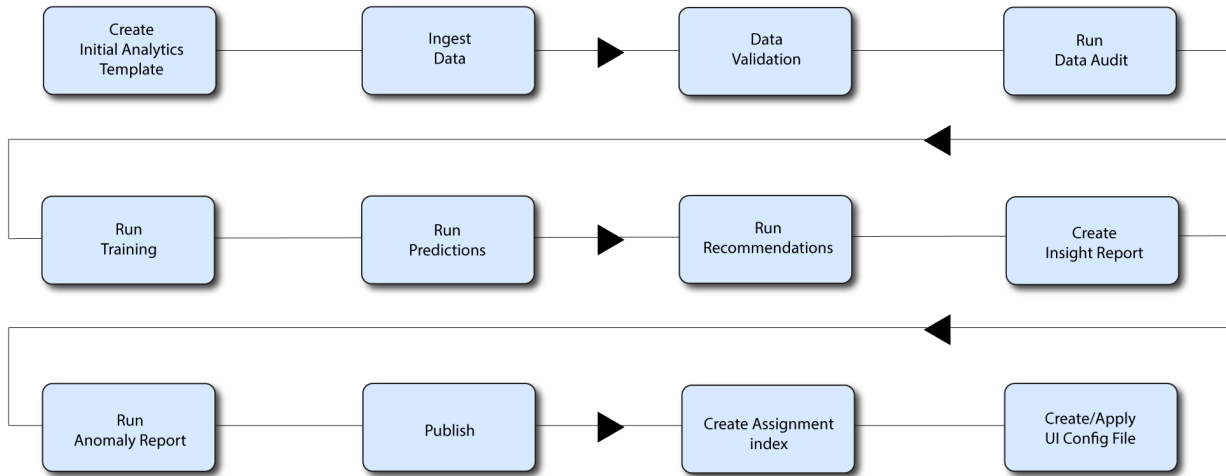
Next, run a job to validate the data, then, when acceptable, ingest the data into the database. After that, run a final audit of the data to ensure accuracy. If everything passes, run the data through its initial training process to create the association rules for each user-assigned entitlement. This 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 database, and then configure and apply any UI configuration changes to the system.

The general analytics process is outlined as follows:

## Autonomous Identity Analytics Pipeline



### 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
<code>analytics create-template</code>	Run this command to create the <code>analytics_init_config.yml</code> configuration file.
<code>analytics apply-template</code>	Apply the changes to <code>analytics_init_config.yml</code> file and create the <code>analytics_config.yml</code> file.
<code>analytics ingest</code>	Ingest data into Autonomous Identity.
<code>analytics validate</code>	Run data validation.

Command	Description
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 insight	Create the Insights report.
analytics anomaly	Create the Anomaly report.
analytics publish	Push the data to the Apache Cassandra backend.
analytics create-ui-config	Create the <code>ui_config.json</code> file.
analytics apply-ui-config	Apply the <code>ui_config.json</code> 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.

## 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 target node, create the initial configuration template. The command generates the `analytics_init_config.yml` in the `/data/conf/` directory.

```
$ analytics create-template
```
2. Edit the `analytics_init_config.yml` file for the Spark machine. Make sure to edit the `user_column_descriptions` and the `spark.total.cores` fields if you are submitting your own dataset. For more information, see "*Appendix A: The analytics\_init\_config.yml File*".
3. 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/
```
4. 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
```

## 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 database.

Ingest the data into the Cassandra database:

1. Make sure Cassandra 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:
/usr/local/lib/python3.6/site-packages/zoran_pipeline_scripts-0.32.0-py3.6.egg/EGG-INF0/scripts/
zinput_data_validation_script.py is successful
```

## Run Data Validation

Next, run the **analytics validate** command to check that the input data is accurate before you run it through the Analytics service.

- Validate the data.

```
$ analytics validate
```

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

```
Script:
/usr/local/lib/python3.6/site-packages/zoran_pipeline_scripts-0.32.0-py3.6.egg/EGG-INF0/scripts/
zinput_data_validation_script.py is successful
```

The validation script outputs a .csv file with 32 pass/fail tests across the seven input files. You can use the validation report to fix any issues with your data files. After the fixes, you can re-run the validation script to check that your files are complete.

The command checks the following items:

- All files contain the correct columns and column names.
- No duplicate rows in the files.
- No null values in the files.
- Check [features.csv](#).
  - No missing `USR_KEY`, `USR_MANAGER_KEY`, `USR_EMP_TYPE` values.

- No duplicate `USR_KEY` values. There should only be one row of entitlements per user. For example, if the user has six entitlements, there should be six rows respectively for each entitlement assignment.
- All `USR_MANAGER_KEYS` should also exist as `USR_KEYS`. This ensures that we have the user attribute information for all managers.
- Check `labels.csv`
  - All `USR_KEY` values in the `labels.csv` should also exist in the `features.csv` file.
- Check `HRName.csv`
  - No duplicate `USR_KEY` values.
  - All `USR_KEY` values in the `HRName.csv` should also exist in the `features.csv` file.
- Check `RoleOwner.csv`
  - No duplicate `ENT` values.
  - All `USR_KEY` values in the `RoleOwner.csv` should also exist in the `features.csv` file.
  - All `ENT` values in the `RoleOwner.csv` file should also exist in the `label.csv` file.
- Check `AppToEnt.csv`
  - No duplicate `ENT` values.
  - All `ENTs` in the `AppToEnt.csv` file should also exist in the `label.csv` file.
- Check `JobAndDeptDesc.csv`
  - No duplicate `USR_KEYS` values.
  - All `USR_KEY` values in the `JobAndDeptDesc.csv` file should also exist in the `features.csv` file.

## 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 as follows:

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 :
/usr/local/lib/python3.6/site-packages/zoran_pipeline_scripts-0.32.0-py3.6.egg/EGG-INF0/scripts/
zinput_data_audit_script.py is successful
```

The script provides the following metrics:

### *CSV File Audit*

File	Description
features.csv	<ul style="list-style-type: none"> <li>• Number of columns/categories (number of features).</li> <li>• Number of rows (number of users).</li> <li>• Number of unique values in each column.</li> <li>• Number of missing values in each column.</li> <li>• 10 most common values in each column.</li> </ul>
labels.csv	<ul style="list-style-type: none"> <li>• Number of rows (number of entitlement assignments).</li> <li>• Number of unique entitlements.</li> <li>• Number of users in entitlement assignments.</li> <li>• 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.</li> <li>• 10 most common entitlements (most assigned entitlements).</li> <li>• 10 most common users (users with the most assignments).</li> </ul>
RoleOwner.csv	<ul style="list-style-type: none"> <li>• Number of entitlements with owners.</li> <li>• Number of entitlement owners.</li> <li>• 10 most common entitlement owners.</li> <li>• Number of users without user attribute information that exist within user entitlement assignments.</li> </ul>
AppToEnt.csv	<ul style="list-style-type: none"> <li>• Number of unique entitlements.</li> <li>• Number of applications.</li> <li>• Number of entitlements per application.</li> </ul>

## 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:

1. Run the training command.

```
$ analytics train
```

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

```
Script : /usr/local/lib/python3.6/site-packages/zoran_analytics-0.32.0-py3.6.egg/EGG-INF0/scripts/ztrain.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 : /usr/local/lib/python3.6/site-packages/zoran_analytics-0.32.0-py3.6.egg/EGG-INF0/scripts/zas_is.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 : /usr/local/lib/python3.6/site-packages/zoran_analytics-0.32.0-py3.6.egg/EGG-INF0/scripts/zrecommend.py is successful
```

## Run the Insight Report

Next, run a report on the generated rules and predictions that were generated during the training and predictions runs.

Run the Analytics Insight Report:

```
$ analytics insight
```

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

```
Script : /usr/local/lib/python3.6/site-packages/zoran_analytics_reporting-0.32.0-py3.6.egg/EGG-INF0/scripts/zinsight.py is successful
```

The report provides the following insights:

- Number of entitlements and assignments received.
- Number of entitlements and assignments scored and unscored.
- Number of assignments scored >80% and <5%.
- Distribution of assignment confidence scores.
- List of the high volume, high confidence entitlements.
- List of the high volume, low confidence entitlements.

- List of users with the most entitlement accesses with an average confidence score of >80% and <5%.
- Breakdown of all applications and confidence scores of their assignments.
- Supervisors with most employees and confidence scores of their assignments.
- Entitlement owners with most entitlements and confidence scores of their assignments.
- List of the "Golden Rules", high confidence justifications that apply to a large volume of people.

## 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 database.

- Run the Anomaly report.

```
$ analytics anomaly
```

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

```
Script : /usr/local/lib/python3.6/site-packages/zoran_analytics_reporting-0.32.0-py3.6.egg/EGG-INF0/scripts/zanomaly.py is successful
```

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.

## Publish the Analytics Data

The final step in the analytics process is to 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 backend.

- Publish the data to the backend.

```
$ analytics publish
```

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

```
Script : /usr/local/lib/python3.6/site-packages/zoran_etl-0.32.0-py3.6.egg/EGG-INF0/scripts/zload.py 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. On the target node, 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
```

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
```

## 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`, and that any UI configuration changes are set in the `ui_config.json` file in the `/data/conf/` directory.

- Run the full pipeline.

```
$ analytics run-pipeline
```

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

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

```
*****
```

```
Pipe Line Ends
```

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

1. analytics ingest
2. analytics validate
3. analytics audit
4. analytics train
5. analytics predict-as-is
6. analytics predict-recommendation
7. analytics insight

8. analytics anomaly
9. analytics publish
10. analytics create-ui-config
11. analytics apply-ui-config

# 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: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
  USR_KEY                                                    # or equivalent)
LINE_OF_BUSINESS,
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:
  # spark base
  logging_level: WARN # Logging level
  # spark config
  config:
    #spark-submit params
    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.

  # other spark configuration properties
  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

```

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

The `ui-config.json` file is a configuration file for the Autonomous Identity user interface. You customize the parameters to allow each user to view certain aspects of the entitlements data.

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:

- Defines the permissions that are applied to each Autonomous Identity user.
- Defines what each user can see in the UI.
- Sets the fields for the UI.

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.

```
{
  "RevokeCertifyAccessConf": {
    "name": "RevokeCertifyAccess",
    "modelDefinition": {
      "fields": {
        "is_processed": "boolean",
        "is_archived": "boolean",
```

```

    "entitlement": "text",
    "user": "text",
    "user_score": "decimal",
    "manager": "text",
    "manager_decision": "int",
    "manager_date_created": "timestamp",
    "role_owner": "text",
    "role_owner_decision": "int",
    "role_owner_date_created": "timestamp",
    "date_created": "timestamp",
    "role_owner_auto_request_reason": "text",
    "role_owner_auto_certify_reason": "text",
    "justification": {
      "type": "frozen",
      "typeDef": "<list<text>>"
    }
  },
  "key": ["is_processed", "entitlement", "user", "date_created", "role_owner_decision",
"manager_decision"],
  "table_name": "revoke_certify_access_request"
}
},
"RecommendPredictionsConf": {
  "name": "RecommendPredictions",
  "modelDefinition": {
    "fields": {
      "usr_key": "text",
      "ent": "text",
      "conf": "decimal",
      "freq": "decimal",
      "frequion": "decimal",
      "rule": {
        "type": "list",
        "typeDef": "<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",
    "modelDefinition": {
        "fields": {
            "type": "text",
            "high": "int",
            "medium": "int",
            "low": "int",
            "null_conf": "int",
            "total": "int"
        },
        "key": ["type"],
        "table_name": "company_view_employee_type"
    }
},
"EntitlementAverageConfScoreConf": {
    "name": "EntitlementAverageConfScore",
    "modelDefinition": {
        "fields": {
            "org": "text",
            "avg_score": "decimal",
            "entitlement": "text"
        },
        "key": ["entitlement", "org"],
        "table_name": "entitlement_average_conf_score"
    }
},
"EntitlementUserScoresConf": {
    "name": "EntitlementUserScores",
    "modelDefinition": {
        "fields": {
            "entitlement": "text",
            "entitlement_name": "text",
            "freq": "decimal",
            "frequion": "decimal",
            "high_risk": "text",
            "user": "text",
            "user_name": "text",
            "score": "float",
            "justification": {
                "type": "list",
                "typeDef": "<text>"
            },
            "app_id": "text",
            "app_name": "text"
        },
        "indexes": [
            "user_name"
        ],
        "key": ["entitlement", "user"],
        "table_name": "entitlement_user_scores"
    }
}

```

```

    }
  },
  "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",
      "user_name": "text"
    },
    "key": ["manager", "user"],
    "table_name": "graph_by_manager"
  }
},
"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"
},
},
"UserConf": {
    "name": "User",
    "modelDefinition": {
        "fields": {
            "user": "text",
            "chief": "text",
            "city": "text",
            "costcenter": "text",
            "department": "text",
            "is_active": "text",
            "jobcode": "text",
            "lob": "text",
            "managername": "text",
            "usr_display_name": "text",
            "usr_emp_type": "text",
            "usr_manager_key": "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": {
      "type": "int",
      "owner_id": "text",
      "group": "text",
      "id": "text",
      "name": "text",
      "user_ids": "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",
      "user_name": "text",
      "entitlement": "text",
      "entitlement_name": "text",
      "app_id": "text",
      "app_name": "text",
      "auto_recert": "text",
      "event_recert": "text",
      "score": "decimal",
      "justification": {

```

```

        "type": "list",
        "typeDef": "<text>"
    },
    "manager": "text",
    "manager_name": "text"
},
"key": ["user", "entitlement"],
"table_name": "master_feed"
}
},
"FullOutputFeed": {
    "name": "FullOutputFeed",
    "modelDefinition": {
        "fields": {
            "user": "text",
            "user_name": "text",
            "entitlement": "text",
            "entitlement_name": "text",
            "app_id": "text",
            "app_name": "text",
            "auto_recert": "text",
            "event_recert": "text",
            "score": "decimal",
            "justification": {
                "type": "list",
                "typeDef": "<text>"
            }
        },
        "manager": "text",
        "manager_name": "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>"
            }
        }
    },
    "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",
        "frequion": "decimal",
        "median": "decimal",
        "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": {
  "thresholds": {
    "top": 1.01,
    "high": 0.75,
    "medium": 0.35,
    "low": 0,
    "autoAccess": 0.5
  },
  "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 Name",
"chief": "Chief",
"city": "City",
"costcenter": "Cost Center",
"department": "Department",

```

```

"is_active": "Active",
"jobcode": "Job Code Name",
"lob": "LOB",
"managername": "Manager",
"usr_display_name": "User Display Name",
"usr_emp_type": "Employee Type",
"usr_manager_key": "Manager"
},
"UIJustifications": {
  "USR_KEY": "User Key",
  "CITY": "City Location Building",
  "USR_DEPARTMENT_NAME": "User Department Name",
  "COST_CENTER": "User Cost Center",
  "JOB_CODE_NAME": "Job Code Name",
  "LINE_OF_BUSINESS": "LOB",
  "CHIEF_YES_NO": "Manager Flag",
  "USR_EMP_TYPE": "Employee Type",
  "USR_DISPLAY_NAME": "User name",
  "MANAGER_NAME": "Manager Name",
  "USR_MANAGER_KEY": "Manager Key",
  "IS_ACTIVE": "Active"
},
"HighRiskConf": {
  "filterValue": "1"
},
"JustificationDelimiter": {
  "justificationDelimiter": "_"
},
"SearchConf": {
  "name": "Search",
  "modelDefinition": {
    "fields": {
      "userdisplayname": "text",
      "user": "text",
      "isentitlementowner": "text",
      "issupervisor": "text"
    },
    "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": {
  "actions": [
    "CERTIFY_ENTITLEMENTS_TO_USERS",
    "CERTIFY_USERS_TO_ENTITLEMENTS",
    "FILTER_ENTITLEMENTS",
    "REVOKE_CERTIFY_ACCESS",
    "SEARCH_USER",
    "SEARCH_USER_ENTITLEMENTS",
    "SEARCH_SUPERVISOR_USER_ENTITLEMENTS",
    "SHOW_ASSIGNMENTS_STATS",
    "SHOW_COMPANY_PAGE",
    "SHOW_COMPANY_COVERAGE_DATA",
    "SHOW_COMPANY_ENTITLEMENTS_DATA",
    "SHOW_COMPANY_EMPLOYEE_PAGE",
    "SHOW_CRITICAL_ENTITLEMENTS",
    "SHOW_EMPLOYEE",
    "SHOW_ENTITLEMENT",
    "SHOW_ENTITLEMENT_AVG_GROUPS",
    "SHOW_ENTITLEMENT_AVG_GROUP_DETAILS",
    "SHOW_ENTITLEMENT_USERS",
    "SHOW_FILTER_OPTIONS",
    "SHOW_ROLE_OWNER_PAGE",
    "SHOW_ROLE_OWNER_USER_DATA",
    "SHOW_ROLE_OWNER_ENT_DATA",
    "SHOW_ROLE_OWNER_AUTO_DATA",
    "SHOW_SUPERVISOR_PAGE",
    "SHOW_SUPERVISOR_DETAILS_PAGE",
    "SHOW_SUPERVISOR_ENT_DATA",
    "SHOW_SUPERVISOR_USER_DATA",
    "SHOW_SUPERVISOR_ENTITLEMENT_USERS",
    "SHOW_SUPERVISOR_USER_ENTITLEMENTS",
    "SHOW_ROLEOWNER_UNSCORED_ENTITLEMENTS",
    "SHOW_SUPERVISOR_UNSCORED_ENTITLEMENTS",
    "SHOW_UNSCORED_ENTITLEMENTS",
    "SHOW_USER",
    "SHOW_ALL_ROLE_OWNER_DATA",
    "SHOW_CERTIFICATIONS"
  ],
  "permissions": {
    "Zoran Admin": {
      "can": "*"
    },
    "Zoran Entitlement Owner": {
      "can": [
        "FILTER_ENTITLEMENTS",
        "SEARCH_USER_ENTITLEMENTS",
        "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",
"CERTIFY_ENTITLEMENTS_TO_USERS",
"CERTIFY_USERS_TO_ENTITLEMENTS",
"REVOKE_CERTIFY_ACCESS"
]
},
"Zoran Executive": {
  "can": [
    "SEARCH_USER",
    "SHOW_ASSIGNMENTS_STATS",
    "SHOW_COMPANY_PAGE",
    "SHOW_COMPANY_COVERAGE_PAGE",
    "SHOW_COMPANY_ENTITLEMENTS_PAGE",
    "SHOW_COMPANY_ENTITLEMENTS_DATA",
    "SHOW_COMPANY_EMPLOYEE_PAGE",
    "SHOW_CRITICAL_ENTITLEMENTS",
    "SHOW_ENTITLEMENT_AVG_GROUPS",
    "SHOW_ENTITLEMENT_AVG_GROUP_DETAILS",
    "SHOW_USER_ENTITLEMENTS"
  ]
},
"Zoran Supervisor": {
  "can": [
    "SEARCH_USER",
    "FILTER_ENTITLEMENTS",
    "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": {
  "can": [
    "SHOW_ENTITLEMENT",
    "SHOW_USER",
    "SHOW_CERTIFICATIONS"
  ]
}
}
}
```

# 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.
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 <a href="#">assignment</a> . 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 <code>conf_thresh</code> 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.