On-prem Connector for Generic Databases

Okta Identity Governance

Deployment Guide

Product Acceleration Team

Overview

This guide provides comprehensive instructions for integrating Okta's On-prem Connector for Generic Databases with Java-compliant database systems. The On-prem Connector for Generic Databases leverages Okta's On-prem SCIM Server to perform operations on various database systems. This guide covers the architecture, system requirements, prerequisites, and detailed steps for deploying and integrating On-prem Connector for Generic Databases with the supported Java-compliant database systems to support user lifecycle management and entitlement management. Additionally, it outlines testing procedures and troubleshooting tips to ensure a successful implementation and ongoing management of user lifecycle within your on-prem Database environment through Okta.

Version Control:

Version	Date	Notes
1.0	Aug' 25	Initial version
1.1	Sep'25	Includes Custom Code scenarios
1.2	Oct'25	Includes Stored Procedures
1.3	Nov'25	Includes MS SQL and MySQL
1.4	Mar'26	Includes IBM DB2

Introduction

On-prem Connector for Generic Databases provides an out-of-the-box solution that connects on-premises databases with Okta Identity Governance. It enables the discovery, visibility, and management of users and entitlements directly within Okta. This integration enhances security, saves time, and simplifies governance by eliminating the need for custom integrations and streamlining user and entitlement management.

The integration of on-prem Database Systems with Okta Platform involves two different resources, which are

1. **On-prem Connector for Generic Databases:** It is an application within the Okta Integration Network, which provides interfaces and functionalities for managing the Identity Lifecycle and Identity Governance operations.
2. **Okta On-prem SCIM Server**: It is a software package deployed on a server within the same network as the on-prem database system. Its purpose is to facilitate CRUD operations for users and entitlements on database systems for Okta.

The On-prem Connector for Generic Databases leverages the power and capabilities of On-prem SCIM Server to achieve Identity Governance across supported on-premise Database Systems.

This document serves as a comprehensive guide for integrating on-prem database systems with Okta. Through this integration, organizations can effectively manage user lifecycle (LCM), provision entitlements, and leverage advanced OIG capabilities such as Access Request, Access Certification, and Segregation of Duties (SoD). This guide will walk you through the necessary steps to achieve a seamless connection and robust identity governance for your on-prem Database Systems.

Deployment Architecture

Okta On-prem SCIM Server and Okta Provisioning Agent work in conjunction to bridge the cloud-based Okta identity platform with your on-premise Database systems, enabling seamless identity governance. These are two primary deployment patterns:

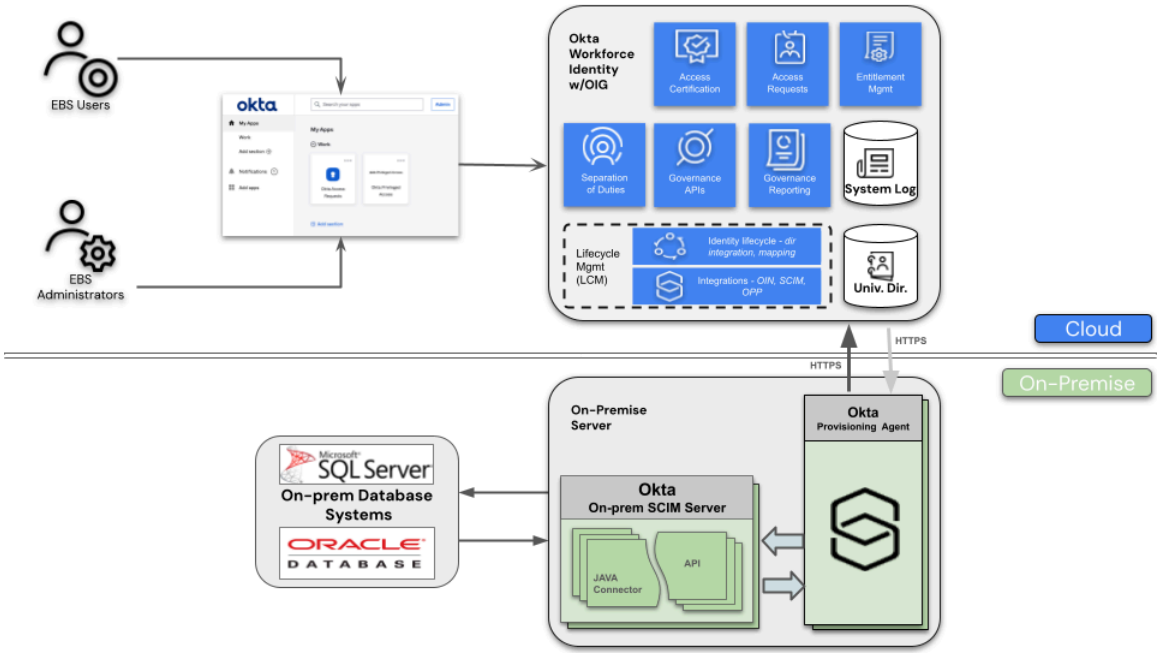
1. **Co-located Deployment**: Where the Okta Provisioning Agent and the Okta On-prem SCIM Server reside on the same server.
2. **Distributed Deployment**: Where the Okta Provisioning Agentand the On-prem SCIM Server resides on different servers.

Note: In this doc we will be using the co-located deployment strategy to deploy, where Okta Provisioning Agent and Okta On-prem SCIM Server will be deployed on same server.

Pattern 1: Co-located Deployment

How it Works:

1. **Okta to Okta Provisioning Agent**: The Okta Provisioning Agent establishes an outbound HTTP connection to the Okta cloud service, using the long polling mechanism. This connection is initiated from *within* your firewall, ensuring security as it's an outbound connection.
2. **Okta Provisioning Agent to Okta On-prem SCIM Server:** Once the Okta Provisioning Agent receives a provisioning or lifecycle management request from Okta, it forwards this request to the co-located Okta On-prem Connector for Generic Databases.
3. **Okta On-prem SCIM Server to Database System**: The Okta On-prem Connector for Generic Databases then translates the request into database operations and connects directly to the On-prem Database Systems. It performs the necessary actions, whether it's creating a user, updating attributes, deactivating an account, or syncing entitlements.
4. **Data Flow Back**: Results and import data flow back in the reverse direction: from Database to Okta On-prem SCIM Server, then Okta On-prem SCIM Server to Okta Provisioning Agent, and finally Okta Provisioning Agent securely transmits this information back to Okta.



Pattern 2: Distributed Deployment

How it Works:

1. **Okta to Okta Provisioning Agent:** Similar to the co-located model, the Okta Provisioning Agent on Server A establishes an outbound connection with Okta via HTTPS (port 443).
2. **Okta Provisioning Agent to Okta On-prem SCIM Server:** The Okta Provisioning Agent on Server A forwards provisioning requests to the Okta On-prem SCIM Server on Server B. This is where the crucial difference lies: this communication now travels over your internal network via TCP port 1443. This requires explicit firewall rules between Server A and Server B.
3. **Okta On-prem SCIM Server to Database System**: The Okta On-prem SCIM Server on Server B connects directly to the Database System, performing the necessary operations.
4. **Data Flow Back**: Data flows back from Database System to Okta On-prem SCIM Server on Server B, then to Okta Provisioning Agent on Server A, and finally back to Okta.

Supported Features

Okta's On-prem Connector for Generic Databases, by utilising the functionalities of On-prem SCIM Server, can help to achieve the following use cases with on-premise Database Systems.

❖ **Lifecycle Management**

It automates joiner, mover, and leaver processes within on-premise environments, enhancing efficiency and security through the utilization of Okta On-prem SCIM Server. Furthermore, this solution centralizes entitlement management within the Okta platform.

The On-prem Connector for Generic Databases provides the following user lifecycle management capabilities. These capabilities are controllable via the following functions, which can be enabled or disabled from the console.

- **Create User**

  It enables the creation of users in the on-premise database systems on the runtime when a user is assigned to the On-prem Connector for Generic Databases application on the Okta Platform.
- **Update User**

It enables the system to propagate any modifications made to a user's profile on Okta Platform to the on-premise database system.

- **Activate User**
  It allows organizations to set the state of the user account in the on-premise database system to Active.
- **Deactivate User**
  It allows organizations to set the state of the user account in the on-premise database system to Inactive.
- **Full Imports**
  It allows organizations to perform a complete import of all the users present in the on-premise database systems into Okta Platform.
- **Scheduled Imports**
  It allows organizations to schedule imports on given time intervals, to bring in users from on-premise database systems into Okta Platform.

❖ Entitlement Management

The On-prem Connector for Generic Databases provides additional capabilities to manage and govern entitlements, enabling least privileged access and automating governance decisions across on-prem databases with seamless LCM integration. These capabilities are controllable via the following functions, which can be enabled or disabled from the console.

- **Add entitlement to user**
  It allows organizations to assign entitlements to users from within Okta Platform and propagate those associations(user-entitlements) to on-premise database systems.
- **Remove entitlement from user**
  It allows organizations to remove entitlements assigned to a user from Okta Platform and propagate those updated associations to on-premise database systems.

❖ Access Request

It automates the process of requesting access to applications and entitlement bundles, which uses an on-prem database as their user repository. It further provides a simplified and frictionless approach that automatically routes user requests to one or more approvers for action.

❖ Access Certifications

It allows you to periodically identify and review users who have access to your critical resources. This ensures that only users who need a resource have access to it and avoid accumulation of elevated or privileged access to a resource.

❖ **Separation of Duties**

It enables defining rules that allow (with or without additional oversight) or block specific entitlement combinations for the database based endpoint.

Supported System

Okta On-prem Connector for Generic Databases provides integration capabilities for below database systems:

1. Oracle Database
2. Microsoft SQL Server
3. MySQL
4. PostgreSQL
5. IBM DB2

System Requirements

The system requirements for deploying Okta's On-prem Scim Server are described below.

Operating System

- RedHat Enterprise Linux - 8, 9 & 10

Hardware

- Processor: 4 or more cores
- Memory: 4 GB RAM (This is enough from a PoC perspective but for production instances, more memory is required.)
- Storage: 10 GB of minimum physical storage is required

Network

The table below provides the required ports which are used by Okta Provisioning Agent and Okta On-prem SCIM Server.

Port	Source	Destination	Description
443	Okta Provisioning Agent	Okta	This outbound connection from the Okta Provisioning Agent to Okta is essential for internet connectivity, enabling it to perform long polling to Okta.

1443	Okta Provisioning Agent	Okta On-prem SCIM Server	Required only when the Okta On-prem SCIM Server and Okta Provisioning Agent are deployed on separate servers.
1521	Okta On-prem SCIM Server	Oracle DB Server	This connection is to send SQL queries to the database for various CRUD operations. **(Subject to customer's environment)**
1433	Okta On-prem SCIM Server	MS SQL Server	This connection is to send SQL queries to the database for various CRUD operations. **(Subject to customer's environment)**
3306	Okta On-prem SCIM Server	MySQL Server	This connection is to send SQL queries to the database for various CRUD operations. **(Subject to customer's environment)**
5432	Okta On-prem SCIM Server	PostgreSQL Server	This connection is to send SQL queries to the database for various CRUD operations. **(Subject to customer's environment)**
5000 /2500	Okta Provisioning Agent	IBM DB2 Server	This connection is to send SQL queries to the database for various CRUD operations. **(Subject to customer's environment)**

Prerequisites

Before beginning the Okta On-prem Connector for Generic Databases deployment, ensure the following prerequisites are met:

- **Linux Server**
  A dedicated Linux server is required for the installation and operation of the Okta Provisioning Agent and Okta On-prem SCIM Server.

- **Okta On-prem SCIM Server**

Version 1.5.0 or later of Okta On-prem SCIM Server is required for the On-prem Connector for Generic Databases.

- **Okta Provisioning Agent**
  Version 3.0.6 or later of Okta Provisioning Agent is required for the On-prem Connector for Generic Databases.

- **Java Installation**
  Ensure that Java Development Kit (JDK) **version 21 or later** is installed on the designated Linux server.

- **Java Database Connectivity (JDBC) Driver**
  Ensure that the relevant JDBC driver required for the specific Database being used is on the designated Linux server.

  Note: The version of the JDBC Driver depends on the version of the respective database version.

- **OpenSSL**
  OpenSSL **version 3 or later**, is required for the deployment of Okta On-prem SCIM Server.

- **Feature Flags**
  There are few feature flags which are required to be enabled for deploying and using the Okta On-prem Connector for Generic Databases in your Okta Org. Ensure the below flags have been enabled:

  - **OPP Agent with SCIM 2.0 Support**
    It enables Okta Org to use the latest OPP Agent with support for SCIM 2.0 APIs and Entitlement Management for app integration.
  - **On-prem Connector for Generic Databases**
    It enables Okta orgs to use On-prem Connector for Generic Databases to manage user lifecycle and entitlement management for their on-premise database systems.

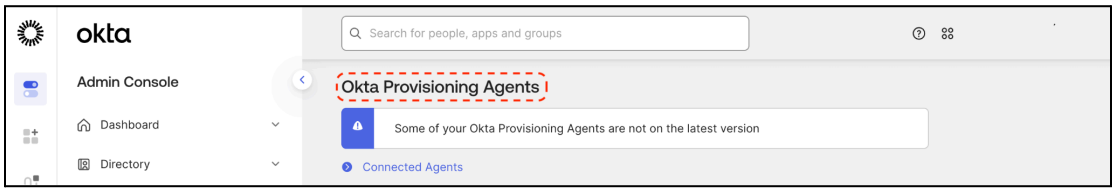
- **Secure Deployment with Okta**
  To ensure a secure and successful deployment, it is crucial to configure your network and firewall to permit communication with Okta's services. This guide details the specific IP ranges that require whitelisting for seamless integration and communication. For the official Okta documentation regarding allowed IP addresses, please refer to this **ARTICLE**.

Before you begin

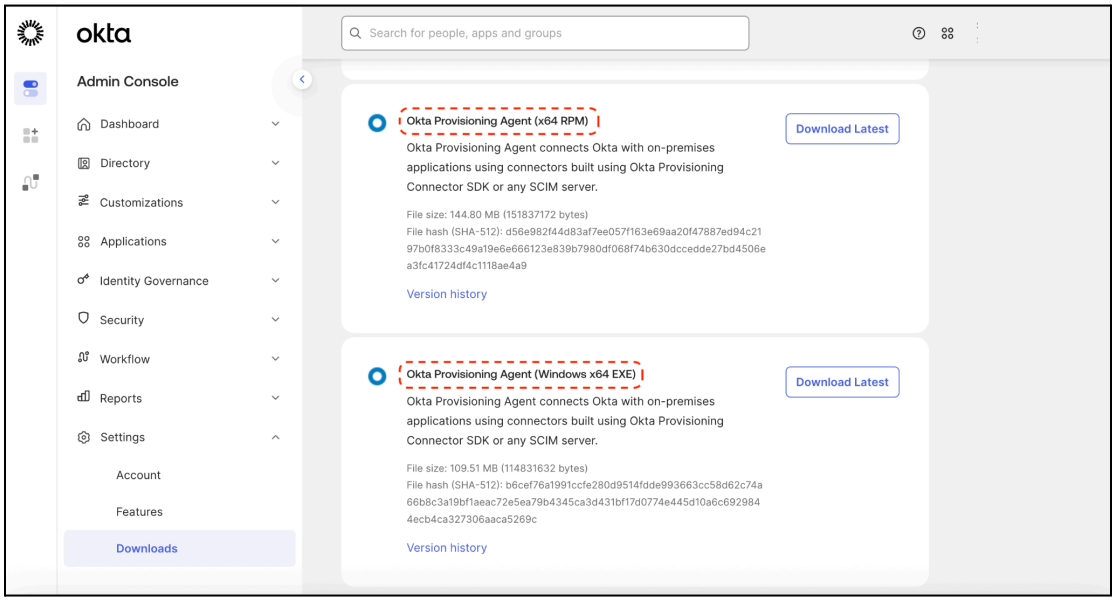
Download Okta Provisioning Agent

The Okta Provisioning Agent installer can be downloaded from the Okta Admin console.

1. Logon to Okta Admin console and navigate to Settings, and then Downloads.
2. Scroll down and locate the Okta Provisioning Agent section.



3. Select the appropriate Okta Provisioning Agent as per your server's operating system.



4. Create a directory, on the linux server as mentioned in pre-requisites, to place all the installable files.
   For e.g. Directory /installers have been created for this.

```
None
[user@ip-X-X-X-X ~]$cd /
[user@ip-X-X-X-X /]$ mkdir /installers/
```

5. Upload the downloaded Okta Provisioning Agent rpm file to host Linux Server, where the agent will be installed.

```
None
user.test@localsystem keys % scp -i <key.pem>
~/Downloads/OktaProvisioningAgent-XXXXX.x86_64.rpm user@X.X.X.X:/installers/
```

6. Verify that the file has been successfully uploaded to the mentioned directory

```
None
[user@ip-X-X-X-X /]$ cd /installers/
[user@ip-X-X-X-X installers]$ ls -ltr
-rw-rw-rw- 1 user user 151837172 Jul 29 12:47
OktaProvisioningAgent-XXXXX.x86_64.rpm
```

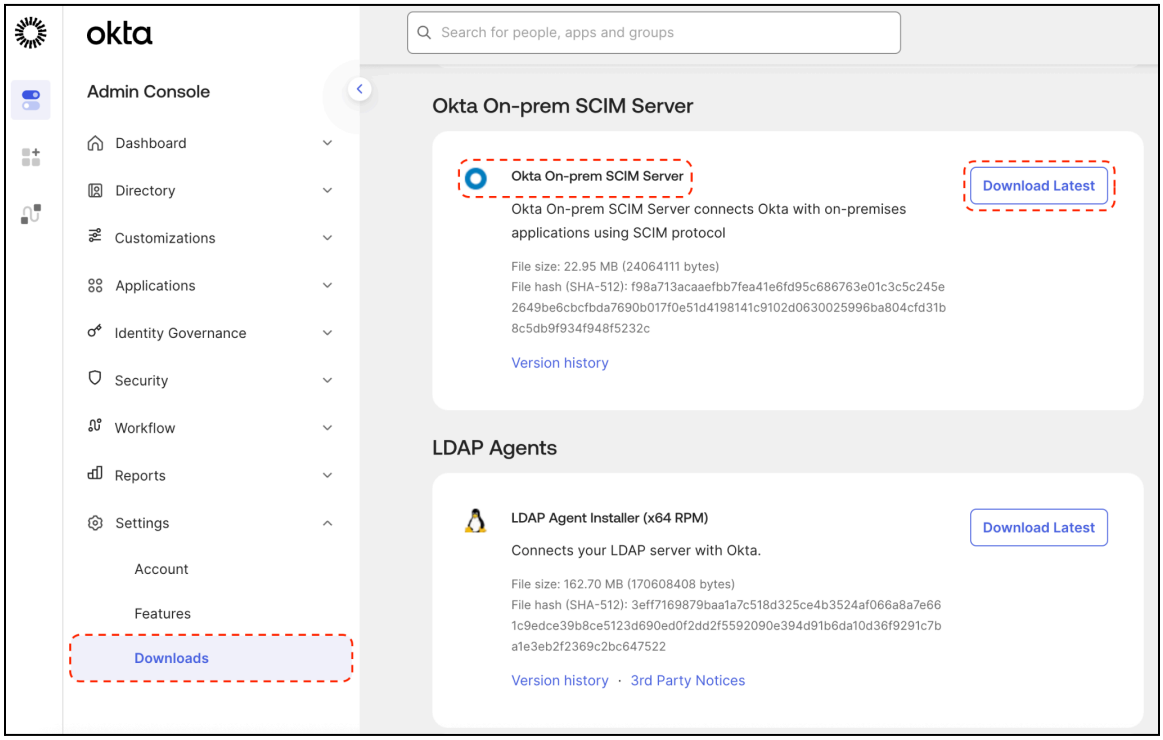
7. Update the permission of the rpm file using the below command.

```
None
[user@ip-X-X-X-X installers]$ chmod 777 OktaProvisioningAgent*
[user@ip-X-X-X-X installers]$ ls -ltr
-rwxrwxrwx 1 user user 151837172 Jul 29 12:47
OktaProvisioningAgent-XXXXX.x86_64.rpm
```

Download Okta On-prem SCIM Server

1. Navigate to Settings, and then Downloads.

2. Search for the Okta On-prem SCIM Server section and select Download Latest.



3. Upload the downloaded file to the designated Linux Server, where On-prem SCIM Server will be installed.
4. Use the directory created in the above section to place the file on the linux server. For e.g. Directory "**/installers**" has been created for this.

```
None
user.test@localsystem keys % scp -i <key.pem>
~/Downloads/OktaOnPremScimServer-XXXXX.rpm  user@X.X.X.X:/installers/
```

Download JDK and JDBC Packages

Okta On-prem SCIM Server requires the Java Development Kit version 21 and Oracle Java Database Connectivity files to be present on the server.

Java Development Kit

To begin with, download the JDK version 21 from here for the appropriate operating system running on the server and place it on the server following below steps:
(Note: The directory created in the step 4 of this section, will be used to place this files on the server)

1. Upload the JDK file to the host server.

```
None
user.test@localsystem keys % scp -i <key.pem>
~/Downloads/jdk-21.0.7_linux-x64_bin.rpm user@X.X.X.X:/installers/
```

2. Verify that the file has been successfully uploaded to the mentioned directory

```
None
[user@ip-X-X-X-X /]$ cd /installers/
[user@ip-X-X-X-X installers]$ ls -ltr
-rw-rw-rw- 1 user user 151837172 Jul 29 12:47 jdk-21.0.7_linux-x64_bin.rpm
```

3. Update permissions for the file.

```
None
[user@ip-X-X-X-X installers]$ chmod 777 jdk-21.0.7_linux-x64_bin.rpm
[user@ip-X-X-X-X installers]$ ls -ltr
-rwxrwxrwx 1 user user 151837172 Jul 29 12:47 jdk-21.0.7_linux-x64_bin.rpm
```

Java Database Connectivity Packages

To facilitate the functionality of the On-prem SCIM Server with various database systems, the requisite JDBC (Java Database Connectivity) packages for each respective database system are essential. These JDBC packages enable the On-prem SCIM Server to establish connectivity with the database systems and execute subsequent operations. This document will detail the specific JDBC packages for different database systems, including guidance on their download and implementation. Follow the steps for the respective database system.

Oracle Database

Oracle Database Connectivity (OJDBC)

The Oracle Database Connectivity (OJDBC) package is specifically designed for seamless integration with Oracle Database systems. As highlighted in this document, the

OJDBC package (version 21 is required) enables the Okta On-prem SCIM Server to establish robust connections and execute database operations on your Oracle Database.

1. Download the OJDBC version 21 driver from the internet or the maven repository and place it on the designated Linux Server.
2. Upload the OJDBC jar file from the local system to the host Server.

```
None
user.test@localsystem keys % scp -i <key.pem>
~/Downloads/ojdbc11-21.5.0.0.jar user@X.X.X.X:/tmp/
```

3. Verify that the file has been successfully uploaded to the mentioned directory

```
None
[user@ip-X-X-X-X /]$ cd /tmp/
[user@ip-X-X-X-X tmp]$ cp /tmp/ojdbc11-21.5.0.0.jar /installers
[user@ip-X-X-X-X tmp]$ cd /installers
[user@ip-X-X-X-X installers]$ ls -ltr ojdbc*
-rw-rw-rw- 1 user user 151837172 Jul 29 12:47 ojdbc11-21.5.0.0.jar
```

MS SQL Server

MS SQL Server Java Database Connectivity (JDBC)

The MS SQL Server JDBC package facilitates robust and secure integration with Microsoft SQL Server database systems. As detailed in this document, the MS SQL Server JDBC package (version 12.10.0 is required) enables the Okta On-prem SCIM Server to establish reliable connections and execute various database operations on your MS SQL Server instance.

Steps to download the package:

1. Navigate to MS SQL Server's download site on the browser.
2. Once successful, it will download the package on your system.
3. Extract the files from the zip file and navigate to the sqljdbc_12.10/enu/jars directory.
4. Locate the mssql-jdbc-12.10.0.jre11.jar file.
5. Upload this file to the Linux Server.

```
None
user.test@localsystem keys % scp -i <key.pem>
~/Downloads/sqljdbc_12.10/enu/jars/mssql-jdbc-12.10.0.jre11.jar
user@X.X.X.X:/tmp/
```

6. Verify that the file has been successfully uploaded to the mentioned directory

```
None
[user@ip-X-X-X-X /]$ cd /tmp/
[user@ip-X-X-X-X tmp]$ cp /tmp/mssql-jdbc-12.10.0.jre11.jar /installers
[user@ip-X-X-X-X tmp]$ cd /installers
[user@ip-X-X-X-X installers]$ ls -ltr mssql*
-rw-rw-rw- 1 user user 102385764 Aug 29 12:47
mssql-jdbc-12.10.0.jre11.jar
```

MySQL

MySQL provides standards-based drivers for JDBC, also known as Connector/J, that facilitates connectivity with the MySQL Database system. As highlighted in this document, the Connector/J version 9.4.0 is required that enables the Okta On-prem SCIM Server to establish robust connections and execute database operations on your MySQL Database. Steps to download the package:

1. Navigate to MySQL Server's download site on the browser.
2. Select Platform Independent from the dropdown box under Select Operation System..
3. Extract the downloaded zip file and within the extracted files locate the "mysql-connector-j-9.4.0.jar".
4. Upload this file to the Linux Server.

```
None
user.test@localsystem keys % scp -i <key.pem>
~/Downloads/mysql-connector-j-9.4.0/mysql-connector-j-9.4.0.jar
user@X.X.X.X:/tmp/
```

5. Verify that the file has been successfully uploaded to the mentioned directory

```
None
[user@ip-X-X-X-X /]$ cd /tmp/
[user@ip-X-X-X-X tmp]$ cp /tmp/mysql-connector-j-9.4.0.jar /installers
[user@ip-X-X-X-X tmp]$ cd /installers
[user@ip-X-X-X-X installers]$ ls -ltr mysql*
-rw-rw-rw- 1 user user 234986109 Sep 21 10:13
mysql-connector-j-9.4.0.jar
```

PostgreSQL

Okta On-prem SCIM Server requires PostgreSQL JDBC Driver version 42.7.8 to connect to a PostgreSQL Database. This standard, database-independent Java code facilitates robust connections and execution of database operations.

Steps to download the package:

1. Navigate to PostgreSQL Server's download site on the browser.
6. Within the Java 8 box, click on **Download**. It will download the latest version of PostgreSQL's JDBC Driver. For e.g. postgresql-42.7.8.jar
7. Upload this file to the Linux Server.

```
None
user.test@localsystem keys % scp -i <key.pem>
~/Downloads/mysql-connector-j-9.4.0/postgresql-42.7.8.jar
user@X.X.X.X:/tmp/
```

8. Verify that the file has been successfully uploaded to the mentioned directory

```
None
[user@ip-X-X-X-X /]$ cd /tmp/
[user@ip-X-X-X-X tmp]$ cp /tmp/postgresql-42.7.8.jar /installers
[user@ip-X-X-X-X tmp]$ cd /installers
[user@ip-X-X-X-X installers]$ ls -ltr postgresql*
-rw-rw-rw- 1 user user 161325978 Sep 23 11:49 postgresql-42.7.8.jar
```

IBM DB2

Okta On-prem SCIM Server requires IBM DB2 JDBC Driver to connect Java programs to an Oracle Database. This standard, database-independent Java code facilitates robust connections and execution of database operations.

Steps to download the package:

1. Navigate to IBM DB2 Support page here.
2. Logon using your IBM Credentials to locate the JDBC Files.
3. Click on the download button for the required JDBC file.

4. Upload this file to the Linux Server.

```
None
[user.test@localsystem keys] % scp -i <key.pem>
~/Downloads/jcc-11.5.9.0.jar user@X.X.X.X:/tmp/
```

5. Verify that the file has been successfully uploaded to the mentioned directory

```
None
[user@ip-X-X-X-X /]$ cd /tmp/
[user@ip-X-X-X-X tmp]$ cp /tmp/jcc-11.5.9.0.jar /installers
[user@ip-X-X-X-X tmp]$ cd /installers
[user@ip-X-X-X-X installers]$ ls -ltr jcc*
-rw-rw-rw- 1 user user 161325978 Feb 23 11:49 jcc-11.5.9.0.jar
```

Collect Environment Details

To facilitate a smooth integration, gather details about your Okta Org and Oracle Database system beforehand. This information will be crucial throughout the deployment process.

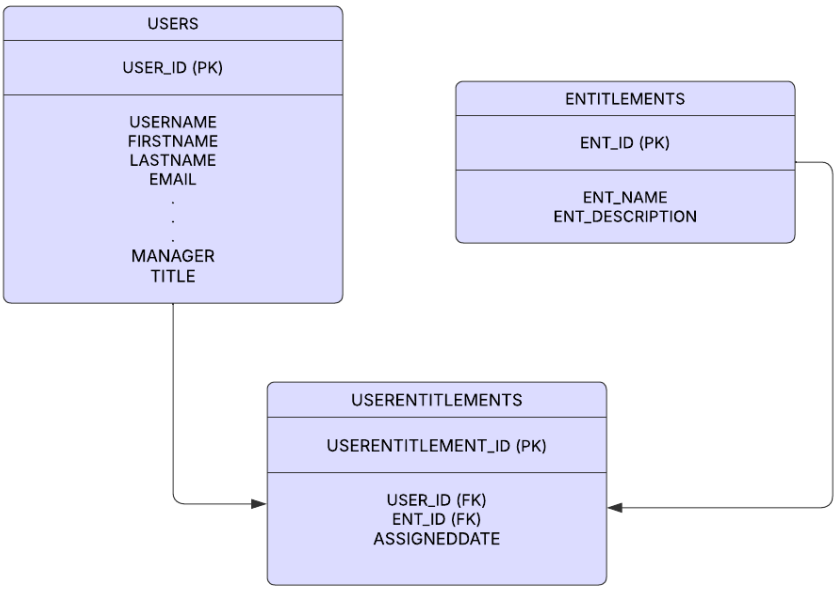
Okta Details

- **URL of Okta Org**
  It is the URL of your Okta org, which is something like https://mycompany.okta.com.
- **Admin Username**
  It is the Administrator's user in the Okta org, which has the privileges to create applications, integrate Okta Provisioning Agent and Okta On-prem SCIM Server.
- **Admin Password**
  It is the password for the above mentioned Administrator User in the Okta org.

Database System Details

Okta's On-prem Connector for Generic Databases will require below information to bind connection with the on-prem database systems and perform necessary operations on it. As it supports various database systems, collect the details for the respective database system which will be integrated using the On-prem SCIM Server.

Below is a sample database schema shown for reference purpose and it will be used to illustrate the integrations and configurations for the On-prem SCIM Server.



Note: The above database schema is only for illustration and reference purposes. Please refer to your Database system schema to configure the On-prem SCIM Server.

1. **Connection Details**
   It constitutes the details of the database system host as well as the database services and its associated objects, which will be used to make connection requests with the database system by Okta Platform.
   - **Username**
     The username of the database user with administrator privileges to execute SQL Queries.
   - **Password**
     The password for the above mentioned database user.
   - **Host Server IP/Domain Name**
     The IP Address or Domain Name of the server where the On-prem Database is deployed.
   - **Host Server Port**
     The port used by the database system for listening to incoming connections.
   - **Database Name**
     The name of the database instance which will be used for the connection to the database system.

2. **Database Column Details**
   The On-prem Connector for Generic Databases application leverages column names within your database tables to accurately identify and extract user and entitlement data. The information extracted from these columns in the database are mapped

against the user IDs, entitlement IDs and entitlement Values within Okta, thereby ensuring the seamless ingestion and storage of relevant information within the Okta Platform.

(Note: For illustration, we will be referring to the schema diagram shown for the database in the Prerequisite section and using the column names mentioned there.)

- **User ID Column**
  This field refers to the name of the column in the user information table that stores the user ID.
  For e.g. In this doc, we will be using the **User_ID** column from the **Users** table in the database schema mentioned above.
- **Entitlement ID Column**
  This field refers to the name of the column in the entitlement table that stores the entitlement's ID.
  For e.g. In this doc, we will be using the **ENT_ID** column from the **Entitlements** table of the database for this.
- **Entitlement Display Column**
  This field represents the name of the column in the entitlement table that stores the entitlement's name.
  For e.g. In this doc, we will be using the **ENT_Name** column from the **Entitlements** table of the database for this.

Construct Database Query

Okta's On-prem Connector for Generic Databases offers three distinct methodologies for importing or provisioning user and entitlement data to and from the database system.

1. SQL Statements
2. Stored Procedures
3. Custom Code

Operations	SQL Statements	Stored Procedures	Custom Code
Discovery & Import from Database System	✅	✅	
Provisioning to Database System	✅	✅	✅

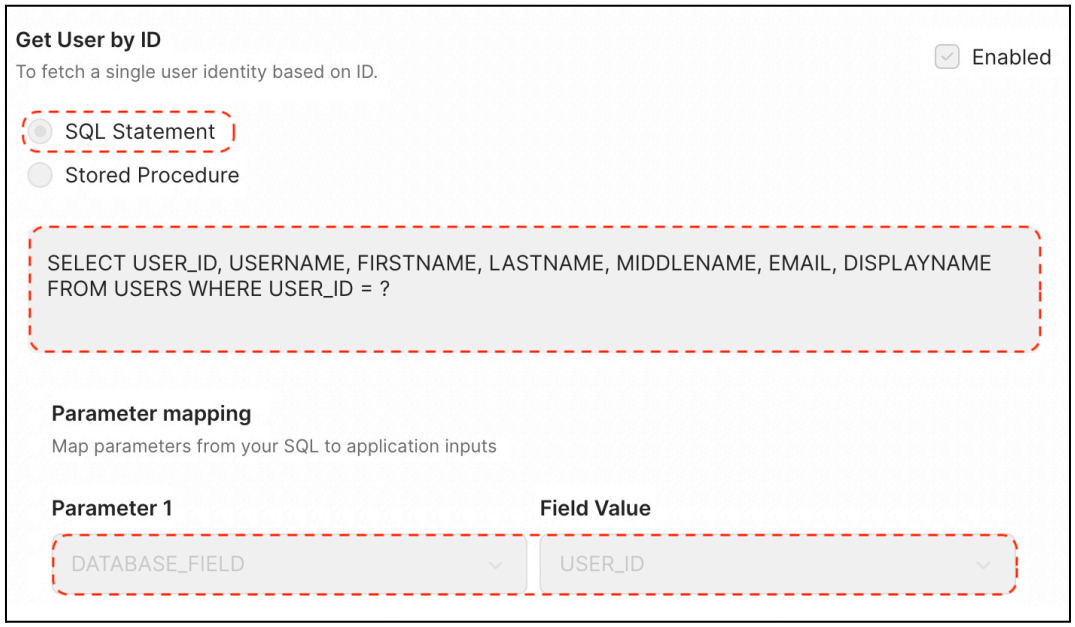
(This provides a comprehensive overview of the operations and their supported methods.)

SQL Statements

It utilizes standard database commands—such as SELECT, INSERT, UPDATE, and DELETE—to perform direct operations on your data tables. The Okta On-prem SCIM Server leverages these custom statements to facilitate both the import of existing user and entitlement data, as well as the propagation of provisioning changes (like creating or updating users) directly to the target database.

To accommodate dynamic provisioning requirements, these SQL statements are constructed at runtime using **Placeholders**. Represented by a question mark ('?'), these placeholders act as variables within the query, allowing the connector to inject context-specific data—such as a user's unique ID or current department—into the command before it is executed.

As illustrated in the configuration, each placeholder corresponds to an entry in the **Parameter Mapping** section. This feature provides the flexibility to map each variable to either a dynamic **Database Field** (drawing from a specific user attribute) or a **Constant** value for static inputs. During execution, the connector automatically substitutes these placeholders with the mapped values to construct and run the final, precise SQL command.



For e.g. if there is an operation being carried out to fetch the information for a user with user_id as "john.doe@example.com", then the final SQL statement will look like below.

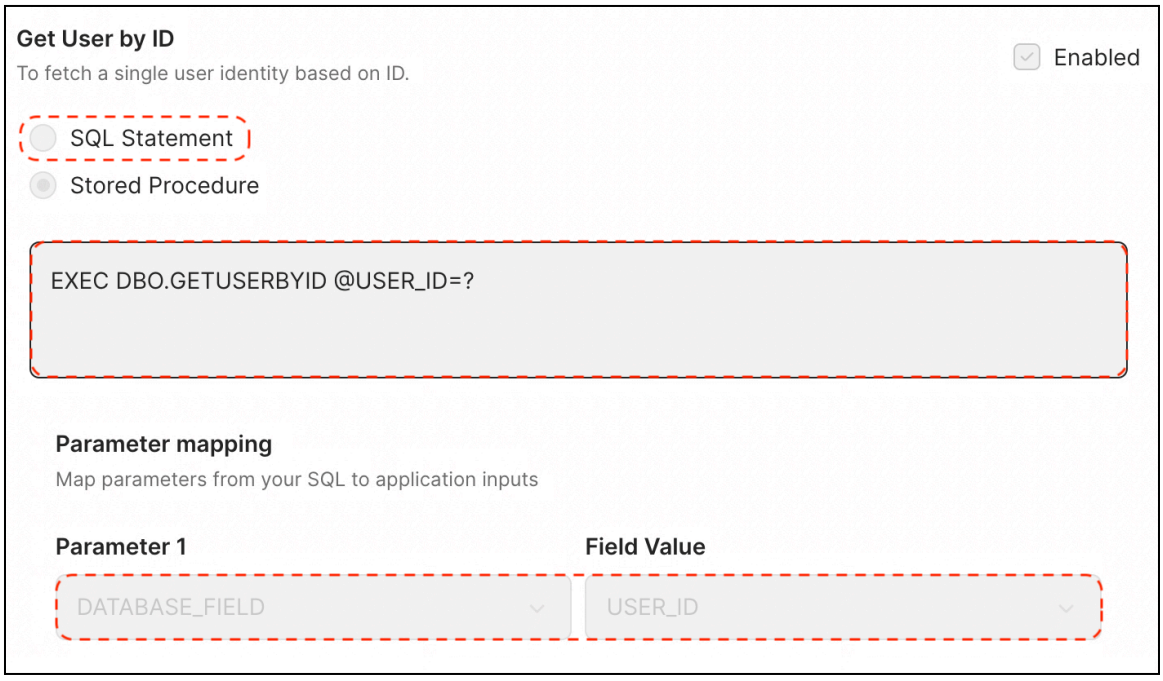
```
SQL
SELECT USER_ID, USERNAME, FIRSTNAME, LASTNAME, MIDDLENAME, HONORIFICPREFIX,
HONORIFICSUFFIX, EMAIL FROM USERS WHERE USER_ID='john.doe@example.com';
```

Stored Procedures

It is a set of pre-compiled SQL statements, saved under a name, and stored within a database, known as a stored procedure. Instead of repeatedly writing and sending complex queries to the database for specific tasks, you can simply invoke the stored procedure by its name. The Okta On-prem SCIM Server leverages stored procedures for both Import and Provisioning operations.

With reference to the below picture, the On-prem Connector for Generic Databases is capable of invoking stored procedures, such as `GETUSERBYID`. The placeholder '?' within the procedure call functions as a variable, configurable within Parameter Mapping, and can represent any column derived from the User or Entitlement Table. This variable is also capable of utilizing a **CURSOR** for parameter passing to the procedure.

To ensure precise data exchange, the connector requires you to define the specific data structure for these parameters, categorized as the 'Cursor Type'. For operations intended to return a complete dataset—such as importing a list of users—the **REFCURSOR** type is utilized, functioning as a direct pointer to the result set generated by the database. For passing or retrieving individual values, such as a user's email address, ID, or creation time, the connector supports standard scalar types including **VARCHAR**, **INTEGER**, **NUMERIC**, **DECIMAL**, **BOOLEAN**, **DATE**, and **TIMESTAMP**. Accurately mapping these types is critical, as it ensures the connector interprets the data exactly as defined by your stored procedure's signature.



Custom Code

Custom Code refers to custom Java code that is used to execute SQL operations on a database, particularly for complex logic. This method enables the provisioning of users and their associated entitlements, allowing for tailored interactions with the database beyond standard SQL statements or stored procedures.

In order to use the custom code option, the Java code needs to be developed to perform the required functions and built into JAR. The JAR file needs to be placed on the server and there are few details which need to be provided on the application interface on the admin console.

The details on how to configure and use the Custom Code option is presented in the Appendix.

Now, we have an understanding of different methods provided by Okta's On-prem Connector for Generic Databases for performing import and provisioning operations on the database systems. We will now explore how we can use these methods to perform the necessary operations to configure and achieve a robust User Lifecycle Management and Entitlement Management.

Integrate Database System with Okta

Okta offers a dedicated app for managing identities stored in an on-prem database system, within the Okta Integration Network (OIN), called as "**On-prem Connector for Generic Databases**". This app provides functionalities to manage User Lifecycle Management, Entitlement Management and Identity Governance for on-premise database systems, by utilizing the Okta On-prem SCIM Server.

As previously stated, the integration of the on-prem Database System with the Okta Platform necessitates the installation of both the Okta Provisioning Agent and the Okta On-prem SCIM Server on the host server. Therefore, we will commence with the installation of the Okta Provisioning Agent on the server.

Install Okta Provisioning Agent

1. Log on to the host server where the Okta Provisioning Agent installation RPM is located (Refer to location for details).
2. Navigate to the directory containing the Okta Provisioning Agent and JDK RPM.
3. Before installing the Okta Provisioning Agent, the JDK needs to be installed. Execute below command to install the JDK:

```
None
[user@ip-X-X-X-X /]$ cd /installers/
[user@ip-X-X-X-X installers]$ sudo yum localinstall jdk-21.0.7_linux-x64_bin.rpmm
Updating Subscription Management repositories.
.
.
.
Running transaction
Preparing        :
1/1    Installing       : jdk-21-2000:21.0.7-8.x86_64
1/1    Running scriptlet: jdk-21-2000:21.0.7-8.x86_64
1/1    Installed products updated.
Installed:
  jdk-21-2000:21.0.7-8.x86_64
Complete!
```

4. Once JDK has been installed successfully, install the Okta Provisioning Agent. Execute the RPM installation command as shown below. When prompted with "Is this ok?", enter "Y".

```
None
[user@ip-X-X-X-X installers]$ sudo yum localinstall
OktaProvisioningAgent-XXXXX.x86_64.rpm
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Examining OktaProvisioningAgent-02.03.01-50e4237.x86_64.rpm:
OktaProvisioningAgent-02.03.01-SNAPSHOT20250630205201.x86_64
Marking OktaProvisioningAgent-02.03.01-50e4237.x86_64.rpm to be installed
Resolving Dependencies
--> Running transaction check
---> Package OktaProvisioningAgent.x86_64 0:02.03.01-SNAPSHOT20250630205201 will
be installed
Transaction Summary
================================================================================
================================================================================
================================================
Install  1 Package (+49 Dependent packages)
Total size: 204 M
Total download size: 11 M
Installed size: 231 M
Is this ok [y/d/N]: y
.
.
Installed:
```

```
OktaProvisioningAgent.x86_64-XXXXX-SNAPSHOT20250630205201
Complete!
[user@ip-X-X-X-X install]$
```

Note: Okta Provisioning Agent requires that you enable version 1.2 of the Transport Layer Security (TLS) protocol.

5. Edit /opt/OktaProvisioningAgent/conf/settings.conf, to update the TLS configurations.

```
None
[user@ip-X-X-X-X installer]$ vi /opt/OktaProvisioningAgent/conf/settings.conf
```

6. Change the arguments passed to the agent by adding **Dhttps.protocols=TLSv1.2** to **JAVA_OPTS**, as shown below, and save the settings.conf file.

```
None
# Arguments passed to the agent JVM
JAVA_OPTS="-Xmx4096m -Dhttps.protocols=TLSv1.2"
JAVA_TOOL_OPTIONS="-Dfile.encoding=UTF8"
```

Note: After TLS configuration has been updated, the agent will be configured to connect with Okta Org.

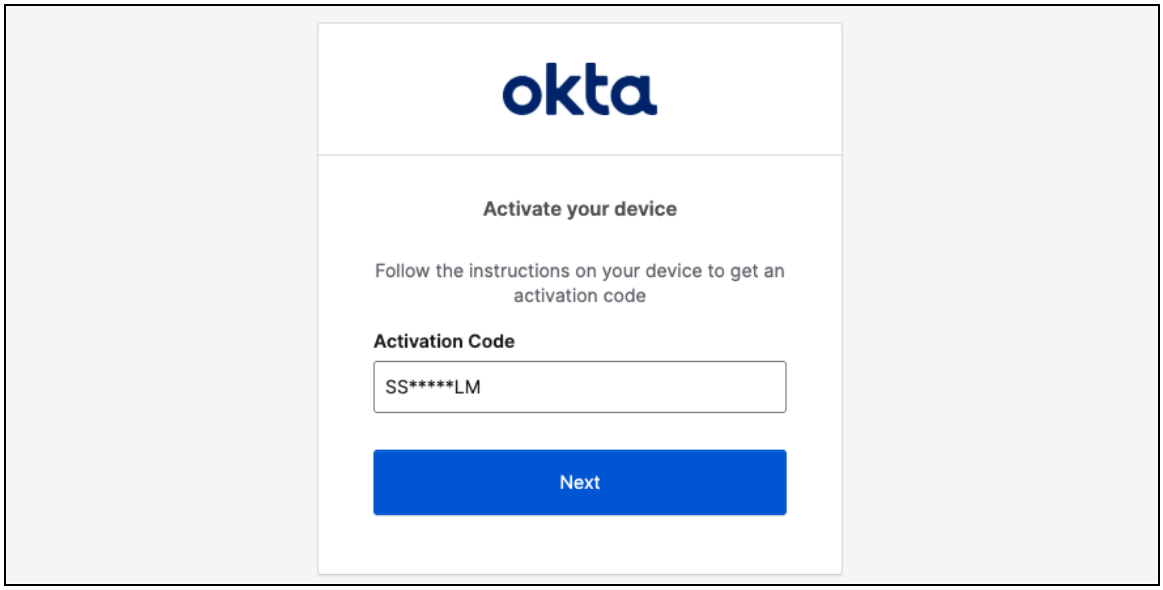
7. Execute the following script /opt/OktaProvisioningAgent/configure_agent.sh and enter the URL of your org at the prompt.

```
None
[user@ip-X-X-X-X install]$ sudo /opt/OktaProvisioningAgent/configure_agent.sh
Welcome to the Okta Provisioning Agent configuration script. This needs to be run
once after the installation to populate required application settings.
Enter the URL of your org. For example: https://mycompany.okta.com
https://test-org.okta.com
Enable proxy (y/n)? [n]: n
```

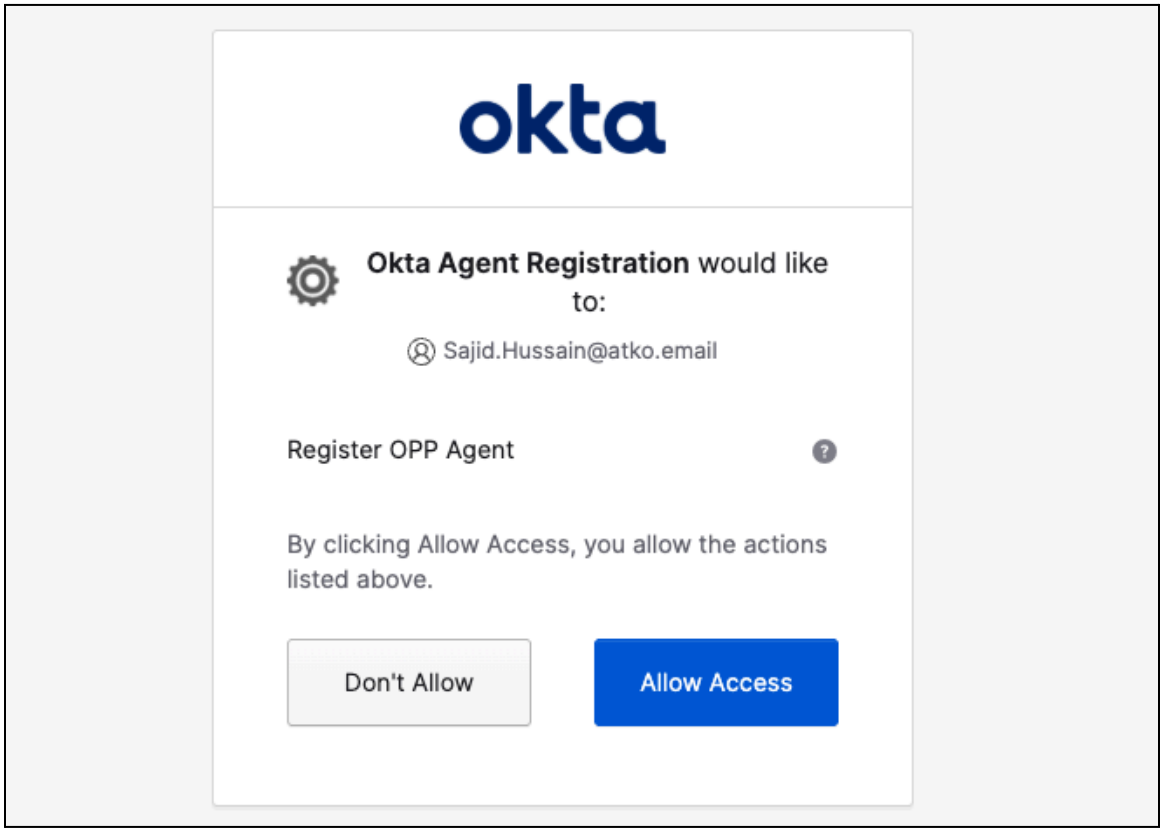
8. On successful execution of the above command, it will provide an activation url which needs to be executed from your browser as well as the activation code.

```
None
[ 2025-09-26 06:16:53.120 ] [ main ] [OppAgentConfigLoader] [  ] [INFO] - To continue, navigate to the following website: [ https://shcorp.oktapreview.com/activate ] and enter the following code: [ SS****LM ]
[ 2025-09-26 06:16:58.419 ] [ main ] [OppAgentRestClient] [  ] [INFO] - OAuth2 Token request is pending for authorization: authorization_pending
```

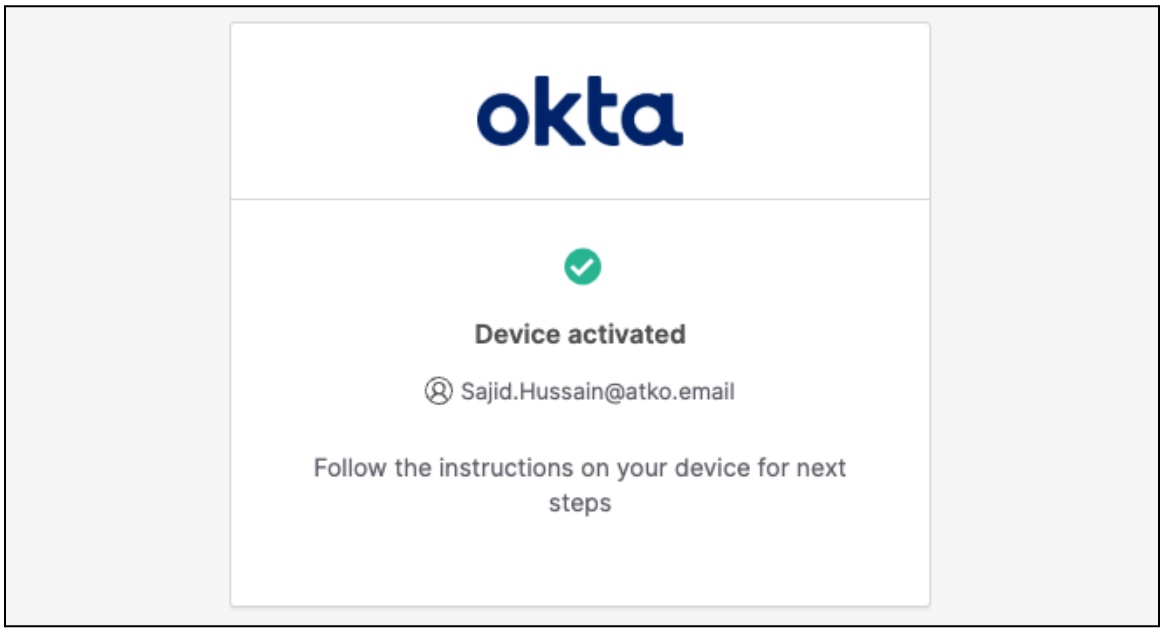
9. On the browser enter the activation url and then provide the Activation Code.



10. To register the Okta Provisioning Agent with your Okta Org, click Allow Access.



11. On successful activation, it will show a message as Device Activated.



12. Now, return to the host server, where you can see the below message after successful configuration.

```
None
Configuration successful.
Service can now be started by typing
systemctl start OktaProvisioningAgent.service
as root.
```

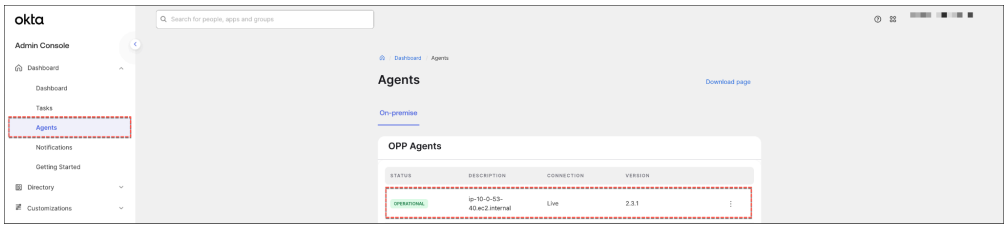
13. Return to the command line and enter the command

```
None
[user@ip-X-X-X-X install]$ sudo systemctl start OktaProvisioningAgent.service
```

14. To confirm that the Okta Provisioning Agent is running, enter the following:

```
None
[user@ip-X-X-X-X install]$ sudo systemctl status OktaProvisioningAgent
● OktaProvisioningAgent.service - SYSV: OktaProvisioningAgent Agent server daemon
   Loaded: loaded (/etc/rc.d/init.d/OktaProvisioningAgent; bad; vendor preset:
disabled)
   Active: active (running) since Tue 2025-07-29 13:25:11 UTC; 1min 58s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 32623 ExecStart=/etc/rc.d/init.d/OktaProvisioningAgent start
(code=exited, status=0/SUCCESS)
 Main PID: 32643 (java)
   CGroup: /system.slice/OktaProvisioningAgent.service
           └─32643 /opt/OktaProvisioningAgent/jre/bin/java
-Dagent_home=/opt/OktaProvisioningAgent -Xmx4096m -Dhttps.protocols=TLSv1.2 -jar
/opt/OktaProvisioningAgent/bin/OktaProvisioningAgent.jar -mode normal -...
Jul 29 13:25:09 ip-X-X-X-X.ec2.internal systemd[1]: Starting SYSV:
OktaProvisioningAgent Agent server daemon...
Jul 29 13:25:09 ip-X-X-X-X.ec2.internal runuser[32632]: pam_unix(runuser:session):
session opened for user provisioningagent by (uid=0)
Jul 29 13:25:11 ip-X-X-X-X.ec2.internal OktaProvisioningAgent[32623]: Starting
OktaProvisioningAgent:[  OK  ]
Jul 29 13:25:11 ip-X-X-X-X.ec2.internal systemd[1]: Started SYSV:
OktaProvisioningAgent Agent server daemon.
```

15. We can also check the status of the Okta Provisioning Agent in the Okta Admin Console.



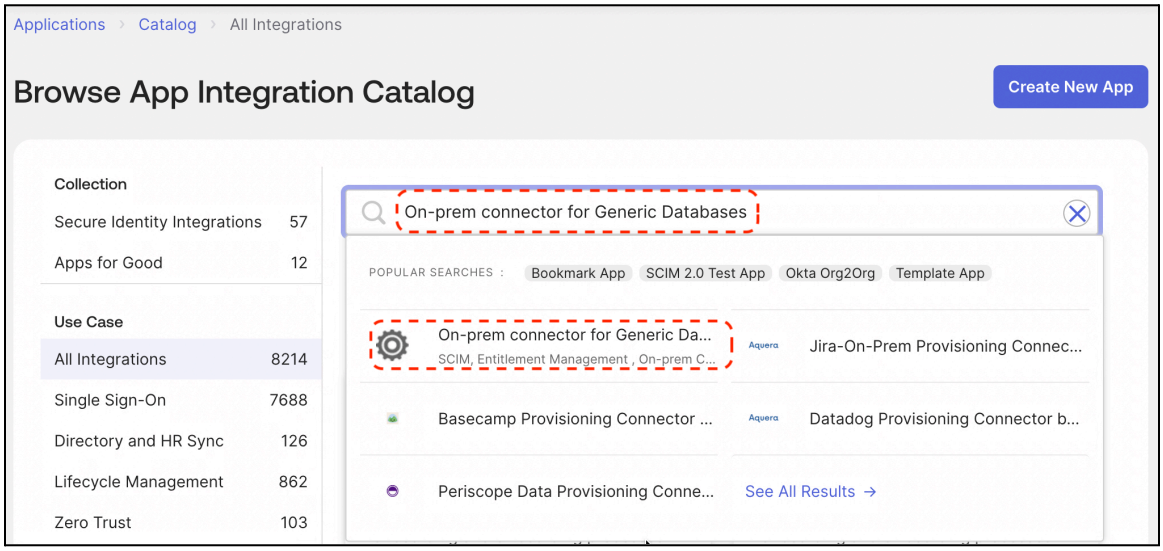
16. Verify that the agent is up and running in the Okta Admin Console, by navigating to Dashboard > Agents.

Now, the Okta Provisioning Agent has been installed and configured successfully with Okta. The next step is to create an **On-prem Connector for Generic Databases** application from Okta Integration Network (OIN).

Create On-prem Connector for Generic Databases application

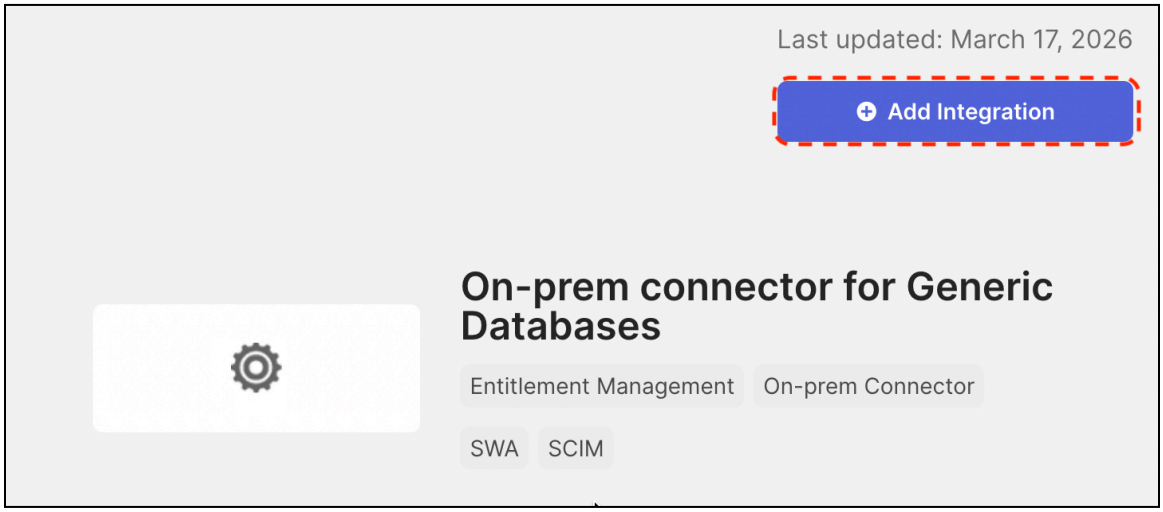
To create the **On-prem Connector for Generic Databases** application in your Okta org, perform the following

1. Logon to Okta Admin Console.
2. Navigate to Applications, select Browse App Catalog.
3. In the search box, type in On-prem Connector for Generic Databases and it would provide the list of database applications, select On-prem Connector for Generic Databases.

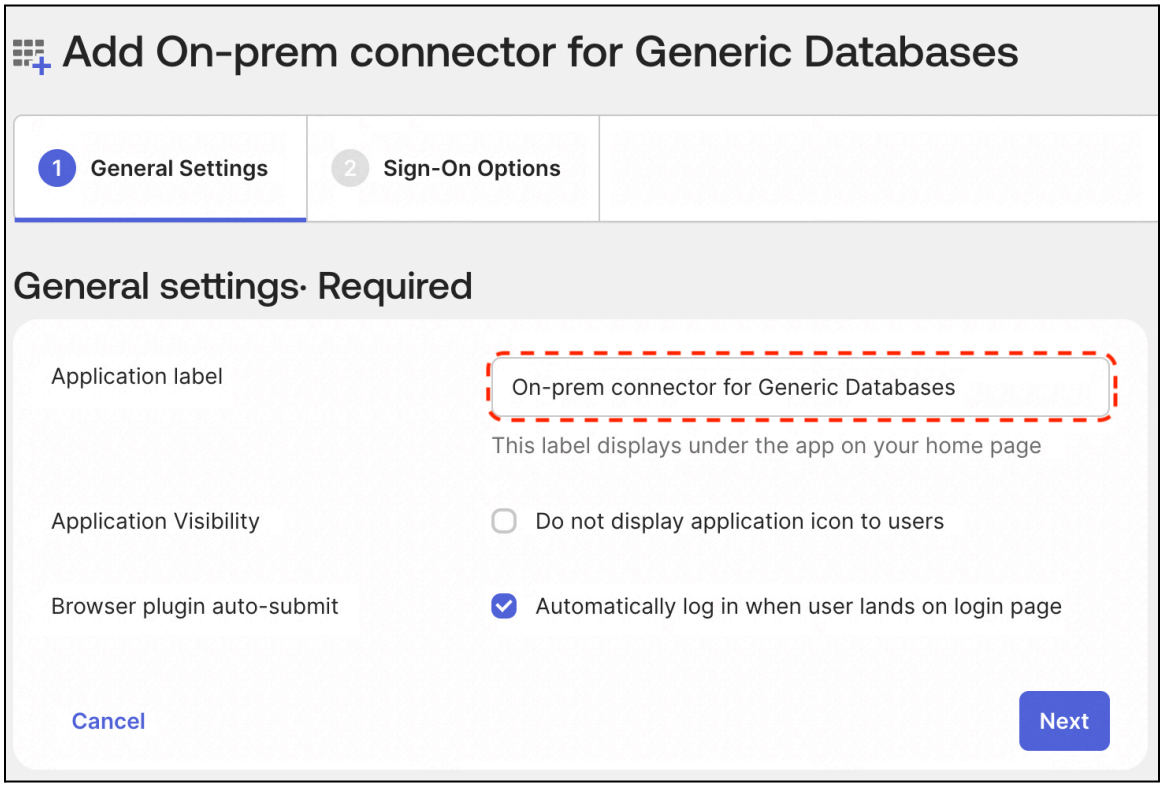


Note: If the application Generic Database Connector is not being shown in App Integration Catalogue then please check the necessary Feature Flags.

4. Select the Add Integration in the Application console.



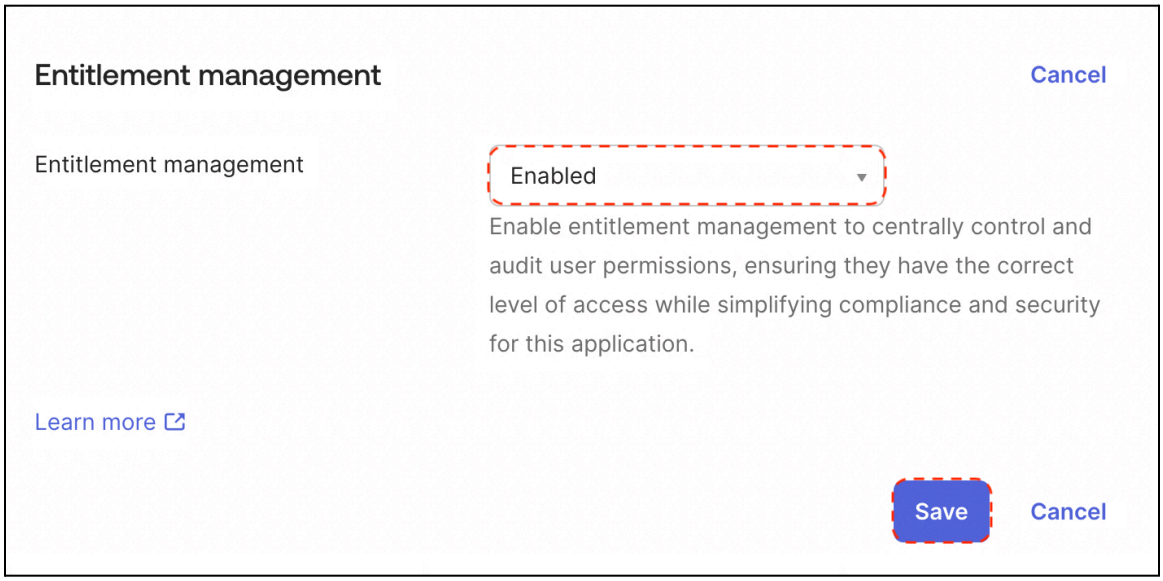
5. Provide a name for the application in the Application Label section.
   By default it provides a name as “On-prem Connector for Generic Databases”



6. Select next and leave the fields to default.
7. Select Done.

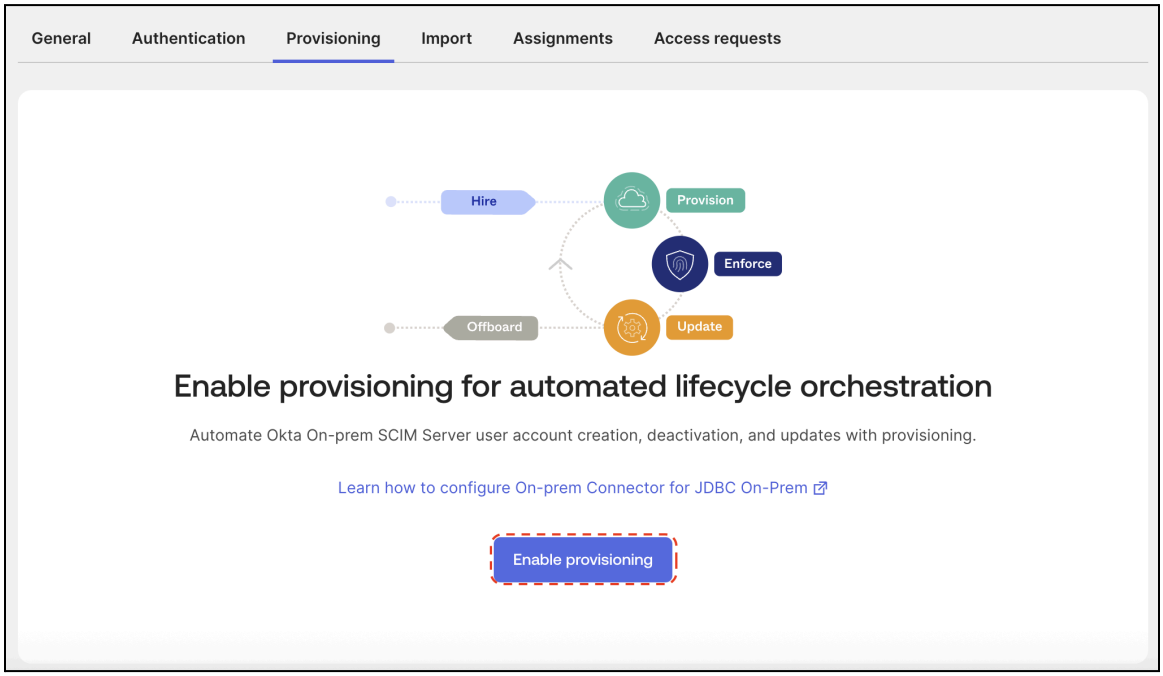
Note: Provisioning for this application necessitates the prior enablement of the Identity Governance engine.

8. In the **General** section, scroll down to the Entitlement Management section, and select **Edit**. From the drop down menu select **Enable** and then select **Save**.



(Once the entitlement management has been enabled, you can notice within Governance tab for the application, that a few sub tabs such as Entitlements, Bundles, etc are present now)

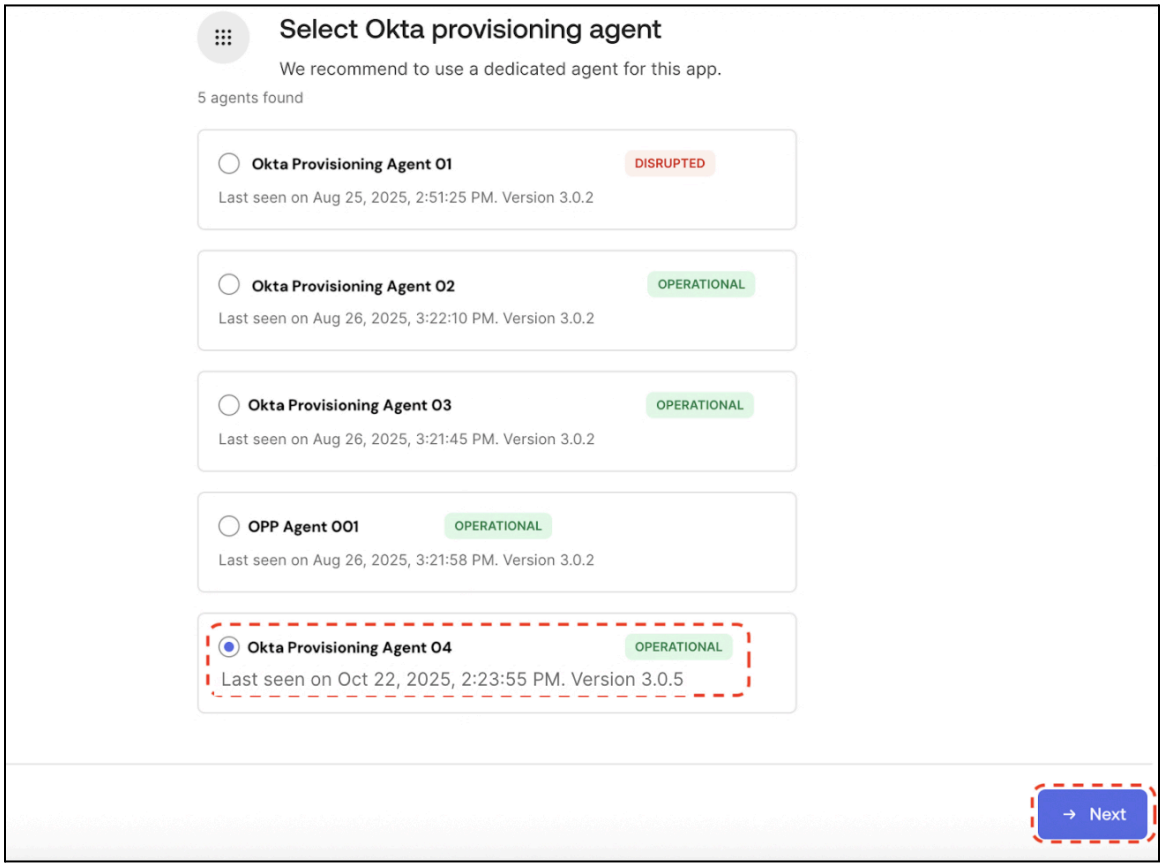
9. Within the same application, select the **Provisioning** tab.
10. Select **Enable Provisioning**.



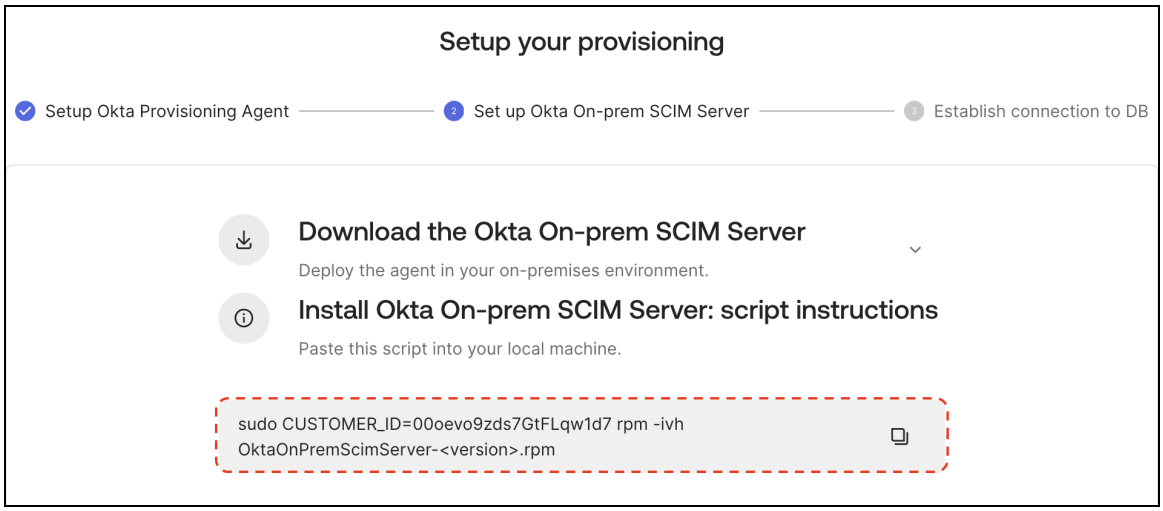
(Once enabled, it provides the option to Download the Okta Provisioning Agent for Windows and Linux.)

> **Note:** Okta Provisioning Agent has already been downloaded and installed on the server. So, in this document we won't be downloading it from here.

11. Select the required Okta Provisioning Agent from the list of agents.
    (This page displays the list of available Okta Provisioning Agents, both active and inactive, within the Okta Org.)



12. Select **Next**.
13. It brings you to the **Okta On-prem SCIM Server** section, take a note of the execution **command**.

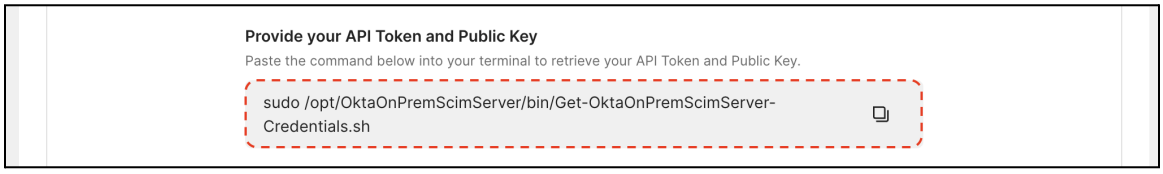


Note: There is option to download the Okta On-prem SCIM Server from this page. Since the On-prem SCIM Server has already been downloaded and placed on server, so we won't be downloading it from here.

14. Formulate the execution command using appropriate details.
    To form the execution command, the OktaOnPremScimServer version is required, which can be found from the Onprem SCIM Server downloaded in this section.
    The execution command would look like this.

```
Shell
sudo CUSTOMER_ID=00oevo9zds7GtFLqw1d7 rpm -ivh OktaOnPremScimServer-<version>.rpm
```

15. Scroll down and take a note of the command under "Provide your API Token and Public Key"



Now, we have the Okta On-prem SCIM Server RPM installer as well as all the commands to install it. So we will proceed with installing the server in the next section and continue with the application configuration once On-prem SCIM Server has been installed successfully.

Install Okta On-prem SCIM Server

1. Logon to the Linux Server, where the Okta On-prem SCIM Server RPM file has been placed.
2. Navigate to the location of the installer.

```
None
[user@ip-X-X-X-X /]$ cd /installers/
```

3. Execute the installation of the On-prem SCIM Server using the below command. (Use the command from Step 14 of the previous section.)

```
None
[user@ip-X-X-X-X installers]$ sudo CUSTOMER_ID=00oevo9zds7GtFLqw1d7 rpm -ivh
OktaOnPremScimServer-1.0.0-1756771200.98a5656.rpm
warning: OktaOnPremScimServer-1.1.0-1757030400.82a12e0.rpm: Header V4 RSA/SHA256
Signature, key ID 7594089a: NOKEY
Verifying...                          ################################# [100%]
Preparing...                          ################################# [100%]
Running %pre. Install type: 1
=====================================================
On-prem SCIM Server EULA - Early Access
.
.
.
If you do not agree to the foregoing you must not download, access or use the
connector.
=====================================================
Do you accept the above terms and conditions? (yes/no): yes
EULA accepted. Continue Installation...
Group 'okscimserver' created.
User 'okscimserver' created.
Completed %pre.
Updating / installing...
   1:OktaOnPremScimServer-0.0.1-SNAPSH################################# [100%]
[2025-09-08T12:04:43Z] First-time installation detected.
[2025-09-08T12:04:43Z] Generating server keypair + self-signed cert for mycompany
[2025-09-08T12:04:43Z] Server keystore created at
/etc/pki/tls/private/OktaOnPremScimServer-00oevo9zds7GtFLqw1d7.p12
(alias=okscimservercert)
[2025-09-08T12:04:43Z] Generating new API key
[2025-09-08T12:04:43Z] Writing config:
/etc/OktaOnPremScimServer/config-mycompany.properties
[2025-09-08T12:04:44Z] OktaOnPremScimServer service enabled and restarted.
```

4. The On-prem SCIM Server starts running after successful installation. Check the status of the service using the below command.

```
Shell
[user@ip-X-X-X-X /]$ sudo systemctl status OktaOnPremScimServer.service
● OktaOnPremScimServer.service - Okta On-Prem SCIM Server
     Loaded: loaded (/usr/lib/systemd/system/OktaOnPremScimServer.service;
enabled; preset: disabled)
     Active: active (running) since Mon 2025-09-08 12:04:44 UTC; 4min 47s ago
   Main PID: 4204 (java)
      Tasks: 35 (limit: 22311)
     Memory: 275.4M
        CPU: 12.747s
     CGroup: /system.slice/OktaOnPremScimServer.service
             └─4204 java -Xms512m -Xmx4096m -XX:+UseG1GC
-XX:+ExitOnOutOfMemoryError
-Dloader.main=com.okta.server.scim.ScimServerApplication -Dloader.path=/opt/Okta>
Sep 08 12:04:44 ip-10-0-0-153.ap-south-1.compute.internal systemd[1]: Started Okta
On-Prem SCIM Server.
.
.
.
[user@ip-X-X-X-X /]$
```

5. Execute the below **command** to generate the API Token and take a note of the **API Bearer Token** value provided in output.

```
None
[user@ip-X-X-X-X installers]$
/opt/OktaOnPremScimServer/bin/Get-OktaOnPremScimServer-Credentials.sh
============================================================
API Bearer Token (use as 'Authorization: Bearer <token>')
============================================================
2338a******************431880
============================================================
Server TLS certificate saved for client trust
============================================================
Path: /tmp/OktaOnPremScimServer-00oevo9zds7GtFLqw1d7.crt
SHA256 Fingerprint:
E1:64:AE:27:D7:3A:41:13:76:C1:1F:2C:20:7D:C7:13:04:8A:CA:52:A0:C4:BA:A1:38:B6:83:D
A:67:25:C7:86
Examples:
  curl --cacert /tmp/OktaOnPremScimServer-mycompany.crt https://<host>:1443/
  keytool -importcert -alias scim -file /tmp/OktaOnPremScimServer-mycompany.crt
-keystore truststore.p12 -storetype PKCS12 -storepass changeit -noprompt
```

6. In the output of the above command, a certificate named

"`OktaOnPremScimServer-00oevo9zds7GtFLqw1d7.crt`" is generated. Download this certificate from the server to the local system.
(Note: This certificate is used by the Okta On-prem SCIM Server to encrypt the connection with Okta Provisioning Agent, thereby enabling secure HTTP.**)**

7. Based on the database which you are integrating with Okta, place the appropriate JDBC driver file into this location **/opt/OktaOnPremScimServer/userlib/.**
(Refer this section for details)

```
None
[user@ip-X-X-X-X /]$ cp /installers/ojdbc11-21.5.0.0.jar
/opt/OktaOnPremScimServer/userlib/
```

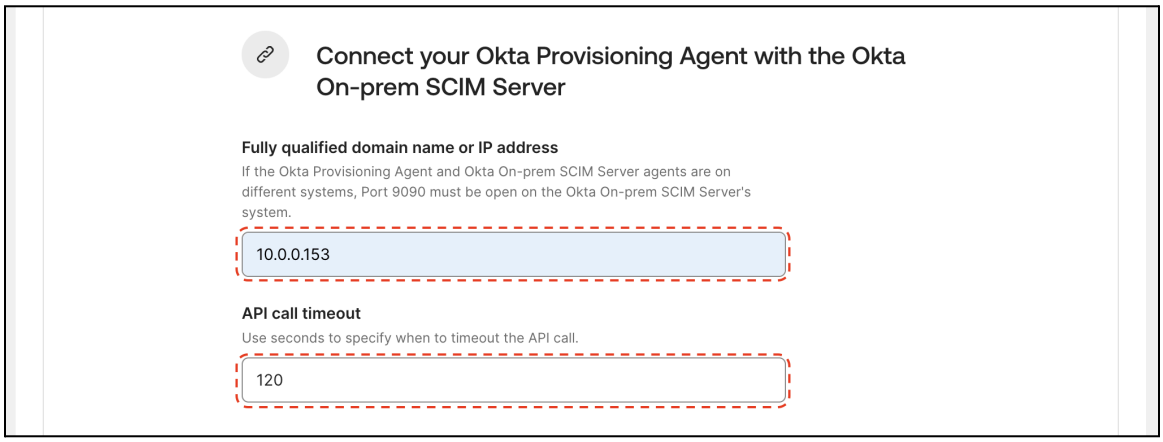
8. Restart the Okta On-prem SCIM Server.

```
None
[user@ip-X-X-X-X /]$ sudo systemctl restart OktaOnPremScimServer.service
[user@ip-X-X-X-X /]$ sudo systemctl status OktaOnPremScimServer.service
```

Once the installation has been completed, we need to continue with the On-prem Connector for Generic Databases application configuration.

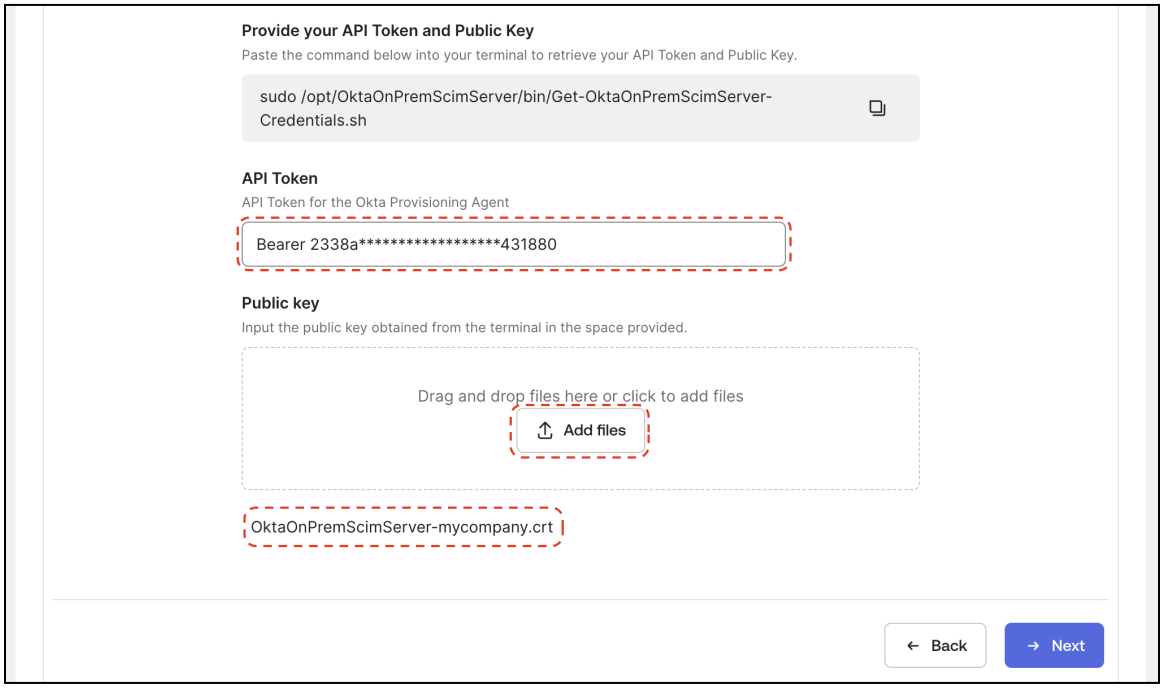
Create On-prem Connector for Generic Databases application (Continued..)

1. Return to the **Provisioning** section of the application on Okta Admin Console.
2. Provide the IP Address of the server where the Okta On-prem SCIM Server is installed, under the **Fully Qualified Domain Name or IP Address**.



3. Input the **API Token** value, appended with **Bearer**, into the API Token field.
(Note: The token value was provided by the **Get-OktaOnPremScimServer-Credentials**

script in the previous section)



4. Under Public Key, select **Add Files.**
5. Navigate to the location where the certificate file was downloaded and select the required certificate file.
   (For e.g. **OktaOnPremScimServer-00oevo9zds7GtFLqw1d7.crt)**
   (Note: The supported certificate files are .crt, .cer and .pem**)**
6. Select **Next**.
7. This section guides through the crucial process of configuring the database for integration. Here, we will be selecting the specific database that will be used for the integration. Follow the steps of the respective database you are configuring. The Database System details to be presented here, have already been collected in this section.

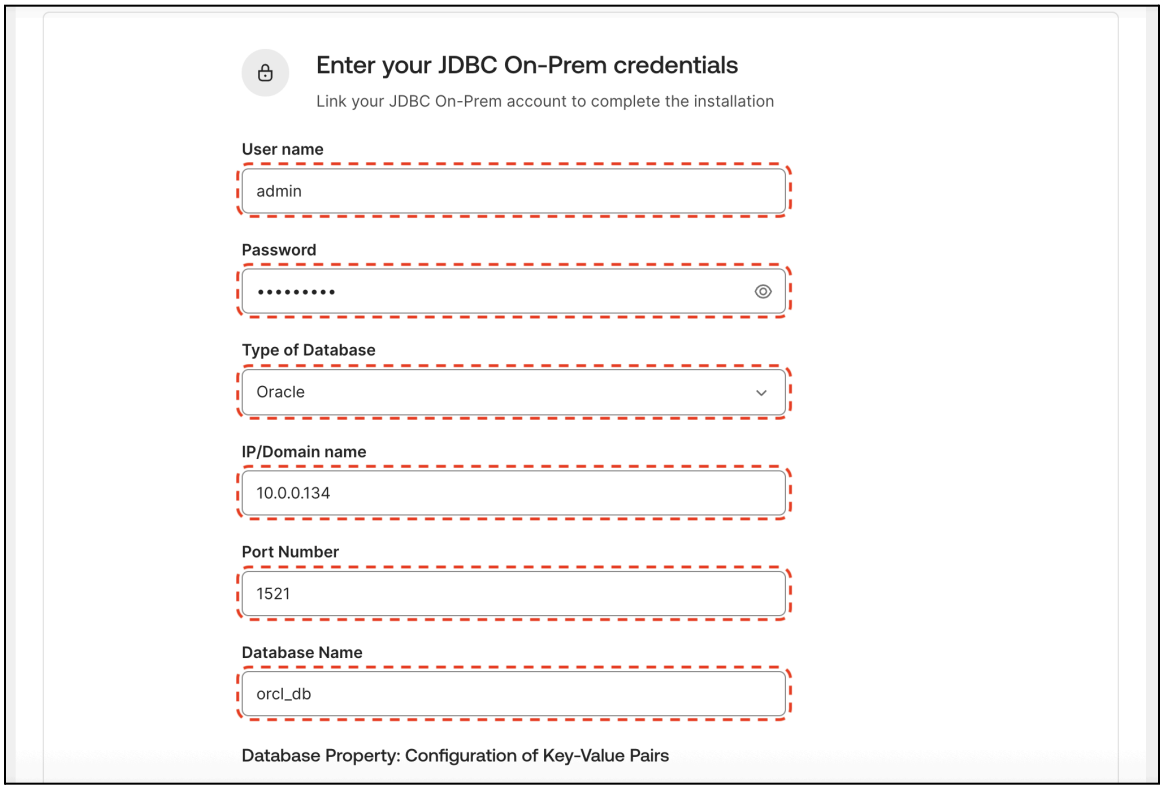
Oracle Database

The Database System details to be presented here, have already been collected in this section.

- Provide the Username of the user in Oracle Database. (For e.g. admin)
- Provide the password for the above user of the Oracle Database.
- Select the Type of Database as **Oracle**
- Provide the IP/Domain Name of the server where Oracle DB is present.

(NOTE: If your database system is using HA configuration then refer to Appendix C)

- Provide the Port Number on which the **Oracle DB** is listening for connections.
- Provide the database Name which will be used from the Oracle Database.



- Select **Connect agents.**

MS SQL Server

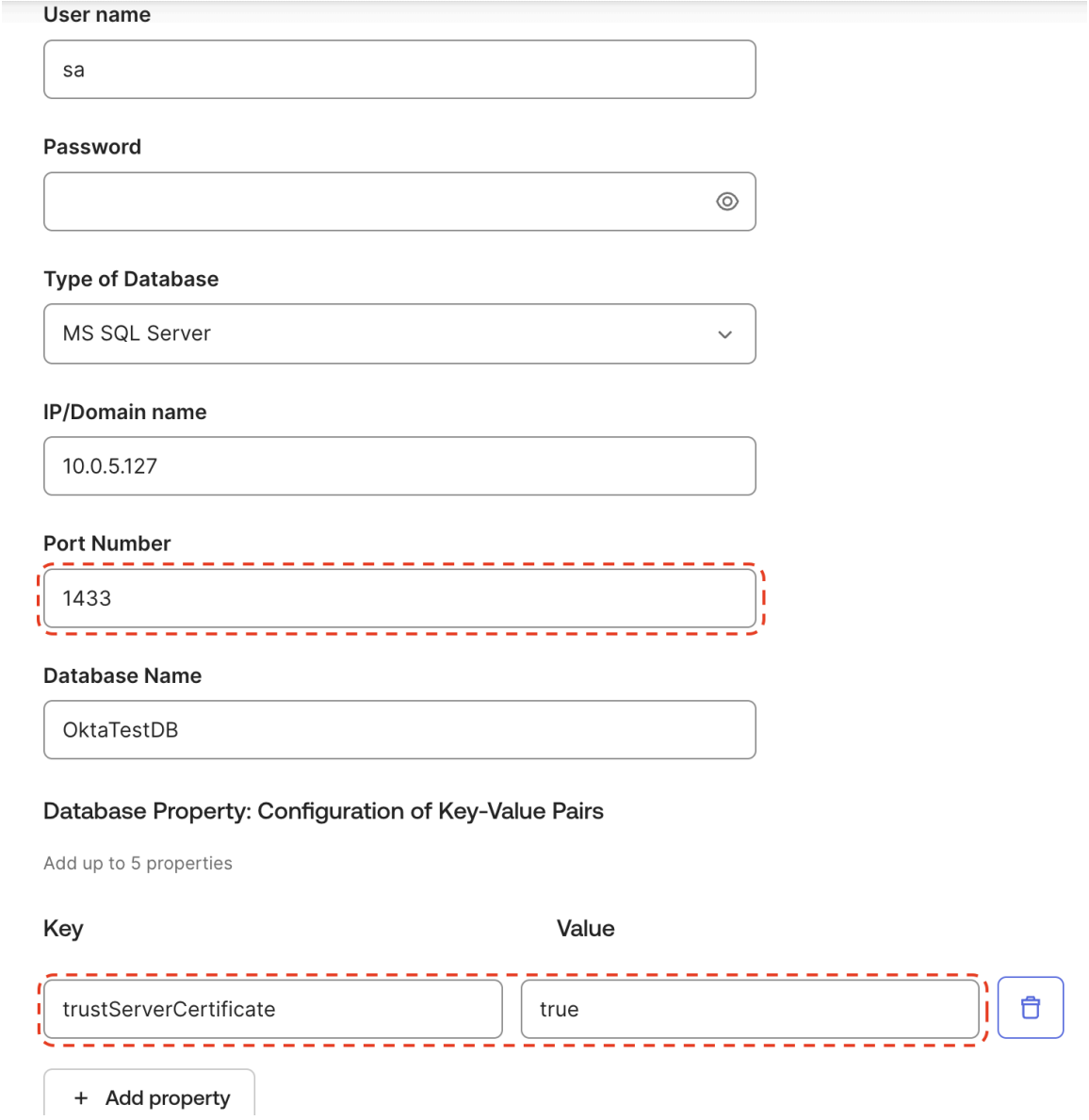
The following steps will guide you to configure the SQL Server Authentication mode, where the On-prem SCIM Server authenticates with MS SQL using its username and password. But if you intend to use the other authentication method available with MS SQL i.e. Windows Authentication Mode then follows this appendix.

The Database System details to be presented here, have already been collected in this section.

- Provide the Username of the user in Oracle Database.
- Provide the password for the above user of the Oracle Database.
- Select the Type of Database as **MS SQL Server**
- Provide the IP/Domain Name of the server where MS SQL Server is present.

(NOTE: If your database system is using HA configuration then refer to Appendix C)

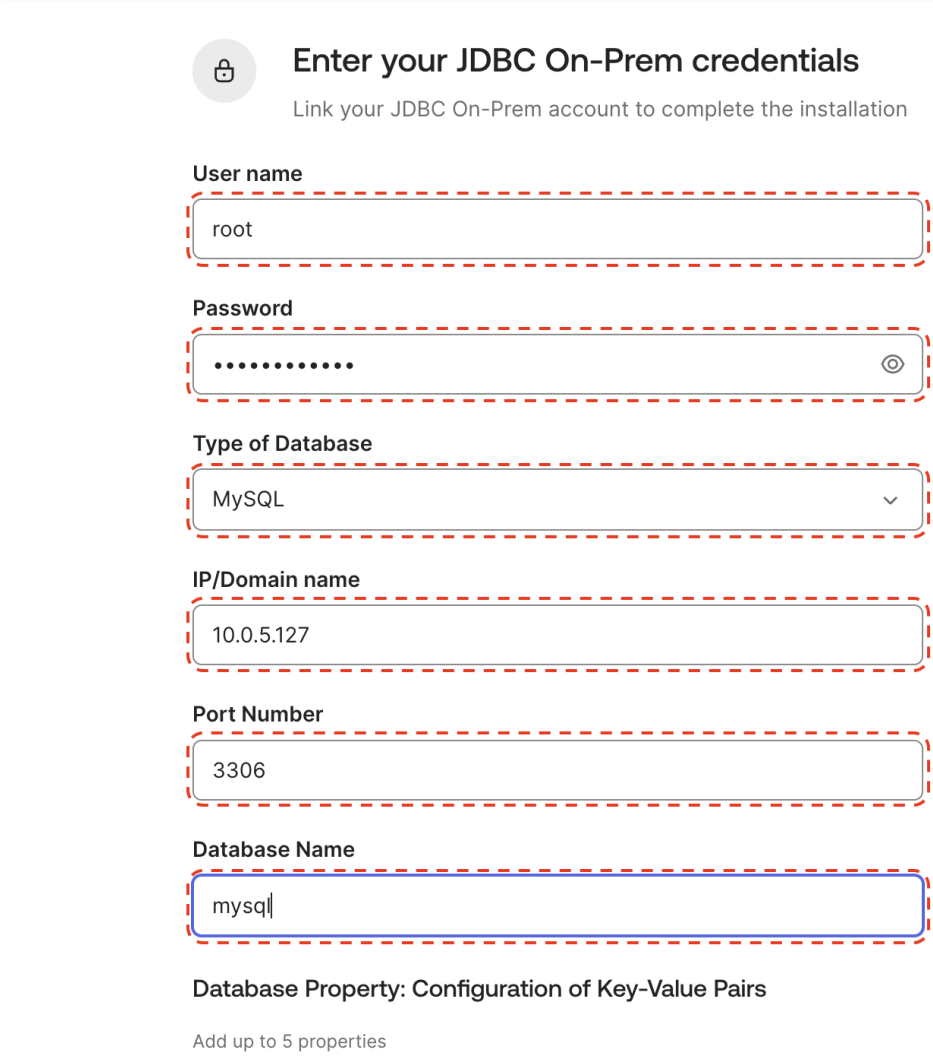
- Provide the Port Number on which the MS SQL Server is listening for connections.
- Provide the database Name which will be used in the MS SQL Server.
- Select **+ Add Property**
- Provide **trustServerCertificate** as Key and **true** as Value for the new field.



- Select **Connect agents.**

MySQL

- Provide the Username of the user in MySQL Database.
- Provide the password for the above user of the MySQL Database.
- Select the Type of Database as **MySQL**
- Provide the IP/Domain Name of the server where **MySQL** is present.
- Provide the Port Number on which the **MySQL** is listening for connections.
- Provide the database name which will be used in the **MySQL** database.

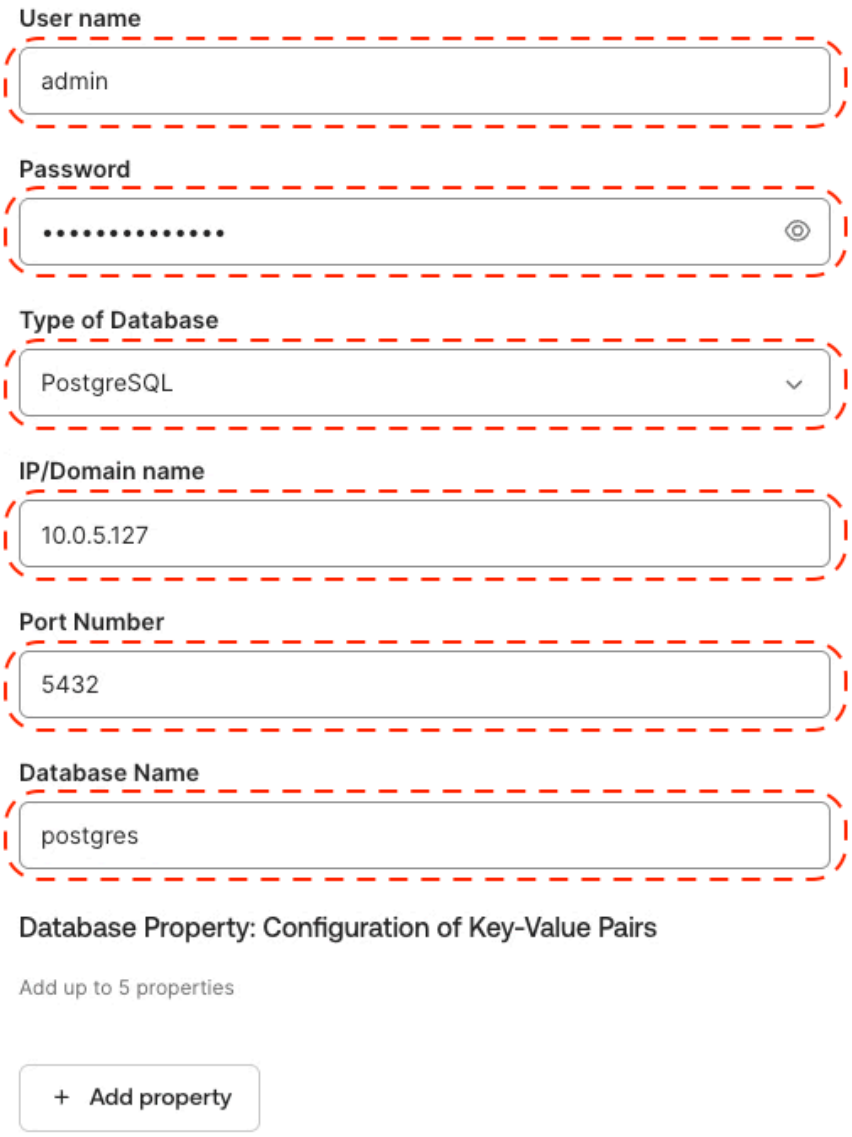


- Select **Connect agents.**

PostgreSQL

- Provide the Username of the user in PostgreSQL Database.

- Provide the password for the above user of the PostgreSQL Database.
- Select the Type of Database as **PostgreSQL**
- Provide the IP/Domain Name of the server where **PostgreSQL** is present.
- Provide the Port Number on which the **PostgreSQL** is listening for connections.
- Provide the database name which will be used in the **PostgreSQL** database.



- Select **Connect agents.**

IBM DB2

- Provide the Username of the user in IBM DB2 Database.
- Provide the password for the above user of the IBM DB2 Database.
- Select the Type of Database as **DB2 LUW**
- Provide the IP/Domain Name of the server where **DB2** is present.
- Provide the Port Number on which the **DB2** is listening for connections.
- Provide the database name which will be used in the **DB2** database.

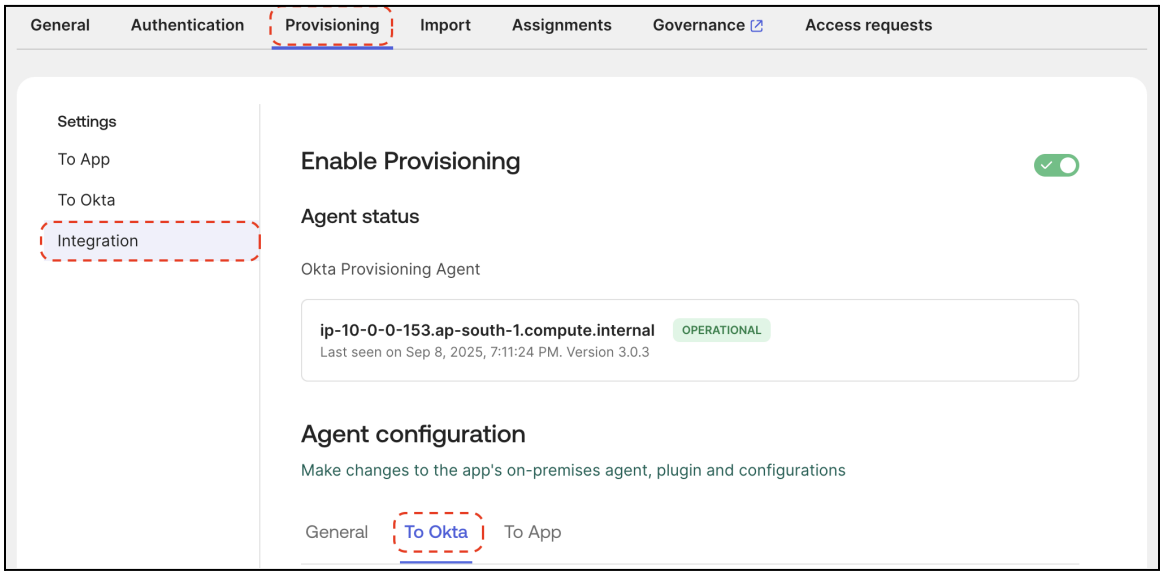
- Select **Connect agents.**

Once the connection is successful and completed, it will land on the Integration tab of the Provisioning section.

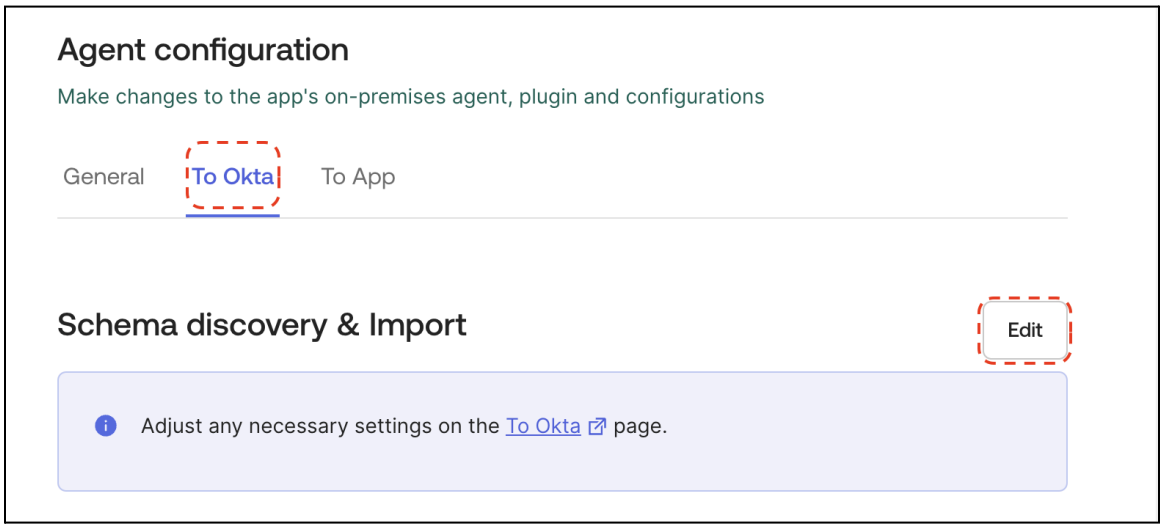
Next, we will configure the **Schema Discovery & Import**. This section provides two options as SQL Statements and Stored Procedures, for performing the necessary operations, which we have discussed in previous sections.

Note: In this document, SQL Statements will be used for Importing and Provisioning operations. To use Stored Procedures or Custom Code for provisioning, please follow the appendix section..

1. Navigate to Provisioning and then Integration. Select **To Okta**.



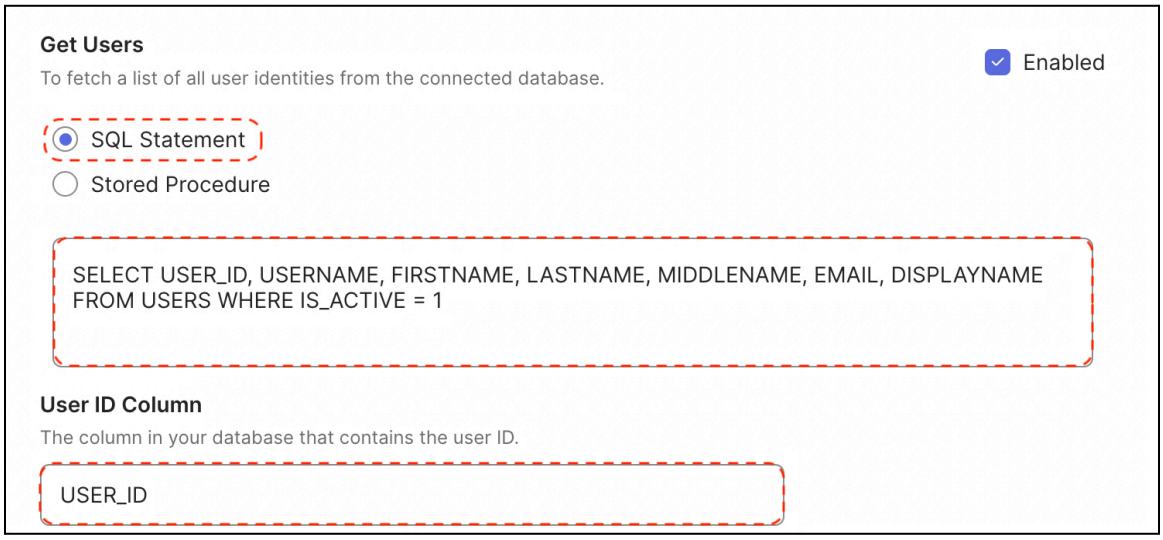
2. Select Edit against **Schema discovery & Import.**



3. Under **Get Users** check the **Enabled** checkbox.
4. Select SQL Statement checkbox.
5. Provide the SQL query for getting user information.
   For example, with reference to the database schema used is this doc, a sample query for this operation would be like below

```
SQL
SELECT USER_ID, USERNAME, FIRSTNAME, LASTNAME, MIDDLENAME, EMAIL, DISPLAYNAME FROM
USERS WHERE IS_ACTIVE = 1
```

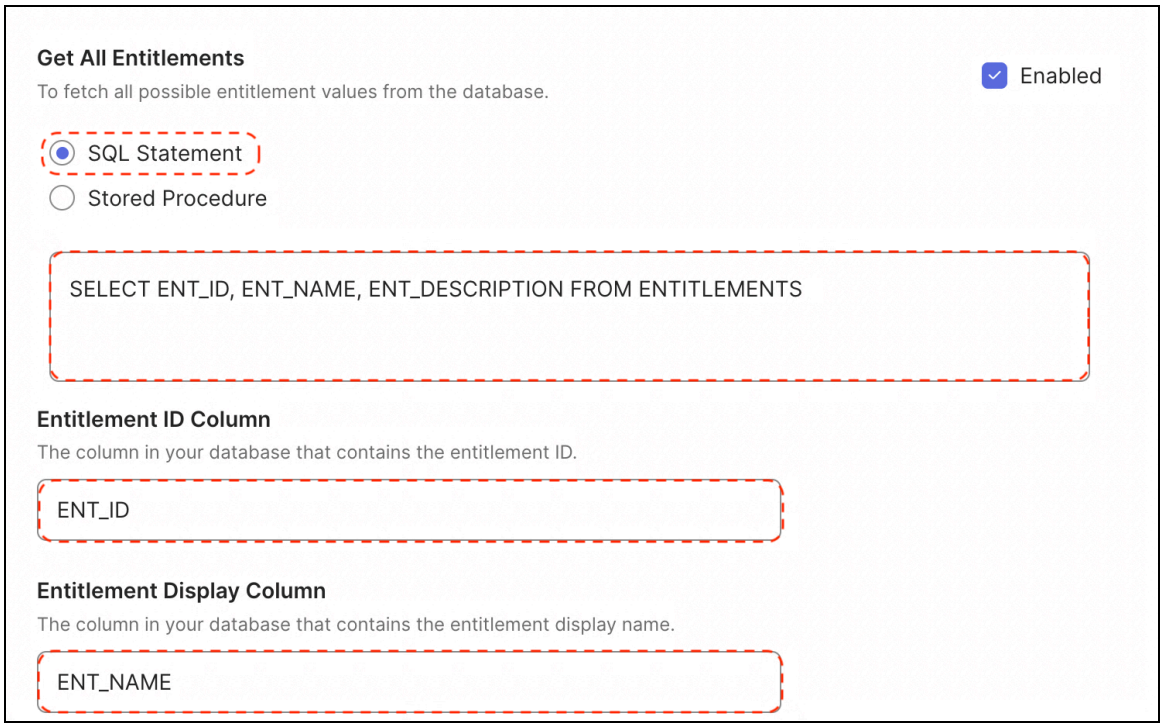
6. Provide the User ID Column name.
   For example, with reference to the database schema used in this doc, the userID column name is User_ID.



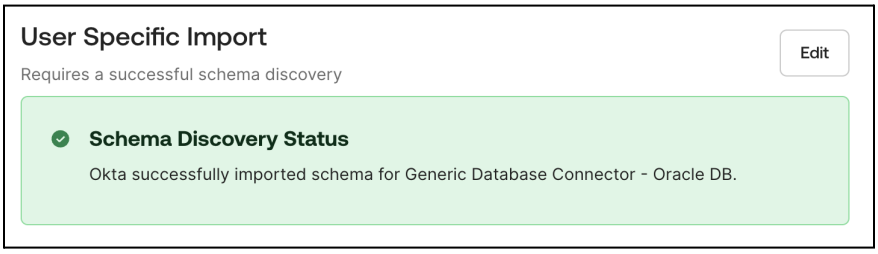
7. Under **Get All Entitlements** check the **Enabled** checkbox.
8. Select SQL Statement checkbox.
9. Provide the SQL query for fetching all the entitlements.
   For example, with reference to the database schema used is this doc, a sample query for this operation would be like below

```
SQL
SELECT ENT_ID, ENT_NAME, ENT_DESCRIPTION FROM ENTITLEMENTS
```

10. Provide the **Entitlement ID** Column.
11. Provide the **Entitlement Display** Column.
    For example, with reference to the database schema used in this doc, the entitlement column names are ENTITLEMENT_ID and ENTITLEMENT_NAME.



Note: Only after the above two steps are completed successfully, then the below steps can be performed, as shown in the below picture.



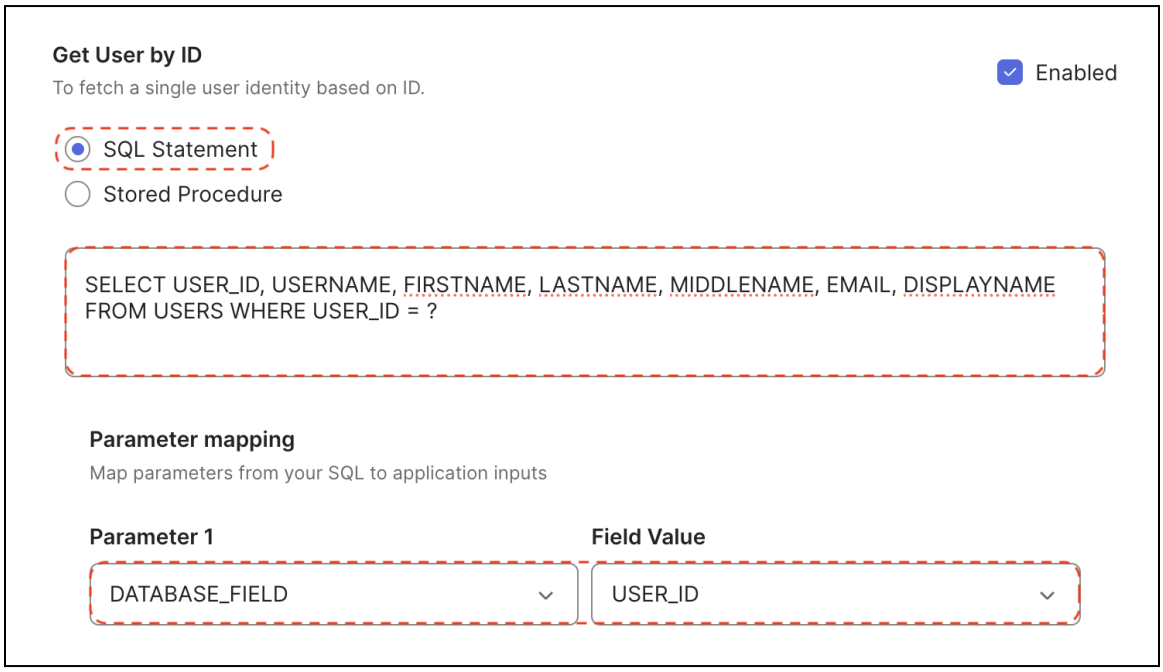
In the upcoming sections, the operations would require construction of dynamic SQL Statements based upon the user for which the operation is being performed, so we will be using the "**?**" placeholders, which are mapped to the attribute of the user and are in a continuous sequence.

8. Under **Get User by ID** check the **Enabled** checkbox.
9. Select SQL Statement checkbox.
10. Provide the SQL query for fetching the details of an individual user.
    For example, with reference to the database schema used in this doc, a sample query for this operation would be like below.
    As the SQL Statement is using placeholder '?' within the statement, it provides the option to map the placeholder with one of the attributes of the table being referred

to in the statement.

```
SQL
SELECT USER_ID, USERNAME, FIRSTNAME, LASTNAME, MIDDLENAME, EMAIL, DISPLAYNAME FROM
USERS WHERE USER_ID = ?
```

11. Provide the user attribute which is used to store the user ID, against Parameter 1.
    For example, with reference to the database schema used in this doc, USER_ID is the attribute used for it.

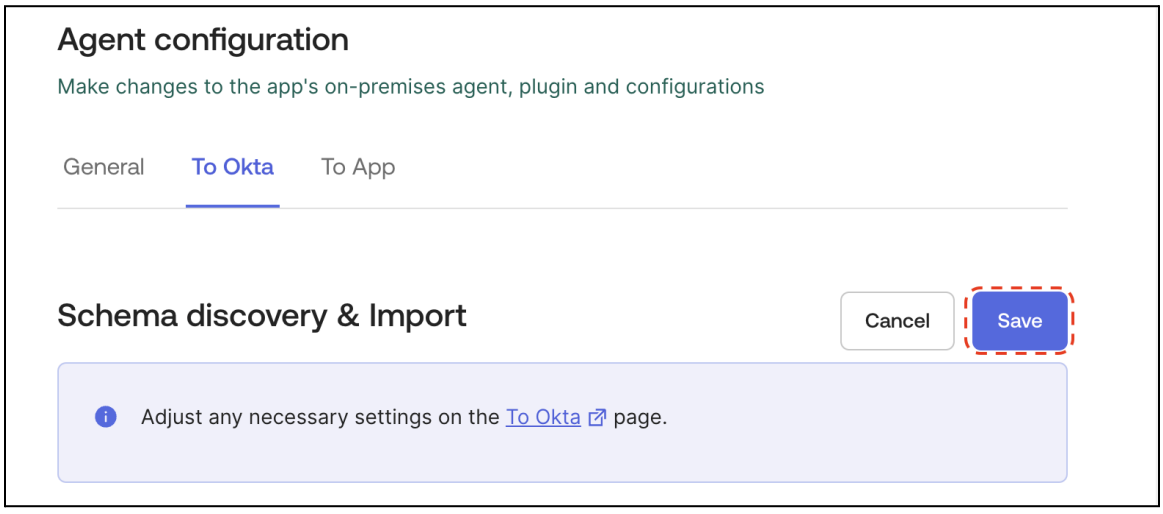


12. Under **Get User Entitlements** check the **Enabled** checkbox.
13. Select SQL Statement checkbox.
    Provide the SQL query fetching the entitlements associated with an individual user. For example, with reference to the database schema used in this doc, a sample query for this operation would be like below.

```
SQL
SELECT UE.USERENTITLEMENTID, UE.USER_ID, U.USERNAME, U.EMAIL, UE.ENT_ID,
E.ENT_NAME, E.ENT_DESCRIPTION, UE.ASSIGNEDDATE
FROM USERENTITLEMENTS UE
JOIN USERS U ON UE.USER_ID = U.USER_ID
JOIN ENTITLEMENTS E ON UE.ENT_ID = E.ENT_ID
WHERE UE.USER_ID = ?
```

14. Provide the user attribute which is used to store the user ID, against Parameter 1. For example, with reference to the database schema used in this doc, USER_ID is the attribute used for it.

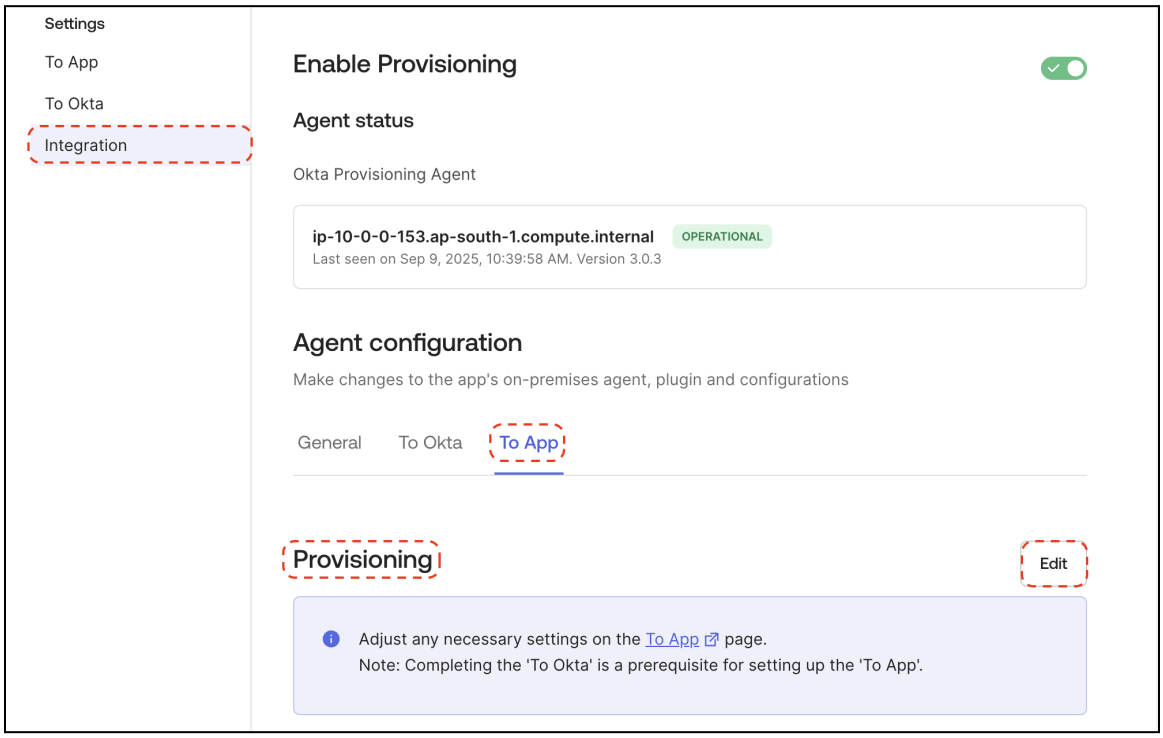
15. Select **Save** button, against Schema Discovery and Import**.**



Once you hit Submit, it will ask to reauthenticate your session using the configured MFA options such as Fastpass etc. This completes the Schema Import and Discovery configurations and we will configure the provisioning section of this application.

In the **Provisioning** section, we will configure all the user and entitlement related operations as discussed in the previous sections.
(**Note**:The On-prem Connector for Generic Databases provides an option to use **Custom Code** for provisioning related functionalities. If there is an intent of using the **Custom Code** for provisioning, then jump to this section.

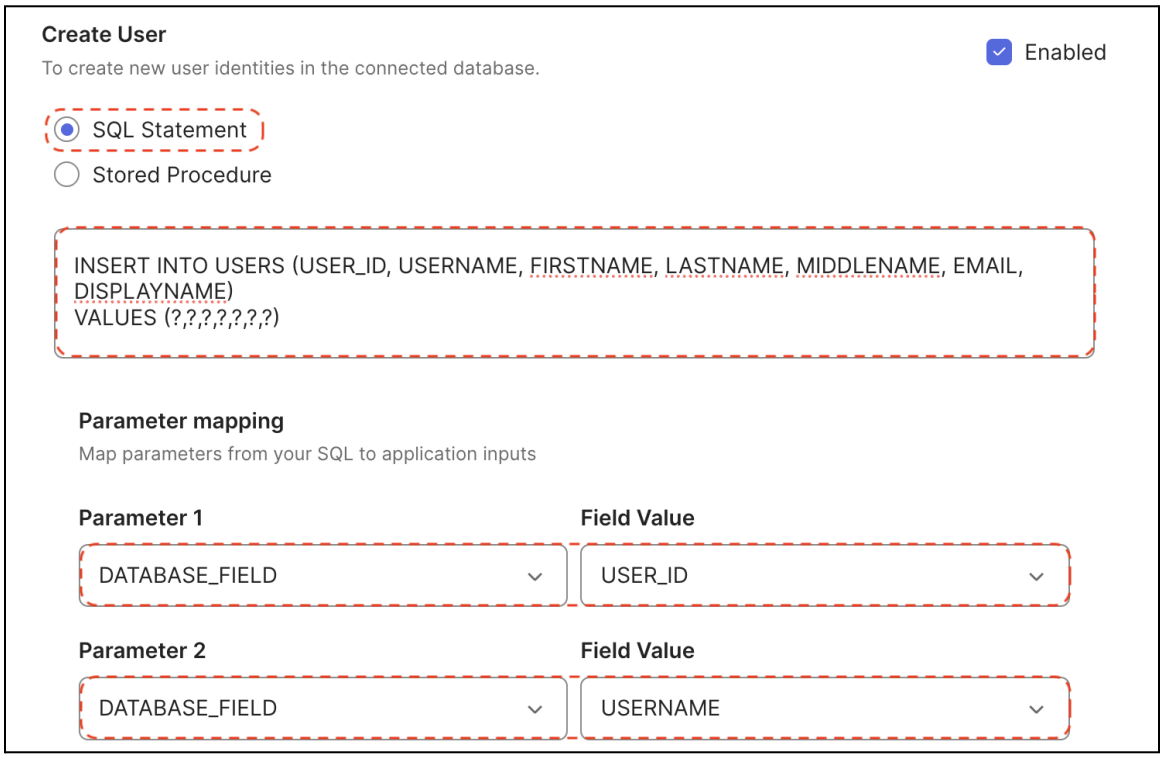
1. Navigate to **To App**, within **Integration** of the **Provisioning** section of this application.
2. Select **Edit** next to Provisioning.



3. Under **Create User** check the **Enabled** checkbox.
4. Select SQL Statement checkbox.
5. Provide the SQL query for creating a user in the database.
   For example, with reference to the database schema used in this doc, a sample query for this operation would be like below.

```sql
INSERT INTO USERS (USER_ID, USERNAME, FIRSTNAME, LASTNAME, MIDDLENAME, EMAIL,
DISPLAYNAME)
VALUES (?,?,?,?,?,?,?)
```

6. Provide the appropriate attribute for each of the placeholders used in the query in the parameters/value mapping.

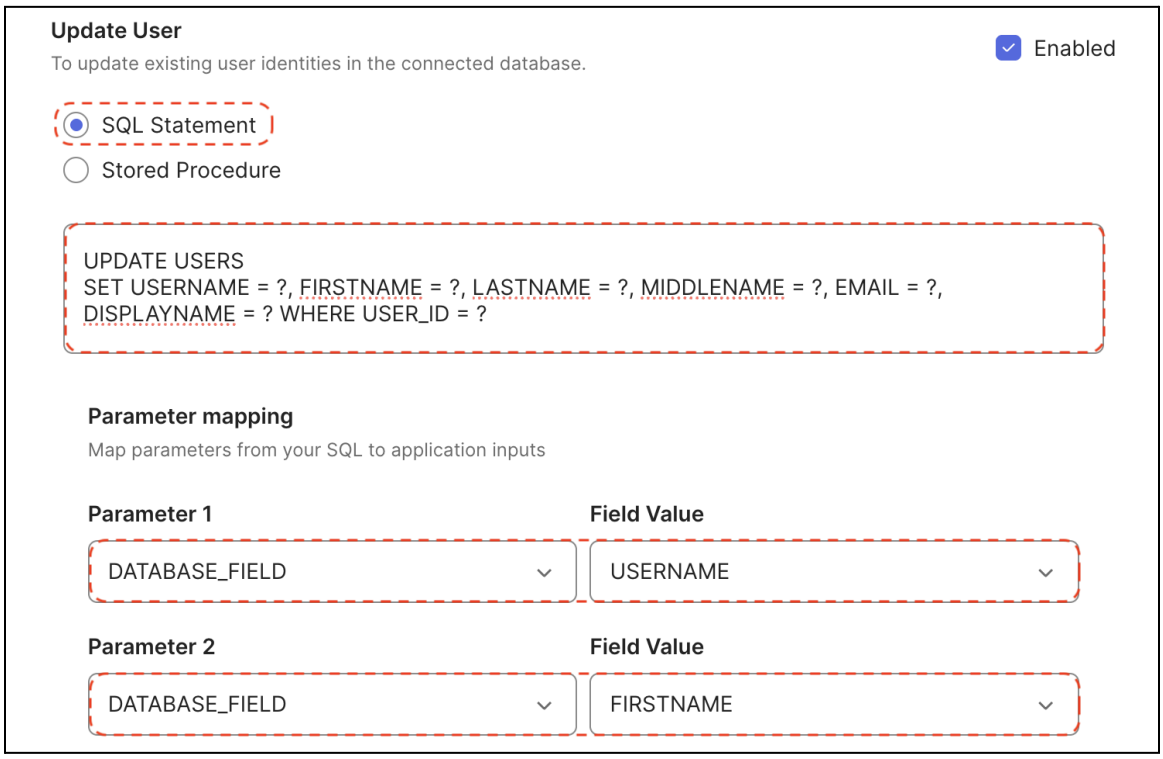


(Note: Since, the attributes used for creating the user are externally sourced attributes so there is an prefix ext_ added to those attributes)

16. Under **Update User** check the **Enabled** checkbox.
17. Select SQL Statement checkbox.
18. Provide the SQL query for updating a user in the database.
    For example, with reference to the database schema used in this doc, a sample query for this operation would be like below.

```SQL
UPDATE USERS
SET USERNAME = ?, FIRSTNAME = ?, LASTNAME = ?, MIDDLENAME = ?, EMAIL = ?,
DISPLAYNAME = ? WHERE USER_ID = ?
```

19. Provide the appropriate attribute for each of the placeholders used in the query in the parameters/value mapping.

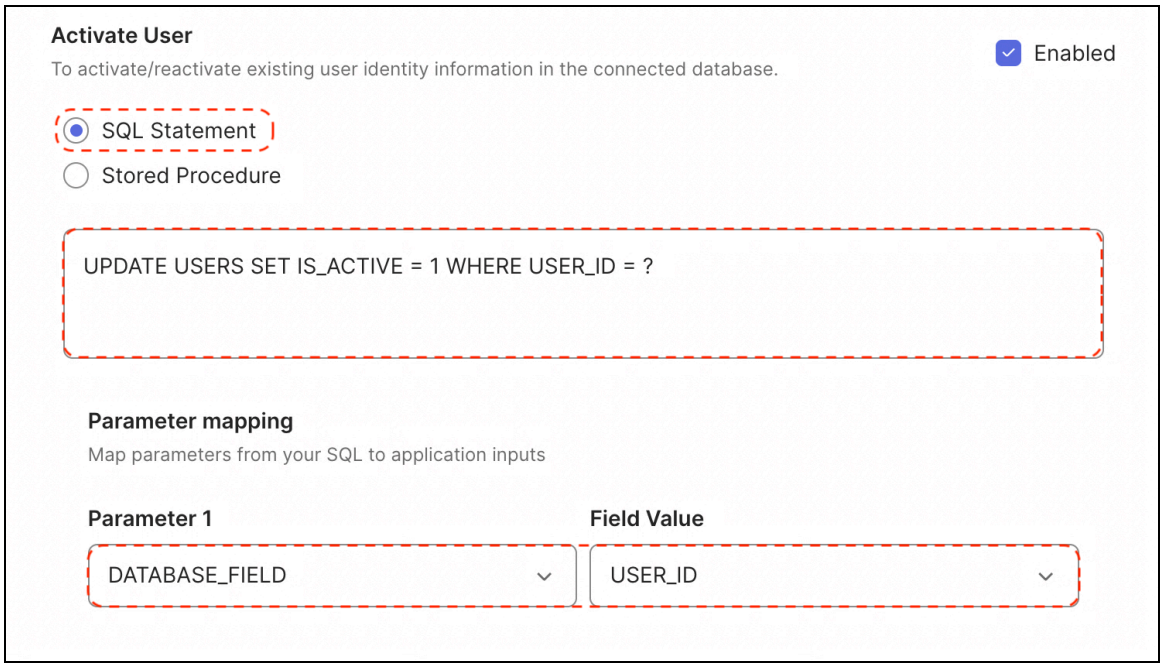


20. Under **Activate User** check the **Enabled** checkbox.
21. Select SQL Statement checkbox.
22. Provide the SQL query for activating a user in the database

    For example, with reference to the database schema used in this doc, a sample query for this operation would be like below.

```sql
UPDATE USERS SET IS_ACTIVE = 1 WHERE USER_ID = ?
```

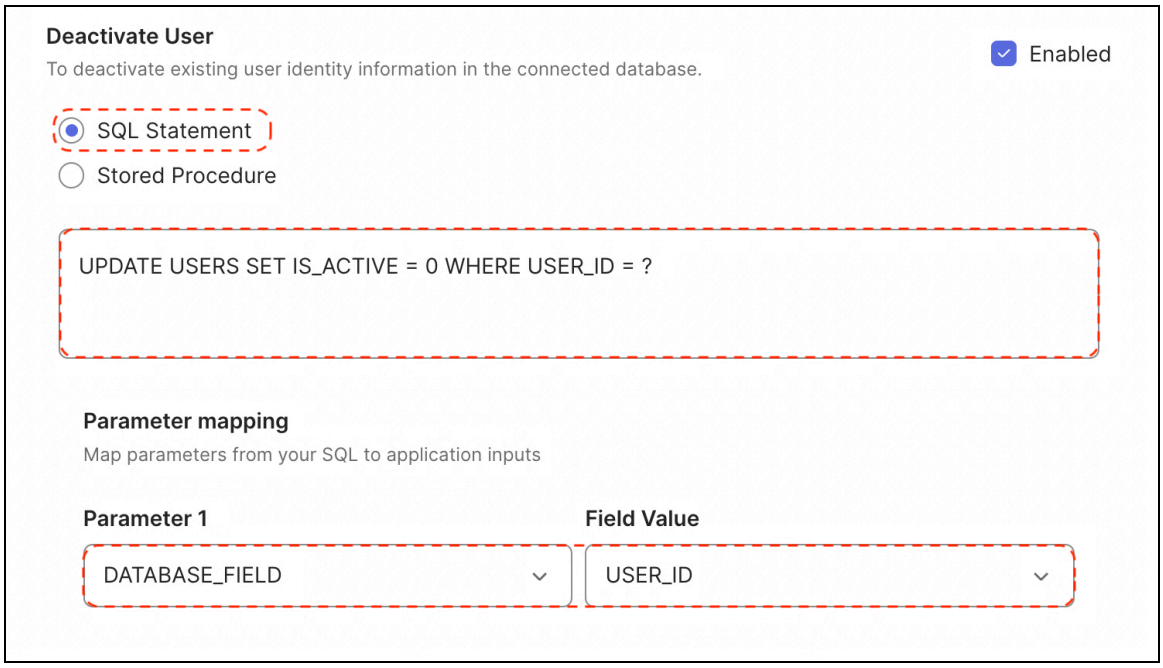
23. Provide the appropriate attribute for each of the placeholders used in the query in the parameters/value mapping.



24. Under **Deactivate User** check the **Enabled** checkbox.
25. Select SQL Statement checkbox.
26. Provide the SQL query for deactivating a user in the database.
    For example, with reference to the database schema used in this doc, a sample query for this operation would be like below.

```sql
UPDATE USERS SET IS_ACTIVE = 0 WHERE USER_ID = ?
```

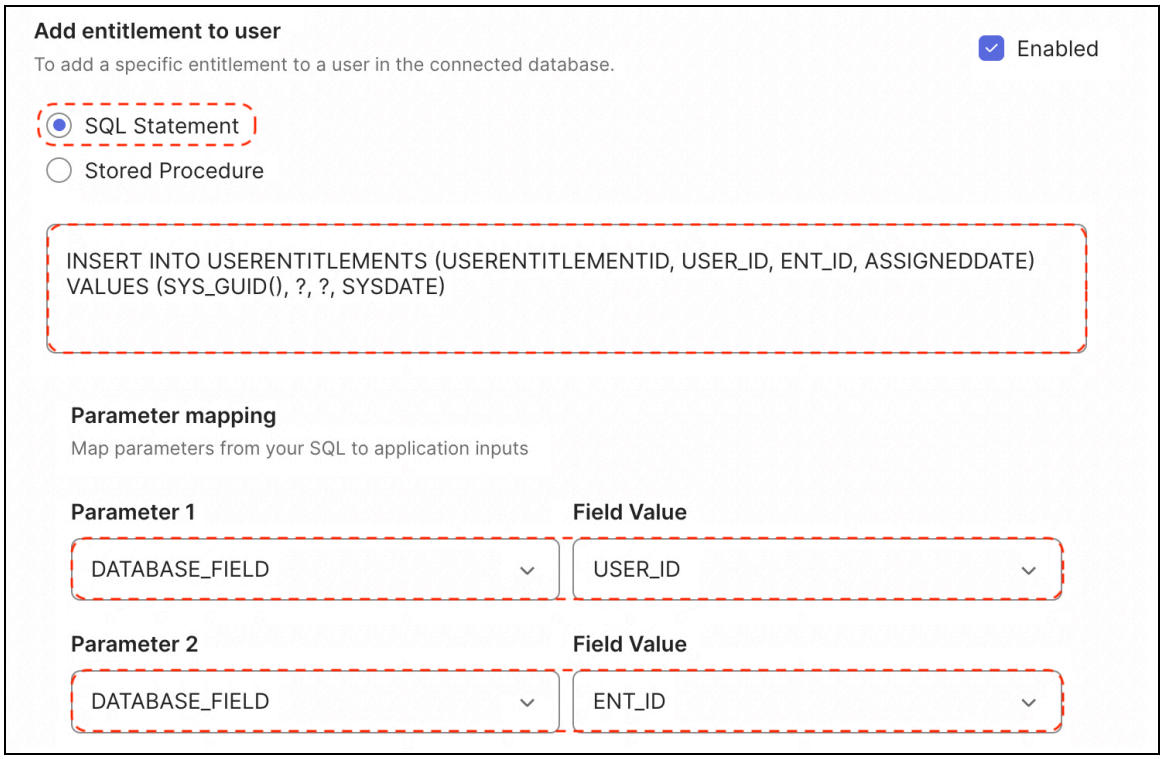
27. Provide the appropriate attribute for each of the placeholders used in the query in the parameters/value mapping.



28. Under **Add entitlement to user** check the **Enabled** checkbox.
29. Select SQL Statement checkbox.
30. Provide the SQL query for adding entitlement to users in the database
    For example, with reference to the database schema used in this doc, a sample query for this operation would be like below.

```sql
INSERT INTO USERENTITLEMENTS (USERENTITLEMENTID, USER_ID, ENT_ID,
ASSIGNEDDATE) VALUES (SYS_GUID(), ?, ?, SYSDATE)
```

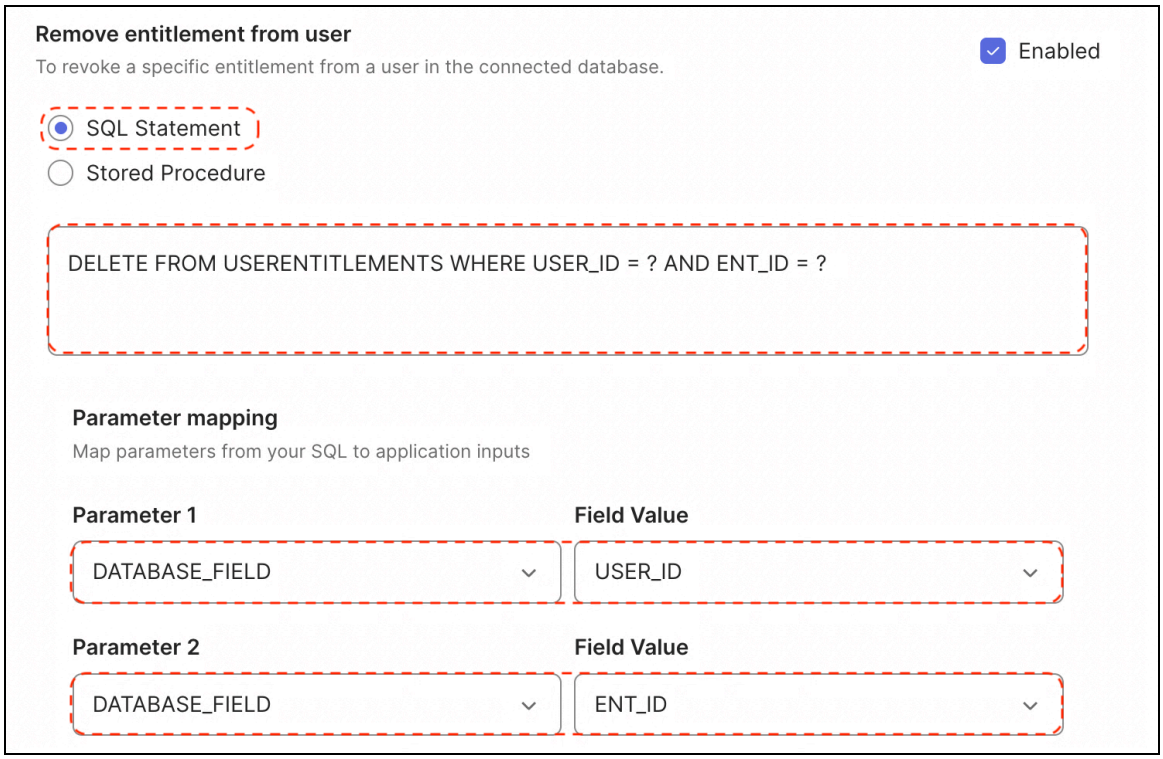
31. Provide the appropriate attribute for each of the placeholders used in the query in the parameters/value mapping.



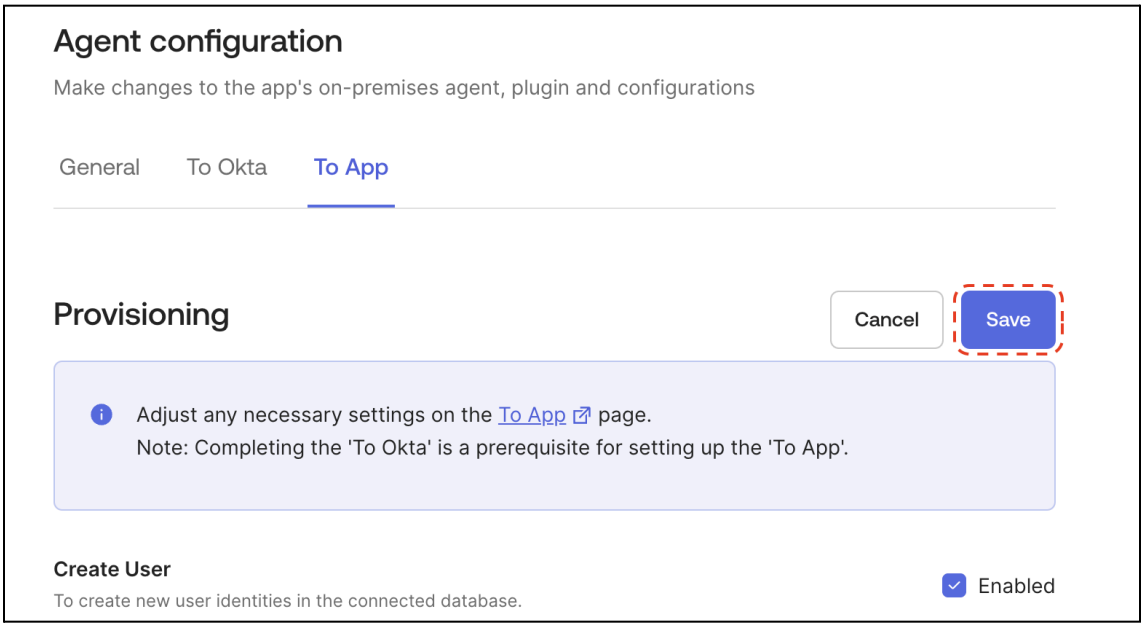
32. Under **Remove entitlement from the user** check the **Enabled** checkbox.
33. Select SQL Statement and provide the SQL query for removing entitlement from users in the database.
    For example, with reference to the database schema used in this doc, a sample query for this operation would be like below.

```sql
DELETE FROM USERENTITLEMENTS WHERE USER_ID = ? AND ENT_ID = ?
```

34. Provide the appropriate attribute for each of the placeholders used in the query in the parameters/value mapping.



35. Scroll to top and Select **Save**.



36. Once you click on Save, it will ask to re-authenticate using configured MFA for the console.

Now, we have the On-prem Connector for Generic Databases loaded with the appropriate SQL Statements and Custom Code to perform the necessary operations. Before the application can start pulling data from the database, we need to configure the user lifecycle management from within the application. In the next section, we will see how to configure those.

Configure User Lifecycle Management

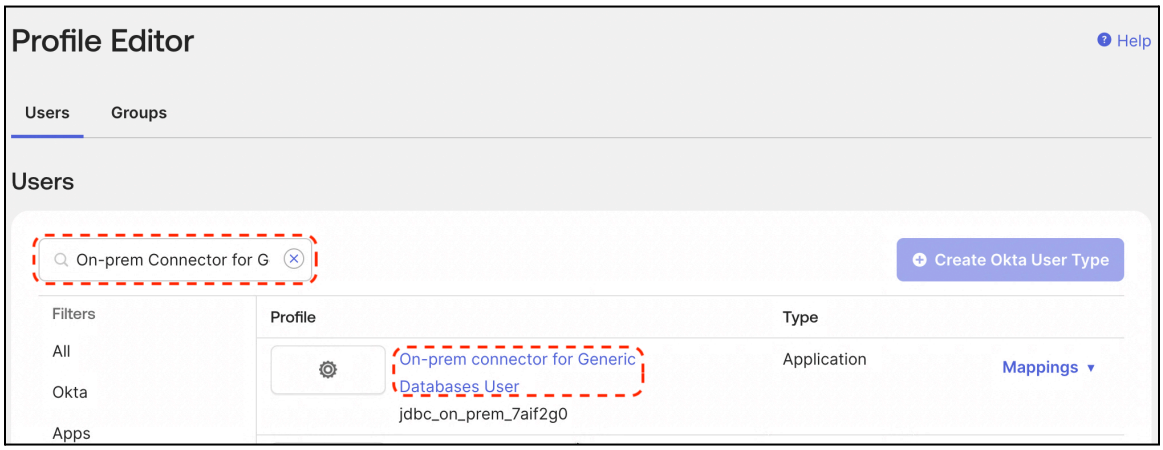
The On-prem Connector for Generic Databases application provides a set of features which can be leveraged to maintain a healthy user lifecycle between Okta and Oracle Database. We will be now exploring these options in this application to configure and streamline provisioning.

Add Custom Attributes to Application User Profile

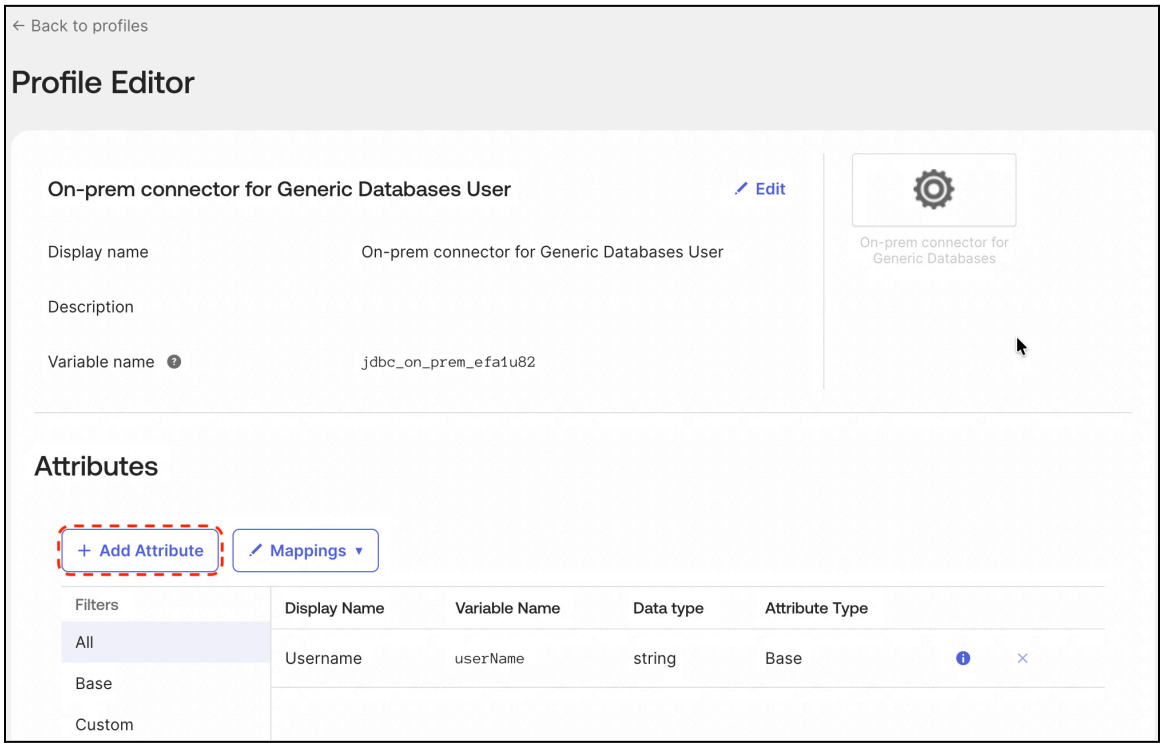
Before beginning with enabling the provisioning configurations, we need to update the attributes for the On-prem Connector for Generic Databases application profile and their mapping to Okta User profile. Since, this application will be bringing users with custom attributes into Okta, we need to add those attributes to the application user profile. (Note: Refer to this article for details on Okta User and Application User profiles.)

To add the attributes from the database into application user profile, follow below steps:

1. On the Okta Admin Console, navigate to Directory and select **Profile Editor.**
2. Search for On-prem Connector for Generic Databases and select the profile with name **On-prem Connector for Generic Databases User**.

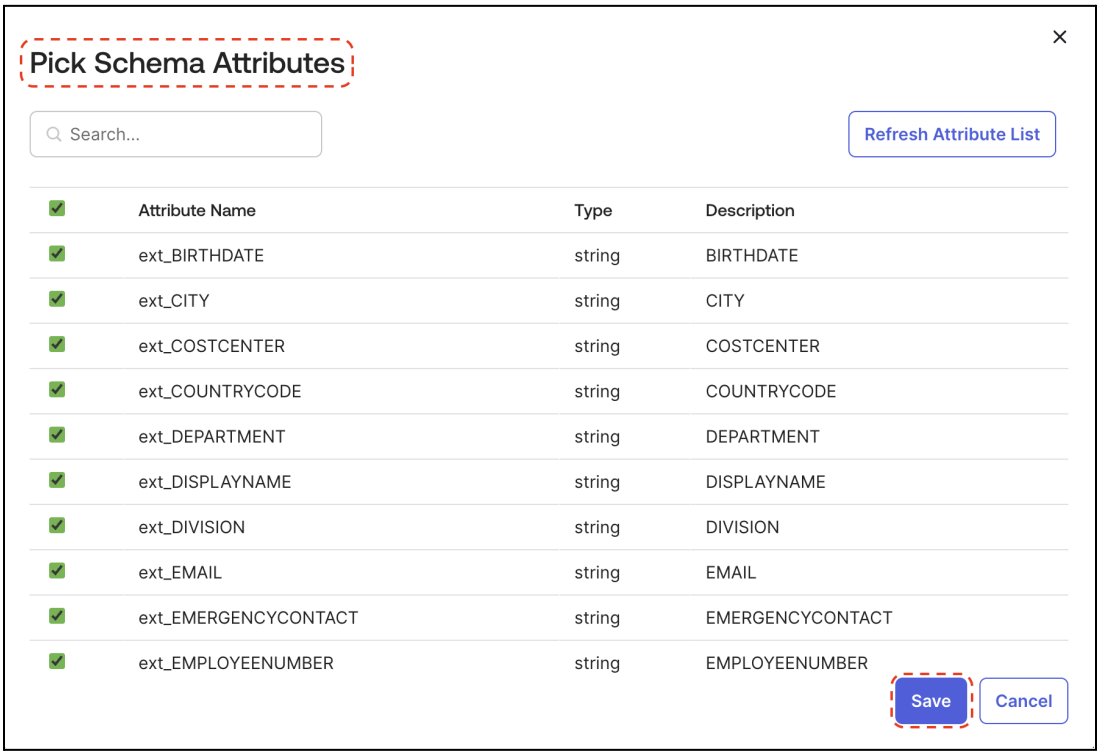


3. Under Attributes, only the Username attribute is present and we need to bring in the attributes from the database. To do so, select **+ Add Attribute**.



4. The next page will show all the attributes imported from the database.

5. Check all the required attributes and select **Save** to add them to the application user profile.



6. The On-prem Connector for Generic Databases user profile now has all the attributes needed for configuring the provisioning operations.

Once we have all the attributes added to the application user's profile, we need to map these attributes against the Okta User's profile attributes.

Mapping User Attributes

Mapping of user attributes has to be done for both user accounts flowing into Okta from the database system as well as for user accounts flowing into the database system from Okta, which are categorised as below

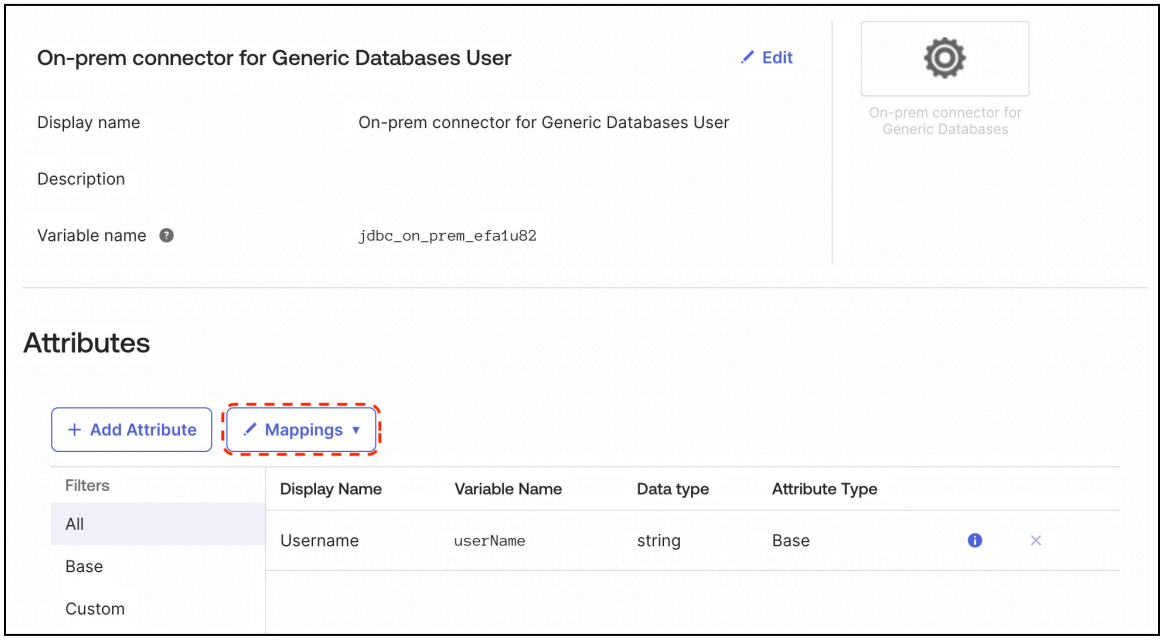
On-prem Connector for Generic Databases User to Okta User

Okta User to On-prem Connector for Generic Databases User

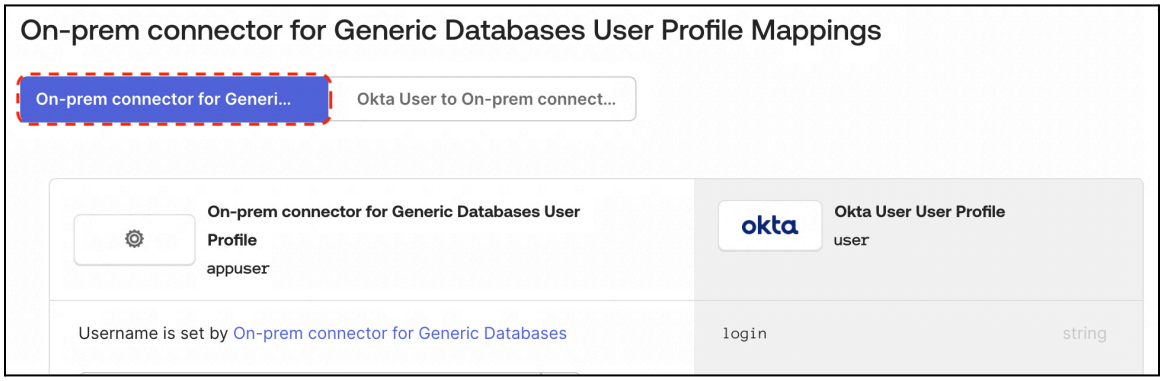
On-prem Connector for Generic Databases User to Okta User

The On-prem Connector for Generic Databases User to Okta User mapping is to govern how the user accounts present in database systems are imported into Okta with each of the attributes in the database system mapped to appropriate attributes in Okta. Follow below step to configure this mapping:

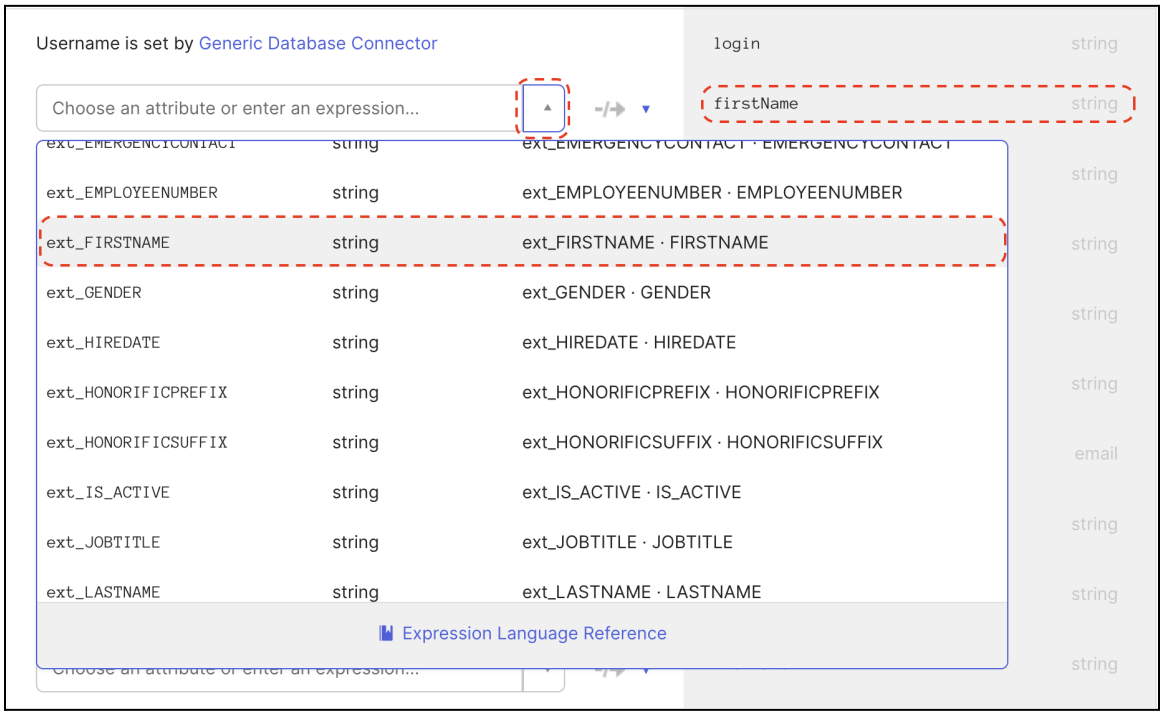
1. Within the On-prem Connector for Generic Databases User profile, select Mappings.



2. By default, the **On-prem Connector for Generic Databases User to Okta User** is selected. Within this, it's visible that the mapping for login is present and the rest are empty. We need to set up the mapping for other attributes as required.



3. Under the On-prem Connector for Generic Databases User Profile, select the drop down menu button within **Choose an attribute or enter an expression**.

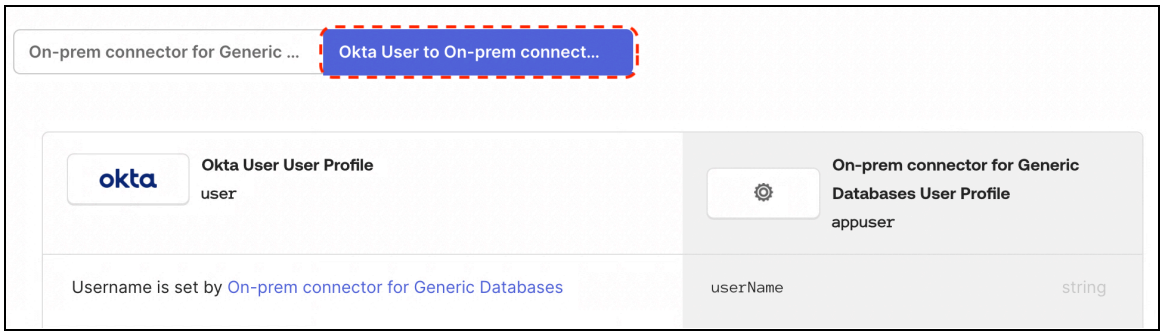


4. The drop down menu provides the list of all the attributes present within the **On-prem Connector for Generic Databases User** Profile, select the appropriate attributes.
   For e.g. In the above picture, it can be seen that for the firstname attribute in Okta User profile, we are selecting the ext_FIRSTNAME from the application user profile.
5. Select the appropriate attributes under On-prem Connector for Generic Databases User profile against the Okta User profile as required.
6. Select **Save** and then select **Apply updates**.

Okta User to On-prem Connector for Generic Databases User

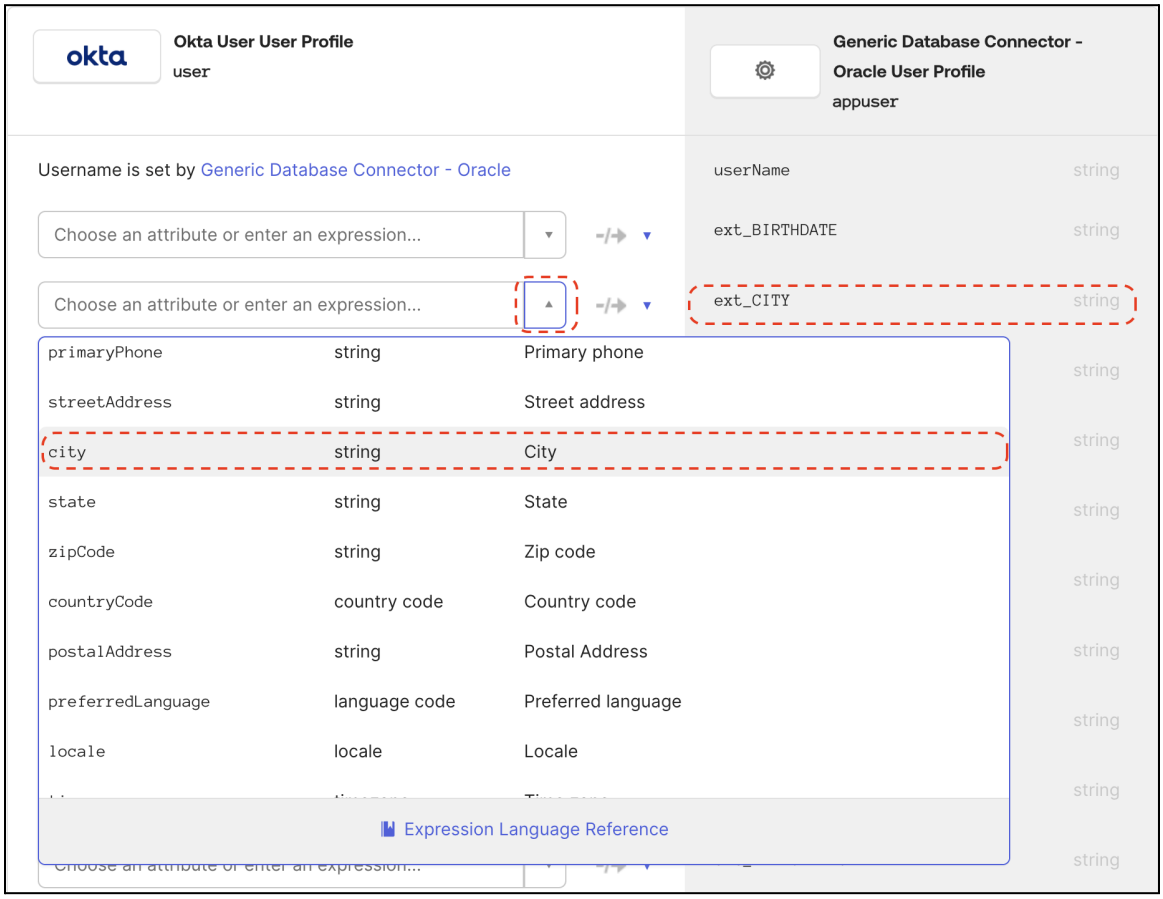
The mapping between Okta users and the On-prem Connector for Generic Databases dictates how user attributes within Okta are correlated with the database user profile. This correlation facilitates the creation or modification of user accounts within the database system. The following steps outline the configuration process for this mapping:

1. Within the On-prem Connector for Generic Databases User profile, select Mappings.
2. Select Okta User to On-prem Connector for Generic Databases User.



3. Under the Okta User to On-prem Connector for Generic Databases User Profile, select the drop down menu button within **Choose an attribute or enter an expression**.
4. The drop down menu provides the list of all the attributes present within the **Okta User** Profile, select the appropriate attributes.

   For e.g. In the below picture, it can be seen that for the ext_city attribute in On-prem Connector for Generic Databases User Profile, we are selecting the city from the application user profile.



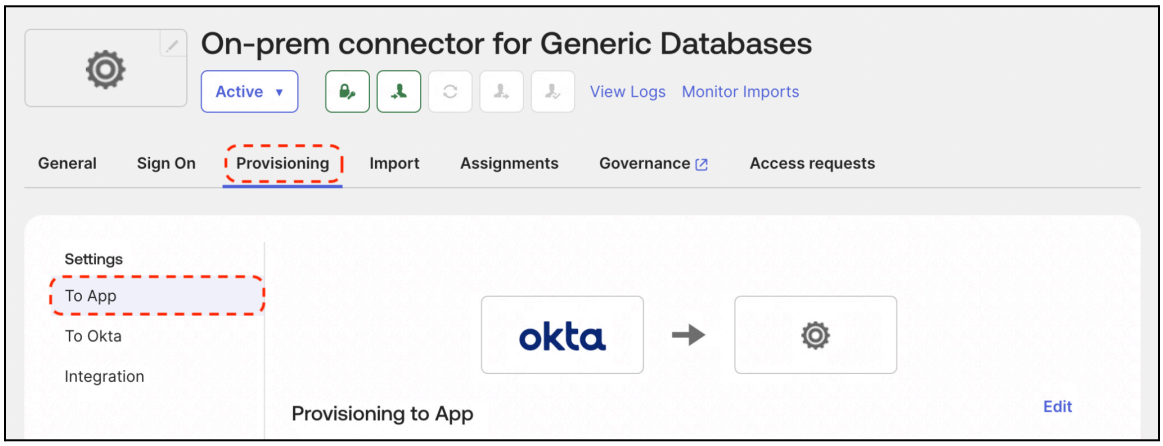
5. Select the appropriate attributes under Okta User profile against the On-prem Connector for Generic Databases User profile as required.
6. Select **Save** and then select **Apply updates**.

Import & Provisioning

Now, we have the mapping of user profiles updated and we can move onto configure the import and provisioning configurations.

Setup Provisioning to Oracle Database

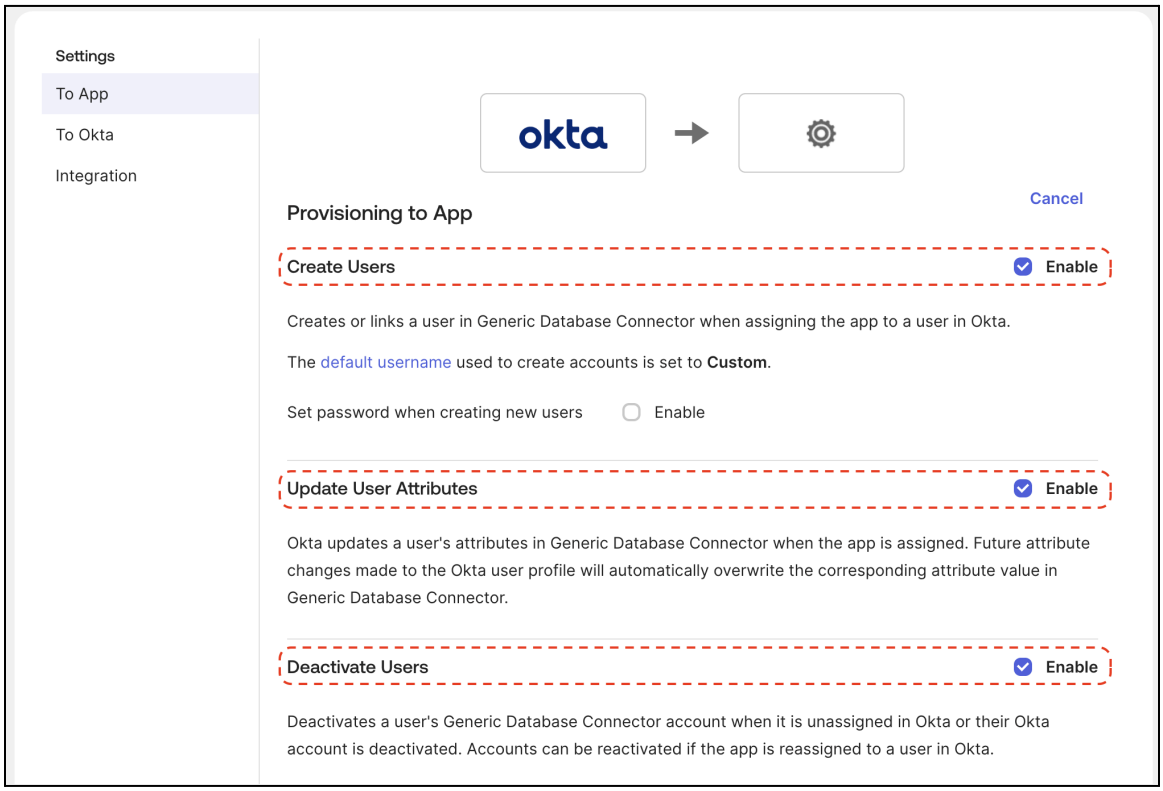
1. In Okta Admin Console, navigate to the Applications and select **On-prem Connector for Generic Databases** application.
2. Select the **Provisioning** section and navigate to **To App** within Settings.



3. Select the **Edit** button on the left side of the console, in the Provisioning to App section.
    a. Create Users
    b. Update User Attributes
    c. Deactivate Users

    (Note: For details on these options please follow this.)

4. Select **Save.**

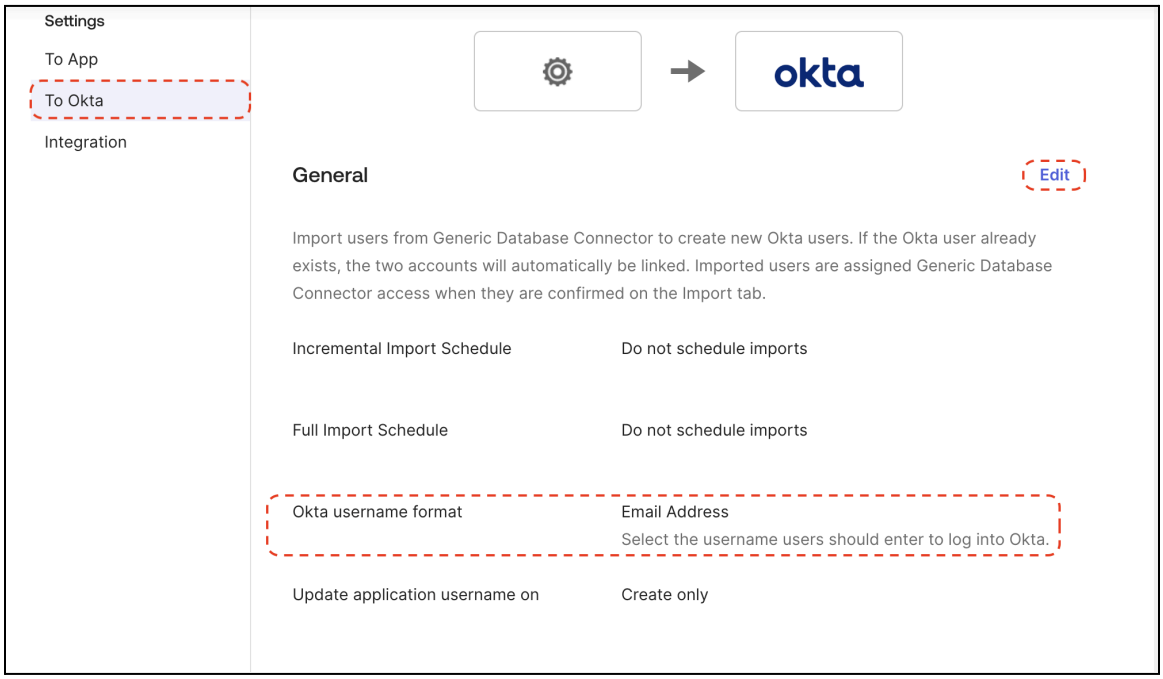


Setup Import

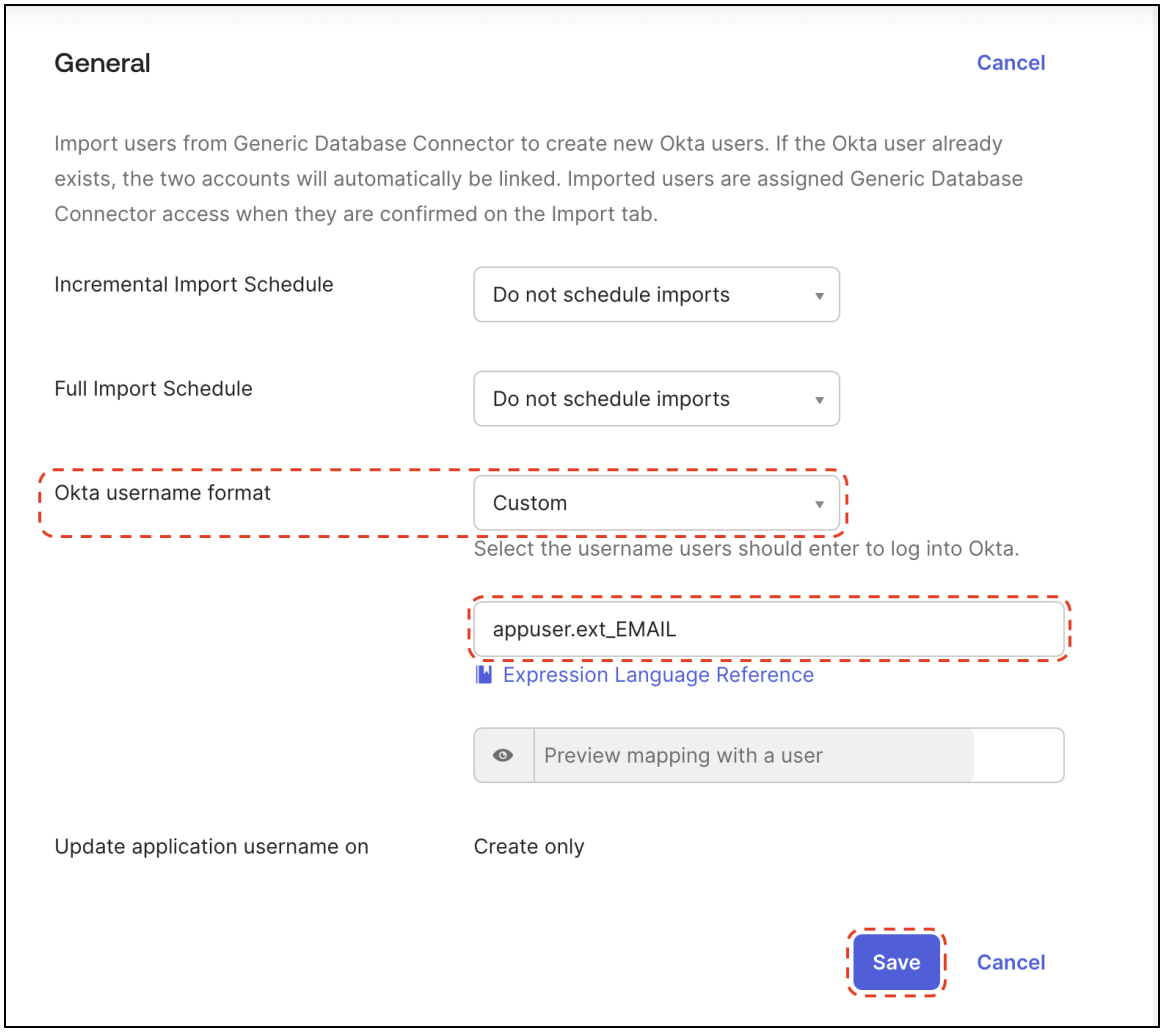
On-prem Connector for Generic Databases application provides the functionality to bring users from databases into Okta using the import feature. To schedule import and configure the mapping of the Okta Username format, follow below steps:

1. Navigate to **On-prem Connector for Generic Databases** application in the Okta Admin Console.
2. Select **Provisioning** and then go to the Provisioning tab.

3. Select **To Okta**, and select **Edit** next to **General**.



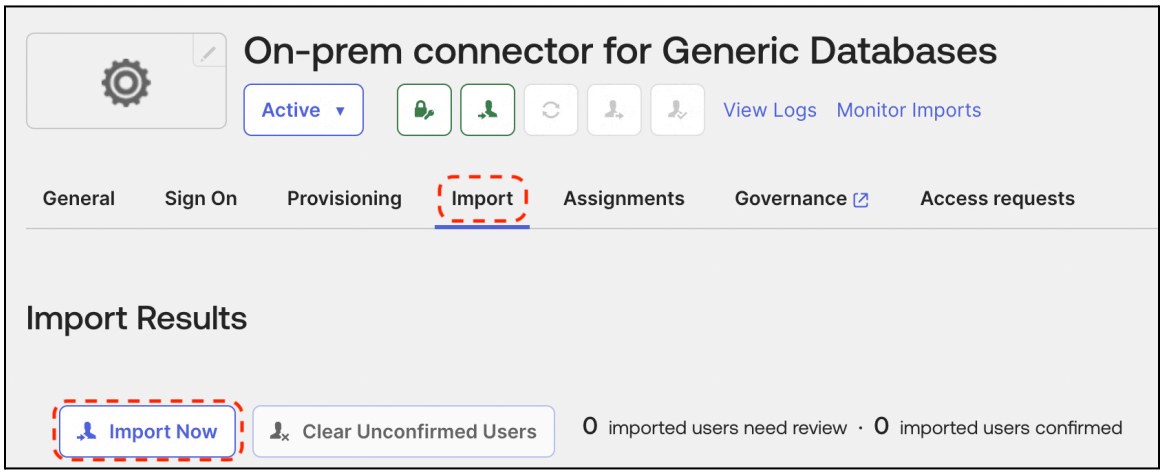
4. Select the required schedule under **Full Import Schedule.**
5. Under Okta username format, select custom from the drop down box.
6. Provide "**appuser.ext_EMAIL**" in the textbox below.
   (For e.g: The username can be in email format, so the required attribute is ext_EMAIL attribute)
7. Select **Save**.


General
Cancel
Import users from Generic Database Connector to create new Okta users. If the Okta user already exists, the two accounts will automatically be linked. Imported users are assigned Generic Database Connector access when they are confirmed on the Import tab.
Incremental Import Schedule
Do not schedule imports
Full Import Schedule
Do not schedule imports
Okta username format
Custom
Select the username users should enter to log into Okta.
appuser.ext_EMAIL
Expression Language Reference
Preview mapping with a user
Update application username on
Create only
Save
Cancel

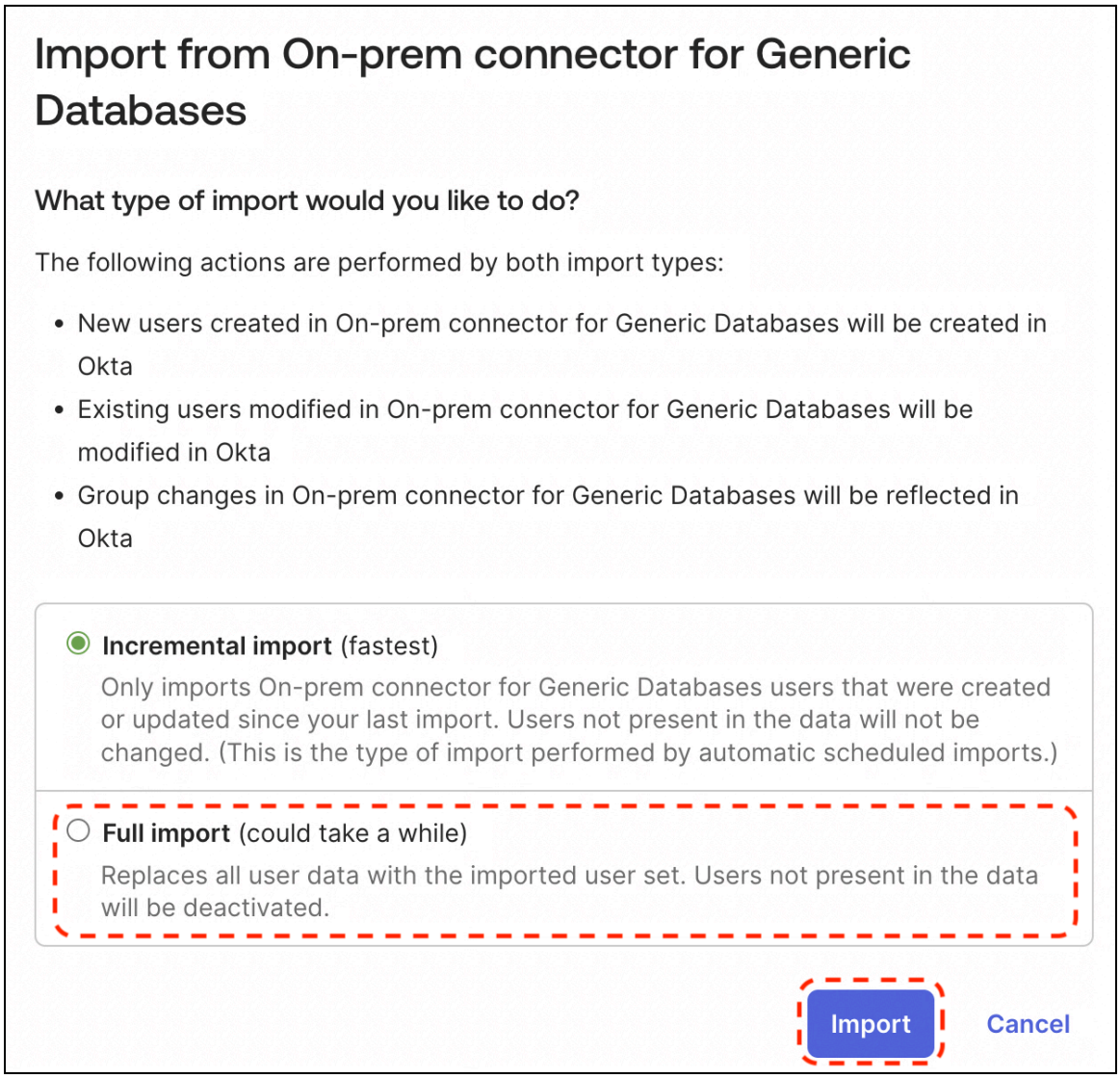
Execute Import

Now the Import configuration has been updated and we can run an import operation for this application to bring the users from the database into Okta. Follow below steps to perform the import:

1. Navigate to the **Import** tab under the **On-prem Connector for Generic Databases** application.
2. Select **Import Now** and it will ask to select the type of import.

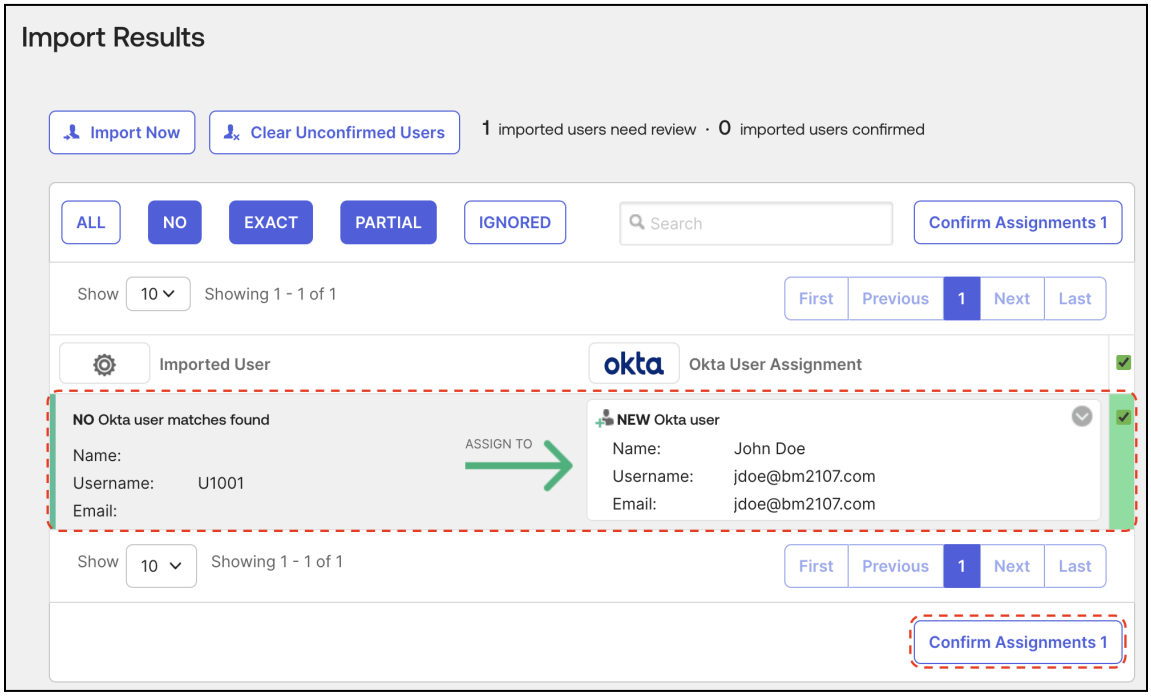


3. Select **Full Import**, and click on the **Import** button.



4. Once the import has successfully completed, it will present an Import Success message.
5. The users imported from the database system into Okta, will be presented. (Since, we have not configured auto confirmation, so we need to manually confirm

these users. These can be changed in the Import Settings)



6. Select the users which need to be imported into Okta and Select **Confirm Assignments.**
7. Select **Confirm**, when asked for reconfirmation.
8. Now, the newly imported user can be verified, by checking under **Assignments** tab.

Testing

Now that we have had the On-prem Connector for Generic Databases application integrated and configured successfully, we can test out its capabilities. We will test the first use case, which is manual assignment of a user to the Oracle Database application.

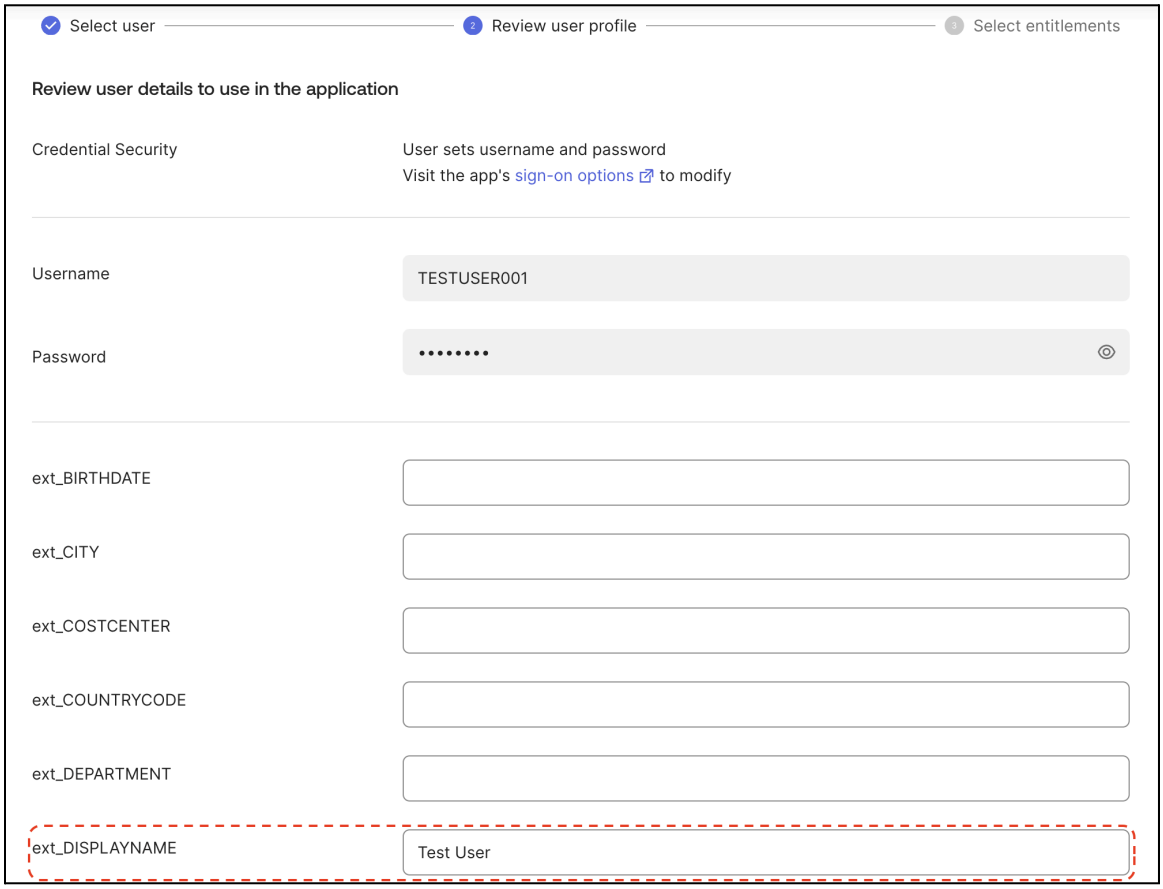
Manual Assignment of User to On-prem Connector for Generic Databases

In this use case, an Okta Administrator will logon to Okta Admin console and assign a user to the newly created On-prem Connector for Generic Databases application. The administrator will also assign some entitlements to this user from within Okta. On successful assignment, this user will be created in the Oracle Database and it should have the assigned entitlements reflected in Oracle database as well.

To assign a new user to On-prem Connector for Generic Databases, follow below steps:

1. Logon to Okta Admin Console.
2. Navigate to Applications and select On-prem Connector for Generic Databases application.

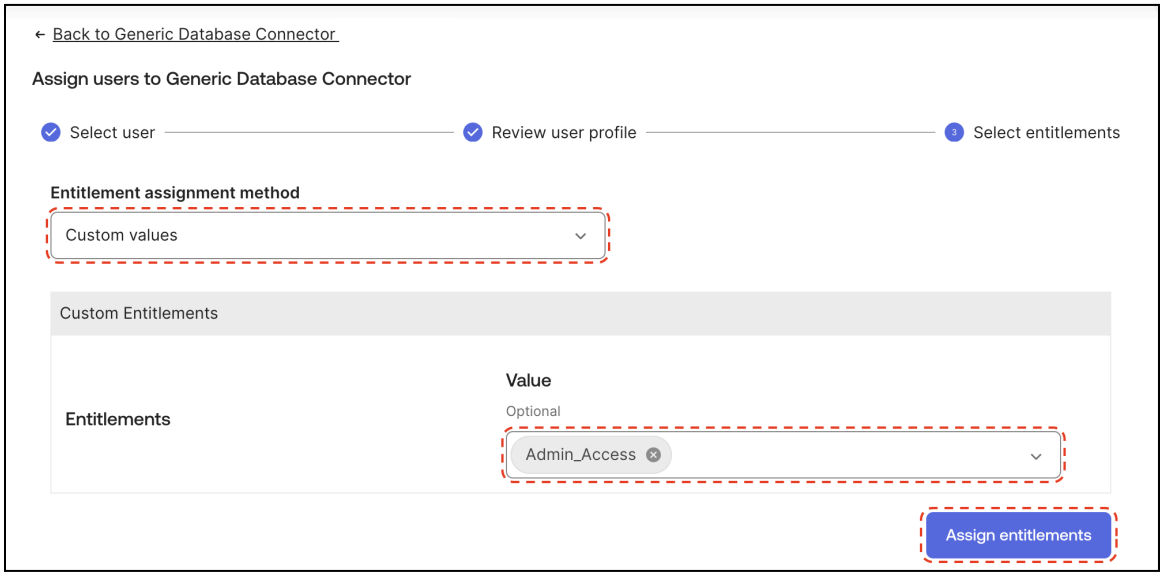
3. Select the Assignments tab within the On-prem Connector for Generic Databases application.
4. Select Assign and then select Assign User.
5. Search for the User.
   In this doc, we are assigning user **TestUser001@mycompany.com** to the On-prem Connector for Generic Databases application.
6. Select the **Assign** button next to the user.
7. In the **Review user details to use in application**, in which the application auto-populates the values for custom attributes as per the mapping we have done.



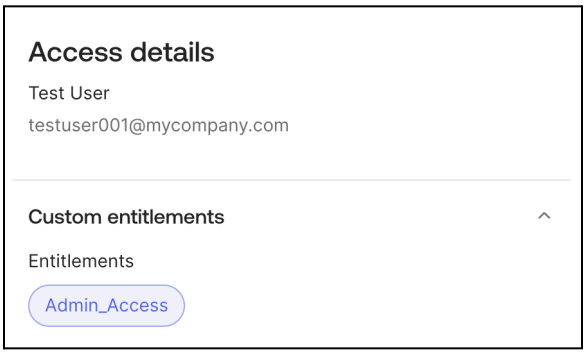
(Note: The empty field can be manually entered into the application.)

8. Select Assign and Continue.
9. The next screen presents the Entitlement Assignment section.
10. Select **Custom Values** under the drop-down menu for Entitlement Assignment method.
11. In the **Entitlements**, select **Admin Access** from the drop-down menu under Values.

12. Select Assign entitlements.



13. The user TestUser001@mycompany.com should be visible now under the Assignments tab.
14. The entitlements assigned to the user can be viewed by clicking the menu button(three vertical dots) beside the user and selecting View access details. It will provide a view of all the responsibilities assigned to the user.



Now to verify that the user has been created in On-prem Connector for Generic Databases and the assigned entitlements in Okta are getting reflected in the Oracle Database system as well, we will logon to the Oracle Database system either via SQL Developer or via SQL*PLUS and verify that the user has been created and necessary entitlements assigned to it using below queries:

```
SQL
SQL> SELECT USERNAME FROM USERS;
USERNAME
--------------------------------------------------------------
Jdoe
testuser001@mycompany.com
```

Troubleshooting

This section is dedicated to providing guidance on resources available to debug issues with On-prem Connector for Generic Databases and its underlying On-prem SCIM Server. It aims to assist administrators in efficiently diagnosing and resolving issues, thereby ensuring the seamless and dependable operation of the Okta Provisioning Agent and Okta On-prem Connector. This builds upon the architectural understanding, system requirements, and detailed integration steps delineated in the preceding sections.

How to check the status of Okta On-prem SCIM Server?

The status of the Okta On-prem SCIM Server service can be checked and verified using below command:

```
Shell
[user@ip-X-X-X-X /]$ sudo systemctl status OktaOnPremScimServer
● OktaOnPremScimServer.service - Okta On-Prem SCIM Server
     Loaded: loaded (/usr/lib/systemd/system/OktaOnPremScimServer.service;
enabled; preset: disabled)
     Active: active (running) since Mon 2025-09-08 12:17:58 UTC; 22h ago
   Main PID: 4827 (java)
      Tasks: 38 (limit: 22311)
     Memory: 433.1M
        CPU: 1min 53.836s
     CGroup: /system.slice/OktaOnPremScimServer.service
             └─4827 java -Xms512m -Xmx4096m -XX:+UseG1GC
-XX:+ExitOnOutOfMemoryError
-Dloader.main=com.okta.server.scim.ScimServerApplication -Dloader.path=/opt/Okta>
Sep 08 12:17:58 ip-10-0-0-153.ap-south-1.compute.internal systemd[1]: Started Okta
On-Prem SCIM Server.
[user@ip-X-X-X-X /]$
```

How to restart the Okta On-prem SCIM Server?

The Okta On-prem SCIM Server service can be restarted using below command:

```
Shell
[user@ip-X-X-X-X /]$ sudo systemctl restart OktaOnPremScimServer
```

What are the logs available for Okta On-prem SCIM Server?

The Okta On-prem SCIM Server provides below application logs as part of the service, which are located **`/var/log/OktaOnPremScimServer/`**

1. service.err
2. service.out
3. application.log

How to uninstall/remove the Okta On-prem SCIM Server?
The Okta On-prem SCIM Server is an rpm based installation, so we can uninstall the rpm using the yum utility. Below is the command to uninstall the server.

(**Caution:** If you uninstall/remove the Okta On-prem SCIM Server, it will stop the connection between Okta and On-prem database system, thereby stopping all the user and entitlement related operations.)

Command: yum remove OktaOnPremScimServer
Example:

```
Shell
[user@ip-X-X-X-X /]$ yum remove OktaOnPremScimServer
This system is not registered with an entitlement server. You can use "rhc" or
"subscription-manager" to register.
Dependencies resolved.
================================================================================
================================================================================
 Package                                Architecture                Version
Repository                        Size
================================================================================
================================================================================
Removing:
 OktaOnPremScimServer                   x86_64
0.0.1-SNAPSHOT20250902071027                    @@System                     26 M
Transaction Summary
================================================================================
================================================================================
Remove  1 Package
Freed space: 26 M
Is this ok [y/N]: y
Running transaction check
Running transaction test
Transaction test succeeded.
.
Installed products updated.
Removed:
  OktaOnPremScimServer-0.0.1-SNAPSHOT20250902071027.x86_64
Complete!
[user@ip-X-X-X-X /]$
```

Appendix A: Using Stored Procedures

A stored procedure is a set of pre-compiled SQL statements, saved under a name, and stored within a database. Instead of repeatedly writing and sending complex queries to the database for specific tasks, you can simply invoke the stored procedure by its name. The Okta On-prem SCIM Server leverages stored procedures for both Import and Provisioning operations.

The On-prem Connector for Generic Databases can invoke stored procedures, using a placeholder '?' within the procedure call as a variable. This variable is configurable within Parameter Mapping and can represent any column derived from the User or Entitlement Table. This method also supports using a CURSOR for parameter passing to the procedure. The CURSOR signifies that the procedure is expected to return data row-by-row, populating a result set that the Connector processes as individual records, which can then be mapped to corresponding Okta attributes. This functionality is beneficial for complex data retrieval scenarios not easily accomplished with basic SQL SELECT statements.

Note: In this document, we will be following steps to configure and use the **Stored Procedures** for **Oracle Database**. The steps will be similar for other database systems as well and can be used for them with little to no modifications.

This appendix details the creation and storage of database procedures, and their configuration within the On-prem Connector for Generic Databases for invocation. This process builds upon the Database Schema outlined in the "Before You Begin" section.

Create and Store Database Procedures

1. Launch the server hosting the database and connect to the database.
2. Initialize the SQLPlus and provide the necessary user credentials.
3. Store the **Get Active Users** procedure in the database system.

```SQL
CREATE OR REPLACE PROCEDURE GET_ACTIVEUSERS(p_cursor OUT SYS_REFCURSOR) AS
BEGIN
  OPEN p_cursor FOR
    SELECT
      USER_ID, USERNAME, FIRSTNAME, LASTNAME, MIDDLENAME, HONORIFICPREFIX,
EMAIL, DISPLAYNAME, NICKNAME, MOBILEPHONE, STREETADDRESS, CITY, STATE,
ZIPCODE, COUNTRYCODE, POSTALADDRESS, TIMEZONE, DEPARTMENT, MANAGERID,
JOBTITLE, WORKLOCATION, EMERGENCYCONTACT, PASSWORD_HASH, IS_ACTIVE,
```

```
COSTCENTER, MANAGER, TITLE, HIREDATE, TERMINATIONDATE, BIRTHDATE,
EMPLOYEENUMBER
    FROM USERS
    WHERE IS_ACTIVE = 1;
END;
```

Note: The Procedures shown in this document are for reference purpose only. Please build and use procedure based on the Database and the schema it uses.

4. Store the **Get All Entitlements** procedure in the database system by executing the below SQL command.

```
SQL
CREATE OR REPLACE PROCEDURE GET_ALL_ENTITLEMENTS(p_cursor OUT
SYS_REFCURSOR) AS
BEGIN
  OPEN p_cursor FOR
    SELECT ENT_ID, ENT_NAME, ENT_DESCRIPTION FROM ENTITLEMENTS;
END;
```

5. Store the **Get User by ID** procedure in the database system by executing the below SQL command.

```
SQL
CREATE OR REPLACE PROCEDURE GET_USER_BY_ID(p_user_id IN VARCHAR2, p_cursor
OUT SYS_REFCURSOR) AS
BEGIN
  OPEN p_cursor FOR
    SELECT USER_ID, USERNAME, FIRSTNAME, LASTNAME, MIDDLENAME,
HONORIFICPREFIX, EMAIL, DISPLAYNAME, NICKNAME, MOBILEPHONE, STREETADDRESS,
CITY, STATE, ZIPCODE, COUNTRYCODE, POSTALADDRESS, TIMEZONE, DEPARTMENT,
MANAGERID, JOBTITLE, EMERGENCYCONTACT, PASSWORD_HASH, IS_ACTIVE,
COSTCENTER, MANAGER, TITLE, HIREDATE, TERMINATIONDATE, BIRTHDATE,
EMPLOYEENUMBER
   FROM USERS WHERE USER_ID = p_user_id;
END;
```

6. Store the **Get User Entitlement** procedure in the database system by executing the below SQL command.

```
SQL
CREATE OR REPLACE PROCEDURE GET_USER_ENTITLEMENT(p_user_id IN VARCHAR2,
p_cursor OUT SYS_REFCURSOR) AS
BEGIN
  OPEN p_cursor FOR
    SELECT UE.USERENTITLEMENTID, UE.USER_ID, U.USERNAME, U.EMAIL,
UE.ENT_ID, E.ENT_NAME, E.ENT_DESCRIPTION, UE.ASSIGNEDDATE
    FROM USERENTITLEMENTS UE
    JOIN USERS U ON UE.USER_ID = U.USER_ID
    JOIN ENTITLEMENTS E ON UE.ENT_ID = E.ENT_ID
    WHERE UE.USER_ID = p_user_id;
END;
```

7. Store the **Create User** procedure in the database system by executing the below SQL command.

```
SQL
CREATE OR REPLACE PROCEDURE CREATE_USER (
  p_user_id IN VARCHAR2,
  p_username IN VARCHAR2,
  p_firstname IN VARCHAR2,
  p_lastname IN VARCHAR2,
  p_middlename IN VARCHAR2,
  p_honorificprefix IN VARCHAR2,
  p_email IN VARCHAR2,
  p_displayname IN VARCHAR2,
  p_nickname IN VARCHAR2,
  p_mobilephone IN VARCHAR2,
  p_streetaddress IN VARCHAR2,
  p_city IN VARCHAR2,
  p_state IN VARCHAR2,
  p_zipcode IN VARCHAR2,
  p_countrycode IN VARCHAR2,
  p_postaladdress IN VARCHAR2,
  p_timezone IN VARCHAR2,
  p_department IN VARCHAR2,
  p_managerid IN VARCHAR2,
  p_jobtitle IN VARCHAR2,
```

```
  p_worklocation IN VARCHAR2,
  p_emergencycontact IN VARCHAR2,
  p_password_hash IN VARCHAR2,
  p_costcenter IN VARCHAR2,
  p_manager IN VARCHAR2,
  p_title IN VARCHAR2,
  p_hiredate IN VARCHAR2,
  p_terminationdate IN VARCHAR2,
  p_birthdate IN VARCHAR2,
  p_employeenumber IN VARCHAR2
) AS
BEGIN
  INSERT INTO USERS (
    USER_ID, USERNAME, FIRSTNAME, LASTNAME, MIDDLENAME, HONORIFICPREFIX,
    EMAIL, DISPLAYNAME, NICKNAME, MOBILEPHONE, STREETADDRESS, CITY, STATE,
ZIPCODE, COUNTRYCODE, POSTALADDRESS, TIMEZONE, DEPARTMENT, MANAGERID,
JOBTITLE, WORKLOCATION, EMERGENCYCONTACT, PASSWORD_HASH, COSTCENTER,
MANAGER, TITLE, HIREDATE, TERMINATIONDATE, BIRTHDATE, EMPLOYEENUMBER)
  VALUES (
    p_user_id, p_username, p_firstname, p_lastname, p_middlename,
p_honorificprefix, p_email, p_displayname, p_nickname, p_mobilephone,
p_streetaddress, p_city, p_state, p_zipcode, p_countrycode,
p_postaladdress, p_timezone, p_department, p_managerid, p_jobtitle,
p_worklocation, p_emergencycontact, p_password_hash, p_costcenter,
p_manager, p_title, p_hiredate, p_terminationdate, p_birthdate,
p_employeenumber);
END;
```

8. Store the **Update Users** procedure in the database system by executing the following SQL command.

```
SQL
CREATE OR REPLACE PROCEDURE UPDATE_USER (
  p_user_id IN VARCHAR2,
  p_username IN VARCHAR2,
  p_firstname IN VARCHAR2,
  p_lastname IN VARCHAR2,
  p_middlename IN VARCHAR2,
  p_honorificprefix IN VARCHAR2,
  p_email IN VARCHAR2,
  p_displayname IN VARCHAR2,
```

```
  p_nickname IN VARCHAR2,
  p_mobilephone IN VARCHAR2,
  p_streetaddress IN VARCHAR2,
  p_city IN VARCHAR2,
  p_state IN VARCHAR2,
  p_zipcode IN VARCHAR2,
  p_countrycode IN VARCHAR2,
  p_postaladdress IN VARCHAR2,
  p_timezone IN VARCHAR2,
  p_department IN VARCHAR2,
  p_managerid IN VARCHAR2,
  p_jobtitle IN VARCHAR2,
  p_worklocation IN VARCHAR2,
  p_emergencycontact IN VARCHAR2,
  p_password_hash IN VARCHAR2,
  p_costcenter IN VARCHAR2,
  p_manager IN VARCHAR2,
  p_title IN VARCHAR2,
  p_hiredate IN VARCHAR2,
  p_terminationdate IN VARCHAR2,
  p_birthdate IN VARCHAR2,
  p_employeenumber IN VARCHAR2
) AS
BEGIN
  UPDATE USERS SET
    USERNAME = p_username, FIRSTNAME = p_firstname, LASTNAME = p_lastname,
MIDDLENAME = p_middlename, HONORIFICPREFIX = p_honorificprefix, EMAIL =
p_email, DISPLAYNAME = p_displayname, NICKNAME = p_nickname, MOBILEPHONE =
p_mobilephone, STREETADDRESS = p_streetaddress, CITY = p_city, STATE =
p_state, ZIPCODE = p_zipcode, COUNTRYCODE = p_countrycode, POSTALADDRESS =
p_postaladdress, TIMEZONE = p_timezone, DEPARTMENT = p_department,
MANAGERID = p_managerid, JOBTITLE = p_jobtitle, WORKLOCATION =
p_worklocation, EMERGENCYCONTACT = p_emergencycontact, PASSWORD_HASH =
p_password_hash, COSTCENTER = p_costcenter, MANAGER = p_manager, TITLE =
p_title, HIREDATE = p_hiredate, TERMINATIONDATE = p_terminationdate,
BIRTHDATE = p_birthdate, EMPLOYEENUMBER = p_employeenumber
  WHERE USER_ID = p_user_id;
END;
```

9. Store the **Activate User** procedure in the database system by executing the following SQL commands.

```
SQL
CREATE OR REPLACE PROCEDURE ACTIVATE_USER(p_user_id IN VARCHAR2) AS
BEGIN
```

```
  UPDATE USERS SET IS_ACTIVE = 1 WHERE USER_ID = p_user_id;
END;
```

10. Store the **Deactivate User** procedure in the database system by executing the following SQL commands.

```
SQL
CREATE OR REPLACE PROCEDURE DEACTIVATE_USER(p_user_id IN VARCHAR2) AS
BEGIN
  UPDATE USERS SET IS_ACTIVE = 0 WHERE USER_ID = p_user_id;
END;
```

11. Store the **Add Entitlement to User** procedure in the database system by executing the following SQL commands.

```
SQL
CREATE OR REPLACE PROCEDURE ADD_ENTITLEMENT_TO_USER(p_user_id IN VARCHAR2,
p_ent_id IN VARCHAR2) AS
BEGIN
  INSERT INTO USERENTITLEMENTS (USERENTITLEMENTID, USER_ID, ENT_ID,
ASSIGNEDDATE)
  VALUES (SYS_GUID(), p_user_id, p_ent_id, SYSDATE);
END;
```

12. Store the **Remove Entitlement from User** procedure in the database system by executing the following SQL commands.

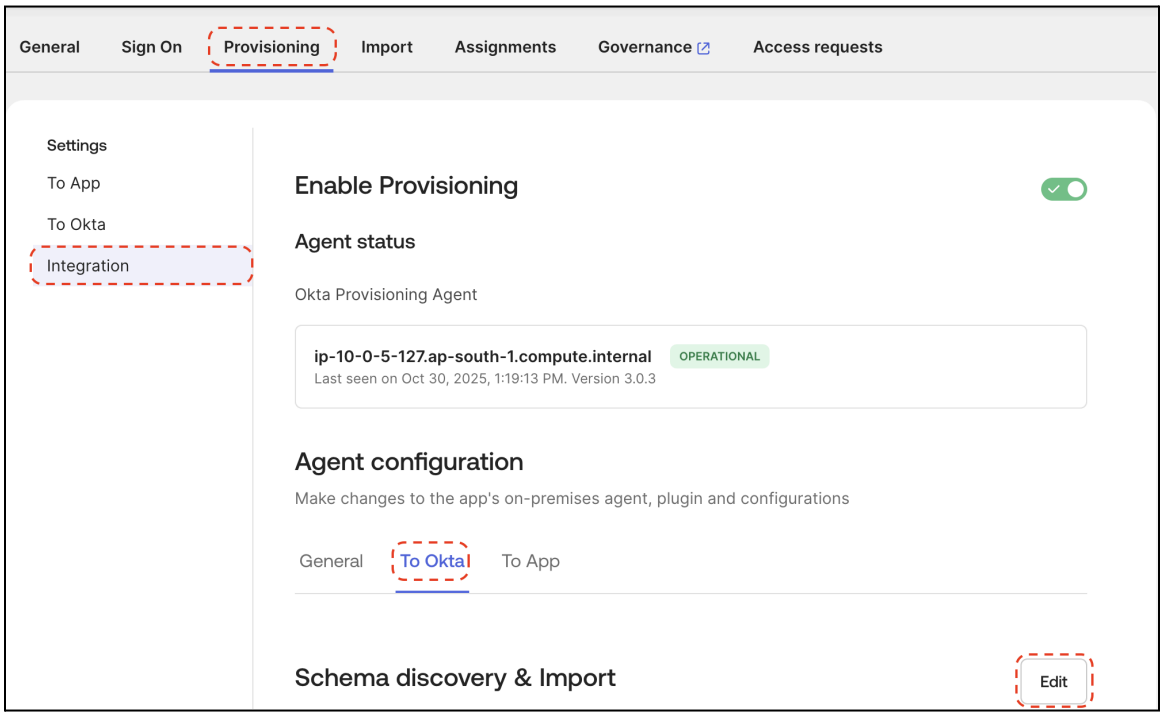
```
SQL
CREATE OR REPLACE PROCEDURE REMOVE_ENTITLEMENT_FROM_USER(p_user_id IN
VARCHAR2, p_ent_id IN VARCHAR2) AS BEGIN
  DELETE FROM USERENTITLEMENTS WHERE USER_ID = p_user_id AND ENT_ID =
p_ent_id;
END;
```

Configure and Execute Stored Procedures

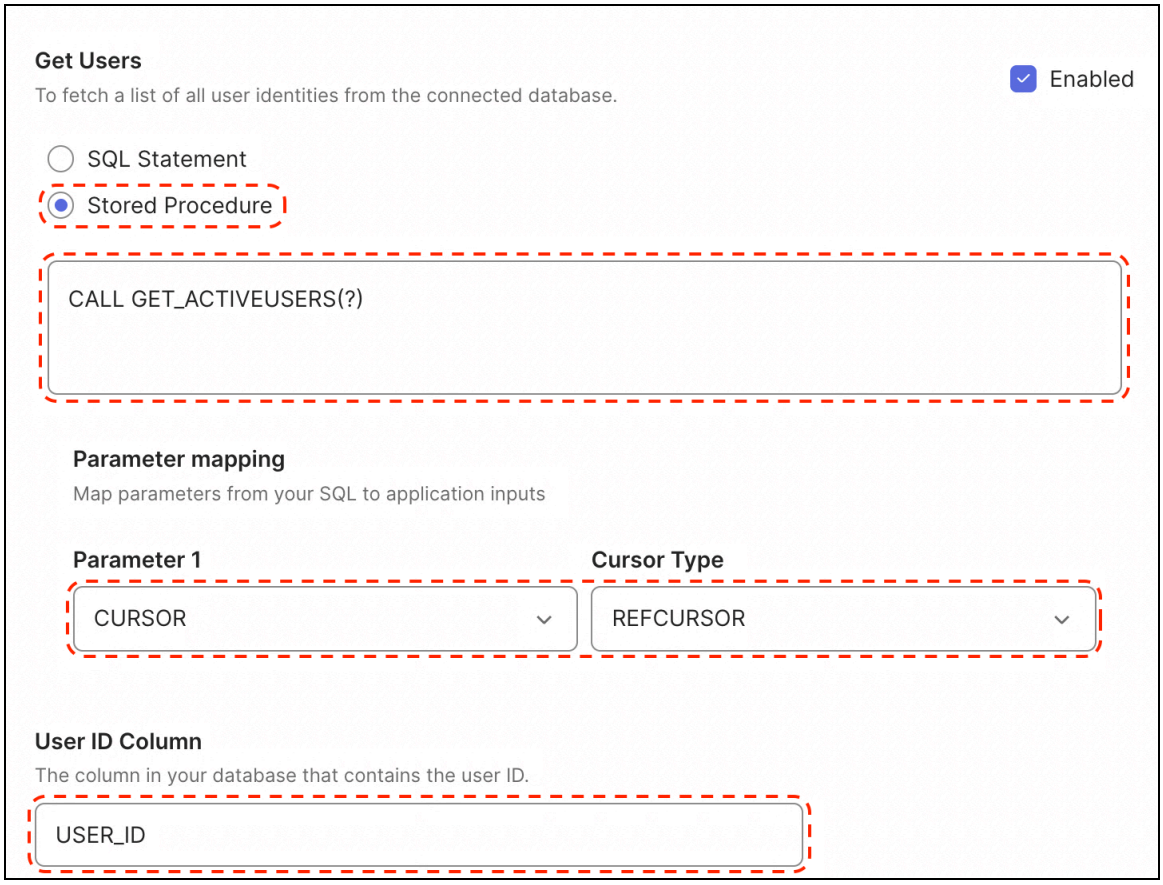
After successfully storing the procedures in the database system, the On-prem Connector for Generic Databases will be configured to call out the Stored Procedure for the respective operations. As there are two different segments of operations involved with the On-prem Connector for Generic Databases, we will be looking at each of them separately.

Import Operations:

1. Navigate to the "**Provisioning**" section of the "**On-prem Connector for Generic Databases**" application on the Okta Admin Console.
2. Go to "**Integration**" and select "**To Okta**" for import operations or "**To App**" for provisioning operations.
3. Select "**Edit**" against "**Schema discovery & Import**".

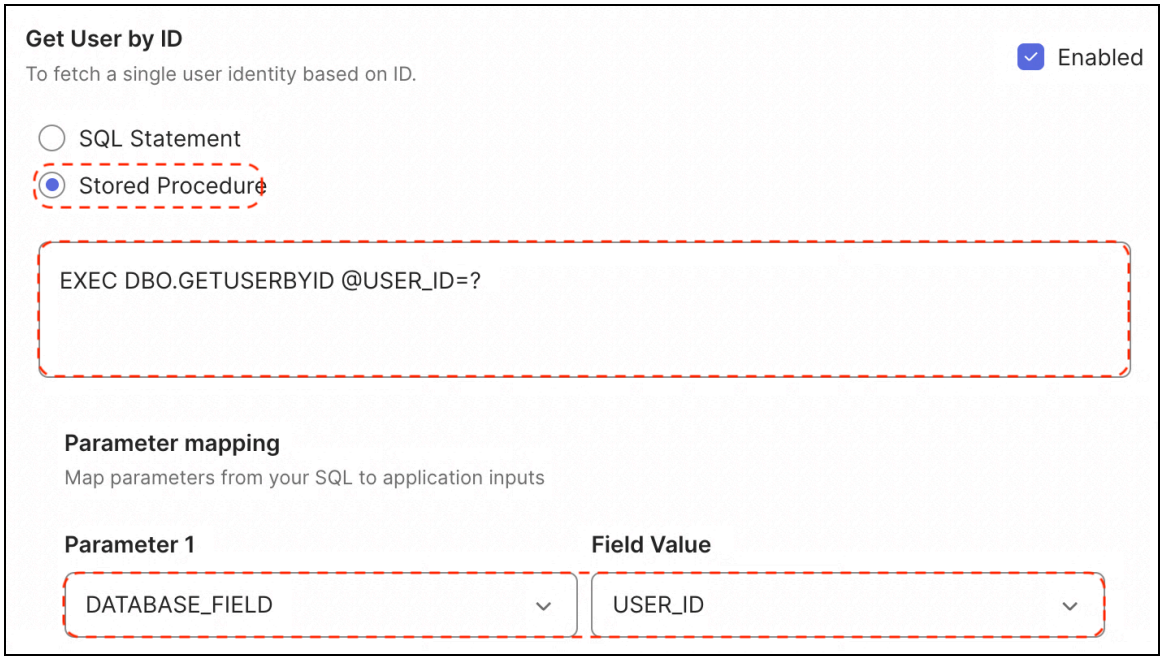


4. For **Get Users** operation, check the "**Enabled**" checkbox.
    a. Select **Stored Procedure**.
    b. Enter "**CALL GET_ACTIVEUSERS(?)**" in the textbox below.
    c. Select "**CURSOR**" from the dropdown menu under Parameter 1.
    d. Select "**REFCURSOR**" from the dropdown menu under CursorType.
    e. Enter "**USER_ID**" in the textbox below the User ID Column.

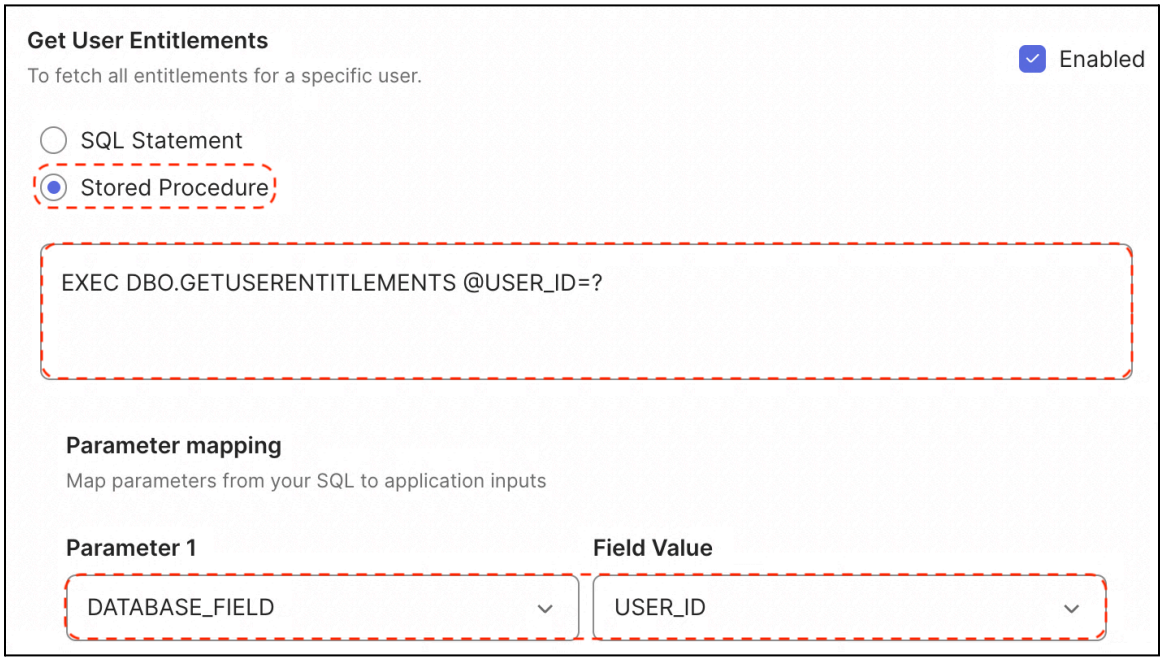


5. For the **Get All Entitlements** operation, check the "**Enabled**" checkbox.
    a. Select **Stored Procedure**.
    b. Enter "**CALL GET_ALL_ENTITLEMENTS(?)**" in the textbox below.
    c. Select "**CURSOR**" from the dropdown menu under Parameter 1.
    d. Select "**REFCURSOR**" from the dropdown menu under CursorType.
    e. Enter "**ENT_ID**" in the textbox below the Entitlement ID Column.
    f. Enter "**ENT_NAME**" in the textbox below the Entitlement Display Column.

6. For **Get User by ID** operation, check the "**Enabled**" checkbox.
    a. Select **Stored Procedure**.
    b. Enter "**CALL GET_USER_BY_ID(?,?)**" in the textbox below.
    c. Select "**DATABASE_FIELD**" in the textbox against Parameter 1.
    d. Select "**USER_ID**" from the dropdown menu under Field Value.

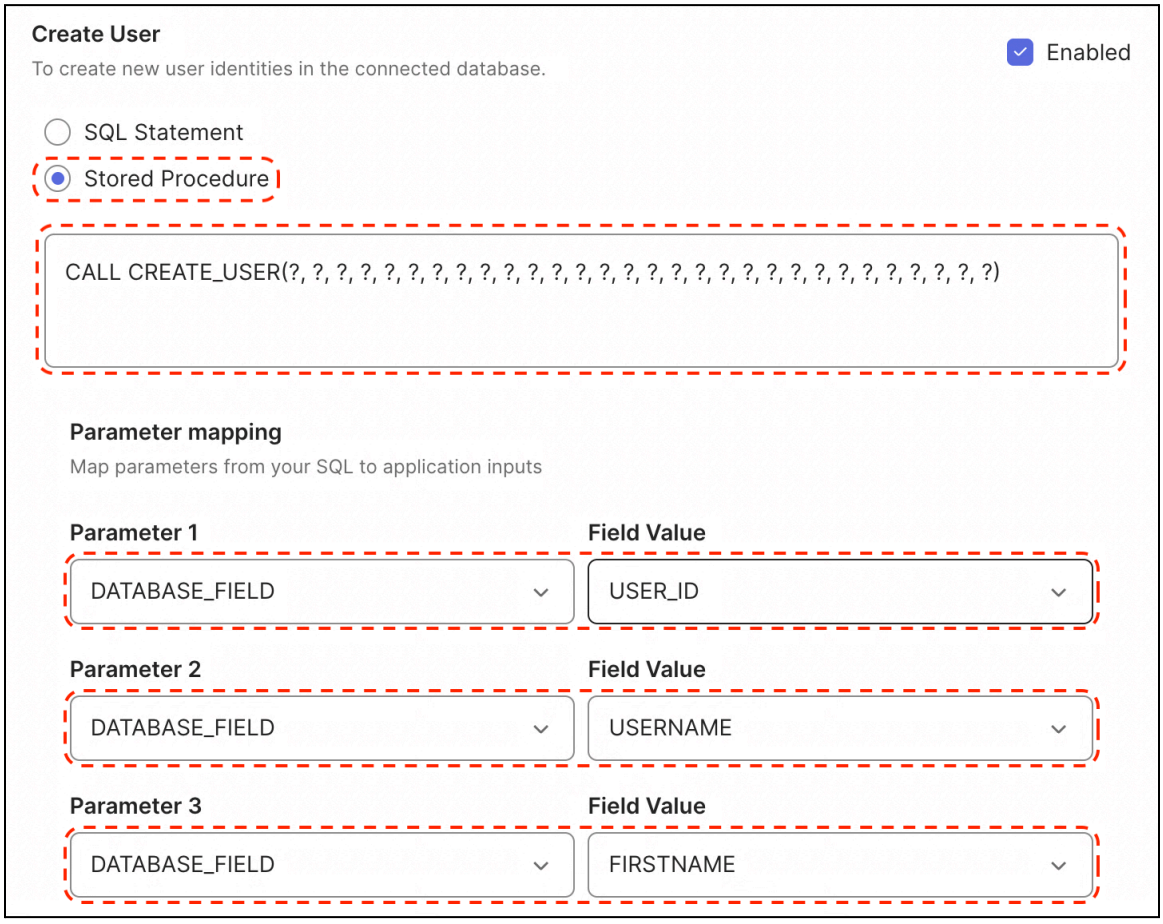


7. For the **Get User Entitlements** operation, check the "**Enabled**" checkbox.
    a. Select **Stored Procedure**.
    b. Enter "**CALL GET_USER_ENTITLEMENT(?,?)**" in the textbox below.
    c. Select "**DATABASE_FIELD**" in the textbox under Parameter 1.
    d. Select "**USER_ID**" from the dropdown menu under Field Value.



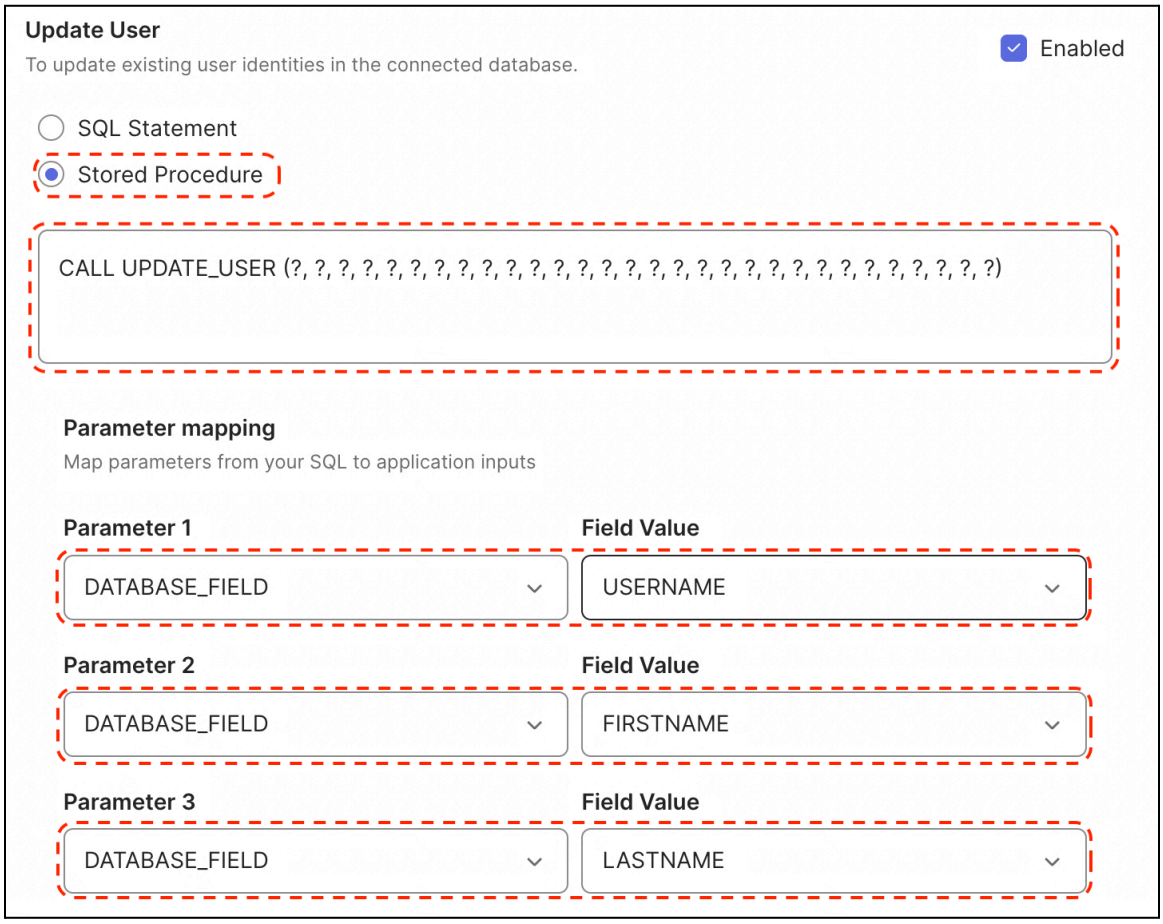
Provisioning Operations:

1. Navigate to "**To App**" within "**Integration**" of the "**Provisioning**" section.
2. Select "**Edit**" next to "Provisioning."
3. For the **Create User** operation, check the "**Enabled**" checkbox.
    a. Select **Stored Procedure**.
    b. Enter "**CALL CREATE_USER(?, ?)**" in the textbox below.
    c. Select "**DATABASE_FIELD**" in the textbox against Parameter 1.
    d. Select "**USER_ID**" from the dropdown menu under Field Value.
    e. In the similar manner, Select "**DATABASE_FIELD**" in the textbox against Parameter and select the respective user attribute under Field Value.

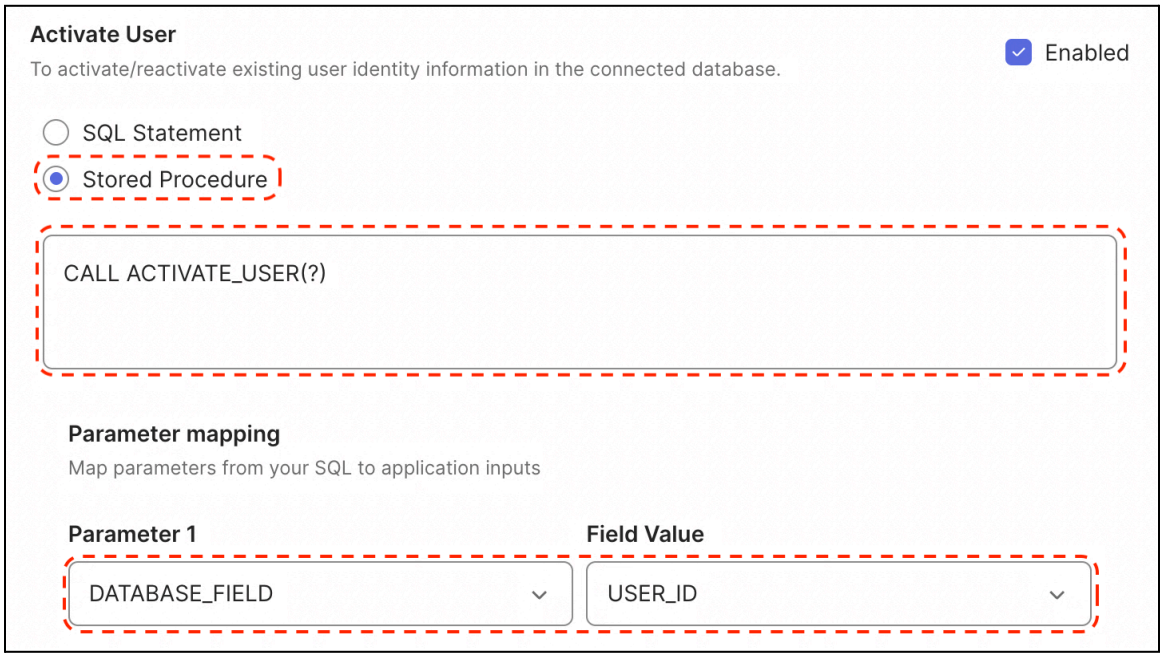


4. For the **Update User** operation, check the "**Enabled**" checkbox.
    a. Select **Stored Procedure**.
    b. Enter "**CALL CREATE_USER(?, ?)**" in the textbox below.
    c. Select "**DATABASE_FIELD**" in the textbox against Parameter 1.

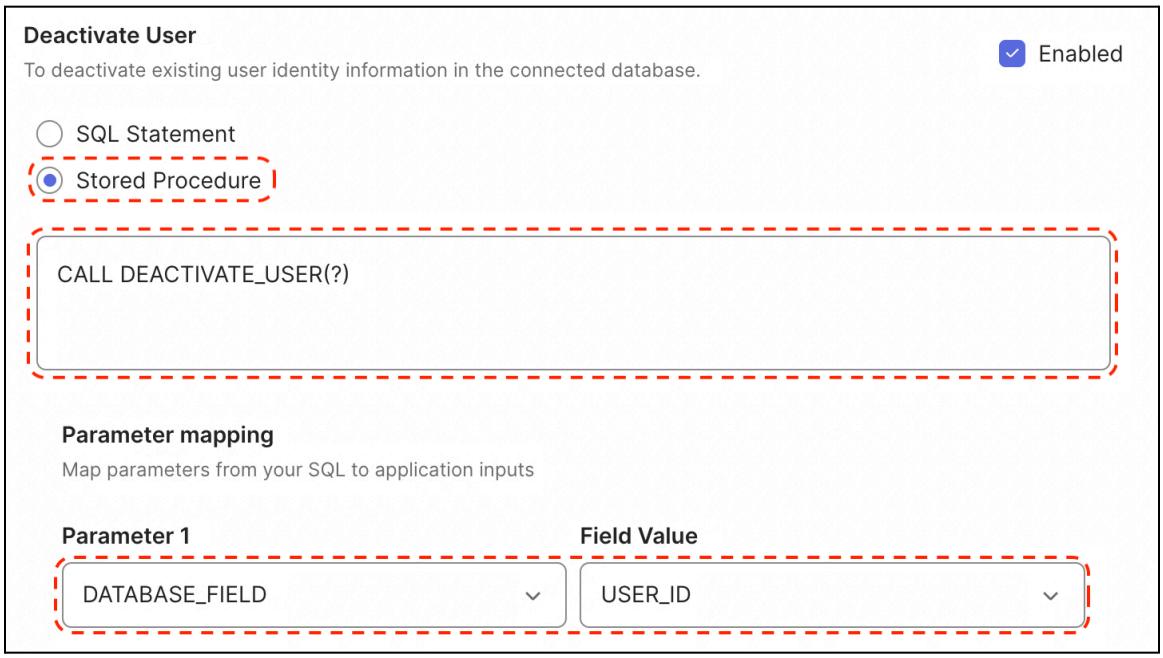
d. Select “**USER_ID**” from the dropdown menu under Field Value.
e. In the similar manner, Select “**DATABASE_FIELD**” in the textbox against Parameter and select the respective user attribute under Field Value.



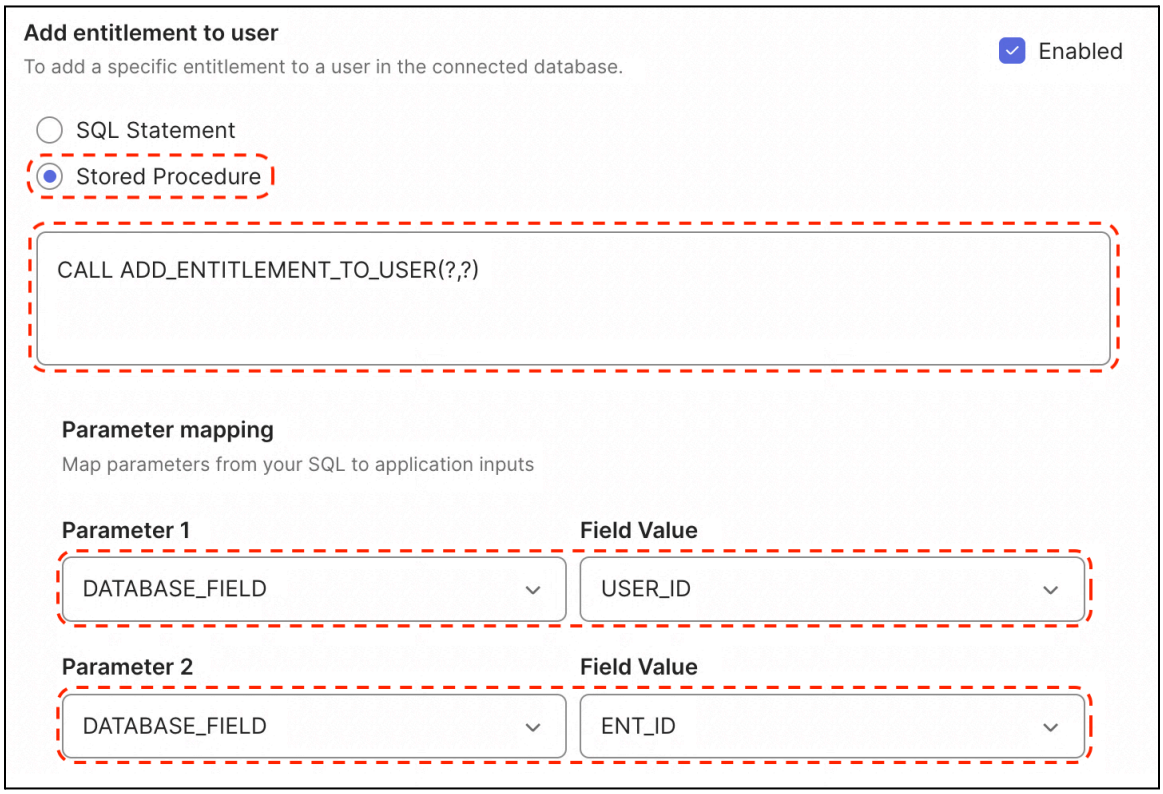
5. For the **Activate User** operation, check the "**Enabled**" checkbox.
   a. Select **Stored Procedure**.
   b. Enter “**CALL ACTIVATE_USER(?)**” in the textbox below.
   c. Select “**DATABASE_FIELD**” in the textbox against Parameter 1.
   d. Select “**USER_ID**” from the dropdown menu under Field Value.



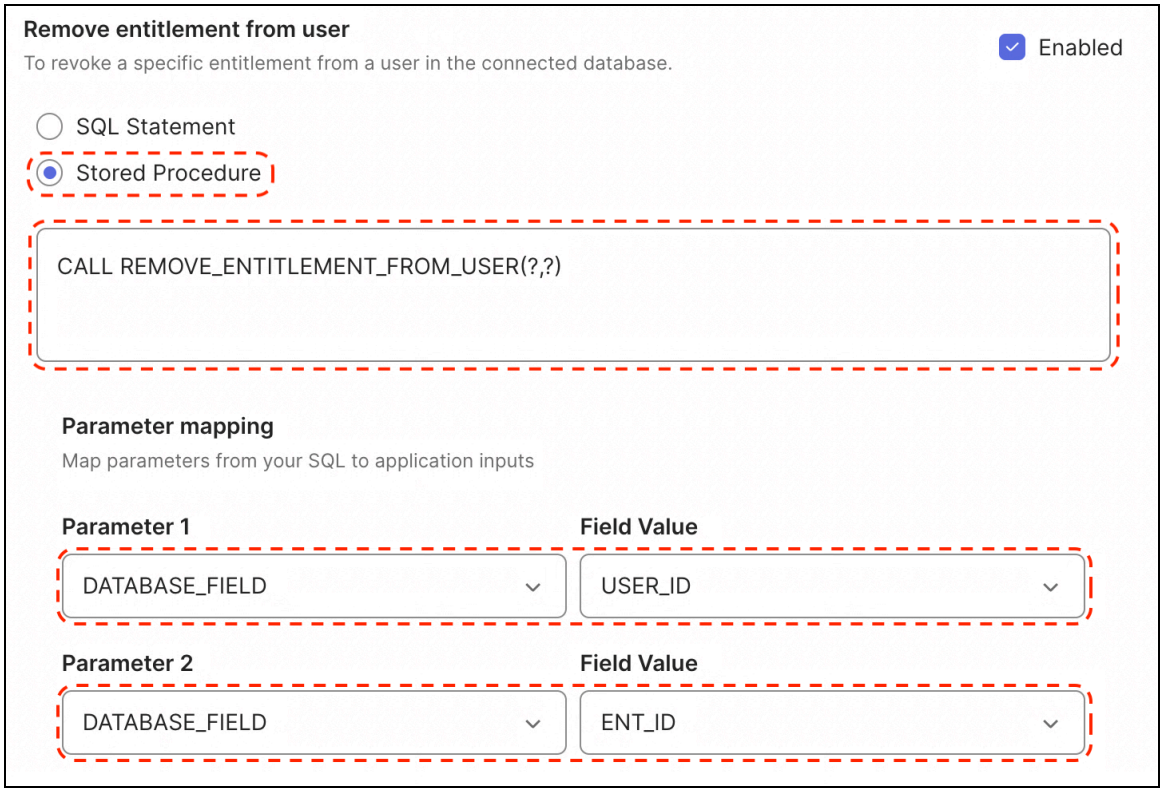
6. For the **Deactivate User** operation, check the "**Enabled**" checkbox.
    a. Select **Stored Procedure**.
    b. Enter "**CALL DEACTIVATE_USER(?)**" in the textbox below.
    c. Select "**DATABASE_FIELD**" in the textbox against Parameter 1.
    d. Select "**USER_ID**" from the dropdown menu under Field Value.



7. For the **Add Entitlement to User** operation, check the "**Enabled**" checkbox.
    a. Select **Stored Procedure**.
    b. Enter "**CALL ADD_ENTITLEMENT_TO_USER(?,?)**" in the textbox below.
    c. Select "**DATABASE_FIELD**" in the textbox against Parameter 1.
    d. Select "**USER_ID**" from the dropdown menu under Field Value.
    e. Select "**DATABASE_FIELD**" in the textbox against Parameter 2.
    f. Select "**ENT_ID**" from the dropdown menu under Field Valu



8. For the **Remove Entitlement from User** operation, check the "**Enabled**" checkbox.
    a. Select **Stored Procedure**.
    b. Enter "**CALL REMOVE_ENTITLEMENT_FROM_USER(?,?)**" in the textbox below.
    c. Select "**DATABASE_FIELD**" in the textbox against Parameter 1.
    d. Select "**USER_ID**" from the dropdown menu under Field Value.
    e. Select "**DATABASE_FIELD**" in the textbox against Parameter 2.
    f. Select "**ENT_ID**" from the dropdown menu under Field Value.


Remove entitlement from user
To revoke a specific entitlement from a user in the connected database.
Enabled
SQL Statement
Stored Procedure
CALL REMOVE_ENTITLEMENT_FROM_USER(?,?)
Parameter mapping
Map parameters from your SQL to application inputs
Parameter 1
Field Value
DATABASE_FIELD
USER_ID
Parameter 2
Field Value
DATABASE_FIELD
ENT_ID

Appendix B: Using Custom Code

This appendix provides detailed instructions on how to configure and use custom Java code for provisioning operations within the Okta On-prem Connector for Generic Databases. While SQL statements and stored procedures offer robust functionalities, custom code allows for the implementation of complex logic and tailored interactions with your database systems, extending the capabilities of user management and entitlement management. It will guide you through the necessary steps to develop, deploy, and integrate your custom Java code (in JAR file format) with the On-prem Connector for Generic Databases application on the Okta Admin Console.

Note: This section and its steps are generic and applicable for all the database systems which are supported by Okta's Generic Database Connector.

Generate Key Pair

The key is used to provide the public key of the customer's certificate. The public key ensures the integrity and authenticity of the custom code by verifying that the JAR file has been signed by the customer's certificate and has not been tampered with since it was built.

(**Note:** If there is a key already present with you, which can be used to sign the JAR then jump to the next section.)

To generate a new key, follow below steps:

1. Navigate to the following location on the Linux Server, where the Okta On-prem SCIM Server is deployed: **/opt/OktaOnPremScimServer/userplugin**

```
None
[user@ip-X-X-X-X /]$ cd /opt/OktaOnPremScimServer/userplugin
```

2. Use the keytool utility to generate a new key pair using the below command.

```
None
[user@ip-X-X-X-X userplugin]$ keytool -genkeypair \
  -alias mykey \
  -keyalg RSA \
  -keysize 2048 \
  -validity 365 \
  -keystore mykeystore.jks \
  -storepass ******* \
  -keypass ******* \
  -dname "CN=User Name, OU=Security, O=CompanyName, C=US"
```

```
Generating 2,048 bit RSA key pair and self-signed certificate (SHA384withRSA) with
a validity of 365 days
      for: CN=User Name, OU=Security, O=MyCompany, C=US
[user@ip-X-X-X-X userplugin]$
```

3. Export the certificate from the keystore.
   When prompted for keystore password, provide the password used in the previous step to generate the key pair.

```
None
[user@ip-X-X-X-X userplugin]$ keytool -exportcert -alias mykey \
  -keystore mykeystore.jks \
  -rfc > mycert.pem
Enter keystore password:  *******
[user@ip-X-X-X-X /]$ ls -ltr
total 8
-rw-r--r--. 1 root root 2662 Sep  9 11:31 mykeystore.jks
-rw-r--r--. 1 root root 1189 Sep  9 11:32 mycert.pem
[user@ip-X-X-X-X userplugin]$ cp mycert.pem /tmp
[user@ip-X-X-X-X userplugin]$ cd /tmp
[user@ip-X-X-X-X tmp]$ ll -ltr mycert.pem
-rw-r--r--. 1 root root 1189 Sep  9 11:32 mycert.pem
[user@ip-X-X-X-X tmp]$
```

4. Download the certificate to the local system and store it in a secure location.
   This certificate will be used during the configuration of Custom Code.

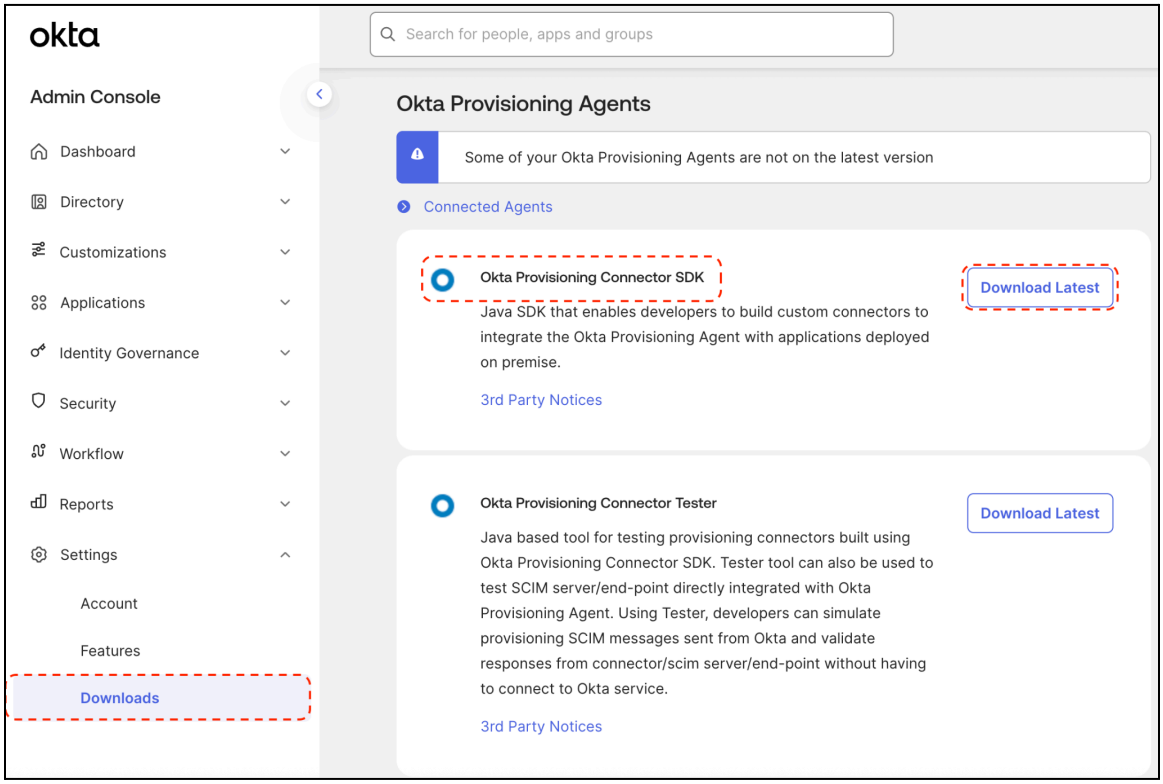
```
None
user@localsystem Downloads % scp -i "linux.pem" user@ip-X-X-X-X:/tmp/mycert.pem
~/Downloads
mycert.pem                                    100% 1189    18.9KB/s   00:00
user@localsystem Downloads %
```

Downloading and Importing SCIM SDK

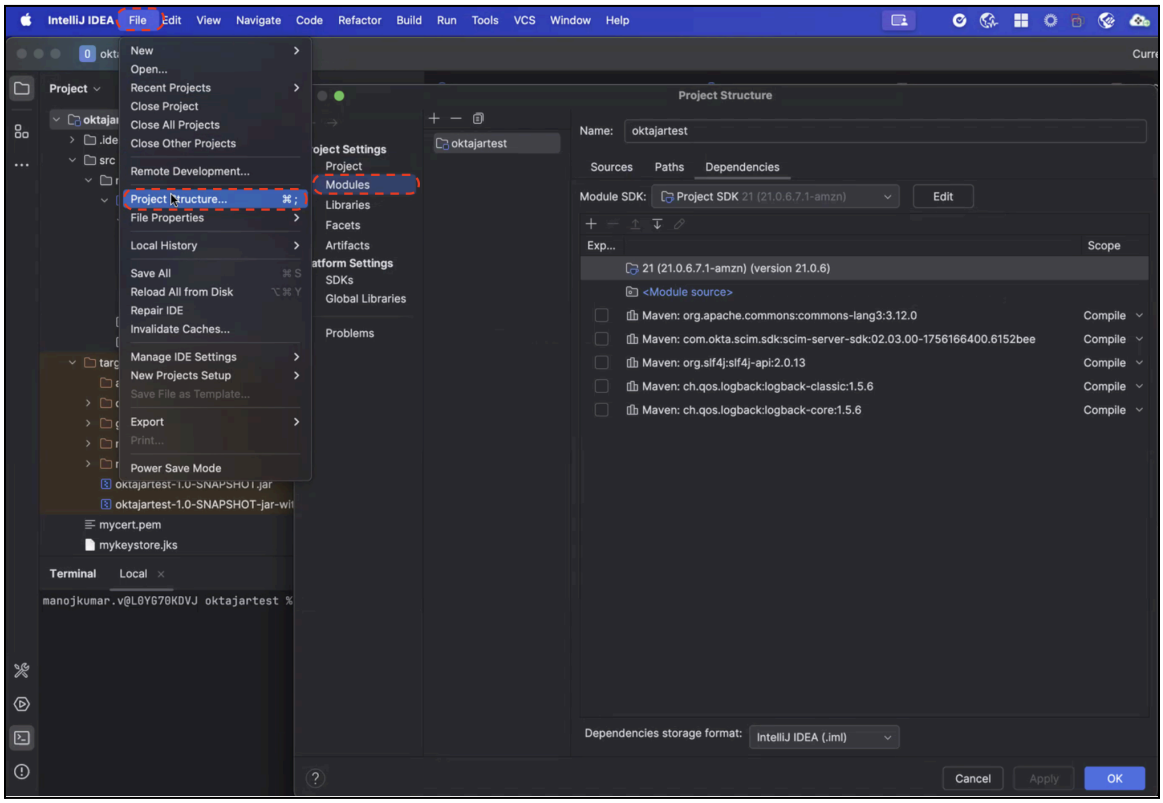
In order to build the custom code package, we need the latest version of SCIM SDK to be integrated with the package. The SCIM Server SDK is part of Okta Provisioning Connector SDK, which is present in Okta Admin Console. Follow below steps to integrate the SDK with your code:

1. Logon to the Okta Admin Console.
2. Navigate to Settings and then select Downloads.

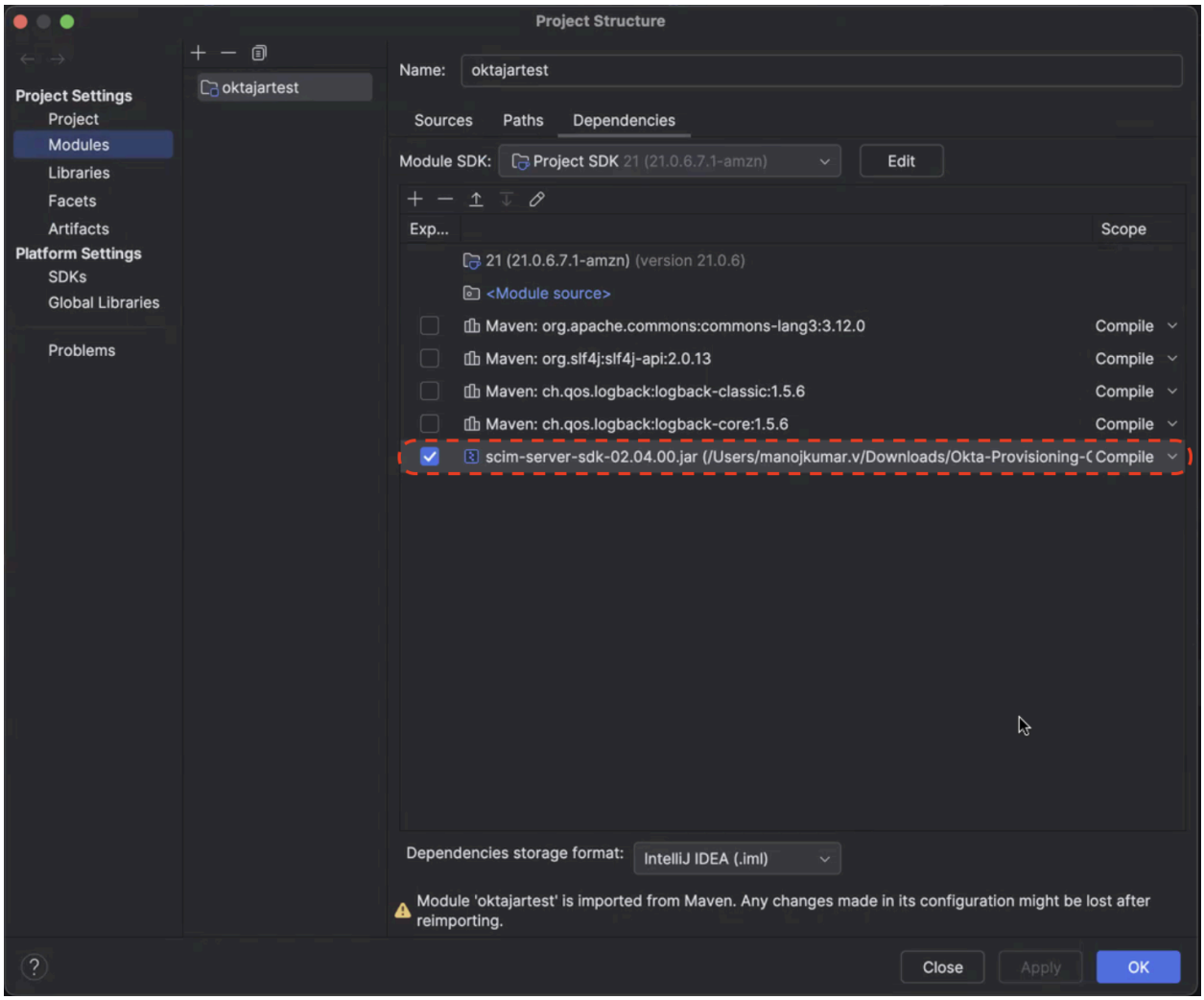
3. Search for Okta Provisioning Connectory SDK and select **Download Latest.**



4. Unzip the **Okta-Provisioning-Connector-SDK**-02.04.00-36817d6.zip on your workstation.
5. Open your IDE being used for developing the custom code.
   (Note: In this doc, we will be referring IntelliJ)
6. Navigate to File > Project Structure > Modules.
7. Select "**+**" to add the new SCIM SDK module.


IntelliJ IDEA
File
Edit
View
Navigate
Code
Refactor
Build
Run
Tools
VCS
Window
Help
New
Open...
Recent Projects
Close Project
Close All Projects
Close Other Projects
Remote Development...
Project Structure...
File Properties
Local History
Save All
Reload All from Disk
Repair IDE
Invalidate Caches...
Manage IDE Settings
New Projects Setup
Save File as Template...
Export
Print...
Power Save Mode
Project
mycert.pem
mykeystore.jks
Terminal
Local
Project Structure
Project Settings
Project
Modules
Libraries
Facets
Artifacts
Platform Settings
SDKs
Global Libraries
Problems
oktajartest
Name:
oktajartest
Sources
Paths
Dependencies
Module SDK:
Project SDK 21 (21.0.6.7.1-amzn)
Edit
Exp...
Scope
21 (21.0.6.7.1-amzn) (version 21.0.6)
<Module source>
Maven: org.apache.commons:commons-lang3:3.12.0
Maven: com.okta.scim.sdk:scim-server-sdk:02.03.00-1756166400.6152bee
Maven: org.slf4j:slf4j-api:2.0.13
Maven: ch.qos.logback:logback-classic:1.5.6
Maven: ch.qos.logback:logback-core:1.5.6
Compile
Dependencies storage format:
IntelliJ IDEA (.iml)
Cancel
Apply
OK

8. In the File Explorer window, navigate to the location where the Okta-Provisioning-Connector-SDK is present.



9. Select **OK**.
10. After adding the SDK to your project, you can build the logic for user provisioning using the Java class that implements the OktaDbOperationPlugin interface.
11. Import the necessary classes as shown below:

    import com.okta.connector.jdbc.OktaDbOperationPlugin;
    import com.okta.connector.jdbc.UserObject;

12. Next, create a class that implements the **OktaDbOperationPlugin** interface and provides the implementation for the createUser and updateUser methods.
    (The example below uses a placeholder for your database connection and logic)

```java
import com.okta.connector.jdbc.OktaDbOperationPlugin;
import com.okta.connector.jdbc.UserObject;

import java.sql.Connection;

public class Test implements OktaDbOperationPlugin {
    @Override
    public UserObject createUser(Connection connection, UserObject user) {
        // Build your custom code
    }

    @Override
    public UserObject updateUser(Connection connection, UserObject user) {
        // Build your custom code
    }
}
```

Build JAR for the Custom Code

Once the java code has been developed and ready to be used, place the code on the server, execute below steps to build the jar with the custom code.

1. Create a directory on the server to store the custom code.

```shell
[user@ip-X-X-X-X /]$ cd /installer/
[user@ip-X-X-X-X installers]$ mkdir oktatestjar
[user@ip-X-X-X-X installers]$ cd oktatestjar
[user@ip-X-X-X-X oktatestjar]$
```

2. Upload and place the java code to below location on the Linux Server.

```shell
[user@ip-X-X-X-X customCodeProject]$ cp /tmp/oktatestjar/* /installer/oktatestjar/
[user@ip-X-X-X-X oktatestjar]$ ls -ltr
total 12
drwxr-xr-x. 3 root root   18 Aug 27 06:33 src
-rw-r--r--. 1 root root 2252 Sep  9 13:35 pom.xml
drwxr-xr-x. 7 root root 4096 Sep  9 14:12 target
[user@ip-X-X-X-X oktatestjar]$
```

3. In the same location, execute the below command to build the JAR package using maven.

   The maven tool must be installed on the server to build the package.

```
Shell
[user@ip-X-X-X-X oktatestjar]$  mvn clean package
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------------< com.example:oktajartest >-----------------------
[INFO] Building oktajartest 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ oktajartest ---
[INFO] Deleting /installers/oktajartest/target
.
.
[INFO] Building jar:
/installers/oktajartest/target/oktajartest-1.0-SNAPSHOT-jar-with-dependencies.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  6.630 s
[INFO] Finished at: 2025-09-09T14:09:35Z
[INFO] ------------------------------------------------------------------------
```

4. Sign the new jar with the previously generated key.

```
None
[user@ip-X-X-X-X oktatestjar]$ jarsigner \
       -keystore mykeystore.jks \
       -storepass ******* \
       -keypass ******* \
       /installers/customCodeProject/oktajartest/target/oktajartest-1.0-SNAPSHOT-ja
r-with-dependencies.jar \
       mykey

jar signed.
```

5. Generate SHA256sum for the new jar. It is used to provide the SHA256 fingerprint of the Java JAR file. This hash ensures the integrity and authenticity of the custom code, verifying that the JAR file has not been tampered with since it was built.
   This hash will be used during configuring the Custom Code option in application.

```
Shell
[user@ip-X-X-X-X oktatestjar]$ sha256sum
/installers/oktajartest/target/oktajartest-1.0-SNAPSHOT-jar-with-dependencies.jar
9015***********b0a19
/installers/oktajartest/target/oktajartest-1.0-SNAPSHOT-jar-with-dependencies.jar
```

6. Place the new jar to the **userplugin** location within the Okta OnPrem SCIM Server installation directory.

```
Shell
[user@ip-X-X-X-X oktatestjar]$ cp
/installers/oktajartest/target/oktajartest-1.0-SNAPSHOT-jar-with-dependencies.jar/
opt/OktaOnPremScimServer/userplugin
```

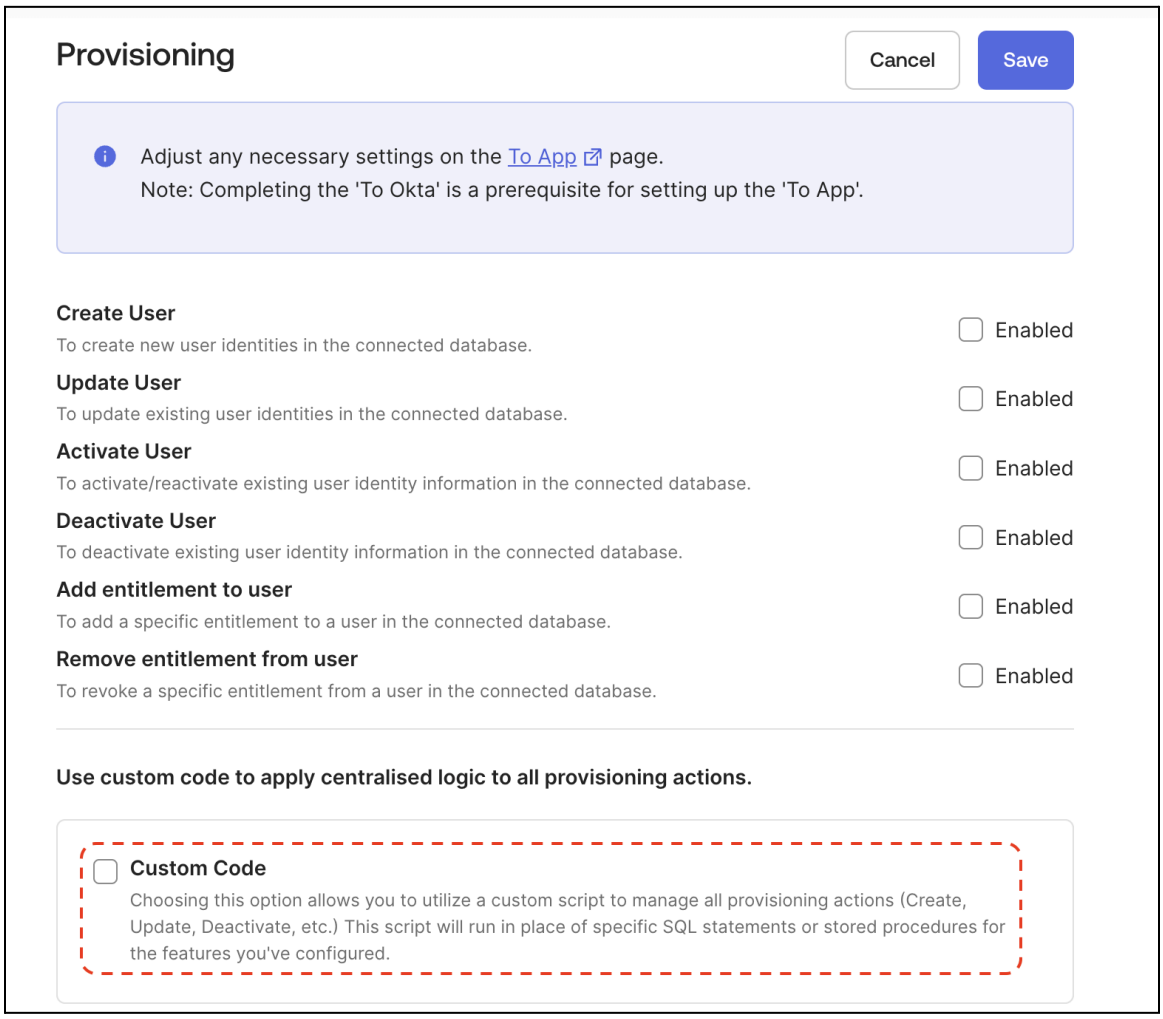
> **Note:** The Okta On-prem SCIM Server does not require a restart for the initial deployment of a JAR file to the **userplugin** location. However, any subsequent changes to the JAR file and its redeployment will necessitate a server restart.

Configuration for Custom Code

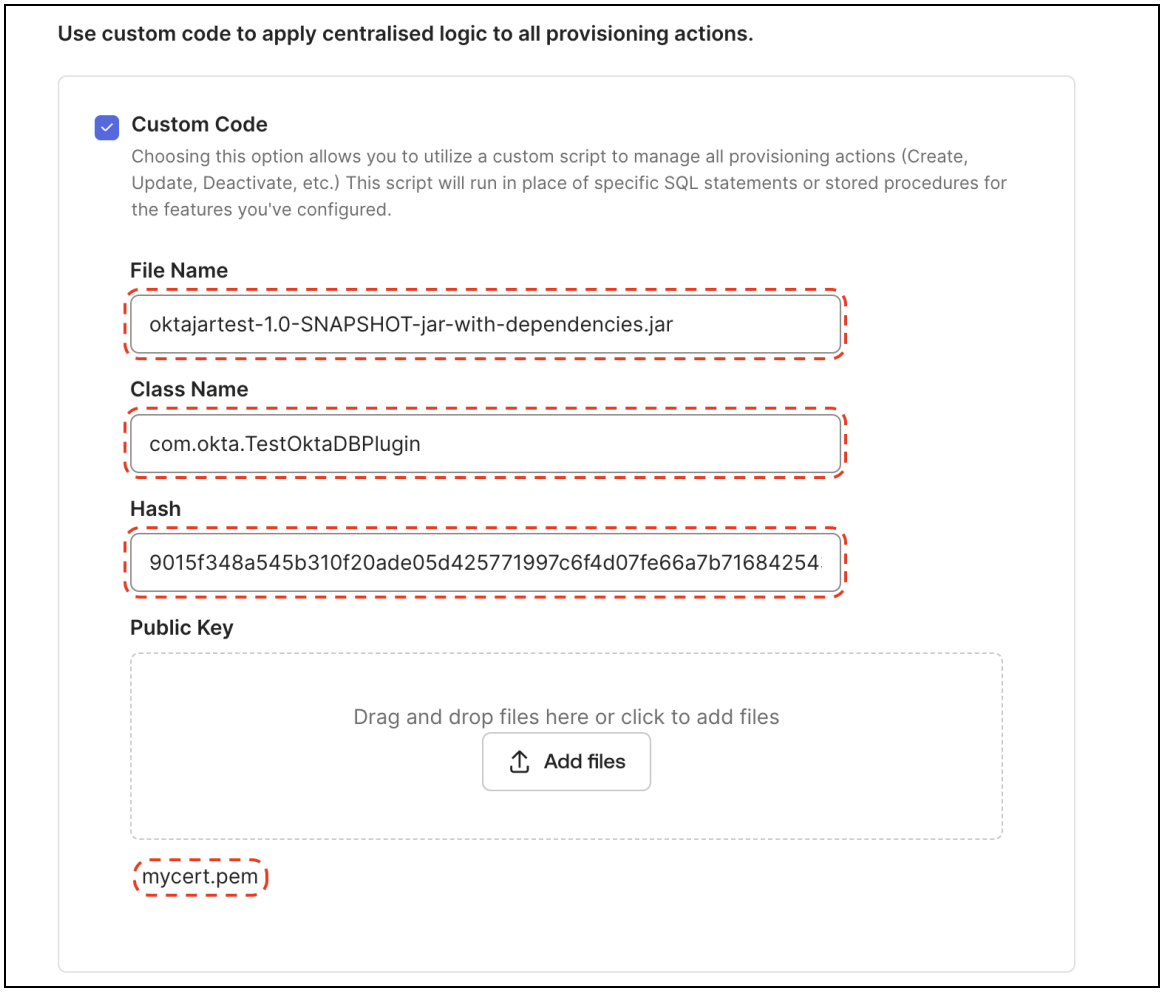
As mentioned earlier, Okta On-prem Connector for Generic Databases provides an option to use Custom Code for performing provisioning operations on the database systems. To use the Custom Code option for the provisioning operations follow below steps:

1. Navigate to **To App**, within **Integration** of the **Provisioning** section of this application.
2. Select **Edit** next to Provisioning.

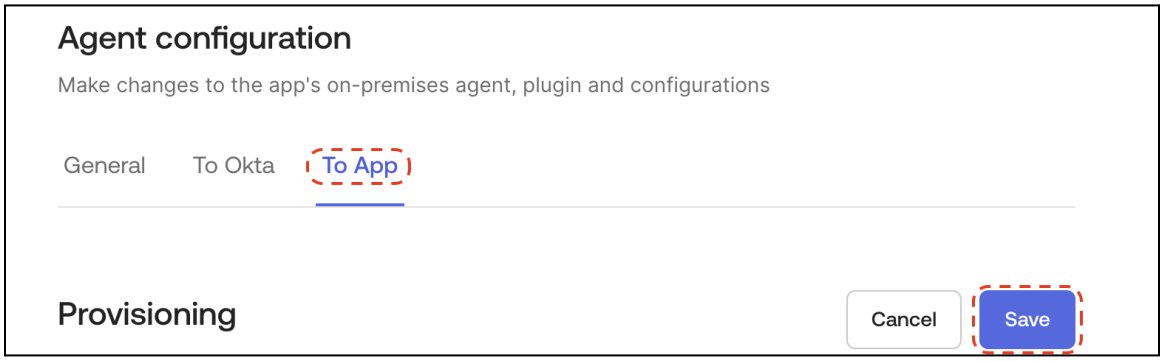
3. Under “Use custom code to apply centralized logic to all provisioning actions.”, check the **Custom Code** checkbox, which allows you to configure it.



4. Under File Name, provide the name of the jar file generated in the previous section(**Build JAR for the Custom Code**).
   (For this document, we have build a jar named “oktajartest-1.0-SNAPSHOT-jar-with-dependencies.jar”)
5. Under Class Name, provide the fully qualified class where your createUser and updateUser methods exist.
   (For this document, we are using the **com.okta.TestOktaDBPlugin**)
6. Under Hash, provide the hash generated in Step 5 of the **Build JAR for the Custom Code** section.
   (In this document, it is **9015**********b0a19**)
7. Under Public Key, select Add files.
8. Navigate to the location where the key file has been downloaded in step 4 of **Build JAR for the Custom Code** section.
   (In this document the certificate is **mycert.pem**)



9. Select Save.



10. On clicking Save, it will ask to re-authenticate using the configured MFA which needs to be completed to successfully save the configurations.

Once the configuration has been completed and the custom code settings have been saved successfully, continue with **Configuring User Lifecycle Management** here.

Appendix C: Using Connection URL for a Database system

There will be scenarios where the On-prem Connector for Generic Databases would need to connect with the on-prem database system using a **Connection URL** instead of IP and Port. For e.g. there are multiple instances of the Database System working together with a load balancer or where you need to connect to DB using Windows Authentication. In such a scenario, we leverage the Connection URL parameter supported by the connector. For configuring the On-prem Connector for Generic Databases using a connection URL, use these alternative steps.

> Note: **Generic Database Connector only allows to use the JDBC Connection URL for Oracle DB and MS SQL Server. Reach out to support to enable your org to use this feature.**

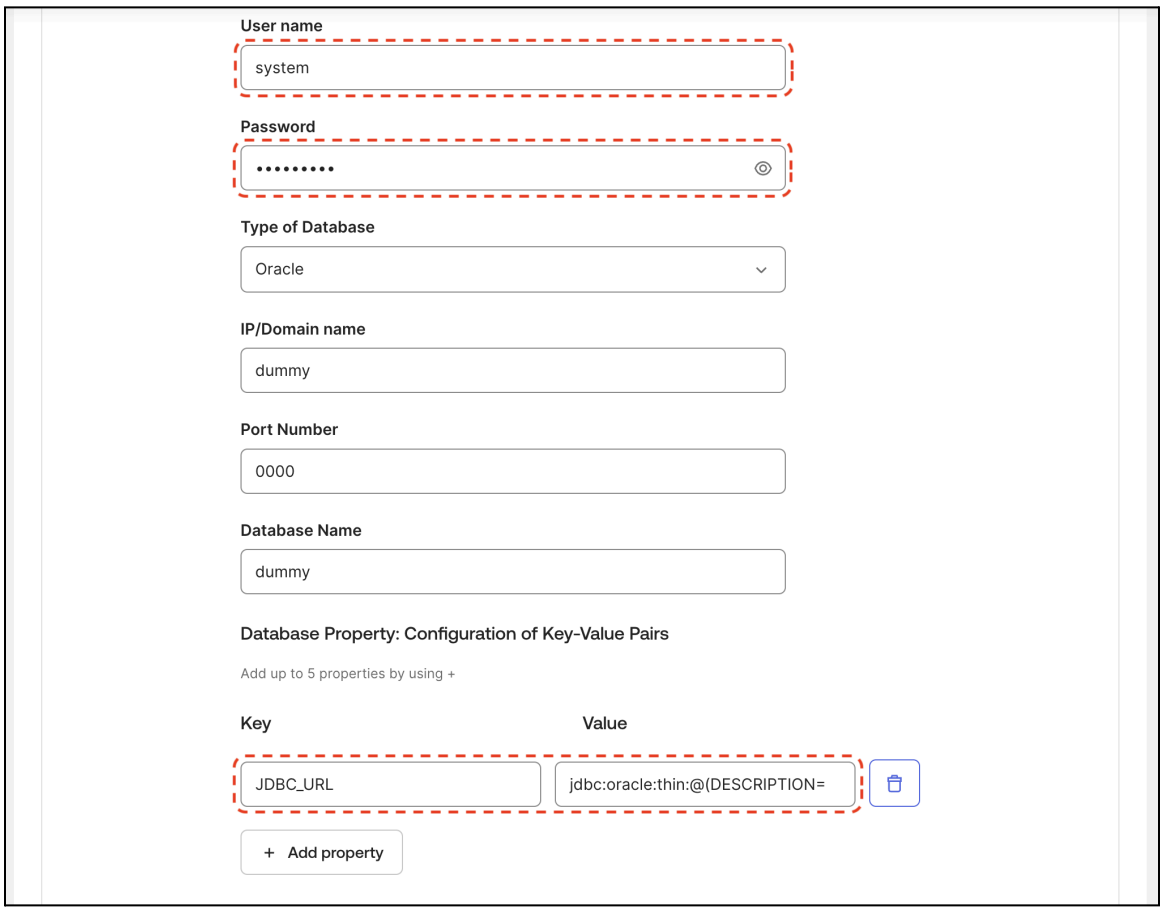
a. Provide username as the admin user which is "**system"**.
b. Provide the password for the "**system"** user.
c. Select type of Database as **Oracle or MS SQL Server**.
d. Provide IP/Domain name as **dummy**.
e. Provide the Port Number as **0000**.
f. Provide the Database Name as **dummy**.
g. Under the Database Property section.
    - Provide the Key as **JDBC_URL.**
    - Provide the value as the JDBC URL of the Database Loadbalancer. Below is a sample of the JDBC URL.

For **Oracle Database**, below is a sample JDBC Connection URL:

```
SQL
jdbc:oracle:thin:@(DESCRIPTION=
        (ADDRESS_LIST=
          (ADDRESS=(PROTOCOL=TCP)(HOST=host1)(PORT=1521))
          (ADDRESS=(PROTOCOL=TCP)(HOST=host2)(PORT=1521))
        )
        (LOAD_BALANCE=on)
        (CONNECT_DATA=(SERVICE_NAME=mydbservice))
    )
```

For **MS SQL Server**, below is a sample JDBC Connection URL:

```
SQL
jdbc:sqlserver://sql-prod.corp.local:1433;databaseName=testDB;encrypt=true;trustSe
rverCertificate=true;
```



- Select **Connect agents.**