Geospatial Content Server System Administration Guide

Release 2.5.0, December 2020
1 Overview

Geospatial Content Server (GCS) provides a comprehensive enterprise solution for hosting, processing, serving, and analyzing terrain, imagery, and other heterogeneous 3D data. This document details how to install GCS, including recommendations for deployment types based on your needs. It also contains maintenance and troubleshooting information. Please contact AGI support with any issues not addressed by this guide.

1.1 Architecture

The GCS system comprises multiple specialized server processes, hereafter referred to as components. There are two categories of GCS deployments: single-machine and distributed. A single-machine deployment is when all GCS components are installed on the same server (virtual or physical). When multiple servers are used to host the GCS components, that is considered a distributed deployment. A fully distributed deployment is when a separate server hosts each GCS component.

GCS System Components

Database server
- Stores data associated with stored assets in a PostgreSQL database
- Hosts binary terrain data as individual SQLite database files (with .terraindb extensions)

Identity server
- Responsible for providing user identities to the system
- Integrates with Active Directory/LDAP servers, or can store user accounts locally

Application server
- Hosts and serves the API and UI portions of the application
- Hosts and serves binary data for 3D models, imagery, and vector assets
- Serves binary data for terrain assets

Processing server
- Converts asset data to streaming-optimized formats, which are then hosted by the application server

Imagery server
Hosts Sentinel-2 Earth imagery (provided with GCS data disk)

Proxy server
Reverse proxy server through which users access the system

Single-Machine Deployment

The diagram below depicts a single-machine GCS deployment. The outer box represents the server hosting the GCS components. The inner boxes represent each GCS component. Even on a single-machine deployment, all communications between GCS components are secured via TLS.

A single-machine deployment is easier to set up than a distributed deployment but limits your ability to scale up as load on the system increases. It is recommended for proof-of-concept activities, but not for production use.
Fully Distributed Deployment

A fully distributed GCS deployment is illustrated in the diagram below.

In this type of installation, you must install each hosted component on a separate virtual or physical machine. You must also set up a shared file server, since the application, processing, and database servers share file locations (see Storage for more details). The NFS configuration section shows one way to set up this shared file server. Since all communications between GCS components are secured via TLS, each server requires a certificate to identify it to the other servers in the system.

This type of deployment is most appropriate for production use, since it is easier to scale up the individual servers as load increases.
2 Prerequisites

Our installation assumes the use of the following technologies. If you use other technologies, please contact AGI support.

- Red Hat Enterprise Linux (RHEL) or CentOS 7, 64-bit
- firewalld (usually installed by default)
- Python 2 or 3 (usually installed by default)

Each server must also have an SSL certificate generated by a trusted certificate authority. You are required to provide the certificate, private key, and certificate chain to the trusted root certificate authority when installing individual components (database, identity server, et al). In a single-machine scenario, the SSL certificate is the same for all components.

---

2.1 Hardware Requirements

The following section details the minimum hardware required to install and run the Geospatial Content Server on a single machine.

<table>
<thead>
<tr>
<th>Size</th>
<th>Number of Users</th>
<th>Hardware Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>25 concurrent or up to 250 intermittent users</td>
<td>16 GB RAM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2x CPU (at least 2.0Ghz)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>512 GB disk space</td>
</tr>
<tr>
<td>Medium</td>
<td>50 concurrent or up to 500 intermittent users</td>
<td>32 GB RAM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4x CPU (at least 2.0Ghz)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 TB disk space</td>
</tr>
<tr>
<td>Size</td>
<td>Number of Users</td>
<td>Hardware Requirements</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------------------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>Large</td>
<td>100 concurrent or up to 1000 intermittent users</td>
<td>32 GB RAM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8x CPU (at least 2.0Ghz)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 TB disk space</td>
</tr>
</tbody>
</table>

**Note:** The values above represent the minimum requirements. After processing, geospatial content can occupy up to four times as much space on the disk as the uploaded source files. If your organization processes large data files, you may require more disk space than is listed in the above requirements.

### 2.2 Extra Packages for Enterprise Linux (EPEL)

To ensure compatibility between our software and your operating system, we install some packages from the Fedora Project’s Extra Packages for Enterprise Linux (EPEL) during our installation process. EPEL is a collection of commonly used software libraries maintained by Red Hat and the Fedora community. Using EPEL ensures binary compatibility of third-party libraries, and enables you to patch vulnerabilities in these libraries immediately, without requiring a new version of GCS. For more information on EPEL, please see their website: https://fedoraproject.org/wiki/EPEL.

**Note:** After you have installed EPEL, you may have to disable verification of metadata. You can find this setting either in the global configuration file (/etc/yum.conf), or in a repository-specific config file (a *.repo file in /etc/yum.repos.d):

```sh
gpgcheck=0
```

Alternatively, if you have set up GPG signing keys for your EPEL instance, you can enable this property:

```sh
gpgcheck=1
```
3 Pre-Installation

Note: All scripts must be executed as a privileged user. Additionally, all of the installation scripts located at the installation root have a --help option. Running the scripts with this flag prints descriptions for all parameters the script accepts.

3.1 GCS License

Before installing GCS, you must obtain a license file from AGI. The installation scripts expect the license to be in a "licenses" folder in the installation root. Alternatively, you can specify an absolute path to a license using the --license-file flag or the LICENSE_FILE configuration file parameter when running "install-app-server.sh" and "install-processing-server.sh" during the Installation Steps.

If you do not yet have a license, you need to request one from AGI support. For a node-locked (local) license file, you will need to find the MAC address (host ID) of the machine on which you will install GCS. You can run print-host-id.sh from the "utils" directory of the installer to find this:

bash utils/print-host-id.sh

There are two types of license files available: node-locked license files and network license files.

3.1.1 Node-locked (local) licenses

A node-locked license allows GCS to run on one dedicated machine. This license type does not require additional servers. If you wish to add additional features or if the license has expired, you will need to replace this file. Note that a separate node-locked license is required for each machine on which GCS will be installed.
3.1.2 Networked licenses

Networked licenses are more flexible than traditional licenses. They can be maintained on a centralized license server, simplifying updates. This license type requires additional infrastructure to host the license server itself.

3.1.3 Deploying license files

If you are performing a new installation, follow the Installation Steps. If you are upgrading an existing installation, follow the Upgrade Steps. If you want to update your license file without installing or upgrading, copy the license files into `/usr/local/gcs-app/licenses` on the application server and into `/usr/local/gcs-processing/licenses` on the processing server.

3.2 Complying with STIG requirements

If your installation needs to comply with Security Technical Implementation Guide (STIG) requirements, you must provide the --harden flag to all server installation scripts or include `HARDEN='true'` in your configuration file. Visit the Defense Information Systems Agency website for more information.

3.3 Generating Internal GCS Certificates

Prior to installing, you need to create certificates by running "generate-certificates.sh". GCS uses these certificates for communication among its internal services. The script will create a "certificates" directory at the installation root containing these generated certificates.

`bash generate-certificates.sh`
Generated certificates are unique to your environment, contain sensitive information, and cannot be replaced if lost. The certificates folder should be backed up securely with limited read access.

**Note:** These certificates are only for GCS internal communication. You will still need machine-specific certificates for the server(s) on which you run the installer.

### 3.4 GCS User Roles and Permissions

The following table describes the different GCS user roles and permissions. The values you provide to the parameters `--gcs-processing-group` and `--gcs-admin-group` will be mapped to the `gcs_processing` and `gcs_admin` roles when installing the identity server.

<table>
<thead>
<tr>
<th>User Permissions</th>
<th>Roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>(no role assigned)</td>
<td>gcs_processing</td>
</tr>
<tr>
<td>User is able to view published assets</td>
<td>X</td>
</tr>
<tr>
<td>User is able to submit assets for processing</td>
<td>X</td>
</tr>
<tr>
<td>User is able to perform administrative tasks, such as viewing system logs</td>
<td></td>
</tr>
</tbody>
</table>

### 3.5 Sentinel-2 Imagery

Starting with GCS 2.4.0, the Imagery Server component of GCS is available to serve high-resolution global Earth imagery collected by the Sentinel-2 mission. Although it is not required, we highly recommend installing the Imagery
Server to take advantage of this imagery. If you choose to install the Imagery Server, follow the steps in Deploying Sentinel-2 Imagery Data. Then, follow the Imagery Server install instructions in the Installation Steps or Installation Steps (Distributed) sections.
4 Preparing Your Installation

There are two options for specifying values to the installation scripts:

- A configuration file (recommended). Using a configuration file to manage parameters for your installer has a few advantages over using command-line arguments, including:
  - Reduced risk of typos.
  - Reduced risk of inconsistent parameters passed to different scripts.
  - Easier to discover what parameters were provided to the install scripts after the fact.

- Command-line arguments.

Note: If an install script is provided both a configuration file and command-line arguments, a command-line argument will override its equivalent configuration file parameter. While this can be useful in certain circumstances, it is recommended that parameters be maintained in a configuration file when installing GCS to a production-like environment.

To prepare for the GCS installation, complete Option 1 or Option 2 to collect all required information.

4.1 Using a Configuration File

A configuration file contains parameter values that are used by the installer.

- Parameters are formatted as:

  NAME='value'

Note: Values must be quoted, since the configuration file is interpreted as a shell script.
By default, the installer looks for a configuration file called `gcs.cfg` in the `conf` directory under the install root. If you do not want to use the default file, use the `--config-file` parameter to point to another file.

To create a starter configuration file:

1. Run the script `create-config-file.sh` in the `utils` directory.
   ```bash
   bash utils/create-config-file.sh
   ``

2. Generate a configuration for single-machine deployment (`single`) or a distributed deployment (`distributed`). The default is `single` if no value is provided.
   ```bash
   --type=['single','distributed']
   ``

3. Define what to include in the configuration file.
   - Omit parameters used for hardening the installation to comply with STIG requirements. This may be helpful for test deployments, but should not be used for production deployments.
     ```bash
     --no-harden
     ``
   - Include all optional parameters.
     ```bash
     --include-optional
     ``

4. Fill in values for all parameters in `conf/gcs.cfg`. Parameters in this file are required for the installation type you selected.

5. Look at the other available parameters in the Installation Parameters appendix and add them to the file if needed.

Continue to the Installation section once your configuration file is complete.

### 4.2 Using Command-Line Arguments

Another way to pass parameters to the installers is to use command-line arguments. All parameters listed in the Installation Parameters appendix can be passed to the installer as command-line arguments by translating from upper case to lower case, replacing underscores with hyphens, and prefixing with two hyphens. For example:
<table>
<thead>
<tr>
<th>Configuration File Parameter / Value</th>
<th>Command-Line Argument / Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>APP_SERVER='app.example.com'</td>
<td>--app-server='app.example.com'</td>
</tr>
<tr>
<td>APP_SERVER_PORT='8445'</td>
<td>--app-server-port='8445'</td>
</tr>
</tbody>
</table>
5 Installation Steps

For a new installation, follow the steps outlined below, being sure to run the commands in the order shown. If you are upgrading an existing installation, please see Upgrade Steps.

Note: This section is for single machine installations. If your organization is installing for a distributed system, refer to the Appendix: Installation Steps (Distributed).

1. Install the Database Server.
   
   bash install-db-server.sh

2. Install the Identity Server.
   
   bash install-id-server.sh

3. Install the Application Server.
   
   bash install-app-server.sh

4. Install the Processing Server.

   Note: You must install the processing server even if you have a serving-only license.

   bash install-processing-server.sh

5. (Optional) Install the Imagery Server.

   bash install-imagery-server.sh

6. Install the Proxy Server.

   bash install-proxy-server.sh
6 Uninstallation Steps

Once GCS is installed, the uninstall scripts will be located at /usr/local/gcs/uninstall. Run the following commands from that location.

1. Uninstall the Proxy Server:
   
   ```bash
   bash uninstall-proxy-server.sh
   ```

2. Uninstall the Imagery Server (if the Imagery Server was installed):
   
   ```bash
   bash uninstall-imagery-server.sh
   ```

3. Uninstall the Processing Server:
   
   ```bash
   bash uninstall-processing-server.sh
   ```

   Optional flags:
   
   ```--keep-logs```

   Prevents processing server logs from being removed.

4. Uninstall the Application Server:
   
   ```bash
   bash uninstall-app-server.sh
   ```

   Optional flags:
   
   ```--keep-logs```

   Prevents application server logs from being removed.

   ```--remove-assets```

   By default, asset data is not removed. Passing this option forces the removal of assets.

5. Uninstall the Identity Server:
   
   ```bash
   bash uninstall-id-server.sh
   ```
6. Uninstall the Database Server:

   bash uninstall-db-server.sh

Optional flags:

   --keep-logs

Prevents database server logs from being removed.

   --remove-database

By default, the database is not removed. Passing this option forces the removal of the database.

   --db-backup-dir=PATH

If a custom backup location was specified during installation, this option must be specified to delete that backup location.
7 Upgrade Steps

When upgrading from an older version of GCS, you must upgrade existing GCS servers. Existing assets and log data will be retained.

Only run the scripts described below; do not execute the install-* .sh scripts.

Note:
- The system will be unavailable during the upgrade process.
- You must backup your systems prior to performing an upgrade.
- You must supply the certificates generated during the original installation.
- Be mindful to use the new installer files when upgrading. Re-running scripts from an old installer will not automatically find the new files.

7.1 Back Up Uninstall Scripts

Run the before-upgrade .sh script on each server to back up the uninstall scripts for the existing version.

bash before-upgrade.sh

7.2 Database Server Upgrade

1. Execute the upgrade script.

bash upgrade-db-server.sh
7.3 Identity Server Upgrade

1. Execute the upgrade script.
   
   ```bash
   bash upgrade-id-server.sh
   ```

7.4 Application Server Upgrade

   **Note:** You can update the license file during the upgrade by supplying the `--license-file` parameter. If you do not supply a value for this parameter, the existing license will be reused.

1. Execute the upgrade script.
   
   ```bash
   bash upgrade-app-server.sh
   ```

7.5 Processing Server Upgrade

   **Note:** You can update the license file during the upgrade by supplying the `--license-file` parameter. If you do not supply a value for this parameter, the existing license will be reused.

1. Execute the upgrade script.
   
   ```bash
   bash upgrade-processing-server.sh
   ```
7.6 Imagery Server Upgrade

GCS 2.5 ships with updated Sentinel-2 Earth imagery. Because the data file is very large (607 GB), the upgrade script does not deploy this new data. You must copy the new data file onto the imagery server machine before continuing with the upgrade. If you cannot upgrade the imagery data, contact AGI support.

1. Locate the data file, s2cloudless-2019_4326_v1.0.0.sqlite on the GCS disk. Relative to the root of the drive, it is in Servers/TiledImageryServer/Sentinel-2.

2. Copy the file s2cloudless-2019_4326_v1.0.0.sqlite into the directory /var/lib/gcs/mapserver/data of the imagery server.

3. Execute the upgrade script.

   bash upgrade-imagery-server.sh

---

7.7 Proxy Server Upgrade

1. Execute the upgrade script.

   bash upgrade-proxy-server.sh
8 Post-Installation Verification

8.1 Database Server

1. Check the status of the service.

    systemctl status postgresql-10

2. Check for the 'Active' state.

   * postgresql-10.service - PostgreSQL 10 database server
     Loaded: loaded (/etc/systemd/system/postgresql-10.service; enabled; vendor preset: disabled)
     Active: active (running) since Thu 2018-06-21 21:37:59 UTC; 3 days ago
     Docs: https://www.postgresql.org/docs/10/static/
     Main PID: 42446 (postmaster)
     CGroup: /system.slice/postgresql-10.service
           └─...
          └
            Jun 21 21:37:58 machine-name systemd[1]: Starting PostgreSQL 10 database server...
            Jun 21 21:37:58 machine-name postmaster[42446]: 2018-06-21 21:37:58.851 UTC [42446] LOG: listening on IPv4 address "0.0.0.0", port 5432
            ...
8.2 Identity Server

1. Check the status of the service.

   systemctl status keycloak

2. Check for the 'Active' state.

   - keycloak.service - WildFly Application Server for Keycloak
     Loaded: loaded (/etc/systemd/system/keycloak.service; enabled; vendor preset: disabled)
     Active: **active (running)** since Thu 2018-06-21 21:41:32 UTC; 16h ago
     Main PID: 46656 (standalone.sh)
     CGroup: /system.slice/keycloak.service
     └─46656 /bin/sh /usr/local/keycloak/bin/standalone.sh -Djboss.bind.address=0.0.0.0
     └─47443 /usr/java/keycloak/bin/java -D[Standalone] -server -Xmx512m...
   
   Jun 21 21:41:32 machine-name systemd[1]: Starting WildFly Application Server for Keycloak...
8.3 Application Server

1. Check the status of the service.

   ```
   systemctl status gcs-app
   ```

2. Check for the 'Active' state.

   ```
   • gcs-app.service - Apache Tomcat Java Servlet Container
     Loaded: loaded (/etc/systemd/system/gcs-app.service; enabled; vendor preset: disabled)
     Active: **active (running)** since Thu 2018-06-21 21:39:24 UTC; 17h ago
     Main PID: 43866 (java)
     CGroup: /system.slice/gcs-app.service
       ├─43866 /usr/java/default/bin/java -Djava.util.logging.config.file=/usr/local/gcs-app/conf/logging.properties
       │  ├─ ...  
       │  └─ Jun 21 21:39:24 machine-name systemd[1]: Starting Apache Tomcat Java Servlet Container...
   ```
8.4 Processing Server

1. Check the status of the service.

   ```bash
   systemctl status gcs-processing
   ```

2. Check for the 'Active' state.

   ```
   • gcs-processing.service - Apache Tomcat Java Servlet Container
   Loaded: loaded (/etc/systemd/system/gcs-processing.service; enabled; vendor preset: disabled)
   Active: **active (running)** since Thu 2018-06-21 21:39:24 UTC; 17h ago
   Main PID: 43866 (java)
   CGroup: /system.slice/gcs-processing.service
   └─43866 /usr/java/default/bin/java -Djava.util.logging.config.file=/usr/local/gcs-
   processing/conf/logging.properties
   ├─ ...  
   └─ Jun 21 21:39:24 machine-name systemd[1]: Starting Apache Tomcat Java Servlet Container...
   ```
8.5 Imagery Server

1. Check the status of the service.

   systemctl status gcs-imagery

2. Check for the 'Active' state.

   - gcs-imagery.service - The Apache HTTP Server
     Loaded: loaded (/etc/systemd/system/gcs-imagery.service; enabled; vendor preset: disabled)
     Active: active (running) since Wed 2020-05-27 15:06:22 UTC; 9min ago
     Docs: man:httpd(8)
     man:apachectl(8)
     Main PID: 4110 (httpd)
     Status: "Total requests: 0; Current requests/sec: 0; Current traffic: 0 B/sec"
     CGroup: /system.slice/gcs-imagery.service
     ⌒ ...
8.6 Proxy Server

1. Check the status of the service.

   systemctl status gcs-proxy

2. Check for the 'Active' state.

   - gcs-proxy.service - The Apache HTTP Server
     Loaded: loaded (/etc/systemd/system/gcs-proxy.service; enabled; vendor preset: disabled)
     Active: active (running) since Wed 2020-05-27 14:36:56 UTC; 2min 26s ago
     Docs: man:httpd(8)
     man:apachectl(8)
     Main PID: 9527 (httpd)
     Status: "Total requests: 0; Current requests/sec: 0; Current traffic: 0 B/sec"
     CGroup: /system.slice/gcs-proxy.service
           ├─...
           └─...

     May 27 14:36:56 machine-name systemd[1]: Starting The Apache HTTP Server...
     May 27 14:36:56 machine-name systemd[1]: Started The Apache HTTP Server.
9 Configuration

The components that make up GCS are controlled by a number of configuration files. GCS has pre-configured some of these files and the installation script will modify others. The following configuration values may be modified in your environment. Prior to modification, we recommend that you use the administrator metrics tab to establish baseline performance to compare with performance post-modification.

9.1 Application Server

9.1.1 REST API

<table>
<thead>
<tr>
<th>info-services.properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location:</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>/usr/local/gcs-app/webapps/api/WEB-INF/classes/META-INF/info</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>heights.maxLocations</th>
</tr>
</thead>
<tbody>
<tr>
<td>The maximum number of locations that can be sent to the heights REST API endpoints. This value must be in the range of 0 to Integer.MAX_VALUE.</td>
</tr>
</tbody>
</table>
10 Storage

The components that make up Geospatial Content Server both require their own storage locations and share files in several locations when transitioning assets from one component to another. As shown below, all of the components share access to `/var/lib/gcs`. On a single-machine installation, this might be on the local disk. In a distributed installation, the location would be mounted from a network share.
## 10.1 Server Locations

<table>
<thead>
<tr>
<th>Location</th>
<th>Description</th>
<th>Growth</th>
</tr>
</thead>
<tbody>
<tr>
<td>/usr/local/gcs-app</td>
<td>App Server Home</td>
<td>Static ~300MB</td>
</tr>
<tr>
<td>/var/log/gcs-app</td>
<td>App Server Logs</td>
<td>Depends on log settings and log retention.</td>
</tr>
<tr>
<td>/etc/httpd</td>
<td>Internal Proxy Configuration</td>
<td>Static ~100KB</td>
</tr>
<tr>
<td>/usr/local/gcs-processing</td>
<td>Processing Server Home</td>
<td>Static ~65MB</td>
</tr>
<tr>
<td>/var/log/httpd</td>
<td>Internal Proxy Logs</td>
<td>Depends on log settings and log retention.</td>
</tr>
<tr>
<td>/var/log/gcs-processing</td>
<td>Processing Server Logs</td>
<td>Depends on log settings and log retention.</td>
</tr>
<tr>
<td>/var/lib/postgresql</td>
<td>Application database and configuration</td>
<td>Increases with the number of users and assets.</td>
</tr>
<tr>
<td>/var/log/gcs-scheduled-jobs</td>
<td>Scheduled database job logs</td>
<td>Depends on log retention and number of assets.</td>
</tr>
<tr>
<td>/usr/local/keycloak</td>
<td>Identity Server and configuration</td>
<td>Static ~295MB</td>
</tr>
<tr>
<td>/usr/local/keycloak/standalone/log</td>
<td>Keycloak Server logs</td>
<td>Depends on log settings and log retention.</td>
</tr>
</tbody>
</table>
10.2 Shared Locations

Note: The storage required for shared locations is based on the files your users are uploading. It is best to err on the side of caution and provide ample storage in order to minimize disruption caused by a lack of disk space. A reference NFS configuration can be found at NFS Configuration.

<table>
<thead>
<tr>
<th>Location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/var/lib/gcs/upload</td>
<td>Upload destination</td>
</tr>
<tr>
<td>/var/lib/gcs/work</td>
<td>Uploaded files are staged here for processing.</td>
</tr>
<tr>
<td>/var/lib/gcs/web</td>
<td>Non-terrain assets are moved here after processing.</td>
</tr>
<tr>
<td>/var/lib/gcs/terrain-work</td>
<td>Uploaded terrain files are staged here for processing. After processing, terrain data is moved to the database server /var/lib/postgresql/terrain-data.</td>
</tr>
</tbody>
</table>
11 Migrating Data from STK Terrain Server

In order to migrate data from STK Terrain Server to GCS, you must have access to the Terrain Server's local file system. The existing terrain data is stored in the `db` folder under the existing STK-terrain install location.

For each `terraindb` file to be imported:

1. Copy the file onto your machine.

2. Click the "Add a new asset" button on the upper right of the user interface to create a new Terrain asset.

3. Enter a name for the new asset. The Cesium Terrain Database file type should be selected.

4. Upload the `terraindb` file. When the upload completes, click the Finalize Asset button to perform the import.

5. Click the Globe View button to make sure the terrain looks correct.

6. If you are satisfied with the results, click the Publish button to make the asset available to others.
12 Managing Roles

12.1 Accessing the AGI Identity Platform Administration Console

The AGI Identity Platform (AIP) is part of the identity server installation shown in the Architecture diagram in the Overview. Using the AGI Identity Platform Administration console, administrators can manage users and sessions and can configure the mapping to a user federation provider. Use of the console requires password authentication and a web browser that supports HTML5 and JavaScript.

1. Open a web browser and navigate to the following URL, replacing <authority> with your AIP host and port number separated by a colon (e.g., id.yourdomain.com:8443):
   
   https://<authority>/auth/admin

2. Enter the username and password for the AIP admin user.
   
   - Username: admin
   - Password: admin
Users, attributes, and group members are maintained in the directory server specified during installation. The AGI Identity Platform uses Keycloak to integrate with your directory and authenticate users within GCS. The identity server installer creates a one-way mapping between your directory and its user store. When a user is authenticated, the user's attributes and group membership are copied to the Keycloak user store.
12.2 Mapping Directory Groups to GCS Roles

GCS uses role mappings to define a user's access to the system.

<table>
<thead>
<tr>
<th>Viewer Users</th>
<th>Authenticated users without any specific role mapping are viewer users with read-only access to published assets.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processing Users</td>
<td>Processing users have the <code>gcs_processing</code> role applied either directly to their user account or indirectly through their group membership. Processing users can upload and process assets. After an asset has been processed, processing users can publish the asset to make it visible to other users. Processing users get the <code>gcs_view_users</code> role applied automatically.</td>
</tr>
<tr>
<td>Administrator Users</td>
<td>Administrator users have the <code>gcs_admin</code> role applied either directly to their user account or indirectly through their group membership. Administrator users can see published assets as well as an administrative tab with recent log data. Administrator users get the <code>gcs_view_users</code> role applied automatically.</td>
</tr>
</tbody>
</table>
12.3 Mapping a Role to an Existing Group in your Directory

1. Click the Groups link on the left menu.

2. Double-click the directory group name to be mapped to a role.
3. Select the desired role (either `gcs_admin` or `gcs_processing`) from the Available Roles list and click the Add selected button to map that role to this group. The role name moves to the Assigned Roles list. You can select both roles in order to grant the rights assigned to each role.

Note: The `gcs_view_users` role will appear in the Effective Roles section when you assign either the `gcs_admin` or the `gcs_processing` role to a group.
13 Troubleshooting

13.1 Log File Locations

<table>
<thead>
<tr>
<th>Component</th>
<th>Log File Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database Server</td>
<td>/var/log/postgres</td>
</tr>
<tr>
<td>Identity Server</td>
<td>/usr/local/keycloak/standalone/log</td>
</tr>
<tr>
<td>Application Server</td>
<td>/var/log/gcs-app</td>
</tr>
<tr>
<td>Processing Server</td>
<td>/var/log/gcs-processing</td>
</tr>
</tbody>
</table>

13.2 Common Issues

Symptom:  
*Cannot convert access token to JSON* (accompanied by a HTTP 401 response from the server).

Cause:  
The public key from Keycloak does not match the oauth-client.properties file in Tomcat.

Components Affected:  
- GCS Application Server
- GCS Processing Server
| Solution: | Copy the public key from Keycloak. In the Keycloak Administration console, select the "agigcs01" realm. Go to Realm Settings > Keys and look for the row with a "Type" of "RSA". In the "Public Keys" column, click "Public Key". On the application server, copy this value into each oauth-client.properties file under ~gcs-app/webapps (there may be several). Replace the lines between

```
-----BEGIN PUBLIC KEY----- 
and 
-----END PUBLIC KEY-----
```

**EXAMPLE:**

```java
jwt.public.key: -----BEGIN PUBLIC KEY-----\
MIIBojANBgkqhkiG9w0BAQEFAAOCAAY8AMIIBigKCAYEA2uw3ZDkqhoc9uVybmkGn\
o67b4dFAzHxcRbZBpcxjW+W8j1822/4ATFf4IvNBydU6efi7LK73v3KcEsTQWloK\
2cbrt+mupiKTuwoP5P1OhFR79SRedOCS0MMRVhFUGchvIy+F9Gy18K/1E+PMzNhC\
k52QNLpMwUxwBZgBK2Kr1kzb5JRknvnniFGBuipoQQBq5opjRqc8P6lwQS4K\
EP594DKbaZ49eu7kFib3nBuDWgDEQRKqqlIgPrkPxlF+alIYrquery0vxxLft84Z\
/hklzqVFL3SarKH+7UWqrpA8HzbCPOLdw4oFpkCA/pW3ePQnA2ofRbOSIFiU5\
fhlsvvlmZrFScvpxCZsrII3k8eQFiaTAMkp+TFpayZZEyg3ak0G/ISyZJYVhg\n/uKVosOEvNg+o3zN06tc3d3m7waXNYkCCXDD92TH3kA+7ULxkVASkE2dQnqEoMaOs\nMqSdQRnqHBxhWxWeKN5vyNc1rWT2TEOwJxIKP3WV4+rAgMBAAE=\

-----END PUBLIC KEY-----
```
Repeat this process on the processing server, using `~gcs-processing/webapps` instead of `~gcs-app/webapps`.

<table>
<thead>
<tr>
<th>Symptom:</th>
<th>The GCS Application Server or the GCS Processing Server does not start. <em>Error calling refresh</em> appears in the catalina.out log file.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cause:</td>
<td>The license file is invalid. This may be due to:</td>
</tr>
<tr>
<td></td>
<td>• An expired license</td>
</tr>
<tr>
<td></td>
<td>• Windows line endings on a Linux system</td>
</tr>
<tr>
<td></td>
<td>• No recognized GCS features</td>
</tr>
<tr>
<td></td>
<td>• An invalid version</td>
</tr>
<tr>
<td>Components Affected:</td>
<td>• GCS Application Server</td>
</tr>
<tr>
<td></td>
<td>• GCS Processing Server</td>
</tr>
<tr>
<td>Solution:</td>
<td>• Run <code>dos2unix</code> on the license file located under gcs-app/licenses and gcs-processing/licenses and restart the services with:</td>
</tr>
<tr>
<td></td>
<td><code>systemctl restart gcs-app</code></td>
</tr>
<tr>
<td></td>
<td>or</td>
</tr>
<tr>
<td></td>
<td><code>systemctl restart gcs-processing</code></td>
</tr>
<tr>
<td></td>
<td>• Manually inspect the contents of the license files, verifying that they contain valid features, are not expired, and have a major version matching the installed version (2.0)</td>
</tr>
<tr>
<td></td>
<td>• Request a new license file from <a href="mailto:support@agi.com">support@agi.com</a>, if needed</td>
</tr>
</tbody>
</table>

In the event that you encounter a different symptom, or the provided solutions do not fix your problem, contact [AGI support](mailto:AGI support).
**A Installation Steps (Distributed)**

*Note:* You must ensure that the GCS group ID is synchronized between the Application, Processing, and Database Servers. See the appendix *Synchronizing Users and Groups*.

For a new installation, follow the steps outlined below, being sure to run the commands in the order shown. If you are upgrading an existing installation, please see *Upgrade Steps*.

The configuration file, licenses, and certificates generated in **pre-installation** must be copied to each of the servers in your distributed installation. The servers use these certificates to communicate securely amongst themselves. In addition, you must ensure that the `/var/lib/gcs` share has been established between the servers prior to executing the install scripts. Please see *Storage* for more information.

*Note:* All file locations should be specified as absolute paths.

1. **Install the Database Server.**
   
   ```bash
   bash install-db-server.sh
   ```

2. **Install the Identity Server.**
   
   ```bash
   bash install-id-server.sh
   ```

3. **Install the Application Server.**
   
   ```bash
   bash install-app-server.sh
   ```

4. **Install the Processing Server. You must install the processing server even if you have a serving-only license.**
   
   ```bash
   bash install-processing-server.sh
   ```

5. **(Optional) Install the Imagery Server.**
   
   ```bash
   bash install-imagery-server.sh
   ```

6. **Install the Proxy Server.**
   
   ```bash
   bash install-proxy-server.sh
   ```
B Integrated Install

To integrate GCS with an existing, shared AGI Identity Platform (AIP), follow these steps to install GCS:

1. Copy a sample configuration file (such as conf/samples/all.cfg) to conf/gcs.cfg and fill in your parameters. In the Identity Server section, fill in the parameters that apply to your existing AIP instance.

2. Install the Database Server as shown in section 5.1.

3. Install the Application Server as shown in section 5.3.

4. Install the Processing Server as shown in section 5.4.

5. (Optional) Install the Imagery Server as shown in section 5.5.

6. Install the Proxy Server as shown in section 5.6.

7. In the utils directory of the installer, find the create-keycloak-realm.sh script. This will export a Keycloak realm configuration to utils/export/gcs-realm.json. (The GCS realm configuration allows your existing AIP to provide user identity information to GCS.)

8. Download the gcs-realm.json file to your local computer.

An AIP/Keycloak administrator must complete the next series of steps.

1. In the Keycloak admin console (web interface), find the realm drop-down in the upper left of the screen. Hover on this to reveal the drop-down menu. Click Add realm.

2. Next to the Import label on the top line, click Select file. Choose the gcs-realm.json file and click Open.

3. Verify that the Name is agigcs01 and Enabled is selected. Click Create.

4. Verify that the new realm, Agigcs01, is displayed on the screen.

You can now log in and use your GCS system.
C Synchronizing Users and Groups in a Distributed Installation

In a distributed environment, the ID of the shared GCS group must be synchronized to allow access to shared files.

1. After installing the database server, use the following command to get the group ID of the gcs group.

   ```bash
   getent group gcs
   ```

   The response should look like the following. In this example, the number 1002 is the group ID. Your group ID should be used in place of GID in the commands below.

   ```bash
   gcs:x:1002:
   ```

2. On the machine that will be the application server, run the following command BEFORE installation, replacing GID with the group ID from above.

   ```bash
   groupadd -g GID gcs
   ```

3. Run the same command on the processing server to create the GCS group (again, BEFORE installation).

   ```bash
   groupadd -g GID gcs
   ```
D  NFS Configuration

Instructions for a reference NFS configuration are listed below.

D.1  Server

1. Configure the host to allow NFS traffic

   firewall-cmd --permanent --zone=public --add-service=nfs
   firewall-cmd --reload

2. Install NFS.

   yum -y install nfs-utils
   systemctl enable nfs-server.service
   systemctl start nfs-server.service

3. Create the shared location.

   mkdir -p /export/gcs

4. Create a /etc/exports file. Replace APP_SERVER, PROC_SERVER, and DB_SERVER with the machine names in your environment.

   /export/gcs APP_SERVER(rw, sync, no_root_squash, no_all_squash) DB_SERVER(rw, sync, no_root_squash, no_all_squash) PROC_SERVER(rw, sync, no_root_squash, no_all_squash)
D.2 Clients

1. Install autofs and nfs-common.

```
yum -y install nfs-common autofs
```

2. Add a `/etc/auto.master.d/gcs` file, replacing NFS_SERVER with the NFS host in your environment.

```
/var/lib/gcs  -fstype=nfs4,rw,nosuid,noexec  NFS_SERVER:/export/gcs
```

3. Add the following line to a file named `/etc/auto.master.d/gcs.autofs`.

```
/- /etc/auto.master.d/gcs --timeout=600
```
E Deploying Sentinel-2 Imagery Data

The Sentinel-2 data is available on the Geospatial Content data disk at Servers/TiledImageryServer/Sentinel-2/s2cloudless-2019_4326_v1.0.0.sqlite. Before installing the imagery server, copy this file into the directory /var/lib/gcs/mapserver/data. If you need to use a different path to this file (e.g., if the storage on which it resides must be mounted on a different path), use the IMAGERY_FILE configuration file parameter when installing the imagery server.
## F Installation Parameters

The following table provides command-line arguments and descriptions for the configuration parameters used by the GCS install.

<table>
<thead>
<tr>
<th>Configuration File Parameter</th>
<th>Command-Line Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSL_CA_FILE</td>
<td>--ssl-ca-file</td>
<td>Full path to a PEM-formatted certificate authority file. This is the certificate authority that signed each SSL certificate for each GCS component.</td>
</tr>
<tr>
<td>LICENSE_FILE</td>
<td>--license-file</td>
<td>AGI-provided license file.</td>
</tr>
<tr>
<td>IP_NETWORK_MASKS</td>
<td>--ip-network-masks</td>
<td>One or more IPv4 address ranges formatted in CIDR notation to control which clients may attempt to connect to the database server.</td>
</tr>
<tr>
<td>APP_SERVER_SSL_CERT_FILE</td>
<td>--app-server-ssl-cert-file</td>
<td>Full path to a PEM-formatted certificate used to identify the GCS app server.</td>
</tr>
<tr>
<td>APP_SERVER_SSL_KEY_FILE</td>
<td>--app-server-ssl-key-file</td>
<td>Full path to the private key that pairs with the public key of APP_SERVER_SSL_CERT_FILE.</td>
</tr>
<tr>
<td>DB_SERVER_SSL_CERT_FILE</td>
<td>--db-server-ssl-cert-file</td>
<td>Full path to a PEM-formatted certificate used to identify the GCS database server.</td>
</tr>
<tr>
<td>Configuration File Parameter</td>
<td>Command-Line Argument</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DB_SERVER_SSL_KEY_FILE</td>
<td>--db-server-ssl-key-file</td>
<td>Full path to the private key that pairs with the public key of DB_SERVER_SSL_CERT_FILE.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ID_SERVER_SSL_CERT_FILE</td>
<td>--id-server-ssl-cert-file</td>
<td>Full path to a PEM-formatted certificate used to identify the GCS identity server.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ID_SERVER_SSL_KEY_FILE</td>
<td>--id-server-ssl-key-file</td>
<td>Full path to the private key that pairs with the public key of ID_SERVER_SSL_KEY_FILE.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMAGERY_SERVER_SSL_CERT_FILE</td>
<td>--imagery-server-ssl-cert-file</td>
<td>Full path to a PEM-formatted certificate used to identify the GCS imagery server.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMAGERY_SERVER_SSL_KEY_FILE</td>
<td>--imagery-server-ssl-key-file</td>
<td>Full path to the private key that pairs with the public key of IMAGERY_SERVER_SSL_CERT_FILE.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROCESSING_SERVER_SSL_CERT_FILE</td>
<td>--processing-server-ssl-cert-file</td>
<td>Full path to a PEM-formatted certificate that identifies the GCS processing server.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROCESSING_SERVER_SSL_KEY_FILE</td>
<td>--processing-server-ssl-key-file</td>
<td>Full path to the private key that pairs with the public key of PROCESSING_SERVER_SSL_CERT_FILE.</td>
</tr>
<tr>
<td>Configuration File Parameter</td>
<td>Command-Line Argument</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PROXY_SERVER_SSL_CERT_FILE</td>
<td>--proxy-server-ssl-cert-file</td>
<td>Full path to a PEM-formatted certificate used to identify the GCS proxy server.</td>
</tr>
<tr>
<td>PROXY_SERVER_SSL_KEY_FILE</td>
<td>--proxy-server-ssl-key-file</td>
<td>Full path to the private key that pairs with the public key of PROXY_SERVER_SSL_CERT_FILE.</td>
</tr>
<tr>
<td>APP_SERVER</td>
<td>--app-server</td>
<td>The fully-qualified domain name of the application server.</td>
</tr>
<tr>
<td>APP_SERVER_PORT</td>
<td>--app-server-port</td>
<td>The HTTPS port for the application server.</td>
</tr>
<tr>
<td>APP_SERVER_CLIENT_AUTH_PORT</td>
<td>--app-server-client-auth-port</td>
<td>The HTTPS port used to accept processing server requests containing client certificates.</td>
</tr>
<tr>
<td>DB_SERVER</td>
<td>--db-server</td>
<td>The fully-qualified domain name of the database server.</td>
</tr>
<tr>
<td>DB_SERVER_PORT</td>
<td>--db-server-port</td>
<td>The port number for the database server.</td>
</tr>
<tr>
<td>ID_SERVER</td>
<td>--id-server</td>
<td>The fully-qualified domain name of the identity server.</td>
</tr>
<tr>
<td>ID_SERVER_PORT</td>
<td>--id-server-port</td>
<td>The HTTPS port for the identity server.</td>
</tr>
<tr>
<td>Configuration File Parameter</td>
<td>Command-Line Argument</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IMAGERY_SERVER</td>
<td>--imagery-server</td>
<td>The fully-qualified domain name of the imagery server.</td>
</tr>
<tr>
<td>IMAGERY_SERVER_PORT</td>
<td>--imagery-server-port</td>
<td>The HTTPS port for the imagery server.</td>
</tr>
<tr>
<td>PROCESSING_SERVER</td>
<td>--processing-server</td>
<td>The fully-qualified domain name of the processing server.</td>
</tr>
<tr>
<td>PROCESSING_SERVER_PORT</td>
<td>--processing-server-port</td>
<td>The HTTPS port for the processing server.</td>
</tr>
<tr>
<td>PROXY_SERVER</td>
<td>--proxy-server</td>
<td>The fully-qualified domain name of the reverse proxy server. This is the domain name that users enter to access the application.</td>
</tr>
<tr>
<td>PROXY_SERVER_PORT</td>
<td>--proxy-server-port</td>
<td>The HTTPS port for the reverse proxy server.</td>
</tr>
<tr>
<td>FIREWALL_ZONE</td>
<td>--firewall-zone</td>
<td>The firewall zone in which rich rules are applied.</td>
</tr>
<tr>
<td>Configuration File Parameter</td>
<td>Command-Line Argument</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SSL_PROTOCOLS</td>
<td>--ssl-protocols</td>
<td>The names of the protocols to support when communicating with clients. Supported protocol strings are defined by Apache (<a href="https://tomcat.apache.org/tomcat-9.0-doc/config/http.html">https://tomcat.apache.org/tomcat-9.0-doc/config/http.html</a>). Multiple protocols may be selected with a comma-separated list (e.g., 'TLSv1.2, TLSv1.3').</td>
</tr>
<tr>
<td>COMPONENTS_LICENSE</td>
<td>--components-license</td>
<td>The full path to the AGI component license file provided by AGI for the application. If not specified, the license file must be in the licenses folder of the installation root.</td>
</tr>
<tr>
<td>LOCALE_LANG</td>
<td>--locale-lang</td>
<td>The language to be used in the application.</td>
</tr>
<tr>
<td>TOMCATInicial_MEMORY</td>
<td>--tomcat-initial-memory</td>
<td>The initial memory allocated to Tomcat (e.g., 2G for 2GB of initial memory).</td>
</tr>
<tr>
<td>TOMCAT_MAX_MEMORY</td>
<td>--tomcat-max-memory</td>
<td>The max memory allocated to Tomcat (e.g., 4G for 4GB of max memory).</td>
</tr>
<tr>
<td>Configuration File Parameter</td>
<td>Command-Line Argument</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------</td>
<td>---------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TOMCAT_LOG_LEVEL</td>
<td>--tomcat-log-level</td>
<td>Specify Tomcat log level. The log levels are SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST, or ALL.</td>
</tr>
<tr>
<td>WEB_MAX_THREADS</td>
<td>--web-max-threads</td>
<td>The maximum number of request processing threads to be created by the Tomcat server.</td>
</tr>
<tr>
<td>DB_BACKUP_DIR</td>
<td>--db-backup-dir</td>
<td>Path to a directory where all database backups will be stored.</td>
</tr>
<tr>
<td>DB_BACKUP_CRON_EXPRESSION</td>
<td>--db-backup-cron-expression</td>
<td>A cron expression to schedule when backup of database cluster will occur. The default schedule runs every night at midnight.</td>
</tr>
<tr>
<td>DB_CONNECTION_LIMIT</td>
<td>--db-connection-limit</td>
<td>The maximum number of connections to the databases being installed. The default is -1, indicating no limit on the number of connections.</td>
</tr>
<tr>
<td>Configuration File Parameter</td>
<td>Command-Line Argument</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-----------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DB_MACHINE_SIZE</td>
<td>--db-machine-size</td>
<td>The size of the machine that the install is running on. This will be used to tune the database settings for optimal performance. The available options are M1, L1, S3, M3 and L3. M1 specifies a single machine deployment on a machine with approximately 8GB of RAM. L1 specifies a single machine deployment on a machine with approximately 16GB of RAM. S3 specifies a multi-machine deployment, where the database is on a machine with approximately 4GB of RAM. M3 and L3 specify multi-machine deployments, with the same amount of RAM specified in M1 and L1, respectively.</td>
</tr>
<tr>
<td>GCS_ADMINGROUP</td>
<td>--gcs-admin-group</td>
<td>Common name (CN) of an LDAP group that has admin access to GCS.</td>
</tr>
<tr>
<td>GCS_PROCESSINGGROUP</td>
<td>--gcs-processing-group</td>
<td>Common name (CN) of an LDAP group that has processing access to GCS.</td>
</tr>
<tr>
<td>Configuration File Parameter</td>
<td>Command-Line Argument</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ID_SERVER_BIND_ADDRESS</td>
<td>--id-server-bind-address</td>
<td>The IP address from which the identity server will accept connections.</td>
</tr>
<tr>
<td>KEYCLOAK_ADMIN_IP_ADDRESSES</td>
<td>--keycloak-admin-ip-addresses</td>
<td>An IPv4 address range formatted in CIDR notation to determine which client hosts may access the identity server admin console. Multiple ranges can be provided by separating each range with a comma (,). Required if HARDEN='true'.</td>
</tr>
<tr>
<td>KEYCLOAK_ADMIN_PASS</td>
<td>--keycloak-admin-pass</td>
<td>Password for the Keycloak administrative user.</td>
</tr>
<tr>
<td>KEYCLOAK_ADMIN_USER</td>
<td>--keycloak-admin-user</td>
<td>Username for the Keycloak administrative user.</td>
</tr>
<tr>
<td>KEYCLOAK_SESSION_TIMEOUT_SECONDS</td>
<td>--keycloak-session-timeout-seconds</td>
<td>Number of seconds until user sessions timeout due to inactivity.</td>
</tr>
<tr>
<td>LDAP_BIND_CREDENTIALS</td>
<td>--ldap-bind-credentials</td>
<td>The password for the LDAP server. Required when LDAP_CONNECT_URL is specified.</td>
</tr>
<tr>
<td>Configuration File Parameter</td>
<td>Command-Line Argument</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>LDAP_BIND_DN</td>
<td>--ldap-bind-dn</td>
<td>Distinguished name (DN) of an account used to bind to the LDAP server. Required when LDAP_CONNECT_URL is specified.</td>
</tr>
<tr>
<td>LDAP_CONNECT_URL</td>
<td>--ldap-connect-url</td>
<td>The hostname of the LDAP server.</td>
</tr>
<tr>
<td>LDAP_GROUPS_DN</td>
<td>--ldap-groups-dn</td>
<td>Distinguished name (DN) of a directory that holds user groups. Keycloak will look for named groups (like GCS_ADMIN_GROUP and GCS_PROCESSING_GROUP) in this location. Required when LDAP_CONNECT_URL is specified.</td>
</tr>
<tr>
<td>LDAP_USERS_DN</td>
<td>--ldap-users-dn</td>
<td>Distinguished name (DN) of a directory that holds user accounts. Required when LDAP_CONNECT_URL is specified.</td>
</tr>
<tr>
<td>MAX_ACTIVE_DB_CONNECTS</td>
<td>--max-active-db-connects</td>
<td>The maximum number of simultaneous active database connections.</td>
</tr>
<tr>
<td>IMAGERY_FILE</td>
<td>--imagery-file</td>
<td>The path to the imagery file to use.</td>
</tr>
<tr>
<td>Configuration File Parameter</td>
<td>Command-Line Argument</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-----------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>IMAGERY_SERVER_BIND_ADDRESS</td>
<td>--imagery-server-bind-address</td>
<td>The IP address from which the imagery server will accept connections.</td>
</tr>
<tr>
<td>MAPCACHE_CONF_FILE</td>
<td>--mapcache-conf-file</td>
<td>The path to a mapcache.xml configuration file to use.</td>
</tr>
<tr>
<td>PROXY_SERVER_BIND_ADDRESS</td>
<td>--proxy-server-bind-address</td>
<td>The IP address from which the proxy server will accept connections.</td>
</tr>
</tbody>
</table>

Some command-line arguments are used as flags to indicate a Boolean "true" value. When these appear in configuration files, they should be given the value 'true' (as a string). The quotes surrounding 'true' are required. All available flags are listed below.

<table>
<thead>
<tr>
<th>Configuration File Parameter / Value</th>
<th>Command-Line Argument / Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HARDEN='true'</td>
<td>--harden</td>
</tr>
<tr>
<td>NO_IMAGERY_SERVER='true'</td>
<td>--no-imagery-server</td>
</tr>
<tr>
<td>INSTALL_APP_DB='true'</td>
<td>--install-app-db</td>
</tr>
<tr>
<td>INSTALL_ID_DB='true'</td>
<td>--install-id-db</td>
</tr>
</tbody>
</table>