SQL Connector

Introduction

Description

The SQL connector allows an easy way to configure and manage relational databases.

Managed System

There are a lot of relational databases, currently, these are the supported databases.

- MySQL
- MariaDB
- PostgreSQL
- Oracle
- Informix
- IBM DB2/400
- Sybase
- ODBC

For more information: List of relational databases

> If your system is not in the previous list, it's possible to include it easily!
>
> For more information to check if your system may be synchronized with this connector you do not hesitate to contact us through our Contact form

Prerequisites

It is needed a user with access and permissions to the schemes and tables required in the scope of the integration.

To configure DB2/400 or Sybase it is mandatory to install the drivers in the lib directory of the Sync Server.

The Java-ODBC bridge is deprecated in Java, and the support will be removed shortly.

Download and Install

The SQL is part of the default connectors, you do not need to install it, but you can upgrade it from the download management section.

> You can visit the Connector Getting started page for more information about the installation process.

Agent Configuration

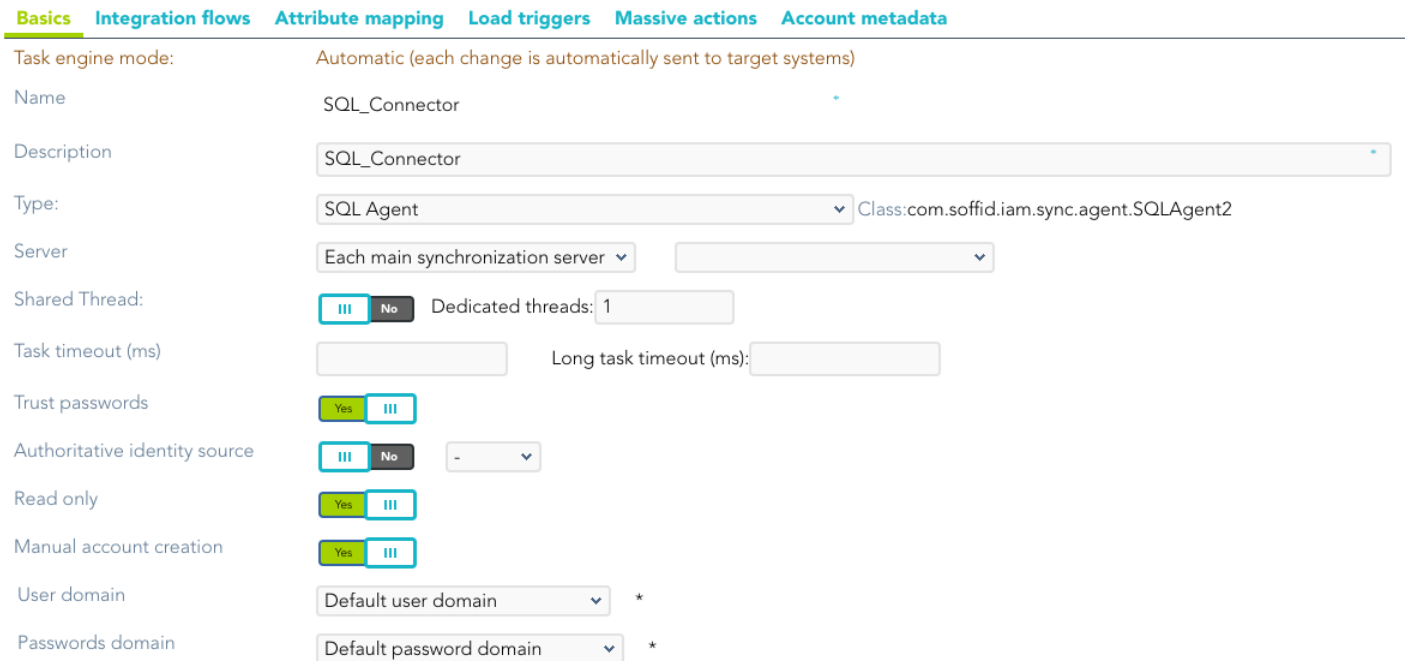
Basic

Generic parameters

After the installation of the addon, you may create and configure agent instances.

To configure this SQL connector you must select "Customizable SQL agent" in the attribute "Type" of the generic parameters section in the agent's page configuration.

> For more information about how you may configure the generic parameters of the agent, see the following link: Agents configuration



Custom parameters

Below there are the specific parameters for this agent implementation:

Parameter	Description
User name	Database user name to authenticate
Password	The password of the database user
Driver	Identifies the driver of the relational database to use. Currently, these are the supported databases: MySQL (& MariaDB), PostgreSQL, Oracle, MS SQL Server, Informix, DB2/400, DB2 Universal, Sybase, ODBC

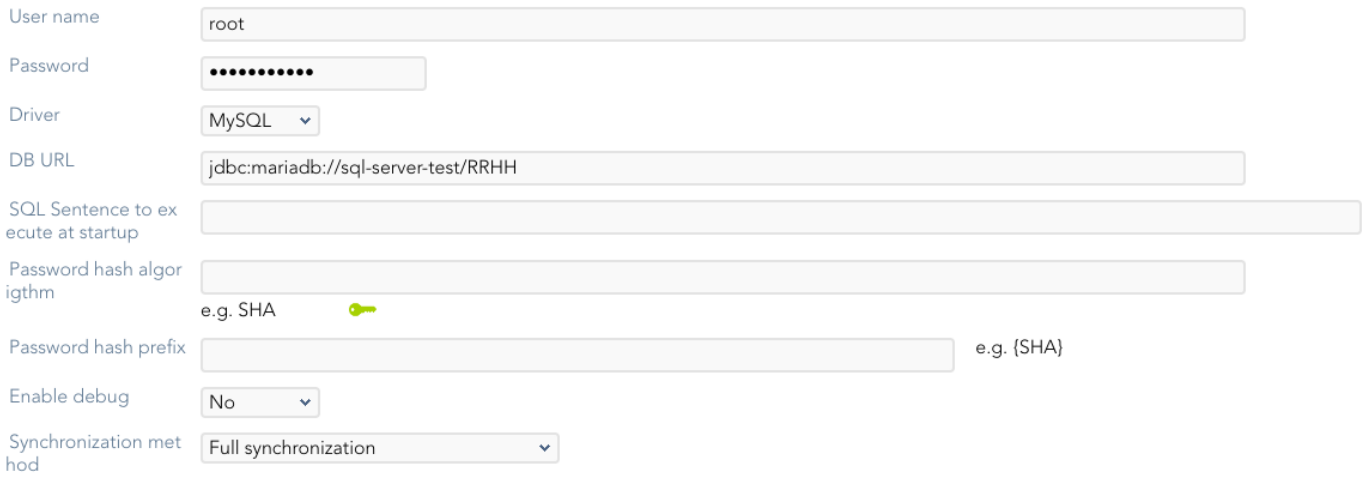
Parameter	Description

DB URL	URL that identifies the connection properties. Please refer to the specific database vendor documentation to build this URL.
	
	`jdbc:mariadb://<HOST>/<DATA_BASE>`
	
	`jdbc:mysql://<HOST>/<DATA_BASE>`
	
	`jdbc:postgresql://<HOST>/<DATA_BASE>`
	
	`jdbc:oracle:<drivertype>:@<database>`
	
	`jdbc:sqlserver://<HOST>;databaseName=<DATA_BASE>`
	
	(*) More documentation about the DB URL

SQL Sentence to execute at startup	Each time the connection to the agent is established, this SQL statement will be executed.
Password hash algorithm	The algorithm is used to encrypt the password. For instance SHA1, SHA256, MD5, etc

Password hash prefix	Prefix to add it to the password.
	
	`{SHA1}BzE/DjIPIsv6Nc/CIFCOs/9FfH4=`
	
	`{SHA256}AIEM+LlNb8ucXeSE077EGHYgs+KHblmquQ2FL+Dxj7Y=`

Enable debug	Two options: **Yes**, and **No**. It enables or not more log traces in the Synchronization Server log
Synchronization method	• **Full synchronization**: persists the changes made in Soffid, regardless of the possible changes made in the final system. • **Incremental synchronization**: this type of synchronization is used to avoid losing changes that have been made to the target system. First, Soffid's changes will be propagated to the target system, and then the changes on the target system will be made in the Soffid system. If the changes are in the same attribute, the Soffid value is the one that will persist. *(**)*



Attribute mapping

This connector can manage users, accounts, roles, groups, and grants.

Properties

Some agents require to configure some custom attributes, you will use the properties section to do that.

Any SQL sentence gets its parameters in three step process:

1. The synchronization engine creates the Soffid object.
2. The Soffid object is translated into a managed system object, using the attribute translation rules.
3. Soffid parser looks for any identifier preceded by a colon (:) symbol. For any symbol found, the symbol is replaced by a parameter whose value is the managed system attribute with the replaced identifier.

Once the SQL sentence has been executed, in the case of SELECT clauses, the column names are used to generate a virtual managed system object. The last step is to apply the attribute translation to generate the Soffid object to be populated.

These are the properties required to map every object of the mapping:

Property	Value

selectAll	SQL sentence that needs to be executed to retrieve all the objects that currently exist on the database.
	• Applies to authoritative identity sources.
	• On non-authoritative identity sources, only the columns needed to calculate the **name** soffid attribute are needed.
	
	You can use this property with the following objects: **user**, **account**, **role**, and **authoritative change**.
	
	`SELECT * FROM USERS`
	
	`SELECT * FROM ROLES`

check	SQL sentence that will return when a single object already exists on the database.
	You can use this property with **all the Soffid objects**.
	
	`SELECT ID FROM USERS WHERE USER=:USER`
	
	`SELECT ID FROM ROLES WHERE ROLE=:ROLE`

insert	SQL sentence to create a new object.
	You can use this property with **all the Soffid objects**.
	
	`INSERT INTO USERS VALUES (:USER, :FIRST_NAME, :LAST_NAME, :MAIL, :GROUP)`
	
	`INSERT INTO USER_ROLES (USETNAME, ROLNAME) VALUES (:USERNAME, :ROLNAME)`

update	SQL sentence to update an existing object.
	You can use this property with **all the Soffid objects**.
	
	`UPDATE USERS SET FIRST_NAME=:FIRST_NAME, LAST_NAME=:LAST_NAME, MAIL=:MAIL, GROUP=:GROUP WHERE ID=:ID`
	
	`UPDATE ROLES SET DESCRIPTION=:DESCRIPTION WHERE ROLE=:ROLE`

delete	SQL sentence to remove (or disable) an existing object.
	You can use this property with **all the Soffid objects**.
	
	`DELETE FROM USERS WHERE ID=:ID`
	
	`DELETE FROM USER_ROLES WHERE ID=:ID`

selectByAccount	SQL sentence to retrieve all the role grants made to an account (for single account information). You can use this property with the following objects: **grant.** `SELECT * FROM USER_ROLES WHERE USERNAME=:USER`
selectByName	SQL sentence to fetch role information based on its name (for single role information). You can use this property with the following objects: **role.** `SELECT * FROM ROLES WHERE ROLE=:ROLE`
updatePassword	SQL sentence to update the user password. You can use this property with the following objects: **user** and **account.** `UPDATE USERS SET PASS=:PASS WHERE USER=:USER`
validatePassword	SQL sentence to check the user password. You can use this property with the following objects: **user** and **account.** `SELET 1 FROM USERS WHERE PASS=:PASS AND USER=:USER`

Attributes

You can customize attribute mappings, you only need to select system objects and the Soffid objects related, manage their attributes, and make either inbound or outbound attribute mappings.

You may map the attributes of the target system with the Soffid available attributes.

- For the target system attributes are required to be accessible to its specification
- For the Soffid attributes, you may follow the next link

> For more information about how you may configure attribute mapping, see the following link: Soffid Attribute Mapping Reference

Example for roles:

Property	Value
delete	DELETE FROM USER_ROLES WHERE ID=:ID
insert	INSERT INTO USER_ROLES (USETNAME, ROLNAME) VALUES (:USERNAME, :ROLNAME)
selectByAccount	SELECT * FROM USER_ROLES WHERE USERNAME=:USER
selectByRole	SELECT * FROM USER_ROLES WHERE USERNAME=:USER

System attribute		Direction	Soffid attribute		
MAIL		←	shortName==null ? attributes{"MAIL"} : shortNa me+"@"+mailDomain		
LAST_NAME		⇄	lastName		
PASS		←	password		
GROUP		⇄	primaryGroup		
FIRST_NAME		⇄	firstName		
USER		⇄	accountName		

Example for accounts:

Property	Value
check	SELECT ID FROM USERS WHERE USER=:USER
delete	DELETE FROM USERS WHERE ID=:ID
insert	INSERT INTO USERS VALUES (:USER, :FIRST_NAME, :LAST_NAME, :MAIL, :GROUP)
selectAll	SELECT * FROM USERS
selectByAccountName	SELECT * FROM USERS WHERE USER=:USER
update	UPDATE USERS SET FIRST_NAME=:FIRST_NAME, LAST_NAME=:LAST_NAME, MAIL=:MAIL, GR OUP=:GROUP WHERE ID=:ID
updatePassword	UPDATE USERS SET PASS=:PASS WHERE USER=:USER
validatePassword	SELET 1 FROM USERS WHERE PASS=:PASS AND USER=:USER

System attribute		Direction	Soffid attribute		
USER		⇄	accountName		
PASS		←	password		
LAST_NAME		⇄	lastName		
FIRST_NAME		⇄	firstName		
GROUP		⇄	primaryGroup		
MAIL		←	shortName==null ? attributes{"MAIL"} : shortNa me+"@"+mailDomain		

Triggers

You can define BeanShell scripts that will be triggered when data is loaded into the target system (outgoing triggers). The trigger result will be a boolean value, true to continue or false to stop.

Triggers can be used to validate or perform a specific action just before performing an operation or just after performing an operation on target objects.

> To view some examples, visit the Outgoing triggers examples page.

Integration flows

Update User

> Visit the Integration flows Update user page for more information

Update Account

> Visit the Integration flows Update account page for more information

---

(*)

https://mariadb.com/kb/en/about-mariadb-connector-j/

https://docs.microsoft.com/es-es/sql/connect/jdbc/building-the-connection-url?view=sql-server-ver16

(**) Soffid provides two synchronization types:

- *Full synchronization*
- *Incremental synchronization*

*The first type, the **full synchronization** method, persists the changes made in Soffid, regardless of the possible changes made in the target system.*

*For the second type, the **incremental synchronization** method, Soffid has developed a synchronization system, using custom internal attributes, to check what changes have been made to the different attributes of an object. Thus, it tries to avoid losing the changes that have been made in the target system. First, Soffid's changes will be propagated to the target system, and then the changes on the target system will be made in the Soffid system. If the changes are in the same attribute, the Soffid value is the one that will persist.*

Revision #41
Created 19 April 2021 15:29:15
Updated 1 March 2024 08:16:21