Oracle® Fusion Middleware

Administering Oracle Radius Agent

F38497-07
April 2022

Oracle Fusion Middleware Administering Oracle Radius Agent,

F38497-07

Primary Author: Oracle Corporation

Contents

Preface

Administering Oracle Radius Agent describes how to install Oracle Radius Agent, configure, and integrate with Oracle Advanced Authentication and Oracle Access Management to provide multi factor authentication capabilities.

Audience

This guide is intended for:

- Administrators responsible for installing and configuring Oracle Radius Agent.
- Administrators responsible for integrating Oracle Access Management and Oracle Advanced Authentication with Oracle Radius Agent for multi-factor authentication.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Documents

For more information, refer to the following documents:

- Online Help
- Administering Oracle Access Management
- REST API documentation
- Administering Oracle Advanced Authentication

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

About Oracle RADIUS Agent

Topics

1.1 Introduction to Oracle RADIUS Agent

Oracle RADIUS Agent is an application layer user authentication service that uses the industry standard RADIUS (Remote Authentication Dial-In User Service) protocol. It also facilitates authorization by fetching groups and user attributes from primary authenticator such as LDAP.

Oracle RADIUS Agent is a RADIUS based authentication service that acts as an intermediary between the client application requiring authentication services and one or more authentication providers. Example application clients are VPN server, Linux server using SSH, Oracle Database, or any RADIUS based client application.

Oracle RADIUS Agent supports both single-factor authentication and multi-factor authentication. You can configure the authentication provider for primary authentication requests. Currently, standard LDAP authentication providers such as Oracle Unified Directory, Oracle Internet Directory, and Microsoft Active Directory are supported. Oracle RADIUS Agent supports multi-factor authentication used in conjunction with Oracle Advanced Authentication. It uses the challenge/response mechanism of the RADIUS protocol for handling multi-factor requests and responses. It also supports user selection of multi-factor authentication mechanisms when more than one factor is available.

Oracle RADIUS Agent makes it easier to support multiple client applications talking to a single Oracle RADIUS Agent instance, with their own specific listener ports and configurations.

The deployment for Oracle RADIUS Agent is based on a container image. The Oracle RADIUS Agent instance is stateless and can be deployed easily for high availability and scalability by using multiple container instances.

1.2 Oracle RADIUS Agent Features

Oracle RADIUS Agent has support for a wide variety of features designed to make it easy to deploy, use, and manage.

- Password-based Primary Authentication Against LDAP

  Oracle RADIUS Agent supports any LDAPv3-compliant servers such as Oracle Unified Directory, Oracle Internet Directory, and Microsoft Active Directory.
- Multi-factor authentication by using Oracle Advanced Authentication, which provides second factor authentication for a multi-factor authentication deployment.

- RADIUS Agent exposes several preferences that can be set based on the deployment requirements. For example, fetching of groups, mappings for groups and users, settings for synchronous mode, and factor preferences for Oracle Advanced Authentication can be configured in preferences. For more information, see Configuration Properties.
- Supports the following second factors:
  - Time-based One-Time Password (TOTP)
  - One-Time Password (OTP) (both SMS and email)
  - Yubikey OTP
- Support for User Selection of Second-Factor Mechanism

  When multiple second-factor authentication mechanisms exist, the user may choose the mechanism to be used at the time of authentication.
- Support for Asynchronous and Synchronous Multi-factor Authentication Modes

  Oracle RADIUS Agent supports both asynchronous and synchronous authentication modes with multi-factor authentication over RADIUS.

  Asynchronous mode is the challenge/response mode where the user is prompted for a second factor based on the preferences configured in Oracle Advanced Authentication. When a user has multiple authentication factors configured, RADIUS interactions are made interactive by showing the user a list of available factors to choose from. The user also has the capability to choose a preferred authentication factor to execute without going through the interactive mode.

  When using legacy applications, which may not have the necessary interfaces for multi-factor authentication, multi-factor authentication may still be executed by adding on the second factor to the primary factor as a combined authentication password and this is feasible for any pre-generated token and takes the format <password>;<second_factor>. This is synchronous mode. The ; delimiter shown here is configurable in Oracle RADIUS Agent.
- Support for User Authorization

  Oracle RADIUS Agent supports retrieving groups and user attributes based on configurable mappings. These can be used by RADIUS clients to perform authorizations based on their needs.
- Support for Multiple RADIUS Client Applications

  You can define global and application-scoped configurations. The application-scoped configurations support overriding specific settings from the global configuration on an application basis.
- IPv6 Support

  Oracle RADIUS Agent supports IPv6 for all networking connections.
- Support for Logging, Metrics, and Auditing

  Oracle RADIUS Agent makes use of java.util.logging. The log files location is configurable and you can configure it during the initial container setup. Oracle RADIUS Agent also supports custom logging.properties. In addition, you can configure log levels dynamically by calling the Oracle RADIUS Agent configuration REST API.

  Oracle RADIUS Agent generates metrics using the MicroProfile Metrics specification. As part of application-specific metrics data, Oracle RADIUS Agent generates metrics related to primary authentication, multi-factor authentication, and listener configuration. The Helidon framework also generates JVM and Helidon specific metrics.

Oracle RADIUS Agent supports file-based audit logs. The name of the logger is `oracle.idm.radius.audit.log.level` and is enabled out-of-the-box. All audit logs are logged into the ora-`audit%g.log` file located in the `logs` directory. You can disable it by changing the log level of this logger to `ERROR`.

- Centralized file-based configuration storage on persistent container volumes.
- Stateless instances

  The Oracle RADIUS Agent instance is stateless and can be easily destroyed and respawned. In addition, it is easy to spawn new instances by pointing to an existing configuration in a shared container volume.
- Load Balancing

  For high availability a number of Oracle RADIUS Agent can be deployed behind a load balancer. The load balancer must be configured for session stickiness.
- Supports coexistence of multiple Oracle RADIUS Agents of different versions sharing the same configuration.

Note:

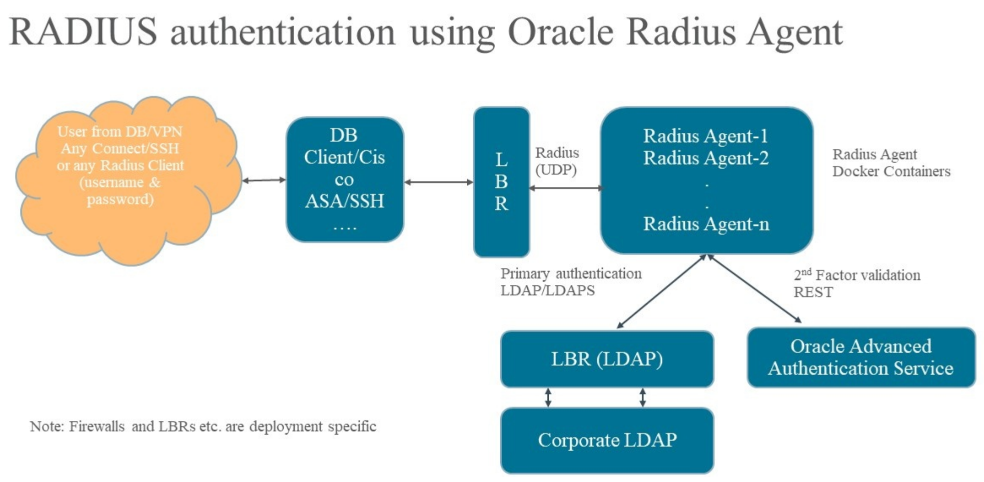
Oracle RADIUS Agent does not support RADIUS Accounting (RFC 2866) in this release.

1.3 Oracle RADIUS Agent Architecture and Deployment Model

A typical infrastructure utilizing Oracle RADIUS Agent consists of one or more application clients, the Oracle RADIUS Agent itself, Oracle Advanced Authentication, one or more back-end authentication providers and persistent storage volumes for use by the Oracle RADIUS Agent.

The Oracle RADIUS Agent architecture enables a flexible and scalable interface to these infrastructure components in order to provide a wide feature set along with high availability and scalability.

The system architecture for Oracle RADIUS Agent is as follows:


RADIUS authentication using Oracle Radius Agent
User from DB/VPN Any Connect/SSH or any Radius Client (username & password)
DB Client/Cisco ASA/SSH ….
L B R
Radius (UDP)
Radius Agent-1
Radius Agent-2
Radius Agent-n
Radius Agent Docker Containers
Primary authentication LDAP/LDAPS
2nd Factor validation REST
LBR (LDAP)
Oracle Advanced Authentication Service
Corporate LDAP
Note: Firewalls and LBRs etc. are deployment specific

The application client communicates with Oracle RADIUS Agent over UDP using the RADIUS protocol. Oracle RADIUS Agent communicates with the primary authentication provider using LDAP.

The following sections provide more information about the component interfaces:

Application Client Connections

The application client connects to the RadiusListener port of Oracle RADIUS Agent and communicates over UDP. The RadiusListener port is randomly allocated by Oracle RADIUS Agent between `1812` and `1830`. This port number needs to be mapped to a local port and exposed based on deployment needs in the container.

Primary Authenticator Interface

Currently, the only supported primary authenticator protocol is LDAP. The connection between Oracle RADIUS Agent and the primary authenticator is over TCP/IP using LDAP.

Multi-factor Authenticator Interfaces

Multi-factor authentication is supported using Oracle Advanced Authentication. The connection between the Oracle RADIUS Agent and Oracle Advanced Authentication uses REST APIs over TCP/IP.

Oracle RADIUS Agent Storage

The Oracle RADIUS Agent container instance uses persistent external storage volumes mounted into the container for storing configuration data, shared secrets from client registrations, and logs. The storage volume may be shared among multiple container instances of the RADIUS agent. Instance data such as logs, is stored under an instance specific directory. You must backup the data store located on the external storage volume for the Oracle RADIUS Agent.

Note:

Instance specific data like logs and auto-generated keystores for HTTPS are stored under an instance directory.

2 Installing and Configuring Oracle RADIUS Agent

Topics

2.1 Prerequisites for Installing Oracle RADIUS Agent

These are the tasks that you must complete before you install Oracle RADIUS Agent.

- The latest container engine is installed and the container service is started. Ensure that an external container volume is mounted in the container for persistent storage of configuration, data, and logs.
- Create a directory on the file system to store the Oracle RADIUS Agent configuration and logs data. The `uid:gid` of the directory must match the user who will start the container. Starting April 2022 release, Oracle RADIUS Agent uses a default `uid:gid` of `14304:14304` for the oracle user running in the container that performs all container operations. The read and write permissions on the directory must be set to `rwxr-x---`. This directory will be mapped to the container volume when you start the container. If you want to use a different user in your deployment, use an override `uid:gid` value at the runtime and set the directory permission to `rwxr-x--`. The following is a sample command to override the `uid:gid`.

  ```
  docker run -d --network rad --name radius1 --user 10001:5001  -v /scratch/
  radius:/u01/oracle/user_projects -p 8000:8080 -p 1812:1812/udp -p
  1813:1813/udp -p 1814:1814 -e INSTANCE_NAME=inst1 container-
  registry.oracle.com/middleware/radius:latest
  ```

- This directory must be backed up for disaster recovery.

- A running instance of the LDAP server against which you want Oracle RADIUS Agent to authenticate and authorize access.
- The LDAP server should contain a user (or users) that acts as a RADIUS Administrator. The administrator user (or users) need to have the necessary ACL configured on the LDAP server to be able to search users and groups.
- If Oracle RADIUS Agent connects to the LDAP server over SSL, then ensure that you have the trusted CA certificate that signed the LDAP server certificate, in PEM format.
- If you intend to use Oracle RADIUS Agent with Oracle Advanced Authentication for multi-factor authentication, then ensure you have a running instance of Oracle Advanced Authentication. See Use Oracle Radius Agent with LDAP as the Primary Authenticator Oracle Advanced Authentication Administrating guide for information about installing Oracle Advanced Authentication.

2.2 System Requirements and Certification

These are system requirements and certification for installing Oracle RADIUS Agent.

Ensure that your environment meets the system requirements such as hardware and software, minimum disk space, memory, required system libraries, packages, or patches before performing any installation.

The minimum system requirements for installing Oracle RADIUS Agent (ORA) is:

- For installing ORA on a standalone host:
  - 4 GB of RAM
  - Disk Space of 10 GB
  - 2 CPU
- For installing ORA on a server:
  - 8 GB of RAM
  - Disk Space of 50 GB
  - 2 CPU (with virtualization support, for example Intel VT)

Refer to the certification documentation on Oracle Technology Network (OTN) for information about the supported installation types, platforms, operating systems, databases, JDKs, and third-party products:

http://www.oracle.com/technetwork/middleware/ias/downloads/fusion-certification-100350.html

2.3 Summary of Steps to Install and Configure Oracle RADIUS Agent

This is a high-level list of steps involved in installing and configuring Oracle RADIUS Agent.

1. Ensure that all the prerequisites for installing Oracle RADIUS Agent are met.
2. Obtain the container image for Oracle RADIUS Agent.

3. Create and start the container and then verify its health status.
4. If you intend to use Oracle RADIUS Agent for multi-factor authentication, then ensure that you have a running instance of Oracle Advanced Authentication .
5. Set up and validate the day-0 or initial configuration.
6. Create global or application-specific configurations as per your requirement.
7. Register the RADIUS client and generate a shared secret.
8. Configure the RADIUS client with the generated shared secret and Oracle RADIUS Agent details.

2.4 Understanding Oracle RADIUS Agent Files and Directories

Learn about the files and directories that Oracle RADIUS Agent stores in the mounted persistent volume of the container.

- All configuration and data is stored in the `/u01/oracle/user_projects` location.
- Instance-specific files (such as instance logs) are stored in the `/u01/oracle/instances` location. If this is not mounted, then instance-specific files will default to the `/u01/oracle/user_projects` location.

To externalize these files and directories, you can mount container volumes (using **-v**) for the above two paths as required, so that they are persisted outside the container. The details for creating the volume and the permissions required are outlined in the Prerequisites for Installing Oracle RADIUS Agent section.

2.5 RADIUS Vendor Specific Attributes

Oracle RADIUS Agent supports the Vendor Specific Attributes (VSA). The RADIUS Vendor Specific Attributes are part of RADIUS Remote Function Call and allow vendors to support their own extended custom attributes. You can define and manage the Vendor Specific Attributes by using Oracle RADIUS Agent's Vendor Specific Attribute endpoint.

Note:

Oracle DB authorizations based on roles requires you to define ORACLE_ROLE VSA, which is included in Oracle RADIUS Agent.

2.6 Understanding Instance-Specific Parameters

Learn about the instance-specific parameters that you can set and configure while starting the container for Oracle RADIUS Agent.

You can pass the following parameters as environment variables while bringing up the Oracle RADIUS Agent container:

- *INSTANCE_NAME*

  This is a mandatory parameter for Oracle RADIUS Agent. Within the location that contains all the Oracle RADIUS Agent files, a directory named *INSTANCE_NAME* is created for each Oracle RADIUS Agent instance. All logs will be present within this *INSTANCE_NAME* directory.

- *java_args*

  JVM Options can be provided. For example:

  ```
  -e java_args='-server -XX:+UnlockExperimentalVMOptions -XX:+UseZGC -
  Xmx1g'
  ```

- *DISABLE_HTTPS*

  If this value is true, then by default REST endpoints will not be https enabled.

- *OVERRIDDEN_INSTANCE_CONFIG*

  This is an optional parameter for specifying instance-specific logging configuration (such as logging levels) that will be overridden during container startup.

  If a particular instance must be started with desired levels of logging, then you can pass a `logging.properties` file as part of this environment variable. This `logging.properties` file should be stored in the volume mapped to the `/u01/oracle/user_projects` location.

  The contents of the `logging.properties` file must be similar to the following entry:

  ```
  oracle.idm.radius.level=FINEST
  ```

- *DEPLOYMENT_NAME*

  This represents name of the Oracle RADIUS Agent deployment. This value is internally used to identify the deployment and its configurations. When multiple Oracle RADIUS Agent instances are running in a cluster sharing the same configuration, the deployment name must be the same for all the instances in the cluster.

  It is recommended to pass deployment name. When it is not passed, a random deployment name is generated and stored in the `deployment.properties` file.

  > **Note:**
  >
  > It is recommended not to manually update deployment name or any other detail in the `deployment.properties` file.

  For example:

  ```
  -e DEPLOYMENT_NAME=oraprod1
  ```

2.7 Installing Oracle RADIUS Agent

Oracle RADIUS Agent is available as a container image. Installing Oracle RADIUS Agent involves pulling the container image, starting the Oracle RADIUS Agent container, and selecting the storage for the configuration files.

1. You can obtain the Oracle RADIUS Agent image either by downloading the zip file from edelivery or by pulling the image from Oracle Container Registry. If downloaded from edelivery, download the Oracle RADIUS Agent zip file from edelivery to the required directory:
   a. Unzip the Oracle RADIUS Agent zip to a directory of your choice.
   b. Load the tar file into the container registry.

      ```
      $ docker load --input <tarfile-path>
      ```

   c. Check the image exists in the container registry.

      ```
      $ docker images
      ```

   Or if using the image from Oracle Container Registry, refer to the documentation there.
2. Tag the image according to deployment environment.

   ```
   docker tag <image name from downloaded image>:<tag> idm-radius-
   server:<tag>
   ```

   Note:

   Downloaded Image Name Example: container-registry.oracle.com/middleware/ora; Tag, Example: 12.2.1.4.210423
3. Create and start the container for Oracle RADIUS Agent by running the following command:

   ```
   $ docker run -d --network [USER_CREATED_NETWORK_NAME] --name
   CONTAINER_NAME -v VOLUME_MAPPING [--user UID:GID] -p
   HOST_CONFIG_PORT:CONTAINER_CONFIG_PORT -p
   HOST_RADIUS_PORT:CONTAINER_RADIUS_PORT [-p
   <HOST_RADIUS_PORT2:CONTAINER_RADIUS_PORT2> . . .] -e <ENV_VAR> -e
   INSTANCE_NAME -e DEPLOYMENT_NAME IMAGE
   ```

   In this command:
   - *USER_CREATED_NETWORK_NAME* is the name of the network bridge and this is only required if you want your Oracle RADIUS Agent container to communicate with other component containers on the same network bridge.
   - *CONTAINER_NAME* is the name of the container for Oracle RADIUS Agent.
   - *VOLUME_MAPPING* is the volume mapping to map a local directory to a container volume.
   - *UID:GID* is the overridden value of the user that will be used by the Oracle RADIUS Agent deployment. This is optional and when this is not provided, Oracle RADIUS Agent will run using default `uid:gid` of `14304:14304`.
   - *HOST_CONFIG_PORT* is the port on the host to be mapped to the Oracle RADIUS Agent container configuration port. This port will be used in the endpoint URLs for all REST API calls.

- *CONTAINER_CONFIG_PORT* is the Oracle RADIUS Agent container configuration port. By default, this is 8080.
- *HOST_RADIUS_PORT* is the port on the host to be mapped to the RADIUS container listener port.
- *CONTAINER_RADIUS_PORT* is the Oracle RADIUS Agent container listener port.
- *ENV_VAR* is any general environment variable to be passed to the JVM that is running Oracle RADIUS Agent. For example, you can use this to set memory limits or language.
- *INSTANCE_NAME* is the name of the Oracle RADIUS Agent instance.
- *DEPLOYMENT_NAME* is the name of the Oracle RADIUS Agent deployment.
- *IMAGE* is the location of the Oracle RADIUS Agent container image in your local image repository.

The following is a sample command to override `uid:gid`:

```
docker run -d --network rad --name radius1 --user 10001:5001  -v /
scratch/radius:/u01/oracle/user_projects -p 8000:8080 -p
1812:1812/udp -p 1813:1813/udp -p 1814:1814 -e INSTANCE_NAME=inst1
container-registry.oracle.com/middleware/radius:latest
```

4. Verify the health status of the container by running the `docker ps` command. Ensure that the status is shown as `healthy`. The output for this command will be similar to the following:

```
CONTAINER ID
IMAGE
COMMAND                        CREATED                   STATUS
PORTS

          NAMES
ef8be733a132        container-registry.oracle.com/middleware/
ora:latest   "entrypoint.sh"          3 weeks ago          Up 3
weeks (healthy)   0.0.0.0:1812-1813->1812-1813/udp, 1814-1830/udp,
0.0.0.0:1814->1814/tcp, 0.0.0.0:8000->8080/tcp radius1
```

> **Note:**
>
> For more information on upgrading Oracle RADIUS Agent, see Upgrading Oracle RADIUS Agent to April 2022 Release or Later.

2.8 Initial or Day-0 Configuration

The initial or day-0 configuration involves setting up a minimum required configuration for Oracle RADIUS Agent container to start.

The information contained in this chapter is explained in detail in the following tutorials: Use Oracle Radius Agent with LDAP as the Primary Authenticator and Use Oracle Radius Agent with Oracle Advanced Authentication for Multi-Factor Authentication .

By default, the container for Oracle RADIUS Agent does not contain any configuration. You must seed in an initial configuration to initialize Oracle RADIUS Agent with your configuration payload.

Seeding in the initial configuration for Oracle RADIUS Agent involves, at a minimum, defining administrator users or groups along with any other configuration required for authentication. These administrator users and groups must preexist in your primary authenticator (LDAP server).

Note:

At least one admin user must be able to bind to your LDAP server otherwise the Oracle RADIUS Agent REST APIs will not be functional.

If you intend to use Oracle RADIUS Agent with Oracle Advanced Authentication for multi-factor authentication, then you must additionally define the required configuration.

You can seed the initial configuration by invoking the following unprotected REST endpoint using the **POST** method:

```
https://<hostname.domain>:PORT/radius-config/v1/init
```

To validate the initial configuration before saving it, invoke the following unprotected REST endpoint using the **POST** method:

```
https://<hostname.domain>:PORT/radius-config/v1/validate
```

In both the endpoints, *PORT* is the host configuration port number (*HOST_CONFIG_PORT*) specified in the `docker run` command. The same port is used in the endpoint URLs for all the REST calls.

After the initial configuration is complete, Oracle RADIUS Agent configuration and administration APIs perform HTTP Basic Authentication as the RADIUS admin user stored in the LDAP server.

The following is the payload that you must pass by using the unprotected REST endpoint for seeding the initial configuration. If you are not using multi-factor authentication with Oracle Advanced Authentication, or you wish to configure at a later time, you may skip the `mfa` section.

```
{
    "radiusAdminGroup": [
              “ADMIN_GRP1”,”ADMIN_GRP2”,”ADMIN_GRPn”
    ],
    "radiusAdminUser": [
              “USERID1 or USER1_DN”,”USERID2 or USER2_DN”,”USERIDn or 
USERn_DN”
    ],
    "authentication": {
        "provider": "LDAP",
        "ldap": {
            "name": "LDAP_SERVER_NAME",
                "dn": "LDAP_SERVER_ADMIN",
                "password": "LDAP_SERVER_ADMIN_PASSWORD",
                "ldapUrl": "LDAPS_URL",
                "baseDN": "LDAP_SERVER_BASE_DN",
                "trustedCertificate": "BASE64_TRUSTED_CA_CERT"
            }
        },
        "mfa": {
            "provider": "OAA",
            "oaa": {
                "oaaUrl": "OAA_RUNTIME_URL",
                "oaaPolicyUrl": "OAA_POLICY_URL",
                "policyUserName":
"POLICY_USERNAME_FROM_OAA_IN_THE_FORMAT_OF_RELEASENAME-oaa-policy",
                "policyUserPassword": "POLICYKEY_FROM_OAA"
            }
        },
        "preferences": {
            "mfaOptions": {
                "defaultGroup": "GROUP_NAME",
                "factorChoices": [
                    "OAA_FACTORS_TO_USE"
                ]
            }
        }
}
```

Note:

The parameters "oaaUrl", "oaaPolicyUrl", "policyUserName", and "policyUserPassword" must be obtained from Oracle Advanced Authentication).

The configuration types such as `authentication`and `mfa` are described in Table 2-1.

In the above example, `trustedCertificate` is used to point to a base64 PEM encoded trusted CA certificate. Alternatively, you can specify the location of the Java keystore (JKS) that contains the certificate. See Configuration Properties for the parameters you need to set when you are using a JKS.

In the `mfa` section above a new Oracle Advanced Authentication (OAA) Agent for RADIUS is created along with an Assurance Level and Policy in one operation. The names assigned to the Agent, Assurance Level and Policy are assigned by OAA directly. If you have previously created your own OAA Agent for RADIUS, along with your own Assurance Level and Policy, then the parameters to pass will change. Refer to the `mfa` parameters in the Configuration Properties section.

The response for this payload will be the initial configuration details that you specified and the port on which Oracle RADIUS Agent listens.

The following is an example cURL command and the payload:

```
curl --header "Content-Type: application/json" --request POST --data-binary
@- https: //localhost:8000/radius-config/v1/init
{
    "radiusAdminGroup": [
        "cn=radiusAdmin1,ou=groups,dc=example,dc=com",
        "cn=radiusAdmin2,ou=groups,dc=example,dc=com"
    ],
    "radiusAdminUser": [
        "uid=user1,ou=people,dc=example,dc=com",
        "uid=user2,ou=people,dc=example,dc=com"
    ],
    "authentication": {
        "provider": "LDAP",
        "ldap": {
            "name": "Corporate LDAP",
            "dn": "cn=Directory Manager",
            "password": "<password>",
            "ldapUrl": "ldaps://ldap.example.com:1636",
            "baseDN": "dc=example,dc=com",
            "trustedCertificate": "-----BEGIN CERTIFICATE-----
\nMIICwzCCAaugAwIBAgIEGfd1kTANBgkqhkiG9w0BAQsFADASMRAwDgYDVQQDEwdteW91ZGRzMB4
XDTIxMDIwODE0MDYyNFoXDTIyMDIwODE0MDYyNFowEjEQMA4GA1UEAxMHbXlvdWRkczCCASIwDQYJ
KoZIhvcNAQEBBQADggEPADCCAQoCggEBAKyx3l/
E4Bt5CLsHSU2UbhPAcSzlsTaOJRZ0V7qAVm6SMlBslokzQthZTuXtlIDlIUcWdCTmMOsk9rZ36E8l
fEkz/
gf5HvXXIaV416Z5O5g4OVQ3MZuPgGvFE17eZRND9NnRdAIv3RWBLjnOGFoD8z7US1i8h7f3fZyZ/
GaQ0VcP4B1ooUTzcQ7MFVymRHMXhlGFVm6cxES5b6EI2R9Wv/BgPY1/
Vypq2kJGJdCoNO8IfXLq1FoGsY2QEbe8tQYJp+cU+WYqAC7cDkNiJ8cxeJ+/
HQ3FFM7BmgzANOrekhNvjCZni9P2PMFXIpO12vEgmtiY4NePoPuyensIznkFySMCAwEAAaMhMB8wH
QYDVR0OBBYEFPJglmgVMkO6FP1ESs32a8HlbO4ZMA0GCSqGSIb3DQEBCwUAA4IBAQAq2whjOzvMaF
TD1m7JK2kzLEtBllJmZ72pwUIz8x0Ju3Kcr4jQeDAq3mOfxR6udWQsJ7+Ovjuvf/
i06HHOxzAbOXOXAyzzS8jbkUX8VjGQueNFdZ7KxumT85gFNkBpe3sDdDmRxgY1pOFIUESFkcie7rL
wCGo1q1z0KvwbqodeZnBprTSFHbePGNAndujVODo4xdH7fIlTrzx6L36BtJKYEKewmrDu9XbhGM1c
8va100WRAHf3IIg8fnrf9Yf3c5+oYdxJoDAr0Y9N8J8ew2Fpdab+I5foQ7kVOCI4OZ23FOLDtGJEy
5mArhTV95EOpqp6+GnE3FnATzUf5ecRChRB\n-----END CERTIFICATE-----",
        }
    },
    "mfa": {
        "provider": "OAA",
        "oaa": {
            "oaaUrl": "https://oaa.example.com:32461/oaa/runtime",
            "oaaPolicyUrl": "https://oaa.example.com:32004/oaa-policy",
            "policyUserName": "oaainstall-oaa-policy",
            "policyUserPassword": "eb3188094918b1bc8e4fd584e8c27f8e7f3fe338"
        }
    },
    "preferences": {
        "mfaOptions": {
            "defaultGroup": "Default",
            "factorChoices": [
                "ChallengeEmail",
                "ChallengeOMATOTP",
                "ChallengeSMS"
            ]
        }
    }
}
```

And the following is the corresponding response:

```
{
    "radiusListener": {
        "port": 1812
    },
    "authentication": {
        "provider": "LDAP",
        "ldap": {
            "name": "Corporate LDAP",
            "dn": "cn=Directory Manager",
            "password": "{AES-GCM}yx8i83DxHhVAKzJ7QE85/
z78ohhFUxmUu6HcGdhxQvkTGaMw",
            "ldapUrl": "ldaps://ldap.example.com:1636",
            "baseDN": "dc=example,dc=com",
            "trustedCertificate": "-----BEGIN CERTIFICATE-----
\nMIICwzCCAaugAwIBAgIEGfd1kTANBgkqhkiG9w0BAQsFADASMRAwDgYDVQQDEwdteW91Z
GRzMB4XDTIxMDIwODE0MDYyNFoXDTIyMDIwODE0MDYyNFowEjEQMA4GA1UEAxMHbXlvdWRk
czCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAKyx3l/
E4Bt5CLsHSU2UbhPAcSzlsTaOJRZ0V7qAVm6SMlBslokzQthZTuXtlIDlIUcWdCTmMOsk9r
Z36E8lfEkz/
gf5HvXXIaV416Z5O5g4OVQ3MZuPgGvFE17eZRND9NnRdAIv3RWBLjnOGFoD8z7US1i8h7f3
fZyZ/GaQ0VcP4B1ooUTzcQ7MFVymRHMXhlGFVm6cxES5b6EI2R9Wv/BgPY1/
Vypq2kJGJdCoNO8IfXLq1FoGsY2QEbe8tQYJp+cU+WYqAC7cDkNiJ8cxeJ+/
HQ3FFM7BmgzANOrekhNvjCZni9P2PMFXIpO12vEgmtiY4NePoPuyensIznkFySMCAwEAAaM
hMB8wHQYDVR0OBBYEFPJglmgVMkO6FP1ESs32a8HlbO4ZMA0GCSqGSIb3DQEBCwUAA4IBAQ
Aq2whjOzvMaFTD1m7JK2kzLEtBllJmZ72pwUIz8x0Ju3Kcr4jQeDAq3mOfxR6udWQsJ7+Ov
juvf/
i06HHOxzAbOXOXAyzzS8jbkUX8VjGQueNFdZ7KxumT85gFNkBpe3sDdDmRxgY1pOFIUESFk
cie7rLwCGo1q1z0KvwbqodeZnBprTSFHbePGNAndujVODo4xdH7fIlTrzx6L36BtJKYEKew
mrDu9XbhGM1c8va100WRAHf3IIg8fnrf9Yf3c5+oYdxJoDAr0Y9N8J8ew2Fpdab+I5foQ7k
VOCI4OZ23FOLDtGJEy5mArhTV95EOpqp6+GnE3FnATzUf5ecRChRB\n-----END
CERTIFICATE-----"
        }
    },
    "mfa": {
        "provider": "OAA",
        "oaa": {
            "name": "Global OAA",
            "oaaUrl": "https://oaa.example.com:32461/oaa/runtime",
            "clientSecret": "9e179ba1-125e-49a4-a0af-a67589031cca",
            "clientId": "f530eea9-c9de-493e-9a39-6fea3b155214",
            "agentgid": "5ff186f1-c291-4940-be38-df58033ab1b4"
        }
    },
    "preferences": {
        "mfaOptions": {
            "defaultGroup": "Default",
            "factorChoices": [
                "ChallengeEmail",
                "ChallengeOMATOTP",
                "ChallengeSMS"
            ],
            "assuranceLevel": "AssuranceLevel-b5f2cca3"
        }
    }
}
```

This completes the initial configuration for Oracle RADIUS Agent and all REST API endpoints are now protected. Any subsequent configuration requires the Oracle RADIUS Agent administrator credentials to perform HTTP Basic Authentication.

2.9 Setting Up Global and Application Configurations

A **global configuration** is a set of configurations that is shared by all RADIUS clients in your Oracle RADIUS Agent instance. In actual deployment environments, different applications can have different configuration requirements. In such scenarios, you can define an **application-scoped configuration** that will override the global configuration for the RADIUS client. Settings not defined in the application configuration are inherited from global configuration.

Topics

- Understanding Global and Application Configurations
- Creating Global and Application Configurations

2.9.1 Understanding Global and Application Configurations

If you intend to set distinct configurations for various RADIUS clients, then you can override specific global configuration settings by using an application configuration with settings that are specific to a given RADIUS client.

For example, suppose your environment consists of two RADIUS clients namely Oracle Database and a VPN client. Both these clients use the same primary and secondary authenticators, but different set of authentication factors for multifactor authentication. In such a scenario, you can define an application configuration for each of the clients to set the respective factors.

Similarly, when you need a dedicated listener for a VPN client that requires higher load and a shared listener for the rest of RADIUS clients in your environments, you can define an application configuration with the dedicated listener port configuration just for the VPN client.

Any other configuration that is not explicitly defined as part of application configuration is inherited from global configuration if it exists. Although you can set different sets of configuration for different RADIUS clients by using application-scoped configuration, admin users or groups set as part of the initial configuration will apply to both global and application-scoped configurations.

2.9.2 Creating Global and Application Configurations

You can create or define a global or an application configuration by making a **POST** request to the following REST API endpoint:

```
https://<host.domain>:PORT/radius-config/v1/configurations
```

Table 2-1 lists the possible values for *`CONFIG_TYPE`*.

Table 2-1   Configuration Types

Configuration Type	Description
`authentication`	Use this configuration type to specify details about the primary authenticator. For example, if you are using an LDAP server as the primary authenticator, then you specify its name, URL, admin user and password, its base DN, and SSL configuration in the `authentication` configuration type section.
`mfa`	Use this configuration type to specify details about Oracle Advanced Authentication to use for multi-factor authentication. For example, you specify details such as the OAA run-time URL, client ID, and client secret in this section.
`preferences`	Use this configuration type to set various preferences based on the deployment requirements. For example, you can specify whether groups must be returned during authentication, specify the type of authentication mode for multi-factor authentication, specify mappings for users and groups in this section.
`radiusListener`	Use this to specify the configuration for your Oracle RADIUS Agent listener port.
`logging`	Use this configuration type to enable or disable logging and also set the log levels. For information on how to set the logging properties refer to Configuring Logging section.
`server`	Use this configuration type for setting configurations related to Oracle RADIUS Agent. For example, you can enable or disable Oracle RADIUS Agent metrics, specify whether dynamic RADIUS clients based on CIDR notation are allowed in this section.

For information about the attributes that you can set and use in each of the configuration types listed above, see REST API documentation.

The following is an example payload to add a new global configuration for the `server` configuration type:

```
{
    "server": {
        "enableMetrics": false,
        "heartBeatInterval": "100000"
    }
}
```

The following is the corresponding response:

```
{
    "server": {
        "enableMetrics": false,
        "heartBeatInterval": 100000
    }
}
```

An application configuration payload is the same as global configuration payload, except that the configuration is enclosed within the `application {}` configuration type and contains 2 additional properties named *`"RADIUS_CLIENT_NAME"`* and `"radiusClientHost"`.

To define an application configuration, use the following format for your payload:

```
{
    "application": {
        "RADIUS_CLIENT_NAME": {
            "radiusClientHost": 
"HOST_NAME_OR_IP_ADDRESS_OF_THE_COMPUTER_HOSTING_THE_RADIUS_CLIENT",
            "CONFIG_TYPE": {
                "PROP1_KEY": "PROP1_VALUE",
                "PROP2_KEY": "PROP2_VALUE",
                "PROPn_KEY": "PROPn_VALUE"
            }
        }
    }
}
```

In this format:

- *CONFIG_TYPE* is the type of configuration that you want to set.
- *PROPn_KEY* and *PROPn_VALUE* are the properties key-value pairs that you can set for each configuration type.
- *`"RADIUS_CLIENT_NAME"`* is any name that you specify to identify the RADIUS client for which you are setting the application configuration.
- `"radiusClientHost"` is a key whose value must be set to the IP address or hostname of the computer hosting the RADIUS client for which you are setting the application configuration.

The following example illustrates the scenario in which you can set an application configuration:

Suppose you have two RADIUS clients namely a database server and a VPN client. Now suppose you want both these clients to use a different set of factors for multi-factor authentication. In such a scenario, you can define an application configuration for each of the clients to define the respective factors.

The following is a sample payload of an application configuration for setting authentication factors for the database RADIUS client. An agent and assurance level for Oracle RADIUS Agent must have been pre-created in Oracle Advanced Authentication.

```
{
    "applicationConfig": {
        "dbRadiusClient": {
            "radiusClientHost": "192.0.2.25",
            "mfa": {
                "provider": "OAA",
                "oaa": {
                    "oaaUrl": "https://oaa.example.com:32461/oaa/
runtime",
                    "clientId": "f530eea9-c9de-493e-9a39-6fea3b15521",
                    "clientSecret": "9e179ba1-125e-49a4-a0af-
a67589031cca",
                    "agentgid": "5ff186f1-c291-4940-be38-df58033ab1b4"
                }
            },
            "preferences": {
                "mfaOptions": {
                    "assuranceLevel": "AssuranceLevel1"
                }
            }
        }
    }
}
```

The following is a sample payload of an application configuration for setting factors for the VPN RADIUS client:

```
{
    "applicationConfig": {
        "vpnRadiusClient": {
            "radiusClientHost": "192.0.2.50",
            "mfa": {
                "provider": "OAA",
                "oaa": {
                    "oaaUrl": "https://oaa.example.com:32461/oaa/
runtime",
                    "clientId": "f530eea9-c9de-493e-9a39-6fea3b15521",
                    "clientSecret": "9e179ba1-125e-49a4-a0af-
a67589031cca",
                    "agentgid": "5ff186f1-c291-4940-be38-df58033ab1b4"
                }
            },
            "preferences": {
                "mfaOptions": {
                    "assuranceLevel": "AssuranceLevel1"
                }
            }
        }
    }
}
```

2.10 Registering a RADIUS Client and Generating the Shared Secret

Whenever you add a new application or service to your network, you must register it as a RADIUS client with Oracle RADIUS Agent and then generate a shared secret for the client to communicate with Oracle RADIUS Agent.

To register a client and generate shared secret, make a **POST** request to the following REST API endpoint:

```
https://radius.example.com:PORT/radius-admin/v1/clients
```

When making the request, enter the Oracle RADIUS Agent admin credentials, configured as part of the initial configuration, for HTTP basic authentication.

The following is the payload that you must pass to the preceding REST API endpoint:

```
{
    "applicationName": "NAME_OF_THE_RADIUS_CLIENT",
    "hostName":
"HOSTNAME_OR_IP_ADDRESS_OF_THE_COMPUTER_HOSTING_THE_RADIUS_CLIENT",
    "applicationType": "TYPE_OF_RADIUS_CLIENT",
    "description": "RADIUS_CLIENT_DESCRIPTION"
}
```

The client details that you specified and the client ID and the shared secret for the registered client is returned in the response. Ensure to note down the shared secret as you must specify this value while configuring your RADIUS client. For example, if you intend to use Oracle Database as your RADIUS client, then you specify the generated shared secret in the database radius.key file.

The following is a sample payload that makes a **POST** request to the `https://<host.domain>:PORT/radius-admin/v1/clients` REST API endpoint:

```
{
    "applicationName": "MyDatabaseClient",
    "hostName": "198.51.100.1",
    "applicationType": "Oracle",
    "description": "Oracle Database"
}
```

And the following is the corresponding sample response:

```
{
    "applicationName": "MyDatabaseClient",
    "applicationType": "Oracle",
    "hostName": "198.51.100.1",
    "id": 2,
    "sharedSecret": "NWNphF-mBec"
}
```

Registering a Dynamic Client

Dynamic client registration allows IP address of the RADIUS client to be specified using CIDR notation.

You can register a dynamic client only if the dynamic client is allowed. To allow a dynamic client, enable the `enableDynamicClients` field in Server Configuration.

The following are the examples for dynamic clients with IP address in CIDR format:

For Day-0, the `enableDynamicClients` flag can be passed in Day-0 init as mentioned in the following example:

```
"server": {
        "enableDynamicClients":true
}
```

For Day-N, if the server configuration is missing, then you can use a POST request to add server configuration as mentioned in the following example:

```
"server": {
        "enableDynamicClients":true
}
```

If server configuration is present, then you can use a PATCH request to update the value of the flag as mentioned in the following example:

```
"server": {
        "enableDynamicClients":true
}
```

If dynamic clients is not enabled, dynamic client registration is not possible and an error is thrown as mentioned in the following example:

```
{
    "errorCode": "IRA-00042",
    "message": "IRA-00042: Dynamic clients not allowed in the server.",
    "timestamp": "2021-04-28T06:46:05.068Z[UTC]"
}
```

To register a dynamic client, the hostName should be given as a range in CIDR notation as mentioned in the following example:

```
{
    "hostName": "192.168.29.244/30",
    "applicationType": "radius",
    "applicationName": "Radius"
}
```

In above example, the CIDR mentioned in the hostName (192.168.29.244/30) includes the IP address range from 192.168.29.244 to 192.168.29.247. The shared secret generated for the CIDR notation will be the shared secret for all these machines and you can use any of these machines to make a RADIUS call.

Any machine which has the IP address in the given CIDR notation range along with the generated shared secret can be used to make a RADIUS call.

2.11 Configuring Logging

Oracle RADIUS Agent logging can be configured during container start up or post Day 0 configuration.

Oracle RADIUS Agent makes use of the *java.util.logging* file. The log files location is configurable and is configured as part of the initial container setup. For more information, see Installing and Configuring Oracle RADIUS Agent section. The Oracle RADIUS Agent also supports setting certain custom logging properties. Log levels can be configured dynamically by calling Oracle RADIUS Agent REST APIs.

Log files are created under `INSTANCE_NAME/logs` directory. The default log names are as follows:

- ora-server0.log - contains the server logs
- ora-access0.log - contains the access logs
- ora-audit0.log - contains the audit logs

Server logs are controlled by setting the `oracle.idm.radius.level`. This can be set to levels defined by `java.util.logging`. The default log level is WARNING.

Access logs are controlled by setting the `oracle.idm.radius.access.log.level`. Access log levels can be ON or OFF. To turn ON, set to `INFO`. To turn OFF, set to any other higher logging level. Access logs are enabled by default.

Audit logs are controlled by setting the `oracle.idm.radius.audit.log.level`. Audit log levels can be ON or OFF. To turn ON, set to `INFO`. To turn OFF, set to any other higher logging level. Audit logs are enabled by default.

To configure log level for logging, make a POST request to the following REST API endpoint:

```
POST https://<hostname.domain>:PORT/radius-config/v1/configurations
```

The following is an example payload that you must pass to the preceding REST API endpoint.

```
{
    "logging": {
        "oracle.idm.radius.level": "FINEST",
        "oracle.idm.radius.access.log.level": "INFO"
    }
}
```

And the following is the corresponding response:

```
{
    "message": "Configuration  is successfully updated.",
    "timestamp": "2021-03-24T15:54:57.502Z[UTC]"
}
```

Using External Logging Properties

You can use the external logging properties for setting additional logging related properties:

- `java.util.logging.FileHandler.limit`

  Logging file handler limits are controlled by setting the `java.util.logging.FileHandler.limit`. This specifies an approximate maximum amount to write (in bytes) to any one file. If this is zero then there is no limit, which is the default.

- `java.util.logging.FileHandler.count`

  Logging file handler count is controlled by setting the `java.util.logging.FileHandler.count`. This specifies how many output files to cycle through. The default value is 1.

  > **Note:**
  >
  > The parameters `java.util.logging.FileHandler.limit` and `java.util.logging.FileHandler.count` will take effect only if they are specified in custom `logging.properties` passed using the `OVERRIDDEN_INSTANCE_CONFIG` variable.

- `oracle.idm.radius.level`

  Use this property to control the server logs.

- `oracle.idm.radius.access.log.level`

  Represents the logger name for access logs and it is enabled by default. Change level of this to "SERVERE" to turn off access logging.

- `oracle.idm.radius.audit.log`

  Represents the logger name for audit logs. Change the level of this to "SERVERE" to turn off audit logs.

- You can use the `FileHandler` with the `LogManager` configuration properties to control the size and count of the server log file.
- The container can be brought up with external `overriden_logging_config.properties` file by specifying it in `OVERRIDDEN_INSTANCE_CONFIG`.

  ```
  docker run -d --network rad --name radius1 -v /home/opc/Radius:/u01/
  oracle/user_projects -p 8000:8080 -p 1812:1812/udp -p 1813:1813/udp
  -p 1814:1814 -e INSTANCE_NAME=inst1 [-e
  OVERRIDDEN_INSTANCE_CONFIG=/u01/oracle/user_projects/
  ```

```
overriden_logging_config.properties] idm-imcs-sandbox.dockerhub-
phx.oci.oraclecorp.com/idm-radius-server:latest
```

2.12 Configuration Properties

The tables below outline the possible configuration properties for Oracle RADIUS Agent.

The table below outlines the Oracle RADIUS Agent configuration properties for the Server Configuration type: `"server:"`

Configuration Parameter	**Description**	**Default Value**
stateCacheEntryTimeout	State Attribute Cache's Entry Timeout	120000
stateCacheConcurrencyLevel	State Attribute Cache's concurrency level	6
validatedTokenCacheEntryTimeout	Validated MFA Token Cache's Entry Timeout	60
validatedTokenCacheConcurrencyLevel	Validated MFA Token Cache's concurrency level	6
customDictionary	Custom RADIUS dictionary file that contain definitions for vendor specific attributes	None
customDictionaryAsStream	Custom dictionary file as stream	None
enableDynamicClients	Indicates if dynamic clients are allowed or not.	None
enableMetrics	Indicates if metrics is enabled.	true
heartBeatInterval	Configured heartbeat thread invocation interval in ms to check availability of authenticator like LDAP.	120000

The table below outlines the Oracle RADIUS Listener configuration properties for the configuration type: `"radiusListener:"`

Configuration Parameters	**Description**	**Default Value**
channelSelectTimeout	UDP NIO channel selection timeout in ms	120000
socketSOTimeout	The underlying socket SO timeout in ms.	1800000
numberOfWorkerThreads	The maximum number of worker threads allowed, this maps to maximum PoolSize value for the underlying ThreadPoolExecutor of the worker threads.	20
coreThreadPoolSize	CorePoolSize value for the underlying ThreadPoolExecutor of the worker threads. It is the minimum number of worker threads the underlying ThreadPoolExecutor maintains.	10

Configuration Parameters	Description	Default Value
threadPoolKeepAliveTime	Thread Pool Keep Alive Time in ms for the underlying ThreadPoolExecutor of the worker threads. When the thread pool more than minimum number of threads configured (coreThreadPoolSize), excess threads will be terminated if they have been idle for more than this threadPoolKeepAliveTime. This provides a means of reducing resource consumption when the thread pool is not being actively used. If the pool becomes more active later, new threads will be constructed. Value 0 means excess threads created will never be cleaned up, so it's recommended to not use 0.	10000
requestCacheEntryTimeout	Request cache's entry timeout value in milliseconds	30000
requestCacheConcurrencyLevel	Request Cache's concurrency level.	6
requestCacheCleanupInitialDelay	Request Cache Cleanup Thread's initial delay.	0
requestCacheCleanupInterval	Request cache cleanup thread's interval.	60000
requestCacheCleanupPoolSize	Request Cache Cleanup Thread Pool Size.	0

The table below outlines the Base Authentication configuration properties. These are applicable to the configuration type: `"authentication:"` and `"mfa:"`

Configuration Parameter	Description	Default Value
name	Name of the configuration	Global Oracle Advanced Authentication for MFAConfig, Global Ldap for LdapConfig, Global Oracle Access Management for OAMConfig
enabled	Flag to indicate if it is enabled	true
retryCount	If configured and a value > 0 indicates that the Operation will be retried for the configured number of times in case of a Connection Failure. Retry can be turned off if value = 0	1
retryInterval	When number of retries is > 1, retry interval between two retries in ms.	0

The table below outlines the Primary Authenticator configuration properties for the configuration type: `"authentication:"`

Configuration Parameter	Description	Default Value
provider	Configured authentication provider type	LDAP
userCacheEntryTimeout	User cache entry timeout value in ms	300000
userCacheConcurrencyLevel	User cache concurrency level	4
userCacheCleanupInterval	User cache cleanup interval	300000
userCacheCleanupPoolSize	User cache cleanup thread pool size	1
userCacheCleanupInitialDelay	UserCache clean up initial delay after which the cleanup thread will start	0
ldap	LDAP configuration details	None

The table below outlines the LDAP Authenticator configuration properties for the configuration type: `"ldap:"`

Configuration Parameter	Description	Default Value
ldapUrl	URL of the LDAP server. It has to be in format ldap://hostname:port(for non-SSL) or ldaps://hostname:port(for SSL)	None
dn	The DN of user which RADIUS agent uses to connect to LDAP server	"" (anonymous bind)
password	The password of user which RADIUS agent uses to connect to LDAP server	"" (anonymous bind)
baseDN	The base DN of the LDAP domain that is used by RADIUS Agent for searching users and groups	""
loginAttr	The login attribute name in LDAP for user. This is used to construct filter to lookup user during login	uid
userFilteringCriteria	Additional filtering criteria for looking up user, this filter when present is ANDed with filter based on loginAttr	None
authWithoutMFACriteria	Based on this filter, matching users are allowed to login using primary factor only when user footprint is not present in MFA.	None
trustedCertificate	Trusted certificate in Base 64 format	None
truststore	Truststore file location in case a JKS file is used for certificates.	None

Configuration Parameter	Description	Default Value
truststorePassword	Truststore password. This is optional and needs to be provided if truststore is used instead of a trustedCertificate.	None
trustedCertificateAliasName	Alias name to be used for referring to trusted certificate in JKS	ldap-server-trusted-cert
keystore	Keystore file location to be used for SSL mutual authentication	None
keystoreCertificateAliasName	Alias name used for referring to the certificate in keystore JKS	None
keystorePassword	Password to the keystore file.	None
keystoreTruststoreType	File type (JKS/PKCS12) when file based truststore/keystore is used. If null, it means certificate itself has been provided.	None
keyFactorAlgorithm	Algorithm with which certificate has been signed	RSA
sslProtocol	The cryptographic protocol RADIUS agent would use for connecting to LDAP server for secure connections. Multiple protocols can be given separated by comma.	None (Uses JVM defaults)
cipherSuites	Default cipher suites supported out of the box	None (Uses JVM defaults)
searchUserBeforeBind	Lookup user in LDAP server to get their DN before initiating user login. This can be disabled if LDAP server supports bind using a mapped ID directly (for example: ID mapper in OUD, UPN bind in Active Directory etc.)	true
connectTimeout	Time limit in milliseconds within which LDAP connection has to be made	5000
readTimeout	Time limit in milliseconds for which RADIUS agent will wait for LDAP server to respond back.	30000
searchTimeout	Time limit in milliseconds for LDAP searches.	30000
referral	Specifies the behavior when a referral is returned by LDAP server. Check JNDI java.naming.referral for possible values.	follow

The table below outlines the LDAP Authenticator - Connection Pool configuration properties for the configuration type: "`ldap:`"

Configuration Parameter	Description	Default Value
initSize	This property indicates the number of connections that are created when connection pool is initialized	5
minSize	The minimum number of connections the pool maintains. If initSize is less than minSize then connection pool is initialized with minSize connections.	5
maxSize	The maximum number of connections the pool maintains. If minSize is greater than maxSize then minSize is set to maxSize.	100
poolMaxWaitTime	Time limit in milliseconds for which client will wait for a LDAP connection to be made available	20000
poolIncrementSize	Number of connections to be made at a time when all existing connections are in use and number of connections are less than maxSize	5
poolMaxConnectionIdleTime	Time in milliseconds after which an idle connection is expired	1500000
poolMaxConnectionReuseTime	Time in milliseconds after which a connection is expired	-1 (No time limit)
poolMaintenanceInterval	Time interval in milliseconds when maintenance thread would run which would enforce above two properties.	600000

The table below outlines the LDAP Authenticator - Connection Socket configuration properties for the configuration type: `"ldap:"`

Configuration Parameter	Description	Default Value
soTimeout	Defines the socket timeout in milliseconds while waiting for data. A value of 0 indicates no timeout	0
soKeepAlive	Enable keep alive probes for socket connection.	true
tcpNoDelay	Data is sent as soon as available	false
reuseAddress	Reuse ports even when it is in TIME_WAIT state	true

The table below outlines the LDAP Authenticator - Group Membership configuration properties for the configuration type: `"ldap:"`

Configuration Parameter	Description	Default Value
memberAttr	LDAP attribute name to be used for group related queries. If not specified then uniquemember and member are used. For AD we need to provide 'member' as its value for nested group searches to work.	None
memberFilteringCriteria	Additional filtering criteria for group searches. This condition when present is ANDed with the group membership query filter. Eg: For AD we need to provide (objectclass=group) for nested group search to work.	None
groupNamingAttribute	Naming attribute for groups in LDAP	cn
maxNestedLevels	Maximum depth search should happen for finding out groups a given user is member of in case if nested groups are configured. (Note: this only applies to the case when group membership query constructed by Oracle RADIUS Agent, and this does not apply to memberof query)	10
enableMemberOfQuery	Enable memberof searches which relies on group membership to be resolved by the backend LDAP server. OID/OUD returns nested group/dynamic groups along with direct membership while AD returns only direct membership of the user.	false
memberOfAttribute	The attribute which returns groups a user is member of. OUD works with ismemberof, AD works with memberof while OID supports both memberof and ismemberof	memberof
maxGroupsToFetch	Sets the maximum number of entries to be returned as a result of the search. A value of 0 indicates no limit. (Note: this only applies to the case when group membership query constructed by Oracle RADIUS Agent, and this does not apply to memberof query)	0

The table below outlines the MFA - Oracle Advanced Authentication configuration properties for the configuration type: "`mfa:`"

Configuration Parameter	Description	Default Value
oaaUrl	Oracle Advanced Authentication Service Base URI	None
timeToLiveInMs	Factor Token's Time To Live in MilliSecs	300000
clientType	Client Type used in Oracle Advanced Authentication for RADIUS Agent	radius
clientSecret	Client Secret in Oracle Advanced Authentication for registered agent for RADIUS Agent	None
clientId	Client ID in Oracle Advanced Authentication for registered agent for RADIUS Agent	None
agentgid	Agent Id in Oracle Advanced Authentication for registered RADIUS Agent	None
connectTimeout	Oracle Advanced Authentication API Connection Timeout in Millisecs	2000
readTimeout	Oracle Advanced Authentication API Read Timeout in Millisecs	10000
oaaPolicyUrl	Oracle Advanced Authentication policy URL	None
policyUserName	Oracle Advanced Authentication policy username	None
policyUserPassword	Oracle Advanced Authentication policy password	None

The table below outlines the Preferences configuration properties for the configuration type: `"preferences:"`

Configuration Parameter	Description	Default Value
returnGroups	Indicates if groups need to be returned during authentication.	false
groupAttrID	RADIUS attribute ID for the group mapping. Groups are returned as part of this RADIUS attribute.	1 (for Oracle)
groupAttrVendorID	RADIUS vendor ID for the group mapping. Groups are returned as part of this RADIUS attribute.	111 (for ORACLE_ROLE attribute)
groupAsSingleString	Returns group details as a single string in response separated by a delimiter that is configured if it's true.	false
groupAsSingleStringDelimiter	Delimiter used when groups are returned as single string.	,
allowTokenInPassword	Indicates if synchronous login mode (Password, delimiter,token concatenation) is enabled/ disabled.	true

Configuration Parameter	Description	Default Value
defaultTokenLength	Length of the token for synchronous login mode. This represents the token length of the DefaultSecondFactor configured.	6
appendDelimiter	Delimiter used for synchronous mode (Password+delimiter+token), for example: password;123456	;
defaultSecondFactor	Default Second Factor. This Factor will be used when no preferred factor is available for the user from the User Preferences of Oracle Advanced Authentication and synchronous mode is used in the RADIUS Request.	ChallengeOMATOTP
mfaOptions	Additional Oracle Advanced Authentication Provider specific options like "assuranceLevel", "factorChoices" (for auto-wiring of Oracle RADIUS Agent and Oracle Advanced Authentication), "defaultGroup" for setting default group name that is passed to Oracle Advanced Authentication.	{"defaultGroup" : "Default"}
allowSpecificFactorInPassword	Indicates if directly invoking Specific Factor by enduser is enabled or not.	false
radiusFactorToMFAFactorMap	RADIUSAgent's Factor to MFA Provider's Factor mapping. These keywords can be used to invoke a specific factor for MFA.	{"totp": "ChallengeOMATOTP", "yubikey": "ChallengeYubicoOTP", "sms": "ChallengeSMS", "mail": "ChallengeEmail"}
userAttrMap	Represents mapping for user attributes from primary authenticator to specified RADIUS Attributes which are to be returned during authentication.	None
groupNameMapping	Mappings to map group names in primary authenticator to different values.	None
factorToTokenLengthMap	RADIUS Factor to factor token length mapping.	{"ChallengeOMATOTP": 6, "ChallengeYubicoOTP": 44}

The table below outlines the Preferences - UserAttrMapping configuration properties for the configuration type: "`userAttrMap:`"

Configuration Parameter	Description	Default Value
vendorId	Vendor ID associated of the mapped RADIUS user attribute	None
attrName	User attribute name in primary authenticator	None
attrId	Attribute id of the mapped RADIUS user attribute	None

2.13 Setting Up Load Balancing

As the Oracle RADIUS Agent Container instance is stateless, you can set up load balancing and high availability by deploying multiple Oracle RADIUS Agent instances behind a load balancer. You can start a new Oracle RADIUS Agent Container instance easily by pointing to the persisted configurations on the volume. This allows easy horizontal scaling.

Session Persistence

When deploying Oracle RADIUS Agent with a load balancer, you need to configure in the load balancer the session persistence based on the IP address of the end user. This is specifically needed since RADIUS is based on User Datagram Protocol and Multi-factor Authentication requests in RADIUS use RADIUS challenge response flow, which uses different RADIUS packets.

3 Administering and Maintaining Oracle RADIUS Agent

Topics

3.1 Managing Global and Application Configurations

Learn how to manage global and application configurations for Oracle RADIUS Agent.

Topics

3.1.1 Updating Global and Application Configurations

To add or modify properties for a given configuration type (for example, mfa, preferences, logging, and so on), make a **PATCH** request to the following protected REST API endpoint:

```
https://localhost:PORT/radius-config/v1/configurations
```

To delete properties for a given configuration type (for example, mfa, preferences, logging, and so on), pass the property value as null and to remove a particular index from a list of values, pass the 'delete' property with index number of the value to be deleted.

To update an existing global or application configuration, make a **PATCH** request to the following protected REST API endpoint:

```
https://localhost:PORT/radius-config/v1/configurations
```

The following is a sample payload for updating the configuration for the `logging` configuration type:

```
{
    "logging": {
        "oracle.idm.radius.level": "FINEST",
        "oracle.idm.radius.access.log.level": "INFO"
    }
}
```

The following is the corresponding response:

```
{
    "message": "Configuration  is successfully updated.",
    "timestamp": "2021-03-02T07:12:01.926Z[UTC]"
}
```

> **Note:**
>
> If you have multiple containers with the shared configuration and one of the containers adds, updates, or deletes the shared configuration at runtime, in that case implicit refresh for other containers does not happen. To get the latest changes either restart other containers or reload the configuration for other containers.
> To perform a reload, run a GET request to `https://<hostname.domain>/radius-config/v1/configurations/reload` . See the Oracle RADIUS Agent REST API guide for more details.

3.1.2 Deleting Global and Application Configurations

To delete an existing global configuration, make a **DELETE** request to the following protected REST API endpoint:

```
https://localhost:PORT/radius-config/v1/configurations/application/
APP_CONFIG_NAME
```

In this endpoint, `APP_CONFIG_NAME` is the name of the application that you want to delete. For example, to delete an application, provide the name of the application such as `DBApplication` as shown in the example below:

```
curl --request DELETE https://localhost:8000/radius-config/v1/
configurations/application/DBApplication --basic -u radadmin1:Welcome1
```

The following is the corresponding response:

```
{
    "message": "Configuration DBApplication is successfully deleted.",
    "timestamp": "2021-03-03T14:31:22.359Z[UTC]"
}
```

To delete global configuration type, make a **DELETE** request to the following protected REST API endpoint:

```
https://localhost:PORT/radius-config/v1/configurations/CONFIG_TYPE
```

In this endpoint, replace:

- *CONFIG_TYPE* with the key of the configuration type that must be deleted. For example, to delete the configuration for multi-factor authentication, replace *CONFIG_TYPE* in the endpoint with `mfa`.

The following is an example endpoint for deleting the configuration mfa:

```
https://localhost:8000/radius-config/v1/configurations/mfa
```

Upon successful execution of the payload, you receive a response that the configuration type is successfully deleted.

The following is the corresponding response:

```
{
    "message": "Configuration mfa is successfully deleted.",
    "timestamp": "2021-03-03T15:35:25.359Z[UTC]"
}
```

3.1.3 Viewing and Reloading Configurations

Viewing Configurations

To view the list of all configurations set in Oracle RADIUS Agent, make a **GET** request to the following protected REST API endpoint:

```
https://localhost:PORT/radius-config/v1/configurations
```

The following is an example GET request to view all the configurations:

```
curl --request GET https://localhost:8000/radius-config/v1/configurations --
basic -u radadmin1:Welcome1
```

The following is the response:

```
{
    "name": "Oracle RADIUS Agent",
    "version": "1.0",
    "oracleRadiusAgent": {
        "radiusListener": {
            "port": 1811
        },
        "authentication": {
            "provider": "LDAP",
            "ldap": {
                "name": "Corporate LDAP",
                "dn": "cn=directory manager",
                "password": "{AES-
GCM}AJ8RXxCvTqlzvY0zlP6xxXUHKN0HukpfdLf4Cq3KCo9wO1u5",
                "ldapUrl": "ldaps://myoudds:1636",
                "baseDN": "dc=example,dc=com",
                "truststore": "-----BEGIN CERTIFICATE-----
\nMIICwzCCAaugAwIBAgIEaNGteTANBgkqhkiG9w0BAQsFADASMRAwDgYDVQQDEwdteW91Z
GRzMB4XDTIxMDEyNzEwMzAwMFoXDTIxMDQyNzEwMzAwMFowEjEQMA4GA1UEAxMHbXlvdWRk
czCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBALHo889pq1F5hgyAFA4rXDP4qww
Zoq7Sg4NIEMrN1elSQmTyXAr2XGtOHtDPojwI4+Tf9dDyr14OSOuUNx8jNA/
viQ3YVmSIn5LiIIJSFNR0GOXAp+ZY2GdEjeAdLkJTd0UHBQPIv3r68TIn7U9UyemDo2PNVU
Rm9f0Xe7DKpj1I1/qYHn0L/
ROs6wS99KsGs8lhij0XGjTKtibaN4XljjKY5uQLD+xTul7B3msirklAKAj72dCfiwmBRn+e
c2bxhfu1DDI0+0gzKpmanCH6s4LjHx572lv0B64Bii5Rzm+rGIUaiiTbODLDFOmxMnfBUYd
Z2GXpx3U3G+BaJvuOZvcCAwEAAaMhMB8wHQYDVR0OBBYEFL8D0CWWPhQLmO8jIL+GECrkAE
ZaMA0GCSqGSIb3DQEBCwUAA4IBAQBeZPiyRc3Xjeuqnwjdu+75zkV52ILXUyvLlMtoXeLRN
g/
NBcZEvtun4MXoODI1pD8g3JWgobCuol7dYVqRLvWGOhhhxVtJ5MDYHQMCQQkvpI2iSdMGF4
+G8H/
kze0xwjPSjg18DxPbFjvDb+5EmfHfk1RpBVbG0sw292iThEJMg9rr25TMy8fhjQmlUdxD/
1yfmd1Qe8xzw+FnEggAUedCGEWyBwEUu/
QaWuLLDpBl98YX1dH9CF4XyGlJ5eNS7AqM4lXjGTTbvkUM7+s5toPUA2z4rVFnMdW0KHjQ1
E6jNDIF+OHwUF61vZeKWxhTATCb83z9HjutqvUZNggd5u+M\n-----END
CERTIFICATE-----",
                "trustedCertificateAliasName": "ldap-server-cert",
                "truststorePassword": "{AES-GCM}gFGvY3D2T+jqn152TU0s/
BSbdaW5eHaa85xZT0OpzhNFLwkd"
            }
        },
        "application": {
            "dbRadiusClient": {
                "name": "Database RADIUS Client",
                "radiusClientHost": "100.100.100.100",
                "radiusListener": {
                    "port": 1815,
                    "numberOfWorkerThreads": 2
                }
            },
            "applicationConfig": {
                "name": "Database RADIUS Client",
                "radiusClientHost": "100.100.100.101",
                "radiusListener": {
                    "port": 1815,
                    "numberOfWorkerThreads": 2
                }
            }
        }
    }
}
```

Reloading Configurations

Reloading configurations will read all the latest configurations and reinitialized Oracle RADIUS Agent. To reload configurations, make a **GET** request to the following protected REST API endpoint:

```
https://localhost:PORT/radius-config/v1/configurations/reload
```

The following is a sample **GET** request that reloads the latest configuration:

```
curl --request GET https://localhost:8000/radius-config/v1/configurations/
reload --basic -u radadmin1:Welcome1
```

This following is the response:

```
{
    "message": "Configuration reloaded successfully.",
    "timestamp": "2021-03-03T13:22:36.498Z[UTC]"
}
```

3.1.4 Validating Configurations

To validate a global or an application configuration for given configuration type without saving it for Oracle RADIUS Agent, make a **POST** request to the following protected REST endpoint:

```
https://localhost:PORT/radius-config/v1/validate
```

4

Tuning and Troubleshooting for Oracle RADIUS Agent

Topics

- Tuning
- Troubleshooting Tips
- Known Issues

4.1 Tuning

The performance of the RADIUS Agent depends on several factors. External factors such as hardware, OS level tunings, network latency, performance tuning of LDAP and Oracle Advanced Authentication (OAA) must be tuned appropriately. This page summarizes only the common important tuning parameters within RADIUS Agent. For complete list of configuration parameters, see Configuration Properties

Topics

- JVM
- RADIUS Agent Configurations

4.1.1 JVM

JVM options can be passed into RADIUS Agent from the docker run command when bringing up a new container. For eg: (Values given here are just for showing a sample):

```
-e java_args='-Xms1g -Xmx4g -Duser.language=en'
```

Any jvm parameter can be provided the same way.

Providing JVM Options via a java.properties file:

Alternatively, if you want to provide a long list of jvm parameters, they can also be provided using a java.properties file that needs to exist within the config/ directory(for example, if the volume is mapped as */home/opc/Radius:/u01/oracle/user_projects*, then java.properties file should pre-exist under */home/opc/Radius/config/*). This file will be automatically picked up when docker container starts.

Example contents of java.properties should be similar to:

```
java_args= -server -XX:+UnlockExperimentalVMOptions -XX:+UseZGC -Xms1g -Xmx4g -
Duser.language=en
```

4.1.2 RADIUS Agent Configurations

You can use the following configuration properties for tuning in the RADIUS Agent:

Listener

- numberOfWorkerThreads - The maximum number of worker threads allowed, this maps to maximum PoolSize value for the underlying ThreadPoolExecutor of the worker threads.
- coreThreadPoolSize - CorePoolSize value for the underlying ThreadPoolExecutor of the worker threads. It is the minimum number of worker threads the underlying ThreadPoolExecutor maintains.
- threadPoolKeepAliveTime - Thread Pool Keep Alive Time in ms for the underlying ThreadPoolExecutor of the worker threads. When the thread pool more than minimum number of threads configured (coreThreadPoolSize), excess threads will be terminated if they have been idle for more than this threadPoolKeepAliveTime. This provides a means of reducing resource consumption when the thread pool is not being actively used. If the pool becomes more active later, new threads will be constructed. Value 0 means excess threads created will never be cleaned up, so it's recommended to not use 0.
- threadPoolMaxQueueSize - Allowed maximum work queue size value for the underlying queue used by ThreadPoolExecutor of the worker threads. 0 means queue is unbounded.

Note:

numberOfWorkerThreads and coreThreadPoolSize need to be configured based on CPU Cores available on the server hardware.

LDAP Connection Pool

- initSize - This property indicates the number of connections that are created when connection pool is initialized
- maxSize - The maximum number of connections the pool maintains. If minSize is greater than maxSize then minSize is set to maxSize.
- minSize - The minimum number of connections the pool maintains. If initSize is less than minSize then connection pool is initialized with minSize connections.
- poolIncrementSize - Number of connections to be made at a time when all existing connections are in use and number of connections are less than maxSize

4.2 Troubleshooting Tips

Topics:

- **RADIUS client unable to connect to Oracle RADIUS Agent**
- **User Primary Authentication Fails in RADIUS Request**
- **User MFA Authentication Fails in RADIUS Request**

4.2.1 RADIUS client unable to connect to Oracle RADIUS Agent

You can use the following troubleshooting tips:

- Make sure Oracle RADIUS Agent host and port is reachable from the RADIUS client host
- Check Oracle RADIUS Agent logs and see if Oracle RADIUS Agent is dropping requests because of RADIUS client registration issues, you can fix the client registration on Oracle RADIUS Agent.
- Make sure RADIUS client application is registered correctly with Oracle RADIUS Agent using the correct hostname or IP address
- Verify the shared secret on RADIUS client side and make sure that it is correct
- Change log level for Oracle RADIUS Agent to FINE or FINEST and check Oracle RADIUS Agent logs

4.2.2 User Primary Authentication Fails in RADIUS Request

You can use the following troubleshooting tips:

- Check Oracle RADIUS Agent logs to find underlying error.
- Verify the shared secret on RADIUS client side and make sure that it is correct. An incorrect shared secret on client side can lead to this error.
- Change log level for Oracle RADIUS Agent to FINE or FINEST and check Oracle RADIUS Agent logs.

4.2.3 User MFA Authentication Fails in RADIUS Request

You can use the following troubleshooting tips:

- Check Oracle RADIUS Agent logs to find underlying error.
- Make sure that Oracle Advanced Authentication is up and running.
- Oracle Advanced Authentication configuration steps are correctly followed and present in Oracle RADIUS Agent configuration.
- Factors are seeded in Oracle Advanced Authentication for user
- Make sure that RADIUS client supports RADIUS challenge, if not, synchronous authentication mode needs to be used.
- Check Oracle Advanced Authentication logs to get further details if error is coming from Oracle Advanced Authentication.
- Change log level for Oracle RADIUS Agent to FINE or FINEST and check Oracle RADIUS Agent logs

4.2.4 RADIUS Request Taking Time

You can use the following troubleshooting tips:

- Check Oracle RADIUS Agent access logs to find which layer is taking time.
- Rule out network latency between RADIUS client and Oracle RADIUS Agent.
- Ensure LDAP and Oracle Advanced Authentication are tuned properly by referring to their corresponding documentations.
- Perform tuning on Oracle RADIUS Agent if needed

4.2.5 Enable Administrator REST Endpoint Using the CLI Utility

You should use the CLI utility only to unblock the administrator to gain access of HTTP-REST administration to manage configuration and the RADIUS client. For example, if the administrator user or the administrator group has changed, then you have to reset either the administrator user or the administrator group using the CLI utility to again invoke the HTTP-REST endpoint. Similarly, if LDAP details are changed then you can use the CLI utility to reset LDAP details so that the administrator user can invoke the HTTP-REST endpoint.

Note:

You need to use the CLI Utility from the docker container.

The command line utility provides the following nine operations:

1. Reset Config
2. Reset Admin Users
3. Reset Admin Groups
4. Print Admin Users and Groups
5. Print Configuration
6. Reset LDAP Config
7. Print HTTPS Secret Store Password
8. Print Bootstrap Secret Store Password
9. Reset Bootstrap Secret Store Password

Note:

The environment variable, `ORACLE_RADIUS_AGENT_COMMON_DIR` should be set to point to the configuration directory location.

Reset Config

The *resetConfig* option resets the configuration in `oracleRadiusAgent.json` . The *resetConfig* option removes all existing configuration in `oracleRadiusAgent.json`.

```
java -cp /u01/oracle/idmradius/oracleradiusagent.jar -
DORACLE_RADIUS_AGENT_COMMON_DIR=/u01/oracle/user_projects
oracle.idm.radius.util.ORACli resetConfig
```

Reset Admin Users

The *resetAdminUsers* option resets the administrator users defined for Oracle RADIUS agent. The administrator users separated by delimiter ';' needs to be passed as argument.

```
java -cp /u01/oracle/idmradius/oracleradiusagent.jar -
DORACLE_RADIUS_AGENT_COMMON_DIR=/u01/oracle/user_projects
oracle.idm.radius.util.ORACli resetAdminUsers
"uid=adminUser,ou=People,dc=example,dc=com;uid=test,ou=People,dc=example,dc=c
om"
```

Reset Admin Groups

The *resetAdminGroups* option resets the administrator groups defined for Oracle RADIUS agent. The administrator groups separated by delimiter ';' needs to be provided as argument.

```
java -cp /u01/oracle/idmradius/oracleradiusagent.jar -
DORACLE_RADIUS_AGENT_COMMON_DIR=/u01/oracle/user_projects
oracle.idm.radius.util.ORACli resetAdminGroups "group1;group2"
```

Print Admin Users and Groups

The *printAdminUsersGroups* option prints the list of users and groups defined for Oracle RADIUS agent.

```
java -cp /u01/oracle/idmradius/oracleradiusagent.jar -
DORACLE_RADIUS_AGENT_COMMON_DIR=/u01/oracle/user_projects
oracle.idm.radius.util.ORACli printAdminUsersGroups
```

Print Configuration

The *printConfiguration* option prints the Oracle RADIUS Agent's configuration details present in `oracleRadiusAgent.json`.

```
java -cp /u01/oracle/idmradius/oracleradiusagent.jar -
DORACLE_RADIUS_AGENT_COMMON_DIR=/u01/oracle/user_projects
oracle.idm.radius.util.ORACli printConfiguration
```

ORACLE_RADIUS_AGENT_COMMON_DIR

Reset Ldap Config

The *resetLdapConfig* resets the primary authenticator details that are present in the `oracleRadiusAgent.json`. The LDAP details such as new `LDAP url`, `dn`, `password`, `truststore` (Base 64 certificate) and `truststore type` (JKS or PKC12, optional for base 64 certificate) along with old `LDAP url` needs to be provided as argument. `Dn` and `password`, if not provided, will be picked from the existing LDAP configuration details along with `loginAttr` and `baseDN`.

```
java -cp /u01/oracle/idmradius/oracleradiusagent.jar -
DORACLE_RADIUS_AGENT_COMMON_DIR=/u01/oracle/user_projects
oracle.idm.radius.util.ORACli resetLdapConfig -oldLdapUrl "ldap://
slc14xzv.us.oracle.com:1389" -newLdapUrl "ldaps://
slc10nty.us.oracle.com:636" -dn "Administrator@ipf.com" -password
"Welcome123" -truststore "-----BEGIN CERTIFICATE-----
\nMIIF4DCCBMigAwIBAgITSwAAAALbKU4kkH5mlgAAAAAAAjANBgkqhkiG9w0BAQUF\nADB
DMRMwEQYKCZImiZPyLGQBGRYDY29tMRMwEQYKCZImiZPyLGQBGRYDaXBmMRcw\nFQYDVQQD
Ew51cy1TTEMxME5UWS1DQTAeFw0yMDA3MjIxMjQxMzdaFw0yMTA3MjIx\nMjQxMzdaMCExH
zAdBgNVBAMTFnNsYzEwbnR5LnVzLm9yYWNsZS5jb20wggEiMA0G\nCSqGSIb3DQEBAQUAA4
IBDwAwggEKAoIBAQC9Ze3otkd+5CLAHEsFK0+rYvqrhfHP\njh46SyANv8qTYzht7k00ijZ
LOftfDw98wUDTUc+8GQ6O/
UZPtDuEdPHpUGVz0G9/\nJZqiMFcFCBMc5gCrc8UYGeQzTVAfFlx6B/
2rbTVDCND3huMZO0pIt+/QM1xJmtuA\ns/
6lUMyJAuK23A+qIjcvhHvzGu196384WGMlr43N85j2Git+u6hEa5xv8aWxO1ne\nFkS1pb1
s153EGqH4V849hJHI+v0nyOz7Ex7mjjl8tUdbfaIgXgWWVDJQCZ4airwk\n3PcK53bXSO1g
xqhAUDKSLK2AYLtyu9fdhDLfZyqQhOHzKaHms+me111LAgMBAAGj\nggLtMIIC6TAvBgkrB
gEEAYI3FAIEIh4gAEQAbwBtAGEAaQBuAEMAbwBuAHQAcgBv\nAGwAbABlAHIwHQYDVR0lBB
YwFAYIKwYBBQUHAwIGCCsGAQUFBwMBMA4GA1UdDwEB\n/
wQEAwIFoDB4BgkqhkiG9w0BCQ8EazBpMA4GCCqGSIb3DQMCAgIAgDAOBggqhkiG\n9w0DBA
ICAIAwCwYJYIZIAWUDBAEqMAsGCWCGSAFlAwQBLTALBglghkgBZQMEAQIw\nCwYJYIZIAWU
DBAEFMAcGBSsOAwIHMAoGCCqGSIb3DQMHMB0GA1UdDgQWBBSuJVQB\nyLox1hotAXeKwWOg
1ZOPgTAfBgNVHSMEGDAWgBSue+W/t+6CQmrn/
6+ySlpXnNu7\npjCByQYDVR0fBIHBMIG+MIG7oIG4oIG1hoGybGRhcDovLy9DTj11cy1TTE
MxME5U\nWS1DQSxDTj1zbGMxMG50eSxDTj1DRFAsQ049UHVibGljJTIwS2V5JTIwU2Vydml
j\nZXMsQ049U2VydmljZXMsQ049Q29uZmlndXJhdGlvbixEQz1pcGYsREM9Y29tP2Nl\ncn
RpZmljYXRlUmV2b2NhdGlvbkxpc3Q/
YmFzZT9vYmplY3RDbGFzcz1jUkxEaXN0\ncmlidXRpb25Qb2ludDCBvAYIKwYBBQUHAQEEg
a8wgawwgakGCCsGAQUFBzAChoGc\nbGRhcDovLy9DTj11cy1TTEMxME5UWS1DQSxDTj1BSU
EsQ049UHVibGljJTIwS2V5\nJTIwU2VydmljZXMsQ049U2VydmljZXMsQ049Q29uZmlndXJ
hdGlvbixEQz1pcGYs\nREM9Y29tP2NBQ2VydGlmaWNhdGU/
YmFzZT9vYmplY3RDbGFzcz1jZXJ0aWZpY2F0\naW9uQXV0aG9yaXR5MEIGA1UdEQQ7MDmgH
wYJKwYBBAGCNxkBoBIEEE/
w10Ud3xFB\nlF4D6wUf4FeCFnNsYzEwbnR5LnVzLm9yYWNsZS5jb20wDQYJKoZIhvcNAQEF
BQAD\nggEBAFIjY72T7aoGYliW+ZTuGC/
Js07cuwB4LYgjO+MKz82sTJyb9AYhfzmmr1fq\nsVz6hCWz6OmDHX9oDRxa21kq4e3aJmGq
Q4NbW+z4hEtBWWWrW7uG4p6CDuB0aAF2\np5IIscXqQpqH0yJC5aABUHCQWJ2225joNPBeR
8vLpa0Wx3raV6GbDhczlxAVWdcA\nGpSkKgajj0MPnhLGHtMvjSJbwcEqw1si6bg5yYUTOE
sURaxk2YMRKu/5GXGCX0RR\n30b/
3Cu9HJap3Gw4a+4bK4zN11qybZqbwLzyloS4I3IMvAA1BD3wMo3D3JDU1xfQ\nzt+1lcqqa
XJ9iuD2/UwJYR7uFI4=\n-----END CERTIFICATE-----"

java -cp /u01/oracle/idmradius/oracleradiusagent.jar -
DORACLE_RADIUS_AGENT_COMMON_DIR=/u01/oracle/user_projects
oracle.idm.radius.util.ORACli resetLdapConfig -oldLdapUrl "ldap://
slc14xzv.us.oracle.com:1389" -newLdapUrl "ldaps://
slc10nty.us.oracle.com:636"  -truststore "/u01/oracle/cert/
orakeystore.p12" -truststoreType "PKCS12" -truststorePassword welcome
```

Print HTTPS Secret Store Password

The *printHTTPSSecretStorePassword* option prints the HTTPS self-signed secret store password.

```
java -cp /u01/oracle/idmradius/oracleradiusagent.jar -
DORACLE_RADIUS_AGENT_COMMON_DIR=/u01/oracle/user_projects
oracle.idm.radius.util.ORACli printHTTPSSecretStorePassword
```

Print Bootstrap Secret Store Password

The *printBootstrapSecretStorePassword* option prints the bootstrap secret store password.

```
java -cp /u01/oracle/idmradius/oracleradiusagent.jar -
DORACLE_RADIUS_AGENT_COMMON_DIR=/u01/oracle/user_projects
oracle.idm.radius.util.ORACli printBootstrapSecretStorePassword
```

Reset Bootstrap Secret Store Password

The *resetBootstrapSecretStorePassword* option resets the password of the bootstrap secret store. The new password must be provided as argument.

```
java -cp /u01/oracle/idmradius/oracleradiusagent.jar -
DORACLE_RADIUS_AGENT_COMMON_DIR=/u01/oracle/user_projects
oracle.idm.radius.util.ORACli resetBootstrapSecretStorePassword
<new_password>
```

4.3 Known Issues

Topics:

- Updating Global and Application Configurations
- LDAP Configuration Changes

4.3.1 Updating Global and Application Configurations

If you have multiple containers with the shared configuration and one of the containers adds, updates, or deletes the shared configuration at runtime, in that case implicit refresh for other containers does not happen. To get the latest changes either restart other containers or reload the configuration for other containers.To perform a reload, run a GET request to `https://<hostname.domain>/radius-config/v1/configurations/reload` . See the Oracle RADIUS Agent REST API guide for more details.

4.3.2 LDAP Configuration Changes

Any LDAP configuration changes results in restart of LDAP connection pool.

Appendices

You may need to consult this supplemental information like the custom database provided supported by Oracle Radius Agent.

This part contains the following appendix:

A
Custom Database Provider

This section provides information on how to install and configure a custom database provider (`ora-custom-db-provider`) for SQL*Plus Clients to support Yubikey OTPs.

Currently, Oracle DB does not allow Yubikey OTPs (44chars) from SQL*Plus Clients to enable multi-factor authentication with the RADIUS Server. To overcome this limitation, a custom UI implementation is provided which you can install on SQLPlus Clients. The custom UI implementation supports multi-factor authentication including Yubikey OTPs.

The custom database provider UI jar file, `ora-custom-db-provider-1.0.0.jar` is packaged in the `idm-radius` docker image and can be copied from the docker container at the following location:

```
/u01/oracle/ora-custom-db-provider
```

To begin using the custom UI implementation, perform the following tasks for SQL *Plus clients:

1. You must update the `sqlnet.ora` file with the following settings. The `sqlnet.ora` file exist in the location, `ORACLE_HOME/network/admin/sqlnet.ora`. It is recommended to use Oracle Net Manager. For more information, see Configure RADIUS on the Oracle Client

   ```
   SQLNET.RADIUS_AUTHENTICATION_INTERFACE = oracle/idm/radius/dbprovider/
   CustomRadiusInterface
   ```

   ```
   SQLNET.RADIUS_CLASSPATH = <LIB_FOLDER>/ora-custom-db-
   provider-1.0.0.jar:<ORACLE_HOME>/network/jlib/netradius8.jar:<LIB_FOLDER>/
   aopalliance-repackaged-2.6.1.jar:<LIB_FOLDER>/hk2-
   api-2.6.1.jar:<LIB_FOLDER>/hk2-locator-2.6.1.jar:<LIB_FOLDER>/hk2-
   utils-2.6.1.jar:<LIB_FOLDER>/jackson-annotations-2.10.1.jar:<LIB_FOLDER>/
   jackson-core-2.10.1.jar:<LIB_FOLDER>/jackson-
   databind-2.10.1.jar:<LIB_FOLDER>/jackson-module-jaxb-
   annotations-2.10.1.jar:<LIB_FOLDER>/
   jakarta.activation-1.2.2.jar:<LIB_FOLDER>/jakarta.activation-
   api-1.2.1.jar:<LIB_FOLDER>/jakarta.annotation-api-1.3.5.jar:<LIB_FOLDER>/
   jakarta.inject-2.6.1.jar:<LIB_FOLDER>/jakarta.ws.rs-
   api-2.1.6.jar:<LIB_FOLDER>/jakarta.xml.bind-api-2.3.2.jar:<LIB_FOLDER>/
   javassist-3.25.0-GA.jar:<LIB_FOLDER>/jersey-client-2.32.jar:<LIB_FOLDER>/
   jersey-common-2.32.jar:<LIB_FOLDER>/jersey-entity-
   filtering-2.32.jar:<LIB_FOLDER>/jersey-hk2-2.32.jar:<LIB_FOLDER>/jersey-
   media-json-jackson-2.32.jar:<LIB_FOLDER>/osgi-resource-
   locator-1.0.3.jar:<ORACLE_HOME>/network/jlib:<ORACLE_HOME>/jdk/jre/
   lib:<ORACLE_HOME>/lib:<ORACLE_HOME>/jdk/jre/lib/
   amd64:<ORACLE_HOME>/jdk/jre/lib/amd64/server/
   ```

   All the required jars mentioned above can be copied into a folder on the file system (LIB_FOLDER). You must provide the Absolute Path in the above setting.

2. For the custom UI to connect and validate the passwords and OTPs, you need to configure the mandatory parameters such as *RADIUS Server Host*, *RADIUS Server HTTPS Port*, and *DB Server Host*. You can configure the mandatory parameters by using the `config.properties` file. By default the `config.properties` file is looked for in `USER.HOME` (`user.home` system property). You can override this location by defining an ENV Variable: IRA_CONFIG_PATH. For example,

   ```
   export IRA_CONFIG_PATH=/home/opc/uiconfig
   ```

   Here, the `config.properties` file would be searched for in the above path.

   > **Note:**
   >
   > If you have defined the hostname aliases for the RADIUS Server and Database Server and the default HTTPS port used by RADIUS Server is 8080, then you need not use the `config.properties` file.

Parameters of `config.properties` File

The parameters in the `config.properties` file is as follows:

- RADIUS_HOST
  Represents the IP address of the RADIUS Server to connect to. If a hostname alias by name, `ora-radius-server` is defined, then by default, it is considered as the RADIUS_HOST. Otherwise, the value for RADIUS_HOST in the `config.properties` file is considered.
- RADIUS_HTTPS_PORT
  Represents the HTTPS port of the RADIUS Server to connect to. By default, 8080 is used. If the RADIUS Server uses a different HTTPS port, then this parameter needs to be defined in the `config.properties` file.
- DB_SERVER_HOST
  Represents the IP address of the DB Server. If a hostname alias by name, `db-server` is defined, then by default, it is considered as the DB_SERVER_HOST. Otherwise, the value for DB_SERVER_HOST in the `config.properties` file is considered.

Sample Value of `config.properties` file Parameters

The sample value of the parameters in the `config.properties` file is as follows:

If `db-server` hostname alias is defined, then DB_SERVER_HOST need not be configured.

```
DB_SERVER_HOST=1.1.1.1
```

If `ora-radius-server` hostname alias is defined, then RADIUS_HOST need not be configured.

```
RADIUS_HOST=2.2.2.2
```

If the default HTTPS port used by RADIUS Server is 8080, then RADIUS_HTTPS_PORT need not be configured.

```
RADIUS_HTTPS_PORT=8090
```

Troubleshooting

A basic logging feature is provided with the capability to only turn on/off logging in case of any issues seen while connecting from the SQL*Plus clients. You can configure the following properties in the config.properties file, if required.

To enable logging, set the following property:

```
ENABLE_LOG=true
```

By default, `ora-custom-mfa.log` is created under the system property, `user.dir`. This location can be customized by defining the following property:

```
LOG_FILE_LOCATION=path_where_logfile_should_be_created
```

If the `custom-db-provider` is not able to connect to RADIUS Server and timing out, configure the following two properties:

- CONNECT_TIMEOUT: Represents the time in milliseconds that the client waits to connect to RADIUS Server before a timeout. Default value is 60000ms (1 min).
- READ_TIMEOUT: Represents the time in milliseconds that the client waits for the server to respond back before a timeout. Default value is 60000ms (1 min).

B Upgrading Oracle RADIUS Agent to April 2022 Release or Later

From the April 2022 release, Oracle RADIUS Agent uses a new default `uid:gid` value of `14304:14304` instead of `1000:1000` for the oracle user running in the container that performs all container operations. If you are upgrading to the April 2022 release from an earlier release, the directory that is mapped to the container volume will be configured to have permission for user 1000:1000. Hence you must change the permission on this directory. The following are the scenarios to illustrate permission change.

Scenario 1: To continue using user `1000:1000`

To continue using the current user with `1000:1000`, you must use the run time option `-user 1000:1000` for starting new containers . No permission changes are required on the directory. For more information, see Installing Oracle RADIUS Agent.

Scenario 2: Switching to a new default user of `14304:14304`

To use the new default user of `14304:14304`, the directory that is mapped to the container volume must have read and write permissions for both users `1000:1000` and `14304:14304` until all Oracle RADIUS Agent instances in the cluster are upgraded. These required permissions on this directory must be set recursively. Once all instances are upgraded to April 2022 release, permissions on this directory for `1000:1000` user is no longer required and can be removed.

Scenario 3: Switching to a different non-default user

To use a different non-default user, the directory that is mapped to the container volume must have read and write permissions for both users `1000:1000` and the custom user until all Oracle RADIUS Agent instances in the cluster are upgraded. These required permissions on this directory must be set recursively. You must use the run time option `-user your_uid:your_gid` while starting a new container. Once all instances are upgraded to April 2022 release, permissions on this directory for `1000:1000` user is no longer required and can be removed.