# JDBC Framework

### Purpose of the Framework

The JDBC Framework is designed to serve as a robust utility for executing a wide array of provisioning tasks based on the provisioning plan provided.

### Supported Operations

1. Create.
2. Add Entitlement.
3. Remove Entitlement.
4. Enable.
5. Disable.
6. Delete.
7. Unlock.

**Prerequisites**

1. Custom object template of JDBC framework.
2. JDBCFramework class or JDBCFramwork Rule Library.

### **Template**

#### The below template is same for enable, disable, delete.



Each operation in the Custom Object template is associated with a specific key, such as 'insert'.

If there are multiple queries to be executed for a single operation, such as "insert," add multiple maps containing the required data in the List.

If the query is a "Stored Procedure," set the value to "true" as shown below:

<entry key="isPreparedStatement" value="true"/>

Add the parameters in the ‘List’. Ensure that every parameter you passing is provided in the ProvisioningPlan. If the parameter is native identity of the user, provide “nativeIdentity” as show in the below. The parameters are index based.

Ex:

“(?,?,?,?,?,?)”   
1st “?” will take input as “nativeIdentity”, 2nd “?”parameter will take “first\_name” as input

<entry key="parameters">

<value>

<List>

<String>nativeIdentity</String>

<String>first\_name</String>

<String>last\_name</String>

<String>full\_name</String>

<String>department</String>

<String>location</String>

</List>

</value>

</entry>

Add you are value to the “query” key as shown below.

<entry key="query" value="SELECT CreateEmployeeProcedure(?,?,?,?,?,?)"/>

#### The below template is same for Entitlement Add, Entitlement Remove, update attribute.



For every attribute in the particular application add an entry in the map as below.

<entry key="bundle\_id">

In the entry if there

If there are multiple queries to be executed for the attribute, such as "bundle\_id" add multiple maps containing the required data in the List.

#### Operations and Their Associated Custom Object Keys

|  |  |  |  |
| --- | --- | --- | --- |
| Account Request Operation | Attribute Request  Operation | Custom Object Key | Description |
| Create | - | insert | Creates a new account with the specified attributes and entitlements. |
| Create | Add | addEntitlement | Assigns a new entitlement to the account during creation. |
| Modify | Add | addEntitlement | Adds one or more entitlements to an existing account. |
| Modify | Remove | removeEntitlement | Removes one or more entitlements from an existing account. |
| Modify | Set | attributeUpdate | Updates specific attribute of an existing account. |
| Enable |  | enable | Enables a previously disabled account. |
| Disable |  | disable | Disables an account, preventing access without deleting it. |
| Delete |  | delete | Permanently deletes the account from the system. |
| Delete |  | removeAllEntitlements | Removes all entitlements from the account as part of the deletion process. |
| Unlock |  | unlock | Unlocks an account that has been locked due to security or policy reasons. |

#### Common Keys and Purpose

|  |  |
| --- | --- |
| Custom Object Key | Description |
| query | Specifies the SQL query or the stored procedure to be executed. |
| isPreparedStatement | Boolean flag indicating whether the query should be executed as a prepared statement (true or false). |
| Parameters | An ordered list of parameters to be supplied to the query or stored procedure at execution time. |

### Usage

The Framework is available as both a Class and Rule Library. This dual availability provides flexibility in how you can integrate and utilize the framework within your system. Both the approaches will return Provisioning Result.

#### Class-Based Approach

The Class-Based Approach involves creating a Java archive file and integrating it into your system. This method is beneficial for those who prefer working directly with Java classes and want to leverage the full capabilities of the Java environment.

1. Create a .jar File:

Compile your Java class into a .jar file. This can typically be done using a build tool or manually using the jar command in the Java Development Kit (JDK).

1. Add the .jar File:

Place the created .jar file into the “identityiq/WEB-INF/lib” folder. This ensures that the Java classes are available to the IdentityIQ application at runtime.

1. Invoke the Provision Method:

Use the following method call to execute the provisioning process. This method requires passing the necessary objects: SailPointContext, Connection, ProvisioningPlan, and CustomObject name.

JDBCOperation.provision(context, connection, plan, “Example Custom Object”);

#### Rule Library

The Rule Library-based approach is more straightforward compared to the Class-based approach, as it is native to the SailPoint environment. This method simplifies integration and leverages the built-in capabilities of SailPoint.

1. Create a New Instance:

Import or copy and paste the necessary code to create a new instance of the Rule Library within your SailPoint environment.

1. Reference the Rule Library:

Ensure that the Rule Library is properly referenced in your code to access the provision method.

1. Invoke the Provision Method:

Use the following method call to execute the provisioning process. This method requires passing the necessary objects: ProvisioningPlan, and CustomObject name.

provision(plan, “Example custom object”);

## Use Case Example

The execution of any use case with the Framework is highly dependent on how the provisioning plan is passed to it. Below is an example scenario illustrating this dependency.

Example 1: Managing Account Entitlements

Scenario:

If an account contains only one entitlement, do not remove it. Instead, mark the account as disabled.

Solution:

1. Count Entitlements:

* Determine the total number of entitlements associated with the identity for the particular application.

1. Modify Account Request:

* If the account contains only one entitlement, change the operation from Operation.Modify AccountRequest to Operation.Disable AccountRequest.

1. Provisioning Plan Handling:

* If the account has more than one entitlement, pass the ProvisioningPlan as it is to the framework for processing.