RDM Development Guide

ataccama

CONFIDENTIALITY NOTICE

Table of Contents

This guide is intended to help advanced RDM users with creating and configuring a fully functional RDM solution.

Topics covered include introductory information, creating an RDM model project, setting up RDM workflows, or configuring the RDM Webapp.

In this guide:

- RDM Product Information
- RDM Configuration
- RDM Reference Documentation
- RDM Backup and Recovery

1 RDM Product Information

1.1 RDM Quickstart Guide

1.1.1 Overview

The RDM solution consists of 4 components:

- **Keycloak** - third party identity management tool
- **RDM Server** - the Ataccama RDM processing engine
- **RDM Webapp** - the Ataccama RDM web application
- **Relational database** (PostgreSQL, MS SQL, or Oracle) which acts as the backend database for the RDM Solution

Each of these components is configured as a Linux service which can be restarted independently and which has its own logs that can be used for identifying issues.

1.1.2 Databases

The RDM supports PostgreSQL, MS SQL, and Oracle relational databases. Only the PostgreSQL Standalone installation is supported by Ataccama. Everything else (Oracle, MS SQL, RDM from Amazon, Azure SQL Services) should be prepared, provided, and maintained by the client.

The description below applies to the maintenance of the PostgreSQL Standalone solution.

1.1.3 Architecture

Depending on the desired architecture, there can be a different number of servers. Due to this, the below information should be interpreted according to your Environment Architecture, and the commands should be executed on the relevant server, where a specific component is installed.

Common architectures are:

- Small schema:
    - **RDM server:** with Keycloak, RDM Server, and RDM Webapp installed
    - **DB Server/Service:** Relational database on a dedicated server
- Medium Schema:
    - **Web server:** with Keycloak and RDM Webapp installed
    - **App server:** with RDM Server installed
    - **DB Server/Service:** Relational database on a dedicated server

- Large Schema:
    - **Keycloak server:** with Keycloak installed
    - **Web server:** with RDM Webapp installed
    - **App server:** with RDM Server installed
    - **DB Server/Service:** Relational database on a dedicated server

1.1.4 Dependencies

- The core element of the RDM solution is the relational database. If the database is not available, none of the further components will be able to start.
- The Keycloak component is a hard requirement, without which the RDM engine and the RDM webapp will not be able to work.
- The RDM engine is a hard requirement without which the RDM webapp will not show any content, as the content should be provided by a project started by the RDM engine.

Using this dependencies flow, troubleshooting can be narrowed down to a specific part of the RDM solution.

1.1.5 Applications as Services

Every application is configured as a Linux service, and its service file can be found in the default Linux folder for services. This folder is `/etc/systemd/system` and the service names are as follows:

- **Keycloak service name:** `keycloak-server.service`
- **RDM Server service name:** `server-rdm.service`
- **RDM Webapp service name:** `rdm-webapp.service`
- **Standalone PostgreSQL (optional) service name:** `postgresql-13.service`

1.1.6 Start/Stop/Restart

As standard Linux services, Ataccama services can be maintained via the `systemctl` command.

Use the following commands:

- **To start the application:** `systemctl start <service-name>`
- **To stop the application:** `systemctl stop <service-name>`
- **To restart the application:** `systemctl restart <service-name>`
- **To see the status of the application:** `systemctl status <service-name>`

Where `<service-name>` is one of the service names mentioned in the paragraph above.

1.1.7 Logs

Each component of the solution has a dedicated folder to store log files.

- **Keycloak logs folder:** `/<path-to-keycloak>/standalone/log` with log file `server.xml`
- **RDM Server logs folder:** `/<path-to-rdm-server>/runtime/server/log` with log file `server.out`
- **RDM Webapp logs folder:** `/<path-to-rdm-webapp>/log` with log file `spring-boot-logger.json.log`

Alternatively, access the component log using the Linux service journal logs command `journalctl`.

For example, if you want to read the entire log of the service, use the following command: `journalctl -u <service-name>`

To see live logs, use the following command: `journalctl -u <service-name> -f`

Log viewing hint

To see logs immediately once you start the service, you can combine the start/stop/restart and log commands in the following way:

```
systemctl start <service-name> && journalctl -u <service-name> -f
```

```
systemctl stop <service-name> && journalctl -u <service-name> -f
```

```
systemctl restart <service-name> && journalctl -u <service-name> -f
```

1.1.8 Configuration files

All configuration files of the solution are provided in the list below and can be used in order to identify component-to-component connection information, connection to the DBs, etc.

Keycloak

The only Keycloak configuration file containing connection details for the relational database is the `standalone.xml` file located in `/<path-to-keycloak>/standalone/configuration`.

For details, see ONE RDM Installation Guide.

RDM Server

Configuration for the RDM server is always provided within the RDM Project created in the `Ataccama ONE Desktop`.

Inside the project, the `Files/etc` folder can be found. This folder contains the two configuration files required to start the RDM Server application:

- **Server Configuration:** `Files/etc/rdm.serverConfig`, contains:
  - The application startup details;
    - TLS options
    - Ports to be used by the application
    - Security Filters (Keycloak Identity Provider)
  - Project configuration:
    - Location of the Workflows folder
    - Location of the Schedulers folder
    - Location of the Resources folder
- **Runtime Configuration:** `Files/etc/rdm.runtimeConfig`, contains:
  - Network connection details:
    - Connection to Keycloak
    - Connection to the relational database
  - Other projects-related configuration

RDM Webapp

The only configuration file used by the RDM webapp is `application.properties` stored in the `/<path-to-rdm-webapp>/etc/` folder.

This file contains the following important properties which may be helpful for troubleshooting:

- `server.port` - property which specifies on which port the RDM webapp is running
- `ataccama.one.rdm.server.url` - the URL on which RDM will search for the project running on the RDM Server
- `ataccama.authentication.keycloak` - set of properties which specify Keycloak connection URL and other Keycloak-related details
- `server.ssl.key-store` - set of modules which contain TLS certificates with which the RDM webapp starts

1.2 Introduction to RDM

Ataccama Reference Data Manager is a tool for managing reference data by supporting formal, defined processes and ensuring central authority over all reference data changes. Reference Data Manager allows not only browsing all reference data, but also changing, creating, and deleting existing data. Its domain-agnostic nature also ensures flexibility to adapt to all types of industries, and its cutting-edge user interface with the latest standards and technology on UI and UX make an easy-to-use product for data managers.

1.2.1 RDM as Part of the Ataccama Integrated Platform

Ataccama Reference Data Manager is used both as a stand-alone fully featured solution for the purpose of managing reference data and in combination with other products for a bigger solution (e.g. Ataccama Master Data Management). Like the rest of the Ataccama platform, RDM excels in integrating with other systems and applications via standard interfaces in various modes.

1.2.2 Business Role

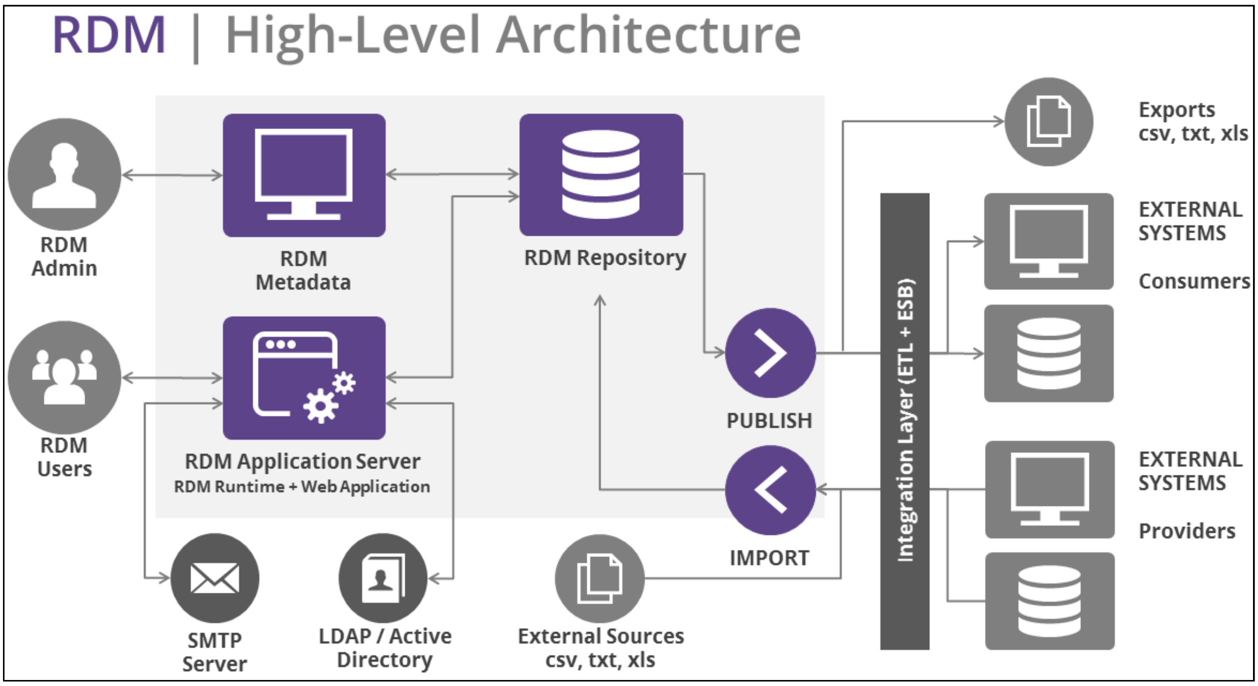
- Provide means for reference data management operations, including a process of validation.
- Easily configurable workflow for business approval processes.
- Synchronization processes across systems, maintaining consistency of the reference data.
- Browse reference data, through hierarchies, views or entity tables, supported by filtering and other advanced navigation features.
- Historical traceability and auditing of data over time.

1.2.3 Business Cases

- Managing all reference data across a diverse range of source systems.
- Providing a central authority for reference data editing (e.g. creation, modification, etc), supported by configurable workflows for a business approval process
- Providing consistent reference data to consuming systems and synchronization with other different reference data sources.
- Providing an easy-to-use application to search and display reference data as required.

1.3 RDM Architecture

The following diagram represents a typical Ataccama Reference Data Manager high-level architecture.



RDM High Level Architecture

Core elements are depicted in purple; elements external to RDM appear in gray. This page describes every core element of the RDM architecture.

1.3.1 RDM Metadata

The RDM metadata is the core configuration of the RDM project. Through the metadata model it defines all entities, structures, workflows, synchronization jobs, and more. This metadata is also the part that represents the RDM implementation.

The implementation is done through a Model Project, which can be generated into a package with all RDM Metadata for the project. The RDM Metadata composing the Reference Data Manager project includes configuration in different domains, as seen below.

Functional Configuration

- **The RDM Logical Model** - this is the central part of the RDM configuration and includes all the configuration related to the canonical model: Tables (also called Reference Data Entities), Hierarchies, Views, Domains as types of data, all types of validation checks and constraint rules over the reference data, and Categories as abstract groups of common elements.
- **Workflow Configuration** - role-based governance workflows support notifications, email templates, multiple workflows based on action type, and workflows-per-entity.

Non-functional Configuration

- **Security** - RDM security can be configured via direct file configuration or LDAP integration. Extensive support for LDAP is available in the application.
- **Application Variables** - all specifics of the environment, technology used for the database, and relevant system metadata.
- **Documentation** - generated documentation based on the defined canonical model.
- **Logging** - logging configuration as seen in the RDM web application for audit and monitoring purposes.

Integration with Other Systems in the Complete Architecture of the Solution

- **Connected Systems** - definition of connected system models and relationships between data and attributes in the canonical model entities and those in the connected systems. This information is used for synchronization of RDM with the systems.
- **Synchronization** - synchronization jobs between the central RDM repository and connected systems.
- **Task Scheduler** - schedules created to automate execution of synchronization over time.
- **Batch Load/Export Interfaces** - interfaces are created automatically based on the configuration and metadata presented above. These interfaces are then available for extension, where more functionality such as transformations, further enrichment, or data preparation can be added.
- **Online Interfaces** - similar to the batch load interfaces, online interfaces offer the possibility of data preparation and further transformations when serving raw data in the default services from the RDM repository.

Once the RDM Metadata is defined, it can be uploaded using the Web Console of the RDM application. The Metadata is then hosted at the web application side, making it independent of the implementation source.

1.3.2 RDM Application Server

Ataccama Reference Data Manager is a web application created on top of the Java technology (using the Google Web Toolkit framework for its development). As seen in the high-level architecture, users work directly with the RDM Web Application, which is connected to the RDM Database.

The Web Application is independent of the RDM Model Project, which is developed and packaged apart, and its configuration is loaded directly into the client. The RDM web application supports model configuration versioning when uploading new configuration and does not require to be connected to the RDM Model Project directly.

The RDM web application requires to be hosted by a web server and have direct access to the database used as the RDM repository. Additional resources might need to be configured from the web server side, such as SMTP server or LDAP-based authentication.

1.3.3 RDM Repository

The RDM database repository contains model configuration, physical structures created automatically based on this model and also reference data stored in these specific structures. Only the RDM web application and its processes can interact with the RDM repository, with no direct external access on data allowed as best practice. As Ataccama is agnostic of the storage technology, this database can be of any RDBMS available on the market, provided a JDBC-compliant driver exists. The database needs to be accessible by the web application, which will connect to it when displaying and executing actions on data.

1.3.4 ONE Desktop Integration

There are two cases in which ONE Desktop Development Guide Integration is used by the RDM application. Both of them are optional and not required for the basic functions of the Reference Data Manager.

Validation and Enrichment Services

Ataccama RDM can use services for validation and enrichment, as seen in RDM Features. These services are usually created using ONE Desktop (as custom DQ services) since they follow some requirements in terms of configuration. If those services are used, integration with ONE Runtime Server is required. This server hosts services that are provided to RDM and can be located in the same machine or a remote machine. The choice of server location is part of the metadata configuration of the project.

Workflows, Interfaces, and Synchronization Jobs

Ataccama RDM can synchronize its master reference data with other systems that have been defined as Connected Systems in the RDM Metadata. The generated workflows as part of the configuration, the schedules from the Task Scheduler, and plans created for the Export and Import interfaces are all hosted and executed by the ONE Runtime Server. In order to take advantage of the exporting and importing features, both batch and online, integration with the ONE Runtime server is needed.

1.4 RDM Features

Ataccama RDM is a tool for managing reference data through a web application front-end. The following are the high-level features offered by the tool.

1.4.1 Model and Metadata Configuration

Ataccama RDM **configuration is metadata-based**, which allows a fast setup. All the configuration revolves around a central reference data model, a canonical reference data model that accommodates all reference data used across the company.

The **central RDM model is used to generate all important** processes in the application, including integration interfaces, synchronization processes, data entity definitions, and relationships.

Model configuration instructions are described in RDM Configuration.

1.4.2 Data Management

Ataccama RDM supports the **full cycle of reference data management**, being suitable as a central and unique authority for all reference data changes, and maintaining consistency of this data across different systems. These operations on data are done through its web application interface.

Such actions as creation, modification, or deletion of the reference data are supported. See Working with Records in RDM for more information.

Workflows

RDM also allows defining processes (also called workflows) dependent on the policy and requirements for a reference data change. See Configuring RDM Workflows for configuration instructions.

Business Dates

RDM also supports the definition of data's effective life, the so-called record business dates.

See Versioning Records in RDM for more information on using business dates. See RDM Tables for configuration instructions.

1.4.3 Browsing and Data Navigation

As one of the main features of reference data management is to display and list reference data, Ataccama RDM allows advanced browsing of data via special views and structures that can be configured at will:

- **Tables** - reference data entities. They are composed of attributes and represent the record structure of reference data tables. This structure is the only mandatory structure during model creation.
- **Hierarchies** - defined by relationships between different entities, hierarchies allow unfolding relationships and exploring data through parent-child trees, providing a different form of inspecting data, suitable for relationship-oriented reference data.
- **Views** - composed of different bits of data across entities joined and united to form a table-oriented data representation. Views are formed through selecting certain attributes or using relationships between different entities of the RDM model.
- **Categories** - a collection of different types of data structures that share a common abstract domain. Hence, categories can be composed of entities, hierarchies, and views, and help structuring and selecting data when dealing with a wide range of reference data structures.
- **Data sets** - custom reports about reference data. Formed via SQL queries, these reports can provide aggregating information about data from several tables connected by foreign keys.

These different structures are fully configurable and are abstractions over the central model defined in the Model Project.

See Creating the RDM Data Model and Creating Additional Reference Data Views for configuration instructions.

See RDM Data Tab Overview for more information.

Viewing Modes

Additionally to the structures that can be defined for browsing data, RDM offers application-wide view modes. These modes help users browse data depending on its state in the publishing cycle:

- **Published** - shows the latest "official" version of reference data used.
- **Edit** - shows records with changes (creation/editing/deletion) made since the latest publish action, if any.
- **History** - shows records in the published state at a given point in time.
- **All history** - shows all changes ever made to all records.

- **Cart** - shows records added to the cart in the EDIT mode. Serves for working with specially marked records.
- **Import** - a mode for importing data. Shows the state of imported records as compared to the latest state of the record (published or edited) or published state at some point in time.
- **Inputs** - a mode for importing data from connected systems. Shows the state of imported records as compared to the latest state of the record (published or edited) or published state at some point in time. Available only to users with system rights.

See RDM Data Viewing Modes for more information.

Navigation Features

Ataccama RDM also offers many other features to help its users navigate vast amounts of reference data and focus only on those pieces that are of interest:

- **Preset filters** - filtering by attribute values. Filters can be combined when searching for particular reference data records.
- **Custom filters** - SQL-type operations that can grant refined filters with great amount of flexibility. While being more complex, they also give more powerful results for refined searches.
- **User-defined cart** - users can earmark recording for further work or publishing by putting them into the cart, in which they can apply bulk operations.

See Filters and Marking Records for Later Use for more information.

1.4.4 Workflows

Workflows are a central feature of RDM as they allow defining formal processes around reference data changes over time. Workflows have the following features:

- Defined on the table basis (per each entity defined in the model).
- Can be set for different types of actions (e.g., creation, modification, or deletion of reference data).
- Support an advanced role-based approach to resolution.

Workflows are also exposed in the web application as a different section where users are notified of their pending actions on the reference data records being modified. Each step of the process can be revoked or approved by a party with proper permissions, and notifications can be set for each transition between workflow states.

See Moving Records Through RDM Workflows for more information.

Notifications

Ataccama RDM supports notifications based on events or actions taken on data going through workflows. Notifications are sent as email messages, and tailored templates for different events and transitions of the data can be created.

1.4.5 Data Quality Validation and Enrichment

Built on top of the Ataccama platform, RDM takes full advantage of its capabilities by adding Data Quality validation and enrichment features to the reference data management process.

Attribute-based Validation

Attributes in a reference data entity can be validated by using data domains. A domain consists on a set of constraints applied to one data attribute, those constraints being the following:

- **Data type validation** - validation by native data types of the Ataccama platform (e.g., boolean, string, integer, float, date, etc.). This basic validation ensures that the given attribute complies with the restrictions of the data type.
- **Regular expression (regex) validation** - for string datatypes, constraints can be set in the form of regular expressions, which define a pattern that must be matched by the input data, otherwise rendering the data as an invalid input.
- **Ranges, formatting, and sizes** - users can define ranges by using maximum and minimum values, as well as defining sizes for string values, maximum number of lines, and further restrictions on the format of the data.
- **Foreign key validation** - attribute integrity can be forced to be validated as a foreign key of another entity, provided that this is also included in the model.

See RDM Domains for configuration instructions.

Record-based Validation

Besides the attribute-based validation, RDM provides a list of different record-based validation methods that allow users to define cross-attribute validation logic.

See RDM Tables for configuration instructions.

Unique and Primary Key Validation

Unique key validation ensures that the attribute or a combination of attributes unequivocally identifies a record in the reference data. Any violation of this unique key is immediately reported when a key appears as a duplicate in the data or when RDM users create a new record in the web application.

SQL Validations

These validations are defined by SQL-type queries that check for custom conditions across columns, rows, and tables.

See SQL Validations for more information.

DQ Expressions

Expressions written in special Ataccama expression syntax can be used to define any combination of conditions that one or several records in a table must satisfy. It is possible to define multiple expressions and the messages to be output whenever the check is violated.

Online Validations

Web services defined from ONE plans. These services provide real-time feedback on user validation, and can be built to create very complex data processing. They can access all values being used by a record and subject them to dictionary checks, third-party services checks, and all sorts of transformations before actual DQ evaluation.

Online Enrichers

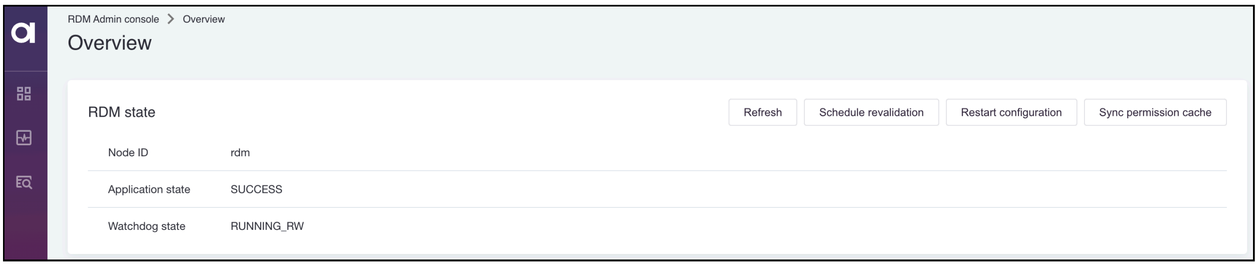
Similarly to creating validation processes, it is also possible to create enrichment web services from ONE plans. Hence, users obtain the benefit of auto-completion based on input data: fields can be filled in or corrected based on the input of other fields.

Record Validation on Application or Configuration Restart

When RDM is first started, all records are validated by default. After this, whenever the application is restarted, validation is performed only if it previously failed on a particular table or if a change has been detected in the table structure. Otherwise, validation is skipped in order to improve the startup time.

If the web app configuration is modified, records are validated following the same logic (that is, validation is only applied to tables that have been altered).

Additionally, it is possible to schedule full validation that will run once the application is restarted again. To do this, navigate to **RDM Admin console** (`http://{rdm_url}:8060/admin`) in your web browser and select **Schedule revalidation**.



Overview

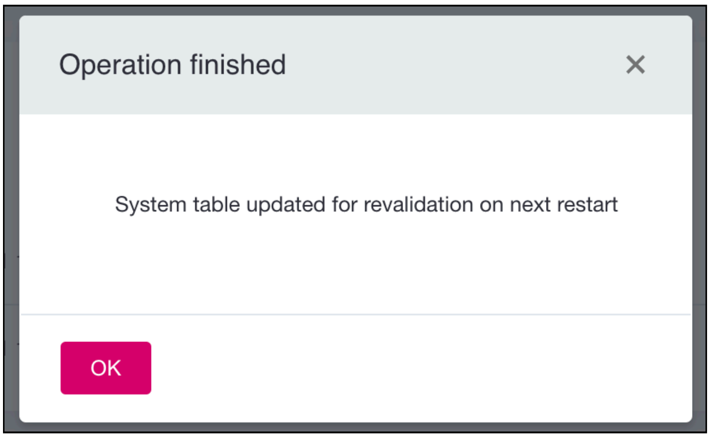
RDM state

Node ID rdm

Application state SUCCESS

Watchdog state RUNNING_RW

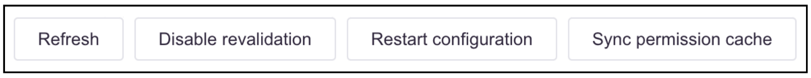
The following message appears to confirm your request:



Operation finished

System table updated for revalidation on next restart

If you change your mind, you can cancel revalidation after you schedule it by selecting **Disable revalidation** from the same screen.



1.4.6 Integration and Provisioning Interfaces

Ataccama RDM, as the rest of the Ataccama platform, excels at integrating with other components of the architecture and systems. This is done through its interfaces, which can be batch or real-time. The interfaces can use all the functionality available in the platform for data transformation or data preparation, adapting to the needs of final exposed interfaces.

Batch Interfaces

Ataccama RDM allows both import and export in batch mode through defined interfaces, which can use additional logic to support data transfer. In terms of integration, the following data formats are supported:

- All types of text files, delimited or fixed-width
- Microsoft Excel files, supporting different versions

- Binary files through a special reader step
- Any database or system with an available JDBC connector

Besides integration with these data formats out-of-the-box, RDM can use third-party connectors to access proprietary, non-relational systems such as CRM/ERP and other legacy systems.

See Configuring RDM Synchronization via Text Files and Configuring RDM Synchronization with External Databases for configuration instructions.

SOA Interfaces

Ataccama RDM also allows defining standard interfaces through services, and these interfaces can be modified to add additional logic and wrap it as a service. The online interfaces are SOAP (over HTTP or over JMS) out of the box and are generated as part of the configuration.

However, other service types are also supported (e.g., JSON services for RESTful interfaces, bulk CSV services, and more).

See Configuring RDM Synchronization via Web Services for configuration instructions.

1.4.7 Synchronization

RDM can be synchronized with external systems in several ways:

1. Synchronization via text files, with a possibility to transfer files to and from a remote server.
2. Synchronization with external databases (in both directions).
3. Writing data to and reading data from selected RDM tables via web services.

In addition, it is possible to configure an on-publish service (based on a ONE plan), which will take place each time a user tries to publish data in the web application.

Connected systems (databases) are independent, and integration to the tool can be done at the attribute level, thus enabling synchronization of systems structurally different from the central reference data model maintained by RDM. All systems can be defined by importing their models to the tool, which will maintain references and relations between the data hosted by the systems and that hosted in RDM.

Synchronization jobs are automatically generated, and information about them is logged and kept by the application. These jobs can be either scheduled, triggered through events, or executed on demand.

See RDM Synchronization for configuration instructions.

See Monitoring RDM Synchronization for information on monitoring the synchronization with external databases.

1.4.8 History and Auditing

RDM offers wide functionality in terms of history and auditing for both reference data itself and activity carried by RDM or its users.

Data History and Auditing

Ataccama RDM maintains historical versions of all reference data, making it possible to trace any reference data changes through the life of the record. Two modes (*History* and *All History*) allow users to:

- access the state of data at a defined point in time (*History* mode*)*.
- define a time range and view the course of changes of data during that time range (*All History*).

See RDM Data Viewing Modes for more information on the *History* and *All History* modes.

Action and Event Auditing

RDM also keeps history logs of all processes and actions performed on the data, making it possible to trace particular events, such as synchronization with systems and changes to reference data by users.

These logs can also be accessed in a persisted form. See RDM Change Log for more information.

1.4.9 Permissions and Data Access

Permissions are managed by RDM using a role-based approach. Users can be granted roles that have permissions over different reference data entities. The permissions can be divided as follows:

- **View permissions**: Permissions to view data in a read-only mode.
- **Creation permissions**: Permissions to create new records.
- **Modify permissions**: Permissions to edit an existing record, but not create or delete.
- **Delete permissions**: Permission to delete reference data (logical delete, as the change has to be published in order to be visible).
- **Publish permissions**: Permission to publish reference data that has been created, modified, or deleted (effective deletion).

Permissions can also be set on attribute level and row level using row rules (implemented as SQL statements).

Permissions can be managed directly from the web application front end by RDM administrators. The roles are also important for the formal processes defined by workflows since access to particular workflow actions are also dependent on the roles mapped to a given user.

The application can also integrate with LDAP-compliant systems for both authorization and authentication.

See Setting Permissions in RDM for more information.

1.5 RDM Supported Platforms and Databases

RDM consists of the following main parts:

1. Ataccama platform core:
    a. **Runtime** installed on a runtime platform. RDM runtime consists of ONE runtime enriched with RDM-specific functionalities.
    b. **ONE Desktop** for RDM solution development and testing on a local machine. RDM solution (RDM model project) is used to generate configuration files used by the RDM runtime.
2. RDM-specific parts:
    a. **Storage database** that stores reference data in their various states.
    b. **Web application** for working with reference data, deployed to an application server.

RDM integration:

- Databases
- Other data sources/targets (e.g., files, Amazon S3)

1.5.1 Ataccama Platform Core

Platforms and databases required for functioning of all Ataccama modules. Shared for the whole platform.

ONE Desktop

Operating system	Recommended	Supported
MS Windows (recommended)	• MS Windows 10	• MS Windows 10, 11 • Vista
Linux	• RHEL 7 • CentOS 7, 8 • SLES 11+	• RHEL 7 • CentOS 6 - 8 • SLES 11+ • 2.6+ kernel
MS Windows Server	• MS Windows Server 2019	• MS Windows Server 2012, 2019
macOS	• macOS Catalina (Version: 10.15)	• macOS High Sierra (Version: 10.13) • macOS Mojave (Version: 10.14)

Virtualization support:

- Run as a virtual application: VMware ThinApp
- Run via a terminal server: Microsoft Remote Desktop / Windows Server 2019 R2 x64

Runtime Platforms

Intel x86 / x86_64 Architecture:

Operating system	Recommended	Supported
Linux	• RHEL 8	• RHEL 7, 8 • CentOS 7, 8

Operating system (non-production use only)	Recommended	Supported
MS Windows		• MS Windows Server 2019

Certain steps and solution components (e.g., Loqate, Loqate Address Validate) use third-party products that may have specific platform and system requirements:

- Loqate
- Address Doctor

3rd Party Software

Platform	Recommended	Supported
Java	• OpenJDK 17 • Oracle Java 17 (LTS)	• OpenJDK 11 (for version 14.5.0), 17 (starting from version 14.5.1) • Oracle Java 11 (for version 14.5.0), 17 (starting from version 14.5.1) for Windows
Keycloak	• Keycloak 15	• Keycloak 12, 13, 14

1.5.2 RDM Specific Platforms and Databases

Browser Support

The latest versions of the following major browsers are supported:

- Google Chrome (recommended)
- Mozilla Firefox (recommended)
- Mozilla Firefox ESR (recommended)
- Internet Explorer
- Microsoft Edge
- Safari (Mac only)

The following browser features must be turned on: Cookies, CSS, Javascript with AJAX processing (XmlHttpRequest).

RDM Application Server

Operating system	Recommended	Supported
Linux	• RHEL 8	• RHEL 7 • CentOS 7

Databases for RDM and RDM Webapp Storage

Database	Recommended	Supported
Oracle	• Oracle 2021	• Oracle 2019
PostgreSQL	• PostgreSQL 14	• PostgreSQL 13.2
Microsoft SQL Server		• SQL Server 2017, 2019
Azure SQL		• ✓
Amazon Aurora PostgreSQL		• ✓

1.5.3 Integration

Databases

All Ataccama modules use a common read/write functionality to read/write data from source/target databases.

Below is a list of databases whose drivers are shipped with the Ataccama products. In addition to these, Ataccama can connect to any database that has a JDBC driver available.

Database	Tested drivers
Microsoft SQL Server	• mssql-jdbc-7.4.1.jre8.jar • jtds-1.3.1.jar
Oracle	• ojdbc8.jar • ojdbc8-12.2.0.1.0.jar
PostgreSQL	• postgresql-42.2.8.jar

2 RDM Configuration

2.1 Introduction

Page not found for multiexcerpt macro.

The page: **DOCTEAM:RDM Web Application User Guide** was not found. Please check/update the page name used in the 'multiexcerpt-include' macro.
This guide is intended to help RDM developers with creating a fully functional RDM solution.

Installing RDM is described in the RDM Development Guide.

Using the RDM web application is described in the the RDM Web Application User Guide.

2.2 Prerequisites

1. Knowledge of working in ONE Desktop: we recommend going through the ONE Desktop Development Guide to get to know ONE Desktop, especially the following articles:
    - Ataccama ONE Desktop
    - Projects
    - Working with Databases
    - Building Plans
    - Working with Component Steps
2. Make sure your understand RDM Architecture.

In this guide:

- RDM Example Project
- Building the RDM Solution from Scratch
- Creating the RDM Data Model
- Creating Additional Reference Data Views
- Updating the RDM Data Model
- Configuring RDM Authorization
- RDM Synchronization
- Configuring Initial Loads to RDM Tables
- Configuring RDM Workflows
- Scheduling RDM Tasks
- RDM App Variables

- Setting up RDM Auditing
- How to Deploy an RDM Web App Configuration
- Configuring RDM Online Services
- Configuring RDM Related Entities
- Enabling WatchDog High Availability
- RDM REST API

2.3 RDM Example Project

The following guide shows you how to install and run the latest RDM example project on **Windows**.

2.3.1 Download Postgres and Keycloak Plugins

For ONE RDM on Windows, Postgres and Keycloak plugins have been provided in order to easily deploy the modules, specifically for the purpose of running the RDM Tutorial.

> Note that the plugins are intended for demonstration purposes only and are not meant for production environment. By using these, you are accepting Keycloak and PostgreSQL licenses. Also keep in mind that the third-party software used in the plugins might not be up-to-date and may therefore pose a security risk.

Download the following packages:

Package
PostgreSQL plugin
Keycloak plugin

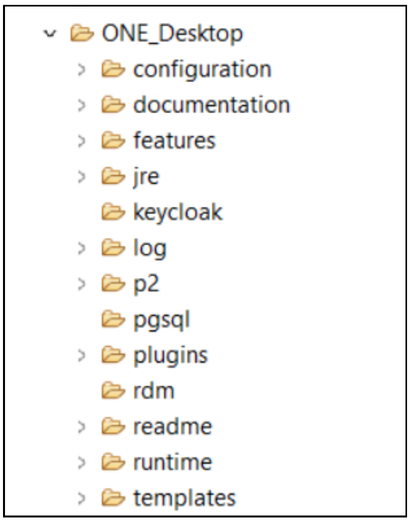
2.3.2 Download Builds

Download the corresponding build packages of RDM and ONE Desktop from the Downloads page.

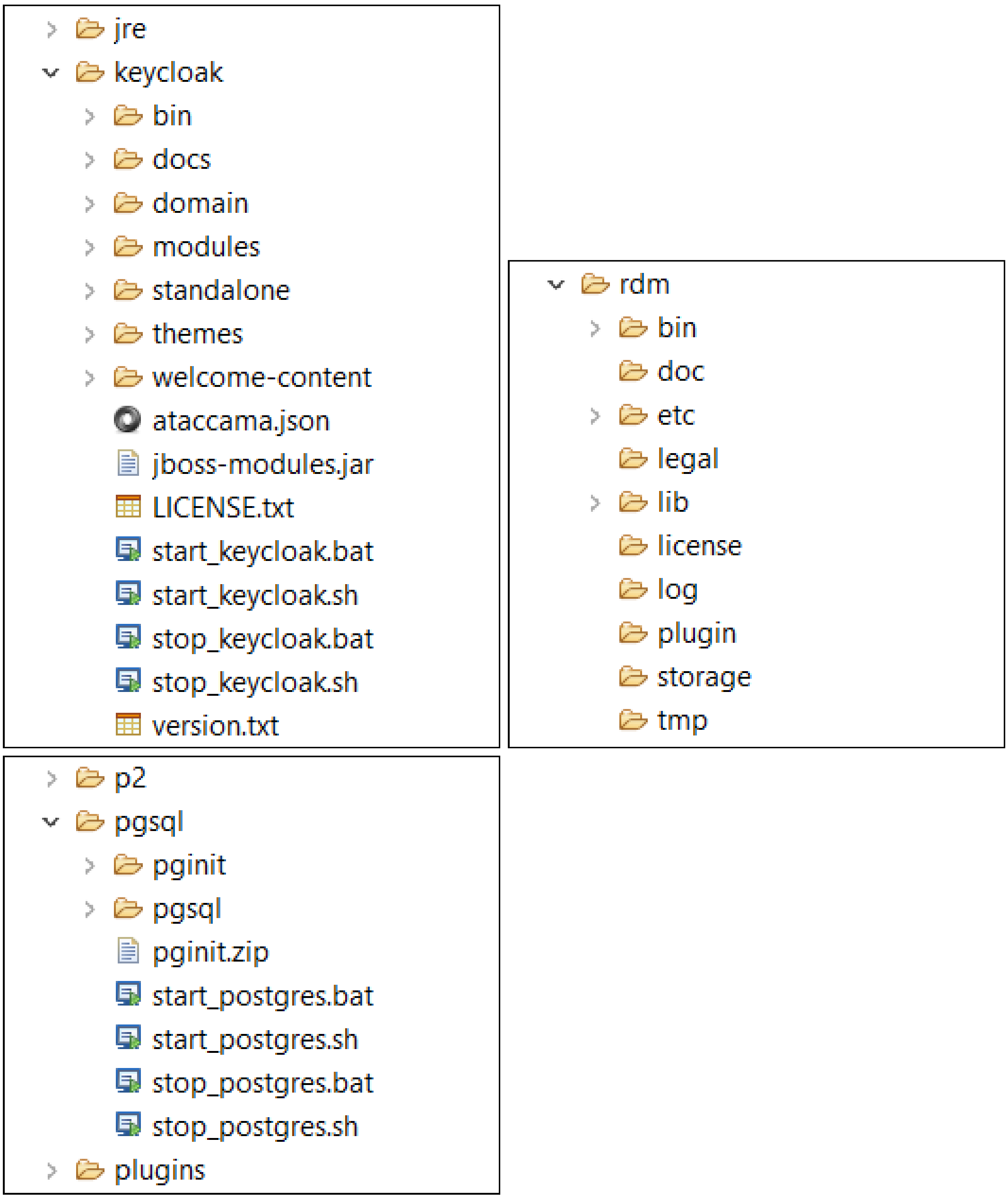
Extract Packages

1. After unpacking ONE Desktop, in **File Explorer** create the following folders:
   - `rdm`
   - `keycloak`

- `pgsql`



2. Extract the downloaded packages to these folders, as follows:
    a. `rdm-assembly-xxxxx.zip` to `rdm`
    b. `keycloak-x.x.x-x-demo.zip` to `keycloak`
    c. `pgsql-xxx.zip` to `pgsql`
    d. Your folder structure should now look like this:



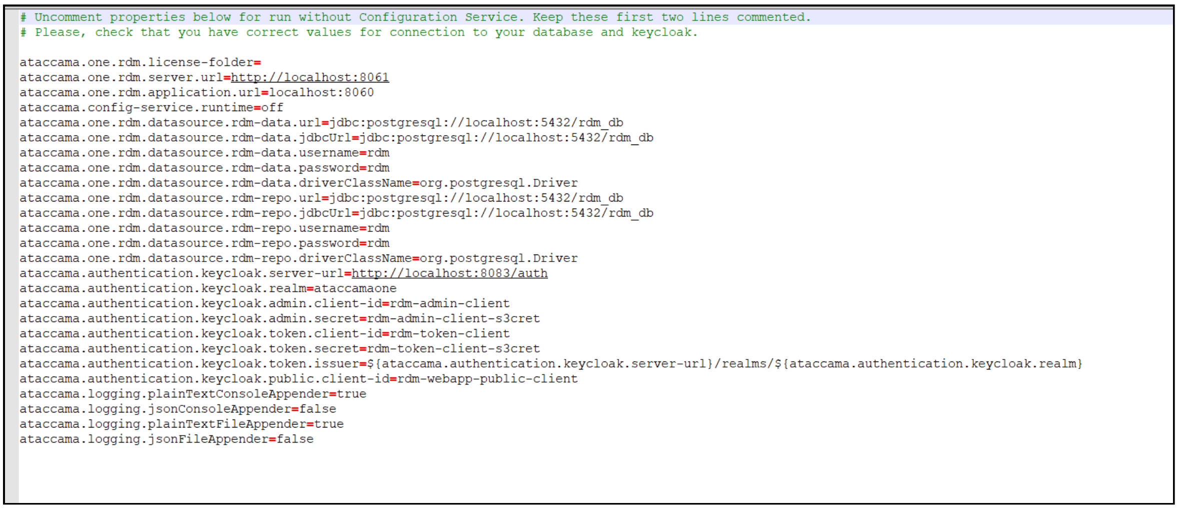
- jre
- keycloak
  - bin
  - docs
  - domain
  - modules
  - standalone
  - themes
  - welcome-content
  - ataccama.json
  - jboss-modules.jar
  - LICENSE.txt
  - start_keycloak.bat
  - start_keycloak.sh
  - stop_keycloak.bat
  - stop_keycloak.sh
  - version.txt
- p2
- pgsql
  - pginit
  - pgsql
  - pginit.zip
  - start_postgres.bat
  - start_postgres.sh
  - stop_postgres.bat
  - stop_postgres.sh
- plugins
- rdm
  - bin
  - doc
  - etc
  - legal
  - lib
  - license
  - log
  - plugin
  - storage
  - tmp

This step is necessary when running the example projects to ensure the scripts in **Files** > **bin** work without issues.

You can now run the RDM example project.

2.3.3 Starting the Example

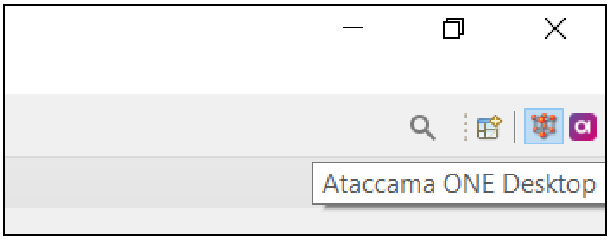
1. Navigate to `rdm/etc` and open `application.properties`. This is the main file for RDM configuration. Uncomment the rows, save, and exit.



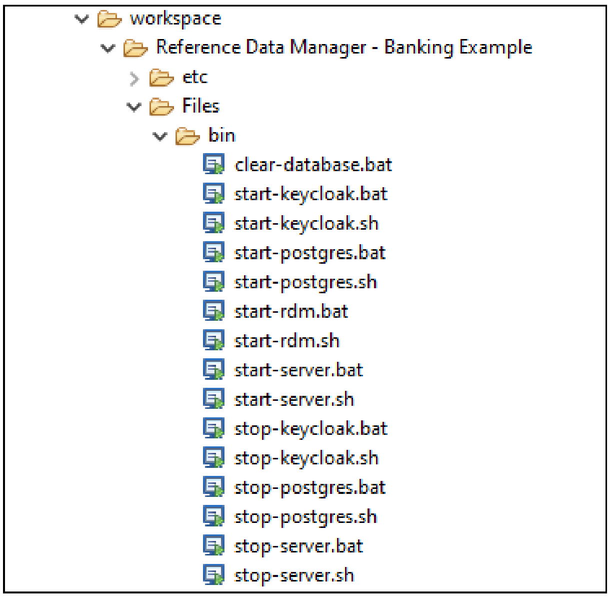
```
# Uncomment properties below for run without Configuration Service. Keep these first two lines commented.
# Please, check that you have correct values for connection to your database and keycloak.

ataccama.one.rdm.license-folder=
ataccama.one.rdm.server.url=http://localhost:8061
ataccama.one.rdm.application.url=localhost:8060
ataccama.config-service.runtime=off
ataccama.one.rdm.datasource.rdm-data.url=jdbc:postgresql://localhost:5432/rdm_db
ataccama.one.rdm.datasource.rdm-data.jdbcUrl=jdbc:postgresql://localhost:5432/rdm_db
ataccama.one.rdm.datasource.rdm-data.username=rdm
ataccama.one.rdm.datasource.rdm-data.password=rdm
ataccama.one.rdm.datasource.rdm-data.driverClassName=org.postgresql.Driver
ataccama.one.rdm.datasource.rdm-repo.url=jdbc:postgresql://localhost:5432/rdm_db
ataccama.one.rdm.datasource.rdm-repo.jdbcUrl=jdbc:postgresql://localhost:5432/rdm_db
ataccama.one.rdm.datasource.rdm-repo.username=rdm
ataccama.one.rdm.datasource.rdm-repo.password=rdm
ataccama.one.rdm.datasource.rdm-repo.driverClassName=org.postgresql.Driver
ataccama.authentication.keycloak.server-url=http://localhost:8083/auth
ataccama.authentication.keycloak.realm=ataccamaone
ataccama.authentication.keycloak.admin.client-id=rdm-admin-client
ataccama.authentication.keycloak.admin.secret=rdm-admin-client-s3cret
ataccama.authentication.keycloak.token.client-id=rdm-token-client
ataccama.authentication.keycloak.token.secret=rdm-token-client-s3cret
ataccama.authentication.keycloak.token.issuer=${ataccama.authentication.keycloak.server-url}/realms/${ataccama.authentication.keycloak.realm}
ataccama.authentication.keycloak.public.client-id=rdm-webapp-public-client
ataccama.logging.plainTextConsoleAppender=true
ataccama.logging.jsonConsoleAppender=false
ataccama.logging.plainTextFileAppender=true
ataccama.logging.jsonFileAppender=false
```

2. Open ONE Desktop and select the ONE Desktop perspective.



3. In Model Explorer, select **New** and then **Model Project**.
4. In **Use template:** select `Reference Data Manager - Banking Example`.
5. Open `Files/bin` folder.



- workspace
  - Reference Data Manager - Banking Example
    - etc
    - Files
      - bin
        - clear-database.bat
        - start-keycloak.bat
        - start-keycloak.sh
        - start-postgres.bat
        - start-postgres.sh
        - start-rdm.bat
        - start-rdm.sh
        - start-server.bat
        - start-server.sh
        - stop-keycloak.bat
        - stop-keycloak.sh
        - stop-postgres.bat
        - stop-postgres.sh
        - stop-server.bat
        - stop-server.sh

6. Double-click **start-keycloak.bat** and wait for Keycloak to start in a separate window titled KeycloakDemoWindow. Wait for the following message to appear in the window: `Keycloak ... started in <number> s. Listening on: http://0.0.0.0:8083`.
7. Double-click **start-postgres.bat**. The database must be running before you execute the server script in the following step.
8. Double-click on **start-server.bat** and wait for the following message to appear in a separate window: `Started ... in <number> seconds`.
9. Double-click on **start-rdm.bat** and wait for the following message to appear in a separate window: `Started ... in <number> seconds`.
10. Once everything starts, open http://localhost:8060 (RDM Web Application) or http://localhost:8060/admin (RDM Admin Console) in your browser.
11. Login into the application with the following credentials:

Username	Password	Can access RDM System Console
admin	admin	✓

12. To access the RDM Online Server Web Console, open http://localhost:8061 in your browser. There you can find the following information:
    - Available online services
    - Configured workflows
    - Defined schedulers

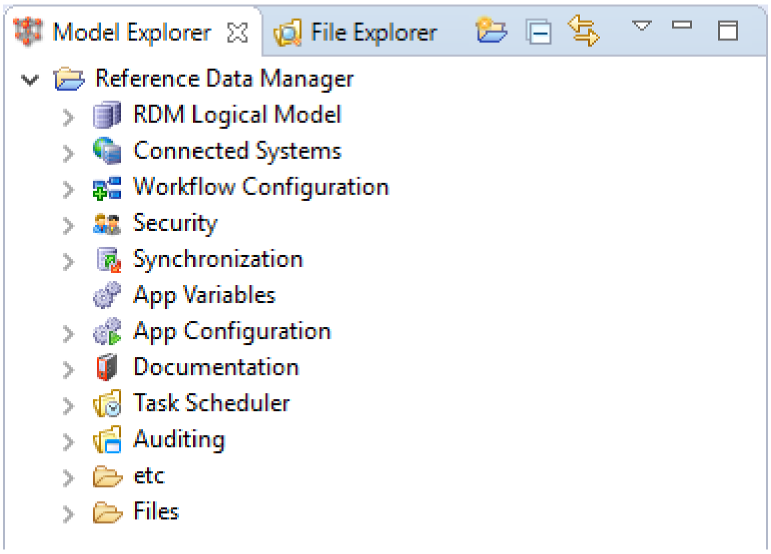
13. For additional work on configuration, define in **File Explorer** perspective following resources:
    - DB resource called `rdm_db` with attributes **Host:** `localhost`, **Port:** `5432`, **Database name:** `rdm_db`, **Username:** `rdm` and **Password:** `rdm`,
    - Server resource called `rdmapp` with attributes **URL:** `http://localhost:8060/admin`, **User:** `admin` and **Password:** `admin`.
    - Server resource called `server` with attributes **URL:** `http://localhost:8061`.

2.3.4 Stopping the Example

1. When you are finished working with RDM Banking Example, close the promts windows.
2. To stop the database, go to `Files/bin` and double-click **stop-postgres.bat**.
3. Stop Keycloak by double-clicking **stop-keycloak.bat**.

2.3.5 Exploring RDM Example Project

The RDM model project contains a set of nodes for configuring the RDM solution:



RDM Model Project

RDM Logical Model

The **RDM Logical Model** node defines all configuration related to the reference data itself, its entities, the relationships and the different ways to categorize and organize reference data for usage.

The data model supports a virtually unlimited number of tables and relationships. Basic functions for data modeling involve adding, editing, and deleting new entities, defining entity attributes and parameters, and establishing relationships between entities.

As an essential part of RDM metadata, complex multi-level hierarchies can be created in the data model. The types of hierarchies supported include balanced, unbalanced, and recursive.

The principal components of the RDM Logical Model are Domains (data types), Tables, and Relationships. See Creating the RDM Data Model.

Auxiliary components – Views, Hierarchies, Categories, and Data sets – allow the using of tables (or specific attributes) to create custom data views or organize tables into categories and hierarchies. See Creating Additional Reference Data Views.

Connected Systems

Connected Systems is where you add databases and servers to import and export data to/from RDM.

The node has the following nodes:

- **Databases** is where you add connections to external databases, with which you want to synchronize your RDM Repository. You also create a data model of tables you want to synchronize with RDM tables and map the attributes of one to the other. See Adding Databases Synchronized with RDM.
- **Servers** is where you add connections to remote servers. These are used for file transfers with reference data for your data imports and exports. See Adding Remote Servers for File Transfers.

Workflow Configuration

Workflow Configuration is where you define your change approval workflows.

The node has the following sub-nodes:

- **Tables** is where you set up table-specific workflows (see Configuring RDM Workflows) and action-triggered email notifications (see Configuring RDM Email Notifications)
- **Statuses** is where you define custom logical states between which records participating in workflows move.
- **Emails** is where you create email templates to be used for per-table record change notifications and workflows and summary notifications. See Creating RDM Email Templates.
- **Summary Notifications** is where you assign email templates to notifications triggered by actions simultaneously performed on multiple records. See Configuring RDM Email Notifications.

Synchronization

Synchronization makes it possible to set up imports and exports to text files, synchronization with external databases, and web service configurations to writing and reading data from RDM. See RDM Synchronization.

App Variables

App Variables is where you set up various web application and data model variables, like database type used, web application GUI language, generated primary key attribute name, and others.
See RDM App Variables.

App Configuration

App Configuration is used to generate a zip file with configuration files to deploy to the RDM web application. It also contains links to files in the `etc` folder. See How to Deploy an RDM Web App Configuration.

Documentation

Documentation contains technical and business documentation about the configured data model.

Task Scheduler

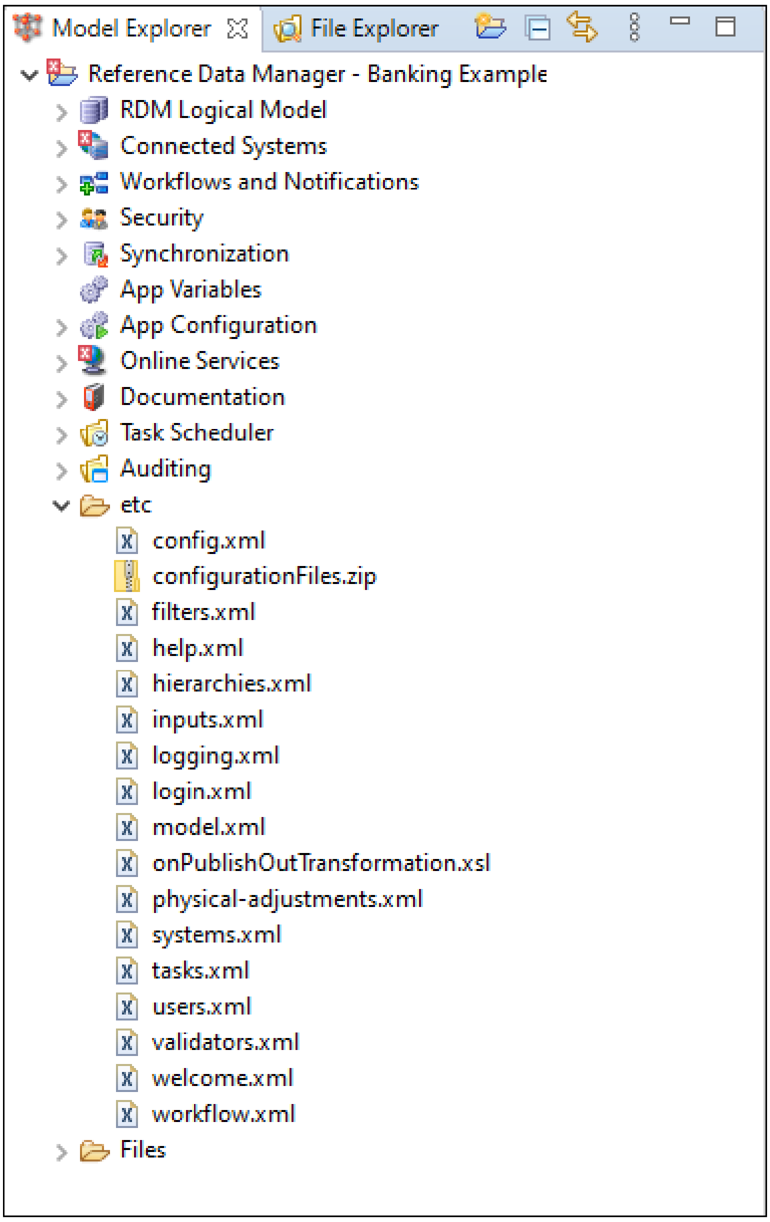
Task Scheduler is where you schedule synchronization or batch export tasks. See Scheduling RDM Tasks.

Auditing

Auditing is where you configure auditing of actions made by web application users. See Setting up RDM Auditing.

etc

etc contains the configuration XML files and the resulting ZIP generated by the **App Configuration** node.



- Reference Data Manager - Banking Example
  - RDM Logical Model
  - Connected Systems
  - Workflows and Notifications
  - Security
  - Synchronization
  - App Variables
  - App Configuration
  - Online Services
  - Documentation
  - Task Scheduler
  - Auditing
  - etc
    - config.xml
    - configurationFiles.zip
    - filters.xml
    - help.xml
    - hierarchies.xml
    - inputs.xml
    - logging.xml
    - login.xml
    - model.xml
    - onPublishOutTransformation.xsl
    - physical-adjustments.xml
    - systems.xml
    - tasks.xml
    - users.xml
    - validators.xml
    - welcome.xml
    - workflow.xml
  - Files

Files

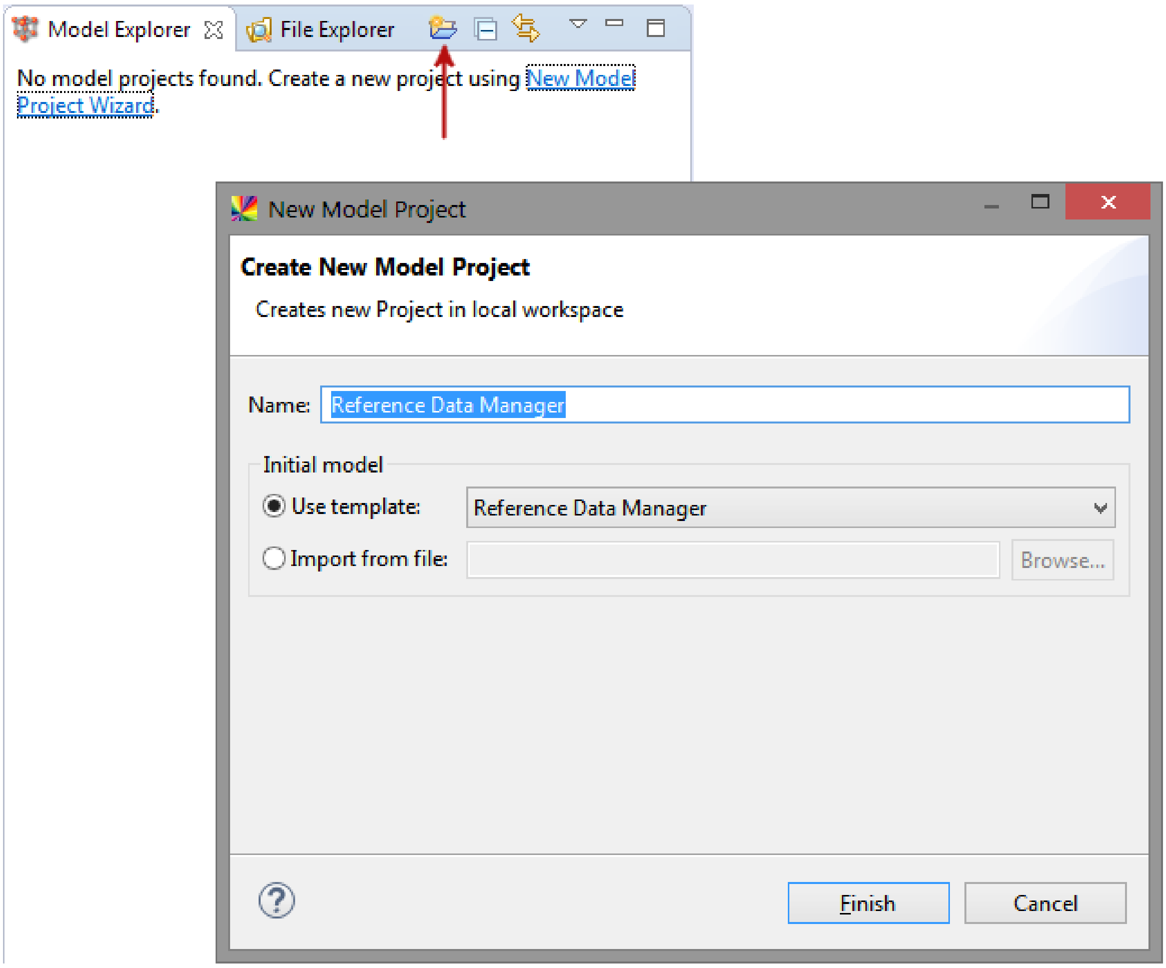
Files contains a data folder containing generated plans, workflow and scheduler files, and HTML documentation files. Any other files, like CSV or TXT batch load data files or custom validation, and enrichment components are recommended to be put here.

2.4 Building the RDM Solution from Scratch

This article is intended to guide you through the creation and development of your first RDM project in ONE Desktop from start to finish, including the deployment of files generated based on your configuration to the RDM web application and runtime.

2.4.1 Create a New Project

1. Start ONE Desktop.
2. Switch to the Model Explorer view and click the **New Model Project** icon. Give the project a name, keep the **Use template** option at *Reference Data Manager* and click **Finish**.



No model projects found. Create a new project using New Model Project Wizard.

Create New Model Project

Creates new Project in local workspace

Name: Reference Data Manager

● Use template: Reference Data Manager

Creating a New Reference Data Manager Project

A new RDM project is created.

2.4.2 Configure Your RDM Project

After you have created your RDM project, it is almost blank. Now you can go ahead and building your RDM solution.

The steps below set forth a suggested course of building the RDM solution designed so that you to complete all prerequisites for each subsequent step. You could, however, go about the task in a different order since most tasks after step 3 are independent of each other but require steps 1,2, or 3 to be completed.

#	Step	Required	Why Configure It
1	Configure required application settings	Yes	Some of the application settings, like the RDM Repository database type and web application URL are required to make the application work. Open to the **App Variables** node and fill in all the fields marked as required.
2	Define RDM User Repositories and Roles	Yes	It is necessary to let RDM know how it should authorize users: against users and roles found in a user directory (LDAP) or added manually.
3	Create the RDM Data Model	Yes	This is the core step of the solution: you define what reference data dictionaries you want to have, their attributes, and relationships between these dictionaries.
4	Create Additional Reference Data Views	No	After creating the data model, you can configure alternative views on your reference data: hierarchical parent-child structure, tables grouped by custom categories, etc.

#	Step	Required	Why Configure It
5	Set up Synchronization with External Systems	No	To propagate data to other systems in your organization, you can set up one several synchronization tasks: database synchronization, text file imports/exports, and online services for reading and writing data
6	Set up Change Approval Workflows	No	If you want to take control of data editing and publishing in RDM, set up change approval workflows for any selected table: design steps, configure step conditions, select participants and columns available for modification in each step, and set up email notifications for completion of steps.
7	Set up Email Notifications	No	If you want to notify selected users about activity in the RDM application, set up email notifications for events like publishing, rejecting, and sending a record for publishing.
8	Configure Initial Loads to RDM Tables	No	Populating your RDM tables with plans is great option if you need to modify/enrich/filter/transform your input data before importing it to RDM.
9	Set up RDM Auditing	No	RDM supports auditing of user actions inside the web application. You can configure what should be logged.

2.4.3 Deploy Web Application Configuration

After setting up everything needed for you RDM solution, you need to generate XML configuration files (packed into a ZIP) and deploy to the RDM web application. See How to Deploy an RDM Web App Configuration.

You need to perform this step every time you make changes to your project.

2.4.4 Deploy Generated Files

Some optional configurations like RDM Synchronization result in generating ONE files like plans (`*.plan`), workflows (`*.ewf`), schedules (`*.sch`), and web service configurations (`*.online`). By default these files are generated (by you) to the `Files` folder of the RDM project. Therefore, you can simply copy the `Files` folder of your RDM project to the server that contains the RDM Runtime.

You need to perform this step every time you generate new versions of files mentioned above.

Check Runtime and Server Configuration

After you copied the `Files` folder to your RDM server, check and correct the paths in the following components in your Server Configuration files:

- Online Services Component
- Workflow Server Component
- Scheduler Server Component

The paths should point to relevant files in the sub-folders of the `Files` folder.

Check whether your Runtime Configuration file contains the definitions of servers and databases that you use for RDM synchronization, initial loads, and validation/enrichment services. The same applies to folder shortcuts. See Importing and Exporting Runtime Configuration to learn how to exports these definitions directly from the IDE.

2.5 Creating the RDM Data Model

The RDM Logical Model can be created in three ways:

- Manually creating the model.
- Importing database metadata (table and attributes names and attribute data types).
- Importing the model from a modeling tool (XMI file).

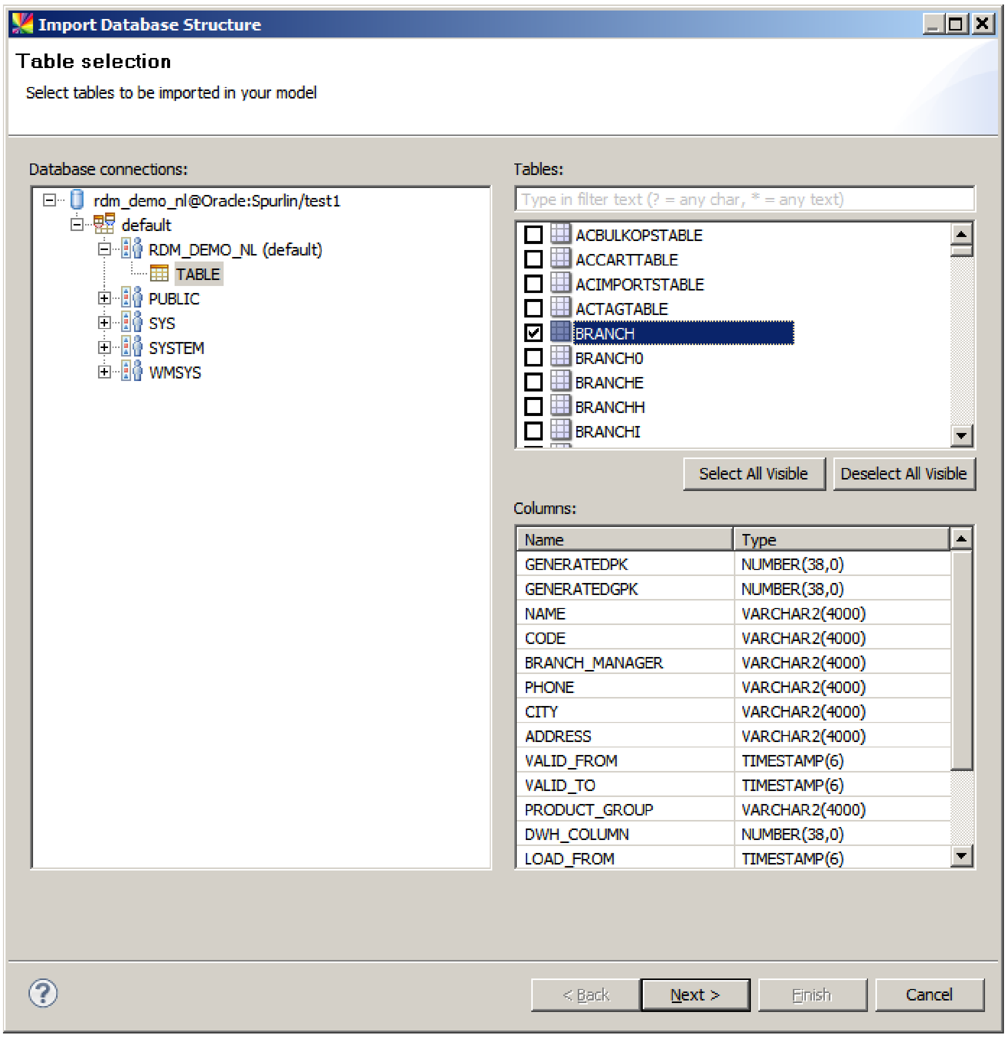
RDM supports incremental importing: when importing a model from an external source, such as an XMI file or a database schema, RDM computes the difference between the external and current RDM model and presents elements to add and delete.

2.5.1 Manual Model Definition

Manual model definition is done by adding domains, tables, and relationships between tables. See RDM Domains, RDM Tables, RDM Relationships.

2.5.2 Importing Database Metadata

1. Right-click **RDM Logical Model > Import database metadata.**
2. Select one of previously added database connections and expand to the needed schema. Then select tables to import in the **Tables** section and click **Next**.


Import Database Structure
Table selection
Select tables to be imported in your model
Database connections:
Tables:
Type in filter text (? = any char, * = any text)
Select All Visible
Deselect All Visible
Columns:
< Back
Next >
Finish
Cancel

Database connections:

- rdm_demo_nl@Oracle:Spurlin/test1
  - default
    - RDM_DEMO_NL (default)
      - TABLE
    - PUBLIC
    - SYS
    - SYSTEM
    - WMSYS

Tables:

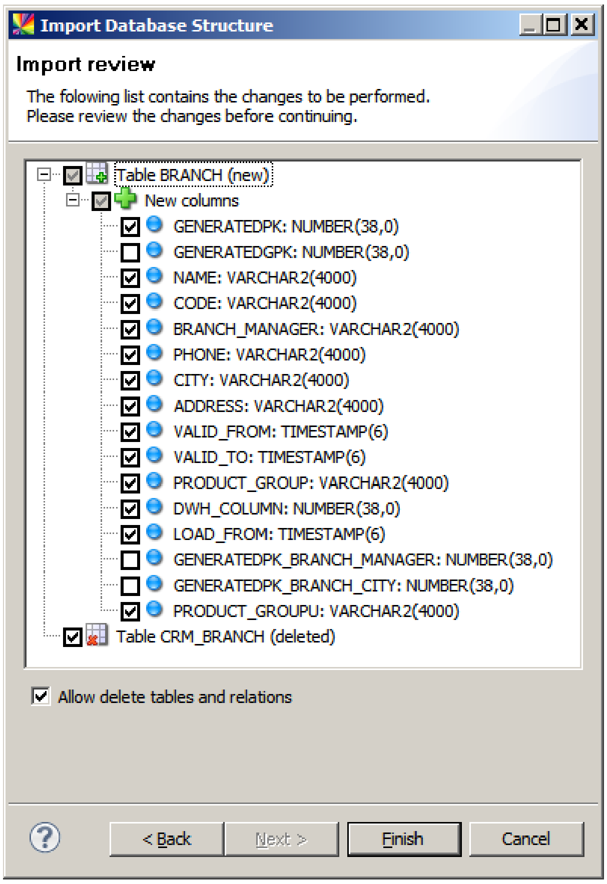
- ☐ ACBULKOPSTABLE
- ☐ ACCARTTABLE
- ☐ ACIMPORTSTABLE
- ☐ ACTAGTABLE
- ☑ BRANCH
- ☐ BRANCH0
- ☐ BRANCHE
- ☐ BRANCHH
- ☐ BRANCHI

Columns:

Name	Type
GENERATEDPK	NUMBER(38,0)
GENERATEDGPK	NUMBER(38,0)
NAME	VARCHAR2(4000)
CODE	VARCHAR2(4000)
BRANCH_MANAGER	VARCHAR2(4000)
PHONE	VARCHAR2(4000)
CITY	VARCHAR2(4000)
ADDRESS	VARCHAR2(4000)
VALID_FROM	TIMESTAMP(6)
VALID_TO	TIMESTAMP(6)
PRODUCT_GROUP	VARCHAR2(4000)
DWH_COLUMN	NUMBER(38,0)
LOAD_FROM	TIMESTAMP(6)

3. Review the columns that will be added and removed and unselect when necessary.

> The **Allow deleting tables and relations** checkbox will let the importer delete unmatched tables and relationships from the current RDM model.



Import review

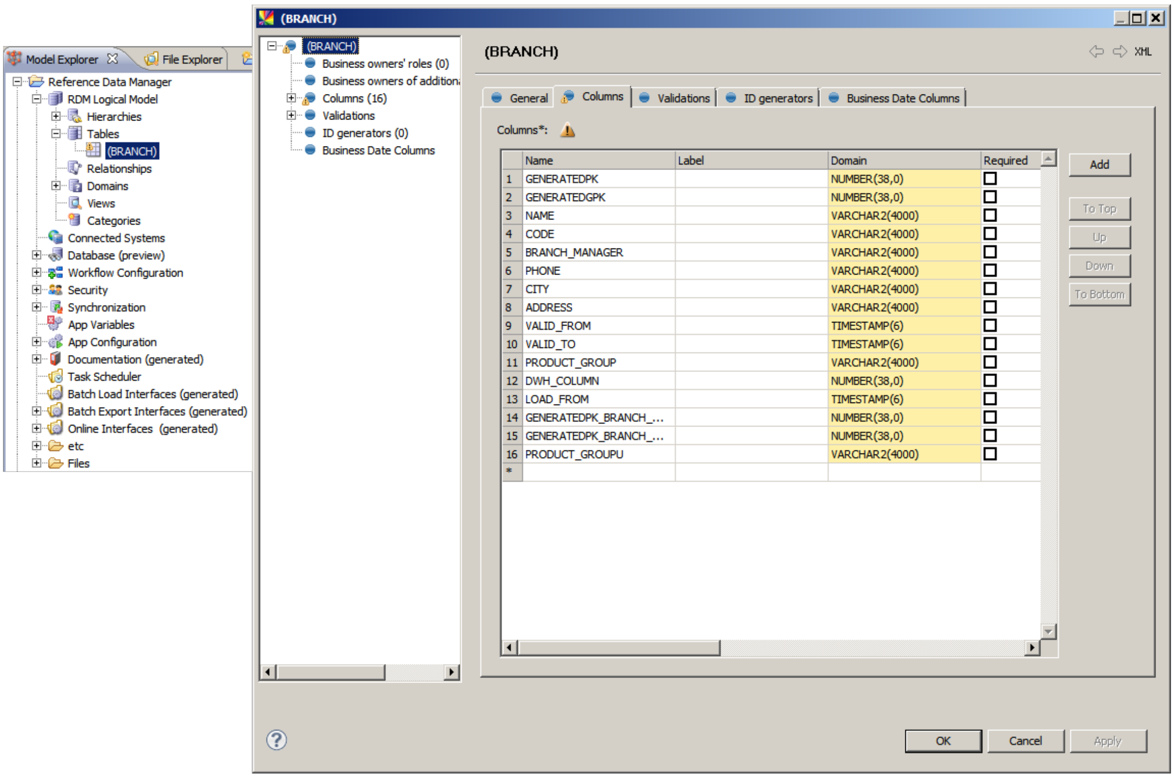
The folowing list contains the changes to be performed.
Please review the changes before continuing.

- ☑ Table BRANCH (new)
  - ☑ New columns
    - ☑ GENERATEDPK: NUMBER(38,0)
    - ☐ GENERATEDGPK: NUMBER(38,0)
    - ☑ NAME: VARCHAR2(4000)
    - ☑ CODE: VARCHAR2(4000)
    - ☑ BRANCH_MANAGER: VARCHAR2(4000)
    - ☑ PHONE: VARCHAR2(4000)
    - ☑ CITY: VARCHAR2(4000)
    - ☑ ADDRESS: VARCHAR2(4000)
    - ☑ VALID_FROM: TIMESTAMP(6)
    - ☑ VALID_TO: TIMESTAMP(6)
    - ☑ PRODUCT_GROUP: VARCHAR2(4000)
    - ☑ DWH_COLUMN: NUMBER(38,0)
    - ☑ LOAD_FROM: TIMESTAMP(6)
    - ☐ GENERATEDPK_BRANCH_MANAGER: NUMBER(38,0)
    - ☐ GENERATEDPK_BRANCH_CITY: NUMBER(38,0)
    - ☑ PRODUCT_GROUPU: VARCHAR2(4000)
- ☑ Table CRM_BRANCH (deleted)

☑ Allow delete tables and relations

4. Click **Finish** to complete the import.

5. Fix any problems that appear in the **Properties** tab of the Status Panel, e.g., invalid domains. See RDM Domains.



Columns*:

	Name	Label	Domain	Required
1	GENERATEDPK		NUMBER(38,0)	☐
2	GENERATEDGPK		NUMBER(38,0)	☐
3	NAME		VARCHAR2(4000)	☐
4	CODE		VARCHAR2(4000)	☐
5	BRANCH_MANAGER		VARCHAR2(4000)	☐
6	PHONE		VARCHAR2(4000)	☐
7	CITY		VARCHAR2(4000)	☐
8	ADDRESS		VARCHAR2(4000)	☐
9	VALID_FROM		TIMESTAMP(6)	☐
10	VALID_TO		TIMESTAMP(6)	☐
11	PRODUCT_GROUP		VARCHAR2(4000)	☐
12	DWH_COLUMN		NUMBER(38,0)	☐
13	LOAD_FROM		TIMESTAMP(6)	☐
14	GENERATEDPK_BRANCH_...		NUMBER(38,0)	☐
15	GENERATEDPK_BRANCH_...		NUMBER(38,0)	☐
16	PRODUCT_GROUPU		VARCHAR2(4000)	☐
*				

6. Add relationships between tables. See RDM Relationships.

2.5.3 Importing from a Modeling Tool

RDM supports importing the data model definition from data modeling tools like ERwin and SAP PowerDesigner.

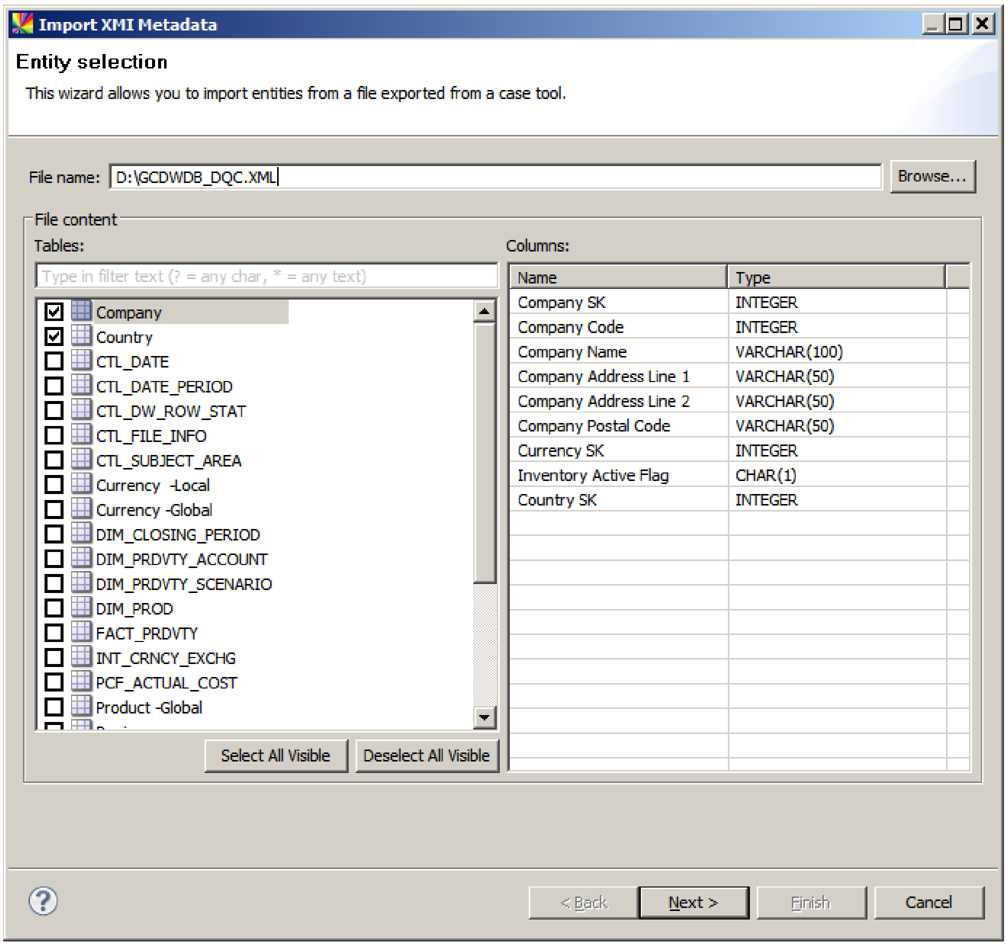
1. Right-click **RDM Logical Model > Import XMI...**

> Some ERwin models may require using **RDM Logical Model > Import from XMI - configurable**. The dialogs will be slightly different than those below.

> For models made in SAP PowerDesigner (`.pdm` extension), it is also possible to import domains in addition to tables and relationships. Use the **RDM Logical Model > Import from XMI - configurable** option.

2. Browse for an XMI/XML file in the **File name** field.

3. Select tables in the **Tables** section and click **Next.**



Entity selection

This wizard allows you to import entities from a file exported from a case tool.

File name: D:\GCDWDB_DQC.XML

Tables:

Type in filter text (? = any char, * = any text)

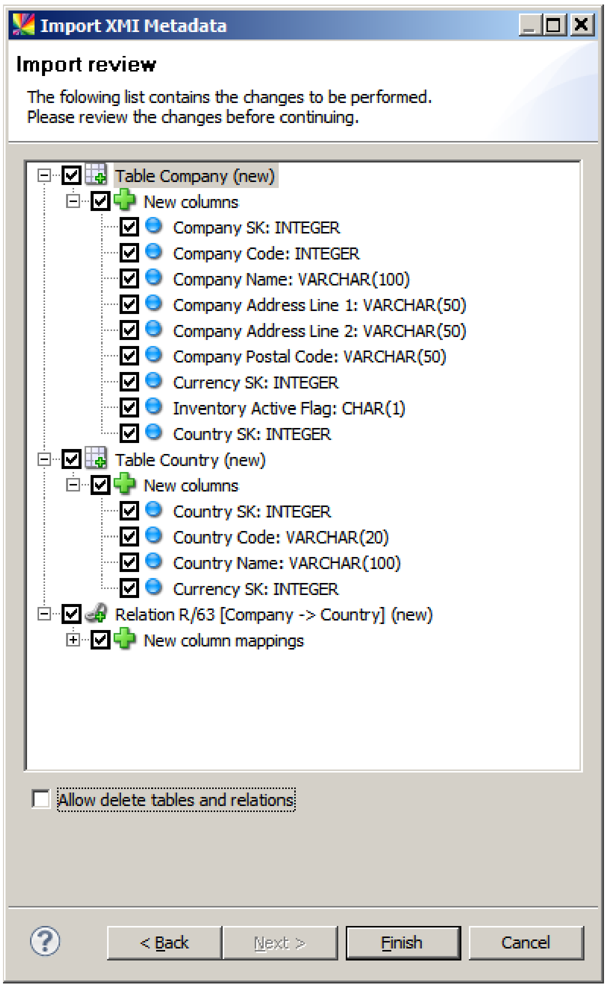
- ☑ Company
- ☑ Country
- ☐ CTL_DATE
- ☐ CTL_DATE_PERIOD
- ☐ CTL_DW_ROW_STAT
- ☐ CTL_FILE_INFO
- ☐ CTL_SUBJECT_AREA
- ☐ Currency -Local
- ☐ Currency -Global
- ☐ DIM_CLOSING_PERIOD
- ☐ DIM_PRDVTY_ACCOUNT
- ☐ DIM_PRDVTY_SCENARIO
- ☐ DIM_PROD
- ☐ FACT_PRDVTY
- ☐ INT_CRNCY_EXCHG
- ☐ PCF_ACTUAL_COST
- ☐ Product -Global

Columns:

Name	Type
Company SK	INTEGER
Company Code	INTEGER
Company Name	VARCHAR(100)
Company Address Line 1	VARCHAR(50)
Company Address Line 2	VARCHAR(50)
Company Postal Code	VARCHAR(50)
Currency SK	INTEGER
Inventory Active Flag	CHAR(1)
Country SK	INTEGER

4. Review the tables, columns, and relationships that will be added and removed (unselect when necessary).

> The *Allow deleting tables and relations* checkbox will let the importer delete unmatched tables and relationships from the current RDM model.



Import review

The folowing list contains the changes to be performed.
Please review the changes before continuing.

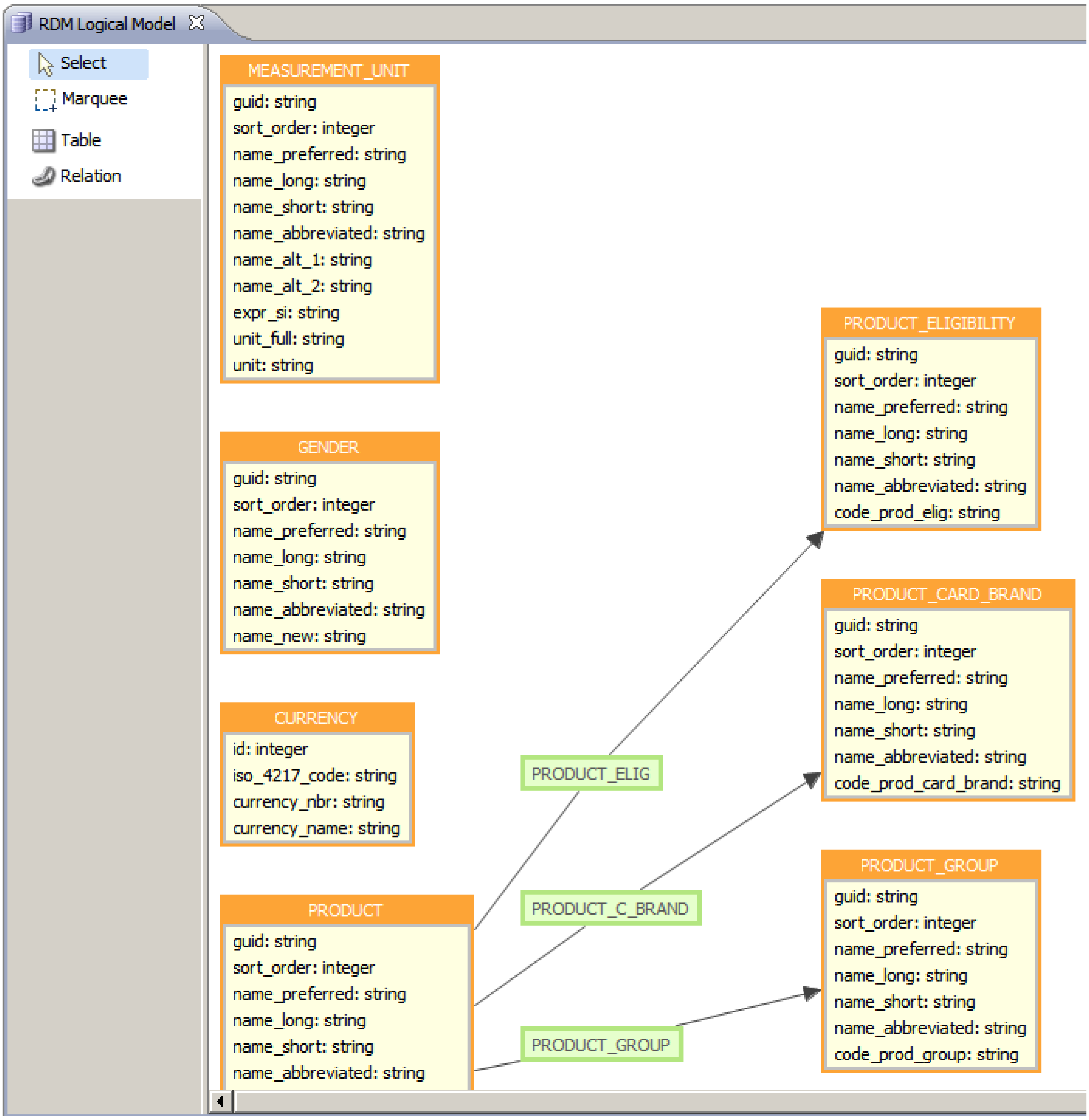
- ☑ Table Company (new)
  - ☑ New columns
    - ☑ Company SK: INTEGER
    - ☑ Company Code: INTEGER
    - ☑ Company Name: VARCHAR(100)
    - ☑ Company Address Line 1: VARCHAR(50)
    - ☑ Company Address Line 2: VARCHAR(50)
    - ☑ Company Postal Code: VARCHAR(50)
    - ☑ Currency SK: INTEGER
    - ☑ Inventory Active Flag: CHAR(1)
    - ☑ Country SK: INTEGER
- ☑ Table Country (new)
  - ☑ New columns
    - ☑ Country SK: INTEGER
    - ☑ Country Code: VARCHAR(20)
    - ☑ Country Name: VARCHAR(100)
    - ☑ Currency SK: INTEGER
- ☑ Relation R/63 [Company -> Country] (new)
  - ☑ New column mappings

☐ Allow delete tables and relations

5. Click **Finish** to complete the import.
6. Fix any problems that appear in the **Properties** tab of the Status Panel, e.g., invalid domains. See RDM Domains.

2.5.4 Viewing the Data Model

You can view the created or imported data model by right-clicking the **RDM Logical Model** node and selecting **Edit schema**.



RDM Logical Model Schema

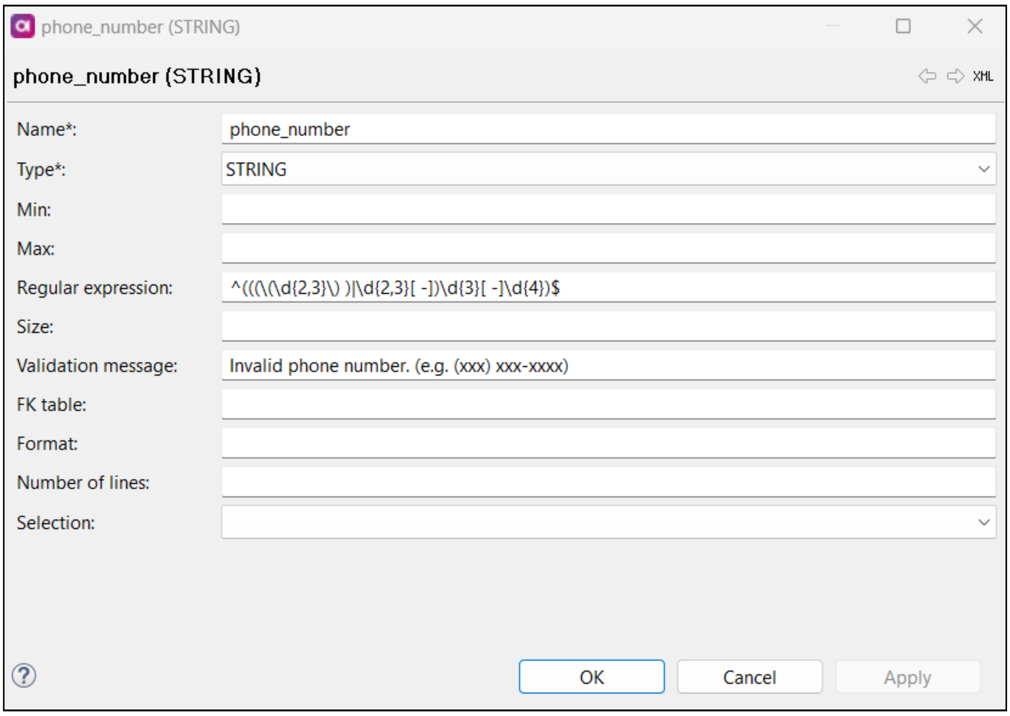
Adding tables and relationships is described in RDM Tables and RDM Relationships.

2.5.5 RDM Domains

Domains are custom constraints, assigned to table columns. For example, you want to limit the values entered into a given column to letters or to a phone number in a particular format (like on the screenshot below). These and other business requirements to dictionary columns are possible to configure via domains by creating custom limitations by format, regular expressions, string size, and values ranges for numeric data types. See Domain Attributes below for all definition options.

Preconfigured Domains

Six default domains are pre-configured in a blank RDM project: BOOLEAN, STRING, INTEGER, LONG, FLOAT, DATE, and DATETIME.



Name*: phone_number
Type*: STRING
Min:
Max:
Regular expression: ^(((\(\d{2,3}\))|\d{2,3}[-])\d{3}[-]\d{4})$
Size:
Validation message: Invalid phone number. (e.g. (xxx) xxx-xxxx)
FK table:
Format:
Number of lines:
Selection:

Example Domain Configuration

Creating a New Domain

1. Expand **RDM Logical Model > Domains.**
2. Right-click **Domains** and select **New domain**.
3. Fill in the attributes (see the following table).
4. Click **OK** to save changes.

The domain is defined and can be assigned to an attribute in an RDM table. See RDM Tables.

Domain Attributes

Attribute	Required	Description
Name	Y	Name of the domain.

Attribute	Required	Description
Type	Y	ONE data type to which the domain is mapped. The following values are possible: • **boolean** - logical values which can be either *true* or *false.* • **string** - sequences of characters that are treated as text. • **integer** - integers ranging from -2^{31} to 2^{31}-1. • **long** - arbitrary-precision signed decimal integers. • **float** - arbitrary-precision signed decimal numbers; the output precision and the precision of division operation can be controlled by the *double.scale* runtime parameter which has the value of 10 by default. • **date** - calendar dates • **datetime** - calendar dates with time fields. • **mnreferences** - RDM pseudo-data type for creating MN relationships between tables. See How to Create an MN Relationship in RDM.
Min	N	Definition of the minimum value.
Max	N	Definition of the maximum value.
Regular expression	N	A regular expression that validates the input. See Regular Expressions for help with regular expressions.
Size	N	Limits the number of characters that can be entered into the field; used primarily for string values; float and integer attributes will be converted to string to check their size.
Foreign key table	N	Name of the referenced table (required for mnreferences type).

Attribute	Required	Description
Format	N	Custom formats to change the way RDM displays domain. See Format Definitions below for information on possible format definitions depending on data type.
Number of lines	N	Determines the number of rows that are displayed in the create/edit dialog in the RDM web application (specified only for string data type). When the number of lines is specified, **Size** must be specified, too.
Selection	N	Defines how Boolean values are displayed in the web application, as a dropdown, a checkbox, or a toggle switch. Possible values: `DROPDOWN`, `CHECKBOX`, `TOGGLE` (default), `NULL`. Can be used only with attributes of Boolean data types.
Validation message	N	A message that appears in the web application when domain-defined constraints are violated. Validation messages are only available for string domains.

The table below specifies which domain attributes are available for each data type:

Domain Type	Min	Max	Regular expression	Size	Validation message	FK table	Format	Number of lines	Selection
boolean	-	-	-	-	+	-	+	-	+

Domain Type	Min	Max	Regular expression	Size	Validation message	FK table	Format	Number of lines	Selection
date	-	-	-	-	+	-	+	-	-
datetime	-	-	-	-	+	-	+	-	-
float	+	+	-	+	+	-	+	-	-
integer	+	+	-	+	+	-	+	-	-
long	+	+	-	+	+	-	+	-	-
mnreferences	-	-	-	-	+	+	-	-	-
string	-	-	+	+	+	-	-	+	-

Format Definitions

This section specifies what custom formats can be configured for each data type, including how data will be displayed in the web application.

Datetime Data Type

Definition of the format for dates uses the following pattern letters:

- **y** - year
- **M** - month
- **d** - day
- **H** - hours
- **m** - minutes
- **s** - seconds
- **S** - milliseconds

The number of pattern letters used for defining each part of the date influences the format. See examples below.

Definition	Result
dd.MM.yyyy	01.02.2015

Definition	Result
dd.MM.yyyy HH:mm:ss	01.02.2015 03:04:05
dd.MM.yyyy HH:mm:ss.SSS	01.02.2015 03:04:05.600
yyyy-MM-dd	2015-02-01
yyyy-MM-dd HH:mm:ss	2015-02-01 03:04:05
yyyy-MM-dd HH:mm:ss.SSS	2015-02-01 03:04:05.600

If the format is not defined, the following default definition (for English set as the web application language) is used from Google Web Toolkit:

Definition	Result
yyyy MMM d HH:mm:ss	2015 Jan 1 01:02:03

Detailed information can be found at the following locations:

- http://www.gwtproject.org/javadoc/latest/com/google/gwt/i18n/client/DateTimeFormat.html
- https://gwt.googlesource.com/gwt/+/release/2.7/user/src/com/google/gwt/i18n/client/constants/

Numeric Data Types

Definition of the format for numbers in the integer, long, and float formats uses the following symbols:

- **#** - stands for one digit. If a displayed number does not contain a digit in the position of a hash, the position will be empty
- **0** - stands for one digit; used to display a zero
  - before the decimal point for fractions like 0.25 instead of .25
  - after the decimal point for setting the number decimal places
- **,** - a thousands separator (the actual separator symbol depends on the RDM web application language settings)
- **.** - decimal mark (the actual decimal mark symbol depends on the RDM web application language settings)

If the format is not defined, the following default definition (for English set as the web application language) is used from the Google Web Toolkit: `#,##0.###`.

Examples for Float

Format	123456789.0	.9	.99	.994	.995	.996
[empty]	123 456 789	0.9	0.99	0.994	0.995	0.996
####.##	123456789	0.9	0.99	0.994	1	1
#,###.##	123 456 789	0.9	0.99	0.99	1	1
#,##0.##	123 456 789	0.9	0.99	0.99	1	1
#,##0.00	123 456 789.00	0.90	0.99	0.99	1.00	1.00
0,000.00	123 456 789.00	0 000.90	0 000.99	0 000.99	0 001.00	0 001.00

Examples for Integer and Long

Format	0	1234	123456789
[empty]	0	1 234	123 456 789
#	0	1234	123456789
#,###,###	0	1 234	123 456 789
0,000,000	0 000 000	0 001 234	123 456 789

Boolean

Boolean values have three possible formats:

Format	Value for True	Value for False
TRUE_FALSE	True	False
VALID_INVALID	Valid	Invalid
YES_NO	Yes	No

If the format is not defined, the domain will take the format of TRUE_FALSE.

2.5.6 RDM Tables

The **Tables** sub-node in the **RDM Logical Model** node allows creating tables of reference data (dictionaries), defining their structure, validations, and various other functions.

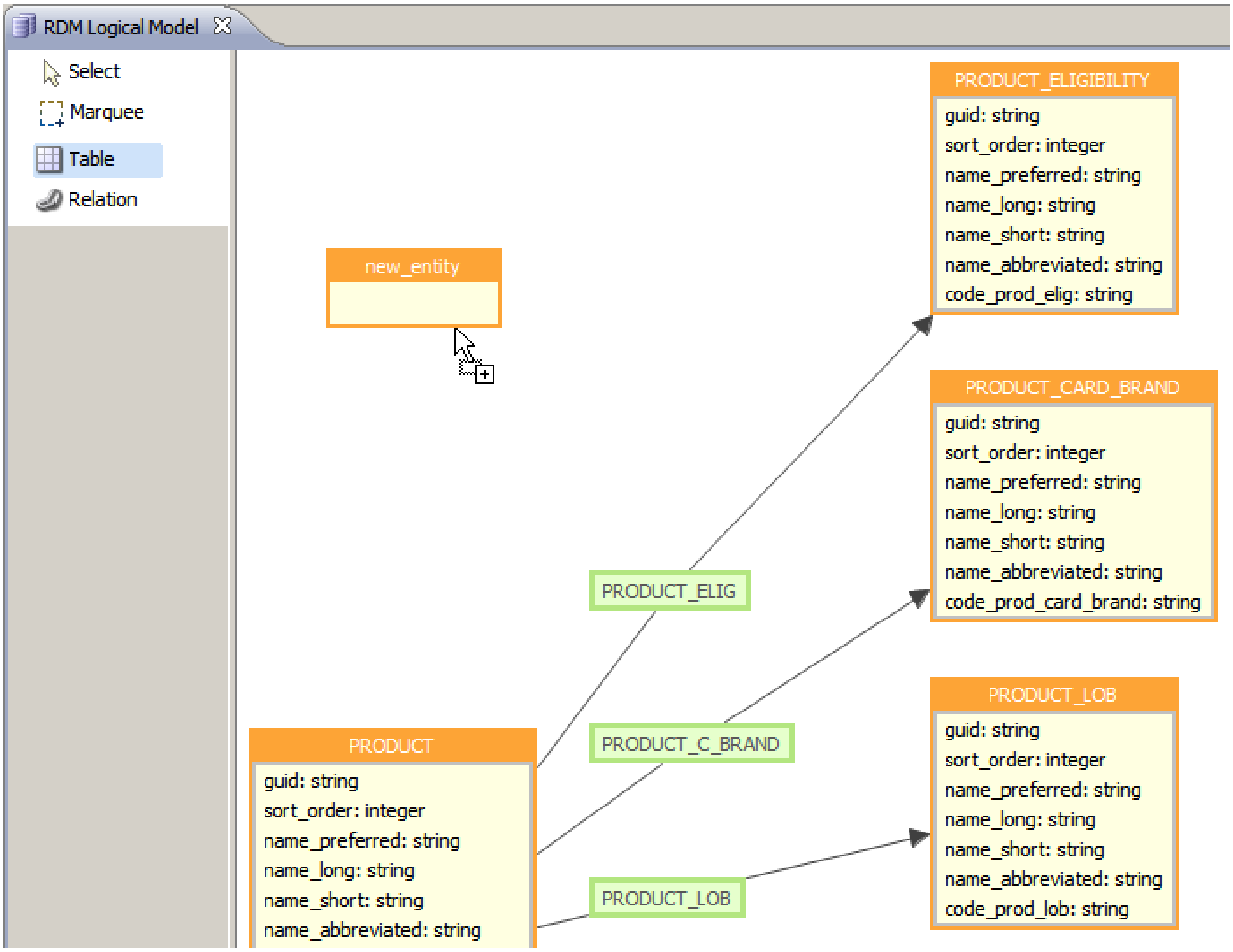
After creating a table in the RDM Logical model and deploying the configuration to the web application, the administrator needs to set the permissions for the table, so that he can view it, edit it and perform other operations like batch import and export.

Adding a New Table in the Model Explorer

1. Right-click **Tables > New table.**
2. Fill in the attributes in all tabs.
3. Click **OK** to save changes.

Adding a New Table in the Data Model Editor

1. Right-click **RDM Logical Model > Edit schema**. The canvas of the Data Model Editor will open, with the elements palette on the left.
2. Click **Table** on the palette and then click on the canvas - an empty table will be created.
3. Double-click the new empty table.
4. Fill in the attributes in all tabs.
5. Click **OK** to save changes.



Adding a Table in the Data Model Editor

Table Attributes

General

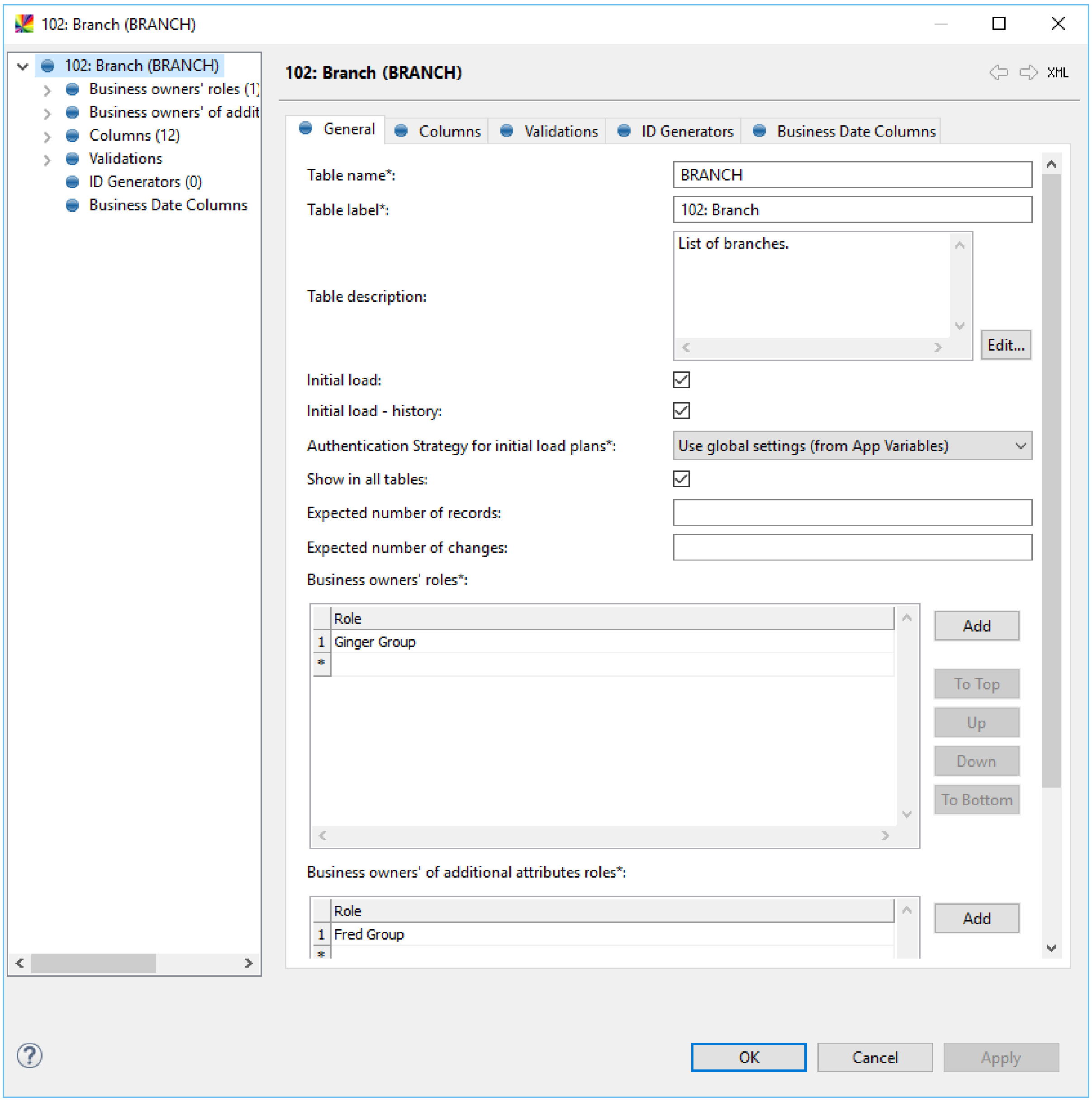


Table name*: BRANCH

Table label*: 102: Branch

Table description: List of branches.

Initial load: ☑

Initial load - history: ☑

Authentication Strategy for initial load plans*: Use global settings (from App Variables)

Show in all tables: ☑

Expected number of records:

Expected number of changes:

Business owners' roles*:

	Role
1	Ginger Group
*	

Business owners' of additional attributes roles*:

	Role
1	Fred Group
*	

Table Definition: General Tab

The **General** tab allows configuring a variety of general table attributes:

Name	Required	Description
Table name	Y	Technical name of the table used in the configuration process (parent keys setup, SQL validations, etc.) and written to the database. The maximum length of the name is determined by the database type. Database type is set in the App Variables node. Some names are reserved and cannot be used. See RDM Reserved Words and Keywords.
Table label	Y	Business name of the table as seen in the RDM web application.
Table description	N	Short description of the table to be displayed in the RDM web application by clicking the **Description** button.
Initial Load	N	If checked, initial load plans are generated for this table. See Configuring Initial Loads to RDM Tables for more information.
Initial load - history	N	If checked, initial data load plans with historical data are generated for this table. See Configuring Initial Loads to RDM Tables for more information.
Authentication Strategy for initial load plans	Y	Determines the authentication configuration for RDM steps in initial load plans. See RDM App Variables for information on authentication strategies.
Show in all tables	N	Determines whether the table will be shown in the table summary accessible by double-clicking the **Tables** node in the web application.

Name	Required	Description
Expected amount of records	N	Approximate number of records in the initial source tables (for documentation purposes only).
Expected amount of changes	N	Approximate number of records incoming into the system (for documentation purposes only).
Business owners' roles	N	Roles that will have editing rights for this table (use content assist to select roles) (for documentation purposes only).
Business owners of additional attributes roles	N	Roles that will have publishing rights for this table (use content assist to select roles) (for documentation purposes only).

Columns

The **Columns** tab allows the adding of columns to a table. Each column has the following parameters:

Name	Required	Description
Name	Y	Technical name of the table used in the configuration process and written to the database. The maximum length of the name is determined by the database type. Database type is set in the App Variables node.
Label	N	Business name of the column as seen in the web application.
Domain	Y	Name of the column's domain. See RDM Domains for information on defining domains.
Required	N	Determines whether the column must possess a value or may remain empty.

Name	Required	Description
Display mode	Y	Sets special display properties for the column in the web application, with the following options: • Normal - the column will not have any special properties. • Label - the column will have the following special properties. ◦ Values of this column will be displayed in combo-box options of a child table in the web application. See Lookups (combo-box lookup) in the RDM Web Application User Guide. ◦ Values of this column will be displayed instead of corresponding foreign keys if **Referenced data display mode** is set to *labels* in Columns setup in the web application. ◦ Values of this column will be displayed instead of corresponding foreign keys when viewing Hierarchies in the RDM web application.
Generated	N	Determines whether the values for the column are automatically generated on the side of the database. If checked, column value will be possible to enter only when creating a record; the column will be inactive when editing the record. In addition, if this column if entered in the RDM Tables tab, it will be inactive even when creating a record. Generated columns cannot be part of a unique key collection, because they are parsed as null before the import.

Name	Required	Description
Default value	N	Default value for the column, which will be pre-filled when creating a new record in the web application. Attributes of the datetime type should have their default value defined in one of the following formats: • yyyy-MM-dd • yyyy-MM-dd HH:mm:ss • yyyy-MM-dd'T'HH:mm:ss.SSS The following variables can be used: • `$username$` in a STRING column will be replaced by the active user (the user that creates the record) — there already is a technical column with username, but if you want to define a validation, approval workflow or filter on it, you need to place the username in a logical column (for example by using the `$username$` variable). • `$now$` in a DATETIME column will be replaced by current date and time. • `$today$` in a DATE column will be replaced by current date. Please note that data type mismatches will cause a validation error.
Value Presenter	N	Assigns one of previously defined value presenters to this column. See RDM Value Presenters.
Description	N	Free text describing the attribute; description will be visible upon hovering over the column name or in the table **Description** dialog.

Validations

The **Validations** tab allows the defining of various instant and web-service-based validations for the current table. The tab has the following sections.

Unique Keys

This section outlines defining unique keys.

A unique key is either one column whose values must be unique in each record or a set of columns whose value combinations should be unique across rows. You can create multiple unique keys. Where multiple unique keys exist, the first unique key to be configured is considered the primary key, unless an alternative primary key is explicitly set. Generated columns cannot be part of a unique key collection, because they are parsed as null before the import.

When creating or editing a record in the web application, records will be tested against this condition when trying to save the record or pressing **Validate**, and if the validation is not passed, a warning will be displayed. Users will not be able to publish this record.

Each unique key has the following attributes:

Name	Required	Description
Name	Y	Name of the unique key
Primary Key	N	Determines whether the unique key is the primary key for the table. Setting a primary key has several uses: 1. A primary key is required for matching records when importing data into the table. 2. A primary key must be defined for a parent table 3. This is the key connecting this table to a child table which has a column with a corresponding references-based domain. See How to Create an MN Relationship in RDM for more information.
Columns	Y	Select the columns composing the unique key to the **Name** column

> Only one unique key can be selected as a primary key.

Expression Validations

This section allows the defining of expressions for validating input data using the Ataccama ONE Desktop syntax. See Commonly Used Functions for the most common ONE Desktop functions. You can define multiple expression validations for each table.

For example, the expression below checks whether the `name` column contains only letters:

```
matches("[a-zA-Z]*",name)
```

When creating or editing a record in the web application, this validation will be performed when trying to save the record or pressing **Validate**, and if the validation is not passed, a warning will be displayed. The warning message will appear next to the column that the user has specified. If no column is specified, the validation message will be displayed as an error for the whole record in the validations panel. Users will not be able to publish this record.

Each expression validation has the following parameters:

Name	Required	Description
Expression	Y	An expression validating input.
Enable	N/A	Lets you easily enable and disable the validation without having to delete and rewrite the expression.
Message	N	The message that will be shown when the validation is violated.
Column	N	Specifies which column will receive a warning message when the validation is violated.

SQL Validations

This section allows defining complex validations that utilize SQL scripts for validations across columns, rows of the table for which it is defined and other tables.

For example, the SQL statement below checks whether there is only one branch manager per city in the `BRANCH` table.

```
select d.generatedPk, 'code' as name
from $BRANCH$ d
where d.CITY in (
    select distinct(CITY)
    from $BRANCH$ c
    GROUP BY CITY
    having count(distinct(BRANCH_MANAGER))>1
)
```

An SQL validation consists of the following parameters:

Field	Required	Description
SQL expression	Y	An SQL SELECT statement that validates a record being created or edited. See Query Guidelines below.
Enable	N/A	Lets you easily enable and disable the validation without having to delete and rewrite the expression.
Validation message	Y	A message shown when the validation is not passed.
Other involved tables	N	If the SQL expression uses other tables in addition to the table to which it is applied, all these table must be listed here.

Query Guidelines

- The SQL query should represent the violation of the condition which is tested, i.e., if the result of the query is not null, the record has not passed the validation.
- The query must always return the *generatedPk* attribute and a column which will display the validation messages in single quotes marked with the *name* alias.

```
select d.generatedPk, 'code' as name
```

- Table names must be wrapped into dollar signs, .e.g, $TABLENAME$ and aliased after the SQL FROM clause.

When the Validation is Performed

- Validation is performed on published and edited data of tables used in the validation.
- The SQL query will be executed every time:
  - The record is saved on creation and editing.
  - The record is validated (the **Validate** button is pressed in the record detail dialog).
  - The record is published.
  - The RDM web application is restarted (i.e., when a new configuration is deployed).

> To run full validation on records, you can schedule revalidation that runs at the next application restart. For more information, see RDM Features, section Record Validation on Application or Configuration Restart.

Online Validations

This section allows selecting previously defined online services for validating input. Online services are created from plans or components and are served by the ONE Runtime Server.

Each online validation has the following parameters:

Name	Required	Description
Name	Y	Name of the online service.
Enable	N/A	Lets you easily enable and disable the validation without having to delete and rewrite the expression.
Location	Y	URL location of the service in the following format: `http://[host]:[port]/[service_name]`.
Soap action	Y	Usually matches the name of the service and describes the SOAP action (services are typically of the SOAP type in ONE Desktop and RDM).
Namespace	Y	Namespace of the service.
Soap version	Y	SOAP version of the defined service (1.1 or 1.2).

Online Enrichers

This section allows selecting previously defined online services to enrich a record being created or edited with data. This feature is called via the **Enrich** button in the **Create/Edit Detail** dialog in the web application. Parameters configuring online enrichers are the same as for online validations.

ID Generators

The *ID Generators* tab allows assigning columns which will have automatically generated IDs; manual assigning or changing of such a column's value is not possible. These columns also need to have the **Generated** option checked (see Columns above).

Business Date Columns

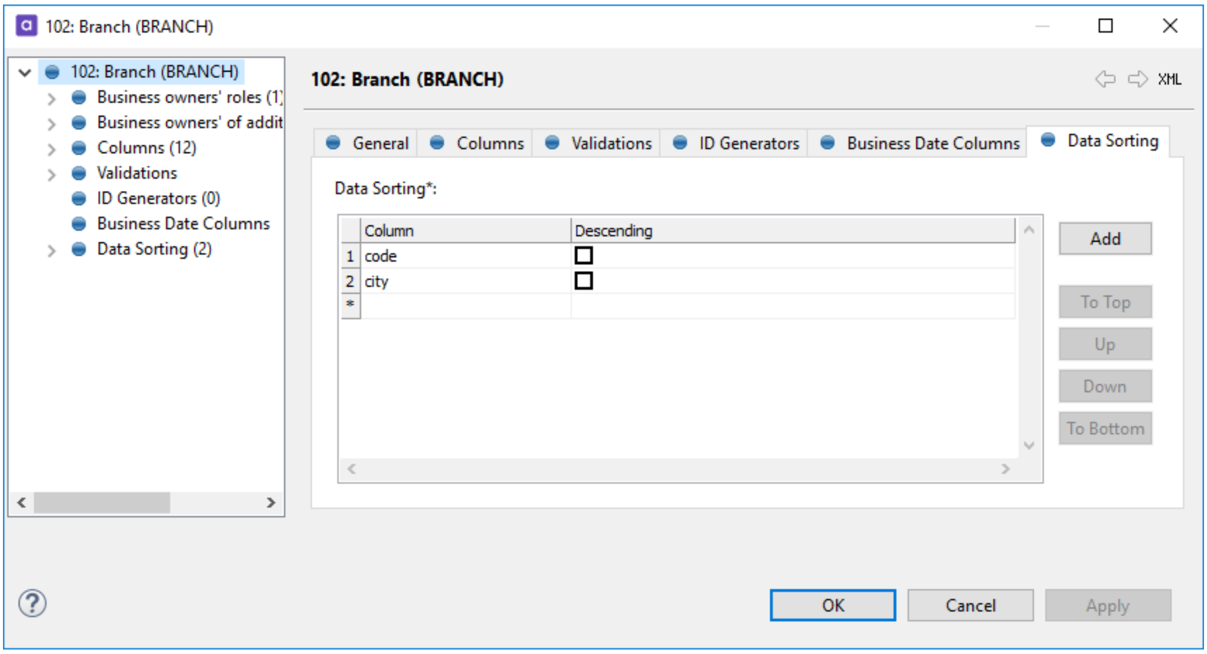
Reference Data Manager supports versioning of records, i.e., a possibility to create multiple versions of the same record with different validity periods.

The **Business Date Columns** allows specifying two columns that will mark the beginning (**Business date FROM** field) and the end (**Business date TO**) of record validity. These columns must have a datetime-based domain.

> Make sure you have specified **Infinity to** and **Infinity from** parameters in RDM App Variables.

Data Sorting

Here you can override RDM's default ordering or records by ID and specify the columns that you want the data to be sorted by. When specifying multiple columns, the records are sorted in the order the columns are listed, similarly to SQL ORDER BY. Checking the **Descending** box next to column results in the records being sorted in the descending order in that column.



Data Sorting*:

	Column	Descending
1	code	☐
2	city	☐
*		

RDM Value Presenters

Value presenters in RDM are used to modify the appearance and content of a column value. Typical examples are clickable URL links with custom link text and image URLs rendered as images. Value presenters consist of HTML code and column names used as value placeholders.

For example, you can make a column value `https://example.com` appear as Example (a clickable link with a custom link text).

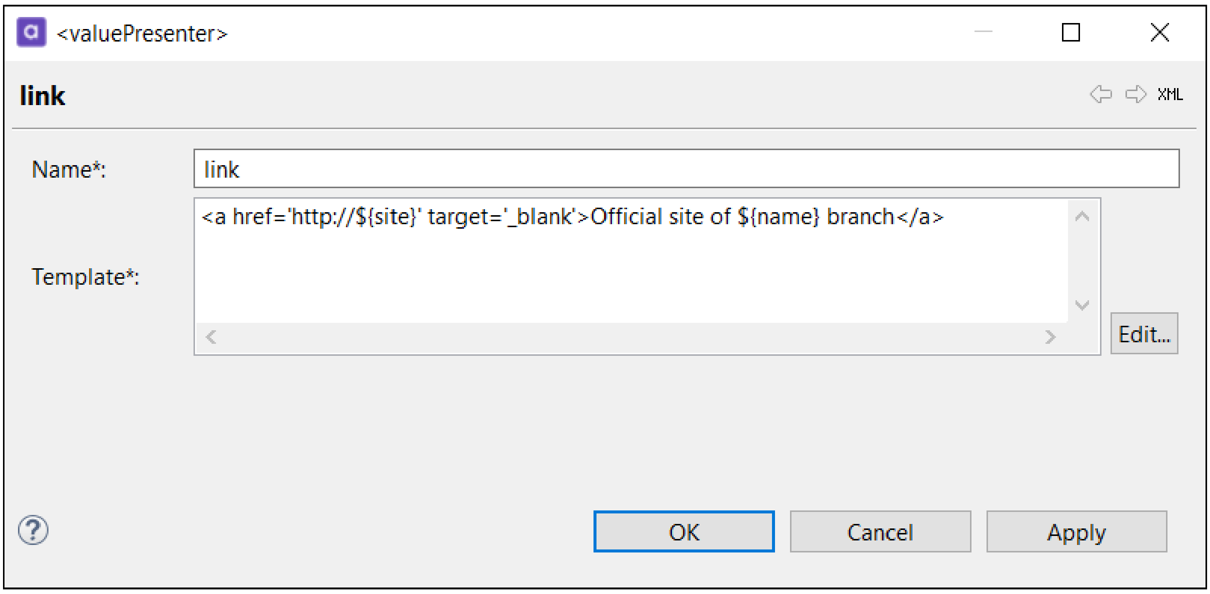
> Using this feature requires basic HTML knowledge.

> In EDIT mode, the raw value of the attribute is displayed. Value presenters are only applied in PUBLISHED and HISTORY modes.

Defining a Value Presenter

Before a value presenter can be assigned to a column to modify its value, it must be defined.

1. Open your RDM project.
2. Expand **RDM Logical Model > Value Presenters**.
3. Right-click **Value Presenters** and click **New value presenter**.
4. Fill in the name and define the template. See RDM Value Presenters for predefined templates.
5. Click **OK**.



Name*: link

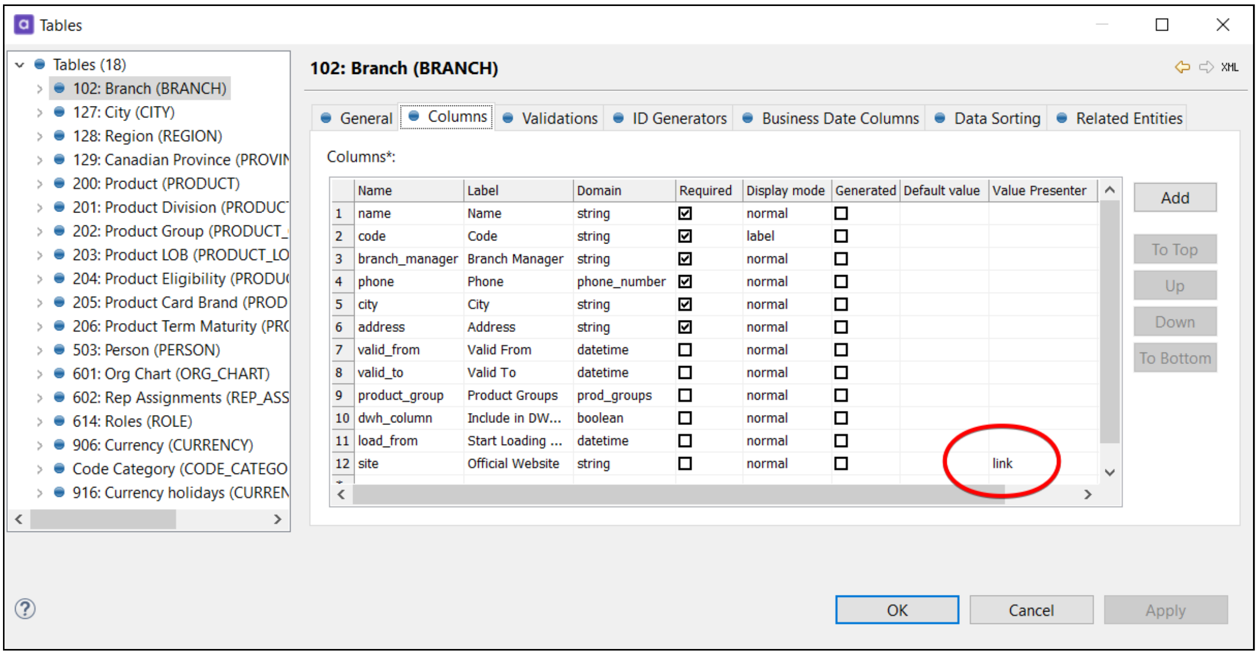
Template*: <a href='http://${site}' target='_blank'>Official site of ${name} branch</a>

Assigning a Value Presenter

Once a value presenter is created, it can be assigned to a column to modify the appearance of its values.

1. Open your RDM project.

2. Expand **RDM Logical Model > Tables.**
3. Double-click a table, in which you want to assign a value presenter.
4. Switch to the **Columns** tab.
5. Double-click the row number with the column, to which you want to assign a value presenter.
6. In the **Value Presenter** field, assign a previously created value presenter.


Tables
Tables (18)
102: Branch (BRANCH)
127: City (CITY)
128: Region (REGION)
129: Canadian Province (PROVIN
200: Product (PRODUCT)
201: Product Division (PRODUC
202: Product Group (PRODUCT_
203: Product LOB (PRODUCT_LO
204: Product Eligibility (PRODU
205: Product Card Brand (PROD
206: Product Term Maturity (PR
503: Person (PERSON)
601: Org Chart (ORG_CHART)
602: Rep Assignments (REP_ASS
614: Roles (ROLE)
906: Currency (CURRENCY)
Code Category (CODE_CATEGO
916: Currency holidays (CURREN
102: Branch (BRANCH)
XML
General
Columns
Validations
ID Generators
Business Date Columns
Data Sorting
Related Entities
Add
To Top
Up
Down
To Bottom
OK
Cancel
Apply

Columns*:

	Name	Label	Domain	Required	Display mode	Generated	Default value	Value Presenter
1	name	Name	string	☑	normal	☐		
2	code	Code	string	☑	label	☐		
3	branch_manager	Branch Manager	string	☑	normal	☐		
4	phone	Phone	phone_number	☑	normal	☐		
5	city	City	string	☑	normal	☐		
6	address	Address	string	☑	normal	☐		
7	valid_from	Valid From	datetime	☐	normal	☐		
8	valid_to	Valid To	datetime	☐	normal	☐		
9	product_group	Product Groups	prod_groups	☐	normal	☐		
10	dwh_column	Include in DW...	boolean	☐	normal	☐		
11	load_from	Start Loading ...	datetime	☐	normal	☐		
12	site	Official Website	string	☐	normal	☐		link

Template Examples

Below are several sample value presenter configurations.

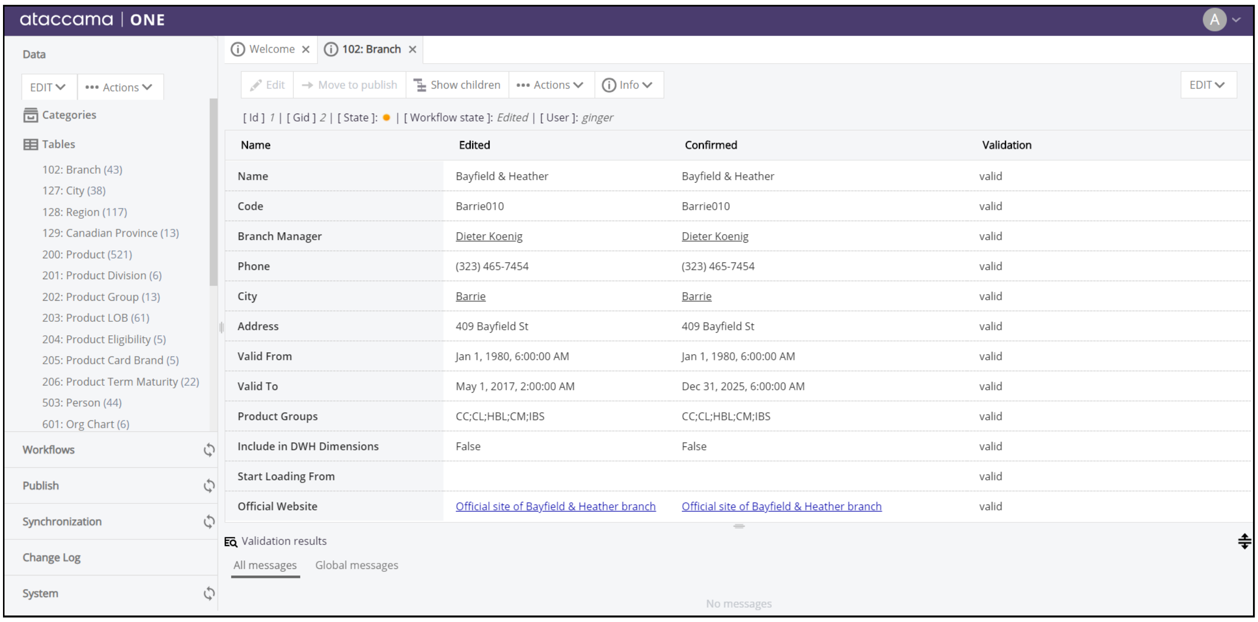
Clickable HTML Link

The following example demonstrates a clickable link with a link text customized for the given table:

```
<a href='http://${site}' target='_blank'>Official site of ${name} branch</a>
```

This template uses the value of the `Official Website` column as the link URL and the value of the `Name` column in the link text.

Here is how it looks in the web application:



[Id] *1* | [Gid] *2* | [State]: ● | [Workflow state]: *Edited* | [User]: *ginger*

Name	Edited	Confirmed	Validation
Name	Bayfield & Heather	Bayfield & Heather	valid
Code	Barrie010	Barrie010	valid
Branch Manager	Dieter Koenig	Dieter Koenig	valid
Phone	(323) 465-7454	(323) 465-7454	valid
City	Barrie	Barrie	valid
Address	409 Bayfield St	409 Bayfield St	valid
Valid From	Jan 1, 1980, 6:00:00 AM	Jan 1, 1980, 6:00:00 AM	valid
Valid To	May 1, 2017, 2:00:00 AM	Dec 31, 2025, 6:00:00 AM	valid
Product Groups	CC;CL;HBL;CM;IBS	CC;CL;HBL;CM;IBS	valid
Include in DWH Dimensions	False	False	valid
Start Loading From			valid
Official Website	Official site of Bayfield & Heather branch	Official site of Bayfield & Heather branch	valid

Email Address to a mailto Link

The following template turns a plain email address into a clickable mailto link:

```
<a href='mailto:${email}' target='_blank'>${email}</a>
```

URL Rendered as an Image

The following template displays an image instead of the underlying URL to that image.

```
<img src='${flag}' height='20' width='40'>
```

2.5.7 RDM Relationships

The **Relationships** sub-node in the **RDM Logical Model** node allows adding, modifying, and deleting parent-child relationships between tables of the RDM logical model.

In the Data Model Editor, the relationship is marked by an arrow from the child table to the parent table, with the relationship name box in between.

> Creating MN relationships is described in How to Create an MN Relationship in RDM.

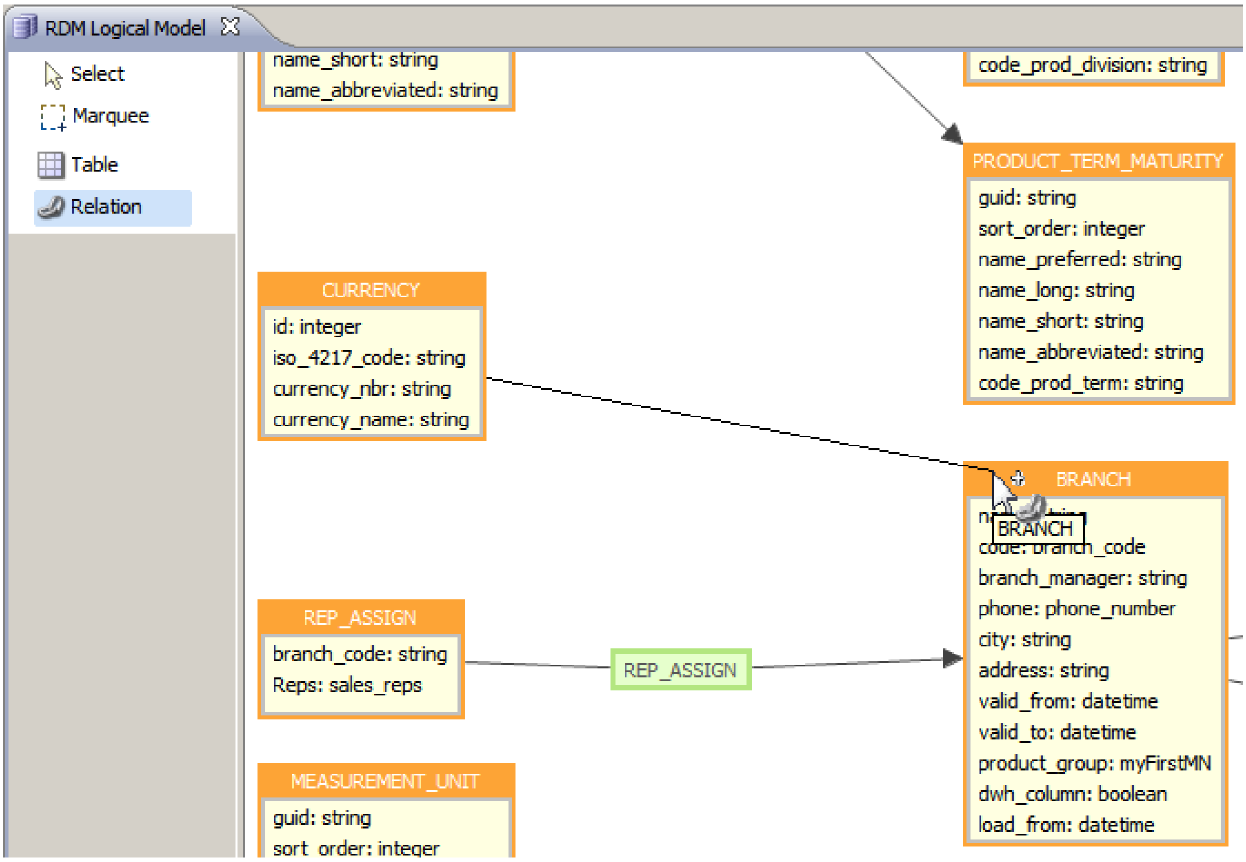
Adding a New Relationship in the Model Explorer

1. Right-click **Relationships > New relationship...**.
2. Fill in the attributes

3. Click **OK** to save changes

Adding a New Relationship in the Data Model Editor

1. Right-click **RDM Logical Model > Edit schema**. The canvas of the Data Model Editor will open, with the elements palette on the left.
2. Select **Relationship** from the palette and connect the child table with the parent table (the arrow should go from the child to the parent)
3. Double-click the newly created relationship
4. Fill in the attributes
5. Click **OK** and save changes



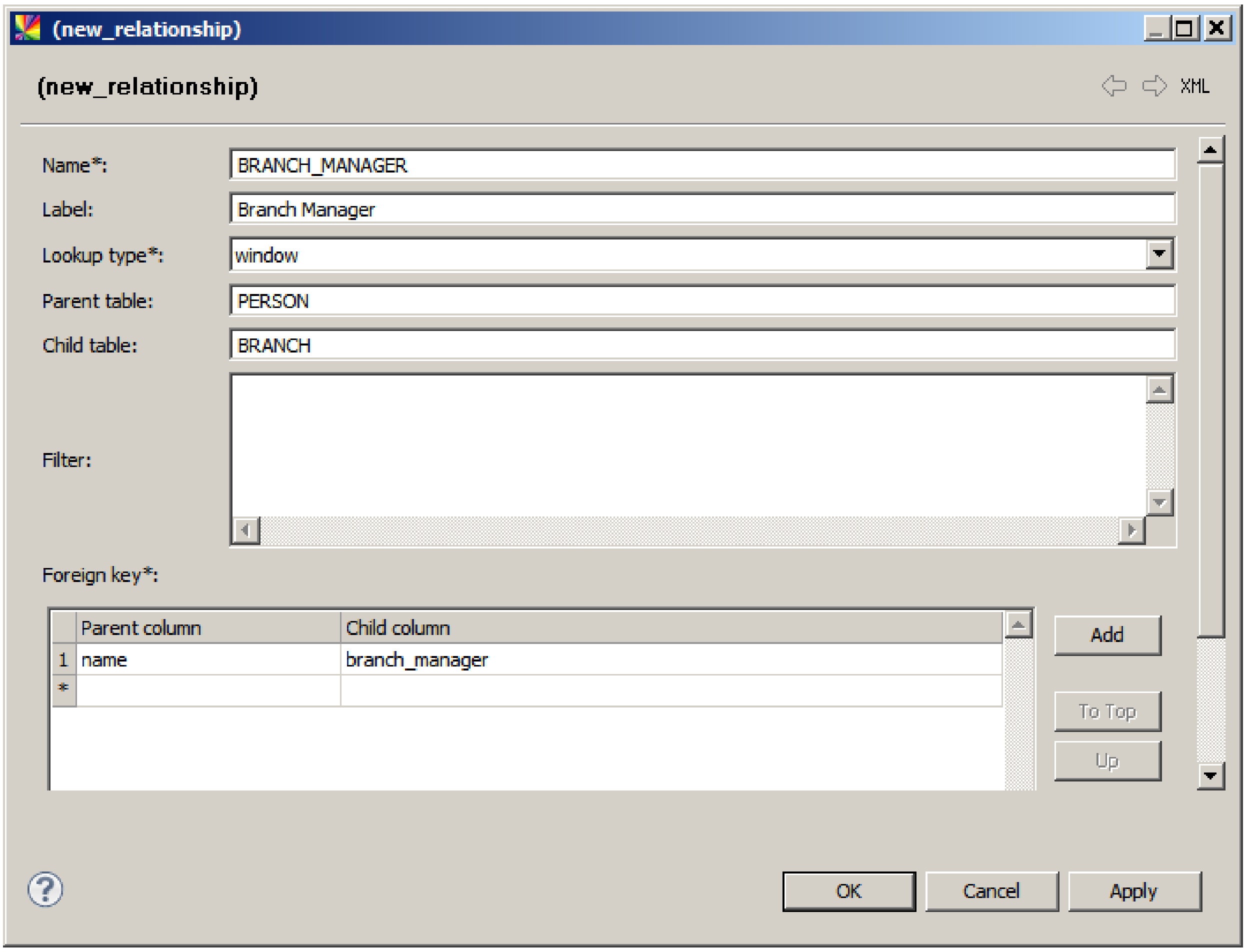
Adding a Relationship from the Data Model Designer

Relationship Attributes

Attribute	Required	Description
Name	Y	Name of the relationship used in the model configuration process. The maximum length of the name is determined by the database type. Database type is set in the App Variables node. Some names are reserved and cannot be used. See RDM Reserved Words and Keywords. In version 12.4.1 using numbers in the Relationship Name will cause an error warning to be shown, this can be disregarded and the model will function as intended. Dashes are not supported.
Label	N	Name of the relationship as seen in the web application.
Lookup type	Y	Determines the appearance of the lookup to the parent table in the **Create/Edit Detail** dialog. See Lookups in the RDM Web Application User Guide. Lookup types: • **combo** - displays a combo-box containing space-separated values of columns having the **Display mode** attribute set to *label*. See Columns. • **window** - opens a new dialog displaying all attributes of the parent table.
Parent table	Y	Name of the parent table (the relationship arrow leads here).

Attribute	Required	Description
Child table	Y	Name of the child table (the relationship arrow starts here).
Filter	N	SQL condition that makes only a subset of parent records satisfying this condition to be available for the child table.
Foreign key	Y	Defines the columns of the parent and child table that create the relationship. • **Parent column** - name of the column in the parent table. • **Child column** - name of the column in the child table.

The condition should contain the part of an SQL query following the `where` logical operator. The syntax generally depends on the database type used, with one important distinction: child table column names should be put in $, e.g., `$child_column_name$`.



(new_relationship)

Name*: BRANCH_MANAGER
Label: Branch Manager
Lookup type*: window
Parent table: PERSON
Child table: BRANCH
Filter:
Foreign key*:

	Parent column	Child column
1	name	branch_manager
*		

Relationship Definition

Multiple Relationships between Two Tables

While one child table can have several relationships to the same parent table, the connection must be via different child columns, i.e., you cannot connect both PARENT_COLUMN A to CHILD_COLUMN A and PARENT_COLUMN B to CHILD_COLUMN A.

Self References

Self-references are supported: just make the **Parent table** and **Child table** the same.

How to Create an MN Relationship in RDM

MN relationships in RDM are not defined in the **RDM Logical Model > Relationships** node. Instead, this type of a relationship is implemented via a MNREFERENCES domain type assigned to a child table.

To create an MN relationship:

1. Create a domain as described in RDM Domains with the **Type** attribute set to MNREFERENCES and **Foreign key table** attribute set to the parent table.
2. Optionally, set a **Validation message.**
3. Assign the created domain to the related column in the child table (**RDM Logical Model > Tables > [Table]** (double-click) **> Columns** tab).
4. Go the parent table, switch to the **Validations** tab.
5. In the **Unique Keys** section, add a new unique key with an arbitrary name and add a column from which the data will be read to the child table.
6. Check the **Primary key**.

Parental Values Encoding in RDM MN References

Overview

Parental keys need to be encoded into the children's binding column to preserve its hierarchical nature. A child can have multiple parents and each parent can be composed of multiple values (columns) in case of a compound key. To ensure proper encoding/decoding of the value, a 2-level encoding has to be used where in the first phase we encode individual key columns and in the second phase we encode/group individual parents. The CSV encoding is performed with the following setup:

```
string qualifier = "

string qualifier escape = "

field separator = ;

use string qualifier for all values = false
```

Which means that only values which contain the `"` or `;` characters will be escaped. This setup is hardcoded in RDM and cannot be changed (as the change of these values would require re-calculation of all parent-binding values in MNChild tables).

The simplest example - 1 parent with single-column key

`parent-value`	initial value

`parent-value`	1st encoding - encode columns of the key
`parent-value`	2nd encoding - encode parents (group of keys)

Simple value example - 1 parent with 2 parents with single-column key

`pkey1 pkey2`	initial value
`pkey1;pkey2`	1st encoding - encode columns of the key
`pkey1;pkey2`	2nd encoding - encode parents (group of keys)

Simple value example - 2 parents with 2-column keys

`pkey11 pkey12 ,pkey21 pkey22`	initial values
`pkey11;pkey12 pkey21;pkey22`	1st encoding - encode columns of the key
`"pkey11;pkey12";"pkey21;pkey22"`	2nd encoding - encode parents - as each parent has 2 keys, we must group parents together using quotes (technically because there's a `;` character in the parent value definition separating individual keys)

Escaping of quotes and separators

If a `"` or `;` character is contained in the key value, the encoding will get a bit more complicated as these characters enforce key value quoting and `"` character requires escaping to recognize value character from quoting character. In the following example, the related quote characters are marked by same color for better visual matching.

Quoted and escaped example - 1 parent with 1-column key

`aaa "abc"`	initial value
`"aaa ""abc"""`	1st encoding - encode keys (there might be multiple), quote inside must be doubled with external (yellow) quoting of the value
`"""aaa """"abc"""""""`	2nd encoding - encode parents (there might be multiple) - each inner quote (from previous step) must be doubled again (blue and yellow quotes) and the resulting value must be quoted again (red quotes)

Quoted and escaped example - 2 parents with 2-column keys

`p;"11" p"12" p"21" p"22"`	initial values
`"p;""11""";"p""12""";"p""21""";"p""22"""`	encoding keys

`"""p;""""11"""""";""p""""12""""""";"""p"""" 21"""""";""p""""22"""""""`	encoding parents

This additional quoting/doubling is needed because:

- if we were not using outer quoting we would have problems with separators (`;`)
- if we were not using inner quoting (doubling inner quotes) we would have problem with proper quotes decoding

Decoding stored values

If you want to decode encoded values back to the parental key values, you must use CSV decoding algorithm with the same setup as when encoding:

```
string qualifier = "

string qualifier escape = "

field separator = ;
```

Example (can be processed e.g. by DQC's TextFileReader):

input	`"""key """"1"""""";""key """"2""""""";"""key """"3"""""";""key """"4"""""""`
output	`"key ""1""";"key ""2""";"key ""3""";"key ""4"""`

You will get CSV encoded values of individual keys of individual parents. To get unencoded values of the keys, perform CSV decoding on that values once again.

Example

input	`"key ""1""";"key ""2""";"key ""3""";"key ""4"""`
output	`key "1"; key"2"; key "3"; key "4"`

2.5.8 RDM Reserved Words and Keywords

The following words are reserved by RDM and have a specific meaning to the application. For this reason, these words cannot be redefined and used to name objects in the logical model such as columns, tables, or indexes.

Reserved Column Names

AC#EDIT_STATE	AC_PNC

AC#ORDER	AC_START
AC#STATE	AC_STATE
AC#STATEMENTS	AC_STATEMENTS
AC_ERROR	AC_VALIDATION_NAME_RESULT
AC_EXPIRYDATE	AC_VALIDATION_RESULT
AC_FINISH	AC_VALIDATION_TYPE_RESULT
AC_FROM_HCN	D_FROM
AC_HCN	D_TO
AC_CHANGE_TYPE	GENERATEDGPK
AC_CHECKED	GENERATEDGPKE
AC_INC	GENERATEDPK
AC_ORDER	GENERATEDPKE

Reserved Table/View/Relationship/Index Names

CS
REP_ROLE
REP_ROLE_ENTITY
REP_ROLE_ENTITY_COLUMN
REP_ROLE_ENTITY_SYSTEM
REP_ROLE_USER
TS

USERNAME
[TABLENAME][E|H|I|Q|R|S|ST|T|U |V] *
[CONNECTED_SYSTEM_NAME][TABLE_NAME] **

* [TABLENAME] is a placeholder for an existing table name in the model. This means that the RDM Logical Model cannot contain a table named PURCHASES and a table named PURCHASESe (PURCHASESE and other suffix letters mentioned above) at the same time.

** [CONNECTED_SYSTEM_NAME] is a placeholder for an existing connected system name in the model. This means that the RDM project cannot contain a connected system named CRM + table named PURCHASES and a table named CRMPURCHASES at the same time.

Also, it is not possible to use common keywords related to a database (SELECT, INSERT, etc.) as a name for an RDM object.

2.6 Creating Additional Reference Data Views

RDM supports viewing data in alternative ways to that provided by tables, and the following artifacts can be created:

- RDM Views - provide users of the RDM web application with a logical subset of chosen tables. Views are composed of two or more tables joined into one record grouping.
- RDM Hierarchies - allow users of the web application viewing data based on parent-child relationships between tables.
- RDM Categories - group related tables, views, and hierarchies into categories and sub-categories.
- RDM Data Sets - custom reports formed by querying the RDM Repository with SQL.

2.6.1 RDM Views

Views provide users of the RDM web application with a logical subset of chosen tables. Views are composed of two or more tables joined into one record grouping. Views are defined in the **RDM Logical Model > Views** node.

Creating a View

1. Right-click **Views > New view.**
2. Fill in RDM Views.
3. Add and configure parent tables under the **Parent tables** section: see RDM Views.
4. Click **OK** to save changes.

General View Attributes

Views are configured via the following attributes:

Field	Required	Description
Name	Y	Technical name of the view used in model configuration, e.g., construction of RDM Hierarchies. Some names are reserved and cannot be used. See RDM Reserved Words and Keywords. The name must be unique across tables, views and data sets.
Label	Y	Name of the view as seen in the web application.
View description	N	Description of the view to be displayed in the web application by clicking the **Description** button.
Show in all tables	N	If checked, the view is shown in the table summary accessible by double-clicking the **Tables** node in the web application.
Base entity	Y	The child table, from which to start building the view (children of this table will not be available, only parents).

Field	Required	Description
Filter	N	Set conditions that restricts the count of rows from the child table to a subset of the table. Both basic and advanced methods are available. For basic conditions use the forms available in the RDM web app to set entities and operators. You can use the advanced filter for situations where the basic filter is not enough. It functions similarly to the WHERE SQL clause and uses SQL-like syntax. To get started: **Step 1:** Use Ctrl+Space or ⌘+Space to get a list of all columns. Column names are case-sensitive and must be in double quotes if they contain more than one word. **Step 2:** Next, set the filtering rules using operators and/or functions. Operators are not case-sensitive, but functions are. Operators appear in `purple` in the advanced condition box. The following are supported: • Operators: `LIKE`, `ESCAPE`, `FALSE`, `TRUE`, `NULL`, `CASE`, `IS`, `IN`, `=`, `<>`, `>`, `>=`, `<`, `<=`, `IS NOT NULL`, `ISNULL`, `NOTLIKE`, `LIKEESCAPE`, `NOTLIKEESCAPE`, `LIST`, `FUNC` • Logical operators: `AND`, `OR`, `NOT` • Functions: `upper`, `lower`, `length`, `concat`, `now`, `to_date`, `substring` **Step 3:** To define the criteria, state what value you are filtering by. Values must be in single quotes and are case-sensitive. They appear in `red` in the advanced condition box.

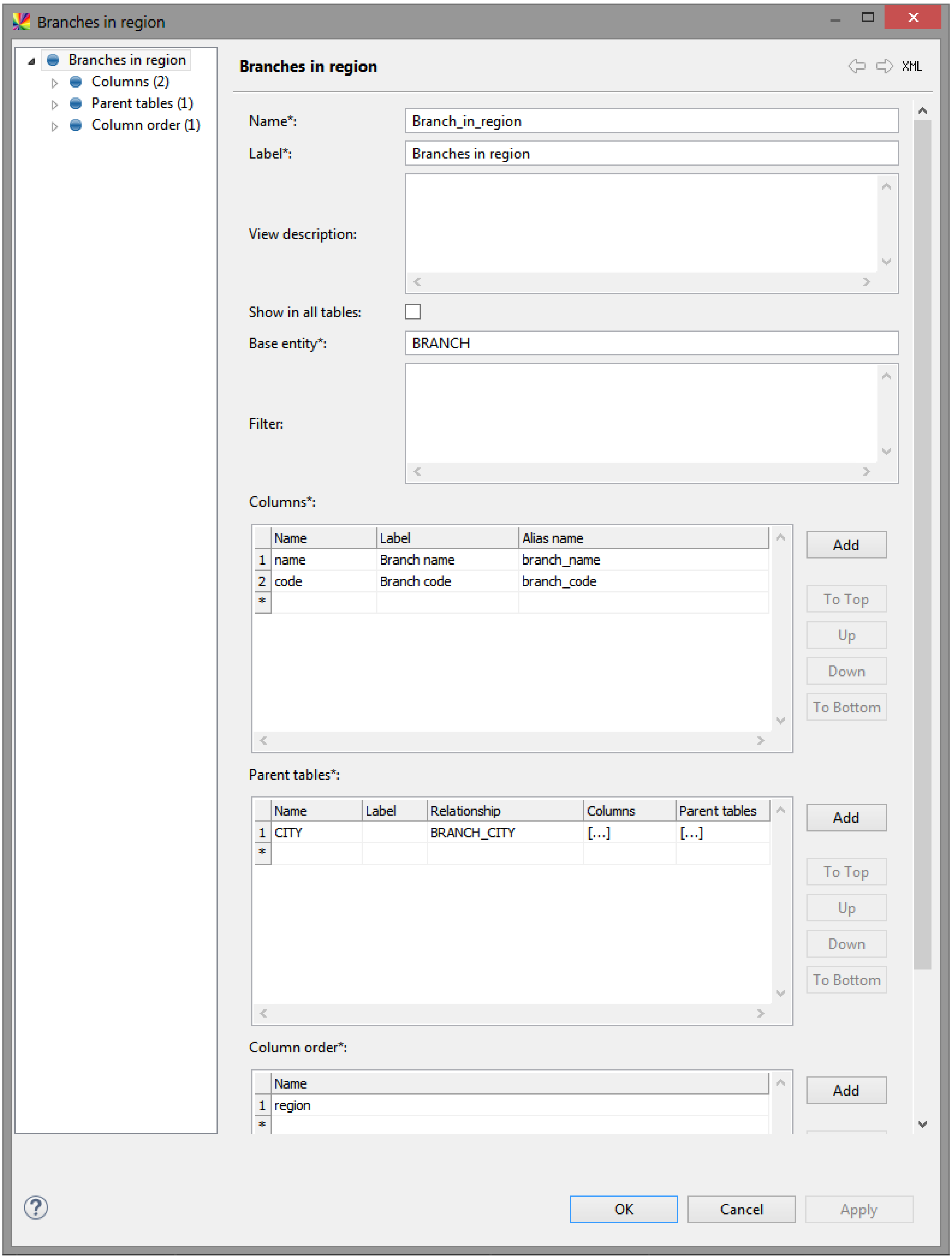
Field	Required	Description
		Native SQL is now supported in row permissions and nativeSql in filter configuration of views and relations.
Columns	Y	Allows selecting columns from the base entity, which will be used to construct the view: • **Name** - column name from the logical model used in the view. • **Label** - (optional) name of the column as seen in the web application. • **Alias** - technical name of the column written to the database; has to be unique across the whole view.
Parent tables	N	Allows selecting parent tables for constructing the view. See RDM Views.
Column order	N	Allows ordering columns in the view. **Alias** names must be used. Come back to this section after you have fully constructed the view: columns from joined parent tables will become available too.

Parent Table Attributes

Field	Required	Description
Name	Y	Technical name of a parent table used in model configuration.
Label	Y	Name of the parent table as seen in the web application by clicking the **Description** button.
Relationship	Y	Name of the relationship between the parent table and the base entity.

Field	Required	Description
Columns	Y	Allows selecting columns from the parent table, which will be used to construct the view: • **Name** - technical name of the column. • **Label** - (optional) name of the column as seen in the web application. • **Alias** - alias of the column as seen in the web application; has to be unique across the whole view.
Parent tables	N	Allows selecting parent tables of the next level.

View Example



Branches in region

Name*: Branch_in_region

Label*: Branches in region

View description:

Show in all tables: ☐

Base entity*: BRANCH

Filter:

Columns*:

	Name	Label	Alias name
1	name	Branch name	branch_name
2	code	Branch code	branch_code
*			

Parent tables*:

	Name	Label	Relationship	Columns	Parent tables
1	CITY		BRANCH_CITY	[...]	[...]
*					

Column order*:

	Name
1	region
*	

RDM View Definition

When referencing foreign key columns in views, you must add the parent table and select the column from there, or validation errors will be thrown.

2.6.2 RDM Hierarchies

Hierarchies allow users of the web application viewing data based on parent-child relationships between tables. See Hierarchies in the RDM Web Application User Guide to see how they work. Hierarchies are defined in the **RDM Logical Model > Hierarchies** node.

Creating a Hierarchy

1. Right-click **Hierarchies > New hierarchy.**
2. Fill in the attributes on all hierarchy levels.
3. Click **OK** to save changes.

Make sure that all tables used in the defined hierarchy have at least one column with **Display mode** set to **label**. See RDM Tables for instructions. Expanded tables will display values of these columns when you browse hierarchies in the RDM web application. If no column has display mode set to label, hierarchy tables will expand with empty values.

Hierarchy Attributes

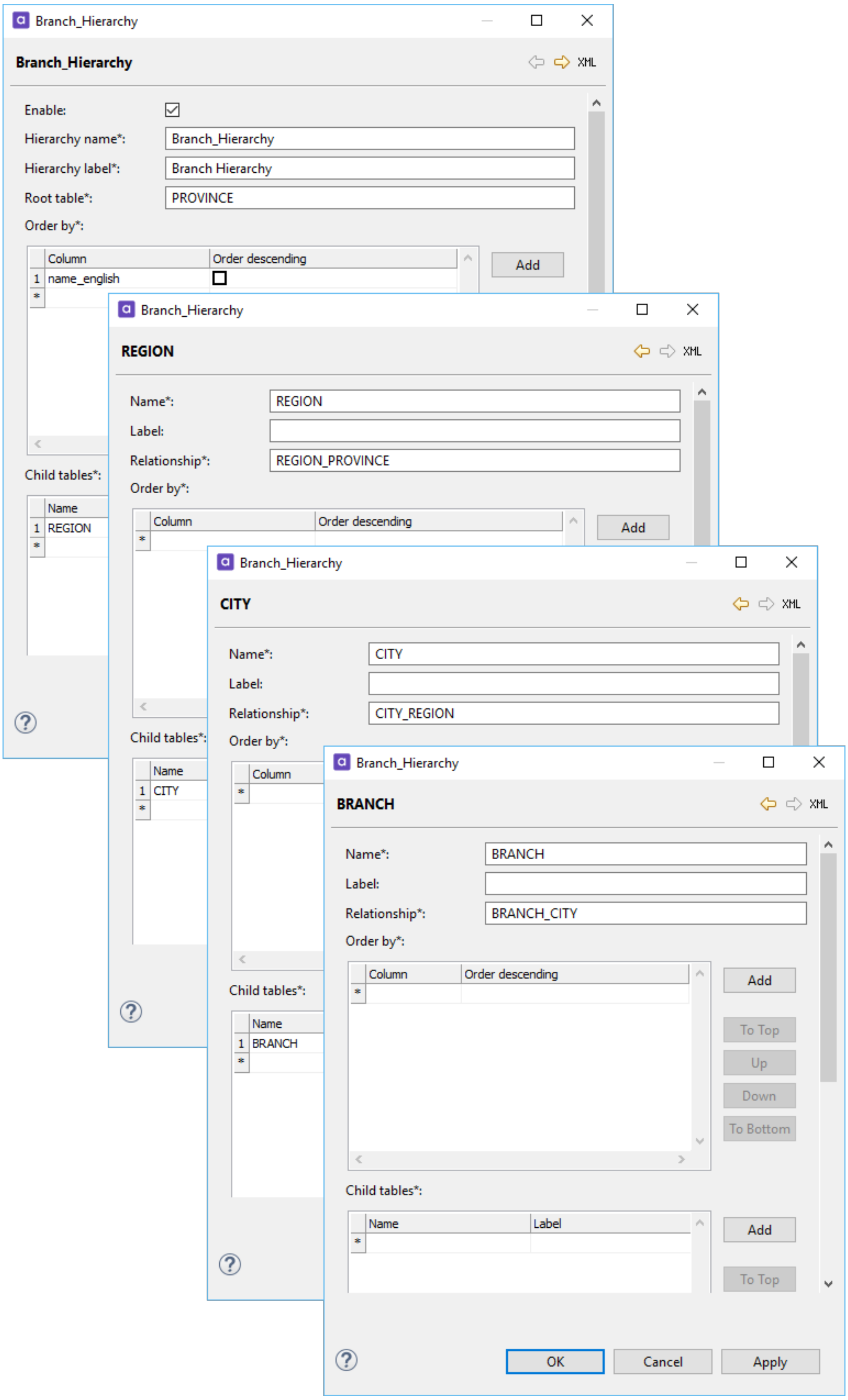
Hierarchies have the following attributes:

Field	Required	Description
Enable	N/A	Easily enables or disable the hierarchy without losing the configuration.
Hierarchy name	Y	Name of the hierarchy to be used in the configuration process.
Hierarchy label	Y	Name of the hierarchy shown in the web application.
Root table	Y	Name of the table that will be the top level node of the hierarchy.

Field	Required	Description
Order by	N	Values are ordered by the column specified here. If this column and the label column (see the note above) are the same, the displayed values will be sorted alphabetically.
Child tables	N	Allows adding child tables into the hierarchy: • **Name** - name of a child table • **Label** - (optional) name of the child table as seen in the web application • **Relationship** - name of the relationship between the parent and the child table • **Order by** - values are ordered based on the column specified here (works in the same way as in the root table). • **Child tables** - allows defining other levels of child tables
Hierarchy views	N	Allows adding an already defined view to the hierarchy. The view should be placed at a level of the hierarchy where the view base table corresponds to either a child table or the parent table on that hierarchy level.

Hierarchy Example

The image below shows all levels of a hierarchy, with the last level having no more child tables.



Branch_Hierarchy

Branch_Hierarchy

Enable: ☑

Hierarchy name*: Branch_Hierarchy

Hierarchy label*: Branch Hierarchy

Root table*: PROVINCE

Order by*:

	Column	Order descending
1	name_english	☐
*		

Child tables*:

	Name
1	REGION
*	

Branch_Hierarchy

REGION

Name*: REGION

Label:

Relationship*: REGION_PROVINCE

Order by*:

	Column	Order descending
*		

Child tables*:

	Name
1	CITY
*	

Branch_Hierarchy

CITY

Name*: CITY

Label:

Relationship*: CITY_REGION

Order by*:

	Column
*	

Child tables*:

	Name
1	BRANCH
*	

Branch_Hierarchy

BRANCH

Name*: BRANCH

Label:

Relationship*: BRANCH_CITY

Order by*:

	Column	Order descending
*		

Child tables*:

	Name	Label
*		

2.6.3 RDM Categories

Categories group related tables, views, and hierarchies. See Categories in the RDM Web Application User Guide to see how they work. Categories are defined in the **RDM Logical Model > Categories** node.

Adding a Category

1. Right-click **Categories > New category.**
2. Fill in the attributes for the main category and sub-categories
3. Click **OK** to save changes

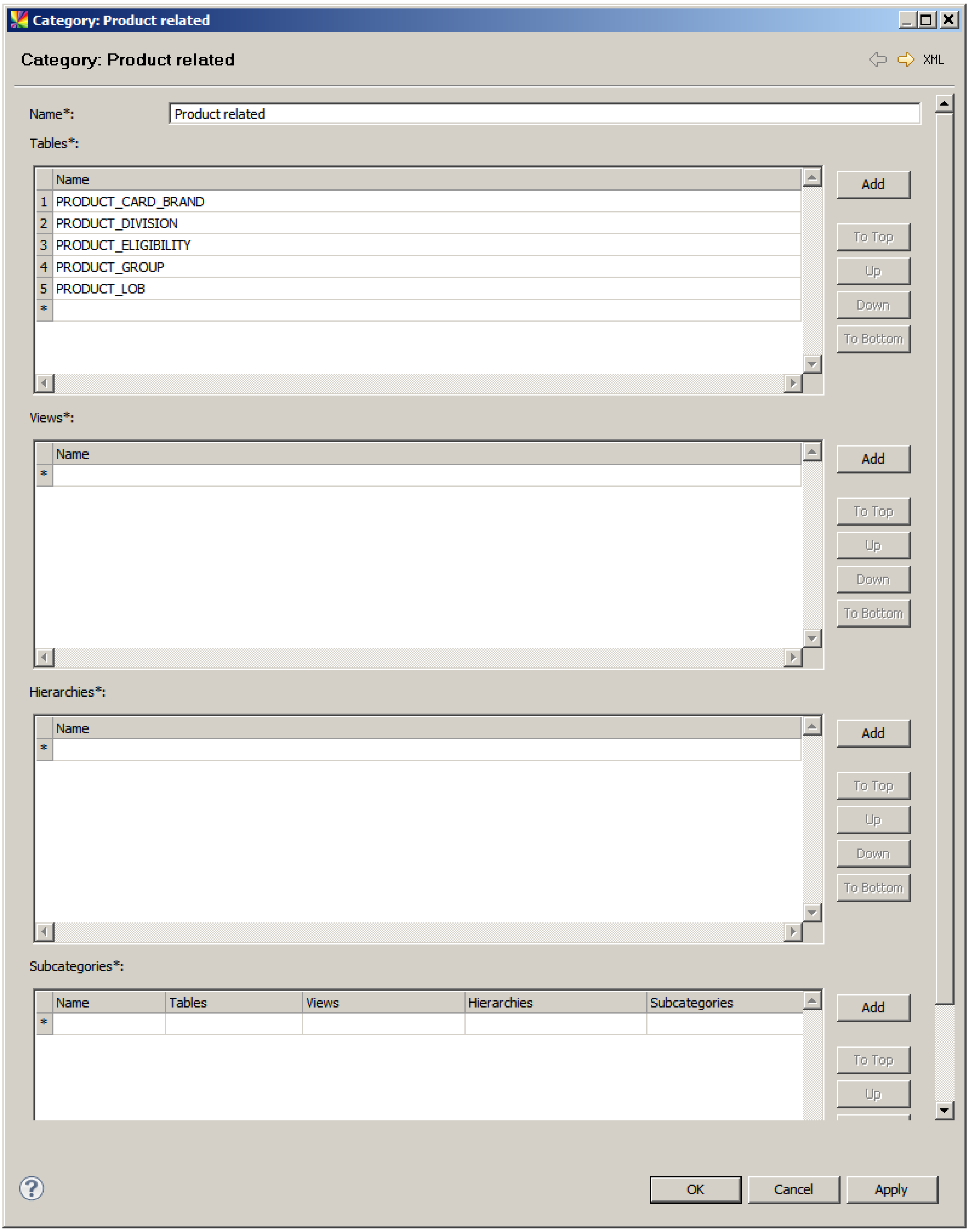
Category Attributes

Categories have the following attributes:

Field	Required	Description
Name	Y	Name of the category.
Tables	N	Allows adding previously defined tables to the category.
Views	N	Allows adding previously defined views to the category.
Hierarchies	N	Allows adding previously defined hierarchies to the category.
Subcategories	N	Allows creating a subcategory.

The same table, view, or hierarchy can be placed into several different categories.

Category Example



Category: Product related

Name*: Product related

Tables*:

	Name
1	PRODUCT_CARD_BRAND
2	PRODUCT_DIVISION
3	PRODUCT_ELIGIBILITY
4	PRODUCT_GROUP
5	PRODUCT_LOB
*	

Views*:

	Name
*	

Hierarchies*:

	Name
*	

Subcategories*:

	Name	Tables	Views	Hierarchies	Subcategories
*					

Category Definition

2.6.4 RDM Data Sets

Data sets are custom reports formed by querying the RDM Repository with SQL. Any SQL functions and clauses for selecting data can be used, like `AVG`, `COUNT`, `SUM`, `JOIN`, etc.

For example, a bank might be interested in the number of branches by region or number of products offered by each branch.

> After creating a data set and deploying the configuration in the web application, the administrator needs to set the permissions for it.

Creating a Data Set

To configure a data set:

1. Navigate to **RDM Logical Model > Data sets.**
2. Right-click **Data sets** and select **New Data set.**
3. In the **General** tab, fill in the attributes:

Field	Required	Description
Dataset name	Yes	Technical name of the data set written to the database. Some names are reserved and cannot be used. See RDM Reserved Words and Keywords. The name must be unique across tables, views and data sets.
Dataset label	Yes	Name of the data set as seen in the web application.
Description	No	Free text describing the data set.

4. Switch to the **SQL Query** tab and specify the SQL query that generates this data set.
5. Switch to the **Columns** tab and specify the attributes of the generated table (based on the SQL query):

Field	Required	Description
Name	Yes	Technical name of the column (or its alias) as written in the SQL query.
Label	Yes	Name of the data set as seen in the web application.
Type	Yes	Data type of the column.
Format	No	Value display format; works in the same way as for domains. See RDM Domains.

6. Switch to the **Order columns** tab and specify the attribute(s) by which the results are sorted.

SQL Query Syntax

While the general format of the query and SQL functions are used, there are a few requirements to the SQL queries used for generating data sets:

1. The resulting column names should conform to the naming of rules of the database used: the safest way is to use letters, numbers, and the underscore. This can be easily achieved by using the SQL Aliases: `as`.
2. Table names should be specified as follows: `$table_[actual_table_name]$`. For example, for table `CITY`, `$table_CITY$` should be used.

Sample SQL Query

```
select r.name as region, count(b.code) as branch_num
from $table_BRANCH$ b
    left join $table_CITY$ c on b.city=c.name
    left join $table_REGION$ r on  c.province=r.abbrev
group by r.name
```

2.7 Updating the RDM Data Model

After the RDM Logical Model (tables and relationships between them) is defined, deployed, and filled with data, most changes to it require adding an additional configuration file `alter-hints.xml` to the configuration ZIP file before deploying changes to the model (see How to Deploy an RDM Web App

Configuration). Alter hints are necessary to preserve permissions and data on existing objects or pre-fill data on a new column that must not be empty.

Using alter hints for making changes to the model and other articles related to updating the model are the focus of this chapter:

2.7.1 How to Use Alter Hints

After the RDM Logical Model (tables and relationships between them) is defined, deployed, and filled with data, most changes to it require adding an additional configuration file `alter-hints.xml` to the configuration ZIP file before deploying changes to the model (see How to Deploy a New Configuration).

Alter Hints Structure

`alter-hints.xml` has the following structure:

```
<alter-hints>
   <entities>
     <entity sourceName="tableA" targetName="tableB">
        <columns>
           <column targetName="columnB" expression="columnA" permissionsFrom="colu
mnA"
           />
        </columns>
     </entity>
   </entities>
</alter-hints>
```

Element	Attribute	Description
entity	sourceName	Old name of the table
entity	targetName	New name of the table
column	targetName	Name of the new column

Element	Attribute	Description
column	expression	Expression used to fill the data. The syntax and available functions depend on the type of database used as the RDM repository. You can use any database native functions which do not violate generic SQL syntax, otherwise RDM will throw an exception
column	permissionsFrom	Name of the column from which to inherit viewing and editing rights

Common Formatting Functions (PostgreSQL)

Function	Description	Example
`to_timestamp(value, format)`	convert string to datetime	`to_timestamp(columnA, 'YYYY-MM-DD HH:MI:SS')`
`to_date(value, format)`	convert string to date	`to_date(columnA, 'DD Mon YYYY')`
`to_char(value, format)`	convert numeric to string	`to_char(columnA, '999D99S')`
`to_number(value, format)`	convert string to numeric	`to_number(columnA, '99G999D9S')`

Integer to float data type change does not require using the `alter-hints.xml` file.

The following sections present practical examples (applicable for PostgreSQL database):

Adding a New String Column

The example below shows `alter-hints.xml` configuration for adding a new column of the string type, filling it with a constant value, and inheriting viewing and editing rights from a different column:

```
<alter-hints>
   <entities>
     <entity sourceName="tableA" targetName="tableA">
        <columns>
           <column
                   targetName="newStringColumn"
                   expression="'string enclosed in single quotes'"
                   permissionsFrom="anotherStringColumn"
           />
        </columns>
     </entity>
   </entities>
</alter-hints>
```

Adding a New Integer Column

The example below shows `alter-hints.xml` configuration for adding a new column of the integer-type domain, filling it with a constant value, and inheriting viewing and editing rights from a different column:

```
<alter-hints>
   <entities>
     <entity sourceName="tableA" targetName="tableA">
        <columns>
           <column
                   targetName="newIntegerColumn"
                   expression="0"
                   permissionsFrom="anotherIntegerColumn"
           />
        </columns>
     </entity>
   </entities>
</alter-hints>
```

Adding a New Date Column

The example below shows `alter-hints.xml` configuration for adding two new columns of the date-type domain, filling them with a constant value, and inheriting viewing and editing rights from a different column. This example can be used when transforming a table to using business dates:

```
<alter-hints>
   <entities>
     <entity sourceName="tableA" targetName="tableA">
        <columns>
           <column
                   targetName="VERSION_FROM"
                   expression="to_date('2015-01-01 00:00:00', 'YYYY-MM-DD
HH24:MI:SS')"
                   permissionsFrom="ID_tableA"
           />
           <column
                   targetName="VERSION_TO"
                   expression="to_date('2015-12-31 00:00:00', 'YYYY-MM-DD
HH24:MI:SS')"
                   permissionsFrom="ID_tableA"
           />
        </columns>
     </entity>
   </entities>
</alter-hints>
```

Changing Column Data Type

The example below shows `alter-hints.xml` configuration for changing a column data type from integer to string. In reality, a new column will be created in the RDM Repository; therefore, viewing and editing rights are configured to be inherited too:

```
<alter-hints>
   <entities>
     <entity sourceName="tableA" targetName="tableA">
        <columns>
           <column
                   targetName="columnA"
                   expression="to_char(columnA)"
                   permissionsFrom="columnA"
           />
        </columns>
     </entity>
   </entities>
</alter-hints>
```

Renaming a Column

The example below shows `alter-hints.xml` configuration for renaming a column. In reality, a new column will be created in the RDM Repository; therefore, viewing and editing rights are configured to be inherited too:

```
<alter-hints>
   <entities>
     <entity sourceName="tableA" targetName="tableA">
        <columns>
           <column
                   targetName="NewColumnA"
                   expression="OldColumnA"
                   permissionsFrom="OldColumnA"
           />
        </columns>
     </entity>
   </entities>
</alter-hints>
```

Renaming the Foreign Key Column

Renaming a foreign key column is necessary when changing the relationship name between tables: child tables contain foreign key columns to parent tables and are named with the following naming convention: **[prefix]_[relationship_name]**, where prefix is set to `GENERATEDPK`.

In version 9.0.3 and older, it was possible to set the prefix in **App Variables** node > **Generated PK name** parameter. If you have upgraded to a newer version of RDM and had a different prefix set in the older version, make sure to use it instead of `GENERATEDPK`.

You can find out the value of the prefix by opening `config.xml` in the **etc** folder of the project: it is stored in the `pkColumnName` attribute of the `model` node.

The example below shows `alter-hints.xml` configuration for renaming a foreign key column:

```
<alter-hints>
   <entities>
     <entity sourceName="tableA" targetName="tableA">
        <columns>
           <column
                   targetName="GENERATEDPK_NewRelationshipName"
                   expression="GENERATEDPK_OldRelationshipName"
           />
        </columns>
     </entity>
   </entities>
</alter-hints>
```

Renaming a Table

The example below shows `alter-hints.xml` configuration for renaming a table. A new table will be created in the database, therefore all table columns will inherit their own data and permissions automatically. For this reason, you should not define the `columns` element:

```
<alter-hints>
   <entities>
     <entity sourceName="tableA" targetName="tableB">
     </entity>
   </entities>
</alter-hints>
```

2.7.2 How to Add a New Column to an RDM Table

These instructions are intended for the case of adding a column to an existing table that already contains data.

Required Column

If the new column is supposed to have the *Required* attribute checked in the column definition, it cannot remain empty when deploying the new model. Such a column can be pre-filled with data using alter hints, with the following options:

- The column will be filled with a constant value.
- The column will be filled with values from a different column.
- The column will be filled with values from a different column processed by an SQL query.

See How to Fill a New Column with Data.

Not Required Column

If the new column is not required to contain data when deploying the new model and can be filled with data later, but the column is supposed to be required later on, the following procedure can be used:

1. Add a new column and specify all necessary attributes as described in Columns, setting the **Required** attribute to false (not checked).
2. Deploy the new configuration. See How to Deploy an RDM Web App Configuration.
3. Fill in the column with data in the RDM web application or by a ONE plan (see How to Fill a New Column with Data) and publish the changes.
4. Open the model and change the **Required** to true for the new column.

5. Deploy the new configuration again.

2.7.3 How to Fill a New Column with Data

When adding a new column to the RDM model, you can pre-fill it with data immediately while deploying the model or construct a ONE plan using the RDM Importer step to fill in the data later.

Using Alter Hints

This is done by creating an XML file `alter-hints.xml` and adding it to the configuration zip file before deploying a new configuration (see How to Deploy an RDM Web App Configuration). See How to Use Alter Hints for detailed explanation of the XML structure and other practical examples.

The code block below shows how to use `alter-hints.xml` for filling the new column with data in three different ways:

- Filling all records with a constant (string and integer example)
- Inheriting values from a different column
- Filling a column with an SQL expression

alter-hints.xml

```
<alter-hints>
   <entities>
     <entity sourceName="tableA" targetName="tableA">
        <columns>
           <!-- Filling all records with a constant -->
           <column targetName="newColumnStr" expression="'string enclosed in
single quotes'"/> <!-- string column example -->
           <column targetName="newColumnInt" expression="0"/> <!-- integer column
example -->

           <!-- Inheriting values from a different column -->
           <column targetName="newColumnB" expression="sourceColumn"
permissionsFrom="sourceColumn"/>

           <!-- Filling a column with an SQL expression -->
           <column targetName="newColumnC" expression="to_char(ColumnA) || '-' ||
ColumnB"/>
        </columns>
     </entity>
   </entities>
</alter-hints>
```

Using a ONE Plan

If you want to make a full import of data into your new column, you can easily do this using a ONE plan.

Step 1 Prepare Data

Prepare a data source (a file, database table, etc.) consisting of the primary key attribute and the new column.

> You can export data from RDM using **Bulk > Export** and then modify the output by adding a new column to it. See Importing and Exporting Data in the RDM Web Application User Guide for more information.

Step 2 Construct the Plan

Construct a ONE plan containing the RDM Importer step . See the plan in Configuring Tasks for Importing Data from Text Files for inspiration.

> The **Columns** tab of the Rdm Importer step must contain the table's unique key column(s), so that the importer is able to perform the incremental merge of data.

2.8 Configuring RDM Authorization

2.8.1 Overview

Keycloak is now the only Identity and Access Management tool available for the RDM web application. Furthermore, all user-role mapping must be carried out in Keycloak according to the instructions found here, and can no longer be done within the web application.

As Keycloak can simultaneously manage roles and users for web applications of multiple Ataccama products, roles in Keycloak are automatically mapped to a specific Ataccama web application using the role prefix defined for the application. For this reason, Keycloak roles for RDM must now have the prefix `RDM_` .

Roles without prefix are intended to be composite roles which comprise of prefixed roles (it describes what roles should apply in each module). For example `admin` in ONE Web Application is a composition of `MMM_admin` , `RDM_admin` and others. For more information on composite roles, see https://www.keycloak.org/docs/latest/server_admin/.

Keycloak Changes

For those already working with Keycloak for web application authorization, the main thing to note is the change in users that will be available by default with the version 13 builds:

RDM Default Users

- `RDM_user` (editing rights)
- `RDM_admin` (admin rights)

Due to configuration changes in Admin Console it no longer allows access to the role `RDM_admin`, but to the role defined in `ataccama.one.rdm.system-group-name` , which is where access to the RDM web application permissions tab is defined. This means the role defined in this property has access to both permissions tab and Admin Console. For access to the permissions tab only, use the property `ataccama.one.rdm.permissions-group-name`.

Otherwise, use the guide below to start working with Keycloak and RDM

Prerequisites

- Java 11 must be installed for successful Keycloak installation
- **Running** PostgreSQL

2.8.2 Installing and Configuring Keycloak

For ease of use, we recommend creating the following folder structure:

one20 > dep > keycloak

This structure is not mandatory, however this is the structure that will be assumed for the instructions below.

1. Download the **Keycloak Standalone Server Distribution** found here.
2. Extract the downloaded file to `C:\\one20\dep\keycloak`.

3. Navigate to `\one20\dep\keycloak\module\system\layers\keycloak\org` and create a new folder structure `postgresql\main`. This will allow you to switch from the default H2 database to postgres.
4. In the new `postgresql\main` folder:
    a. Download JDBC PostgreSQL driver version 42.2.14.
    b. Create a new file called `module.xml` with the following content:

```
<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.3" name="org.postgresql">

    <resources>
        <resource-root path="postgresql-42.2.14.jar"/>
    </resources>

    <dependencies>
        <module name="javax.api"/>
        <module name="javax.transaction.api"/>
    </dependencies>
</module>
```

> The `path` attribute in the `resource-root` node must match the name of the JDBC driver (`postgresql-42.2.14.jar`). The module name (`org.postgresql`) needs to correspond to your folder structure that you have created in previous steps (`\one20\dep\keycloak\modules\system\layers\keycloak\org\postgresql\main`)

5. Open `\one20\dep\keycloak\standalone\configuration\standalone.xml` and make the following changes:
    a. In `drivers`, declare the new PostgreSQL driver:

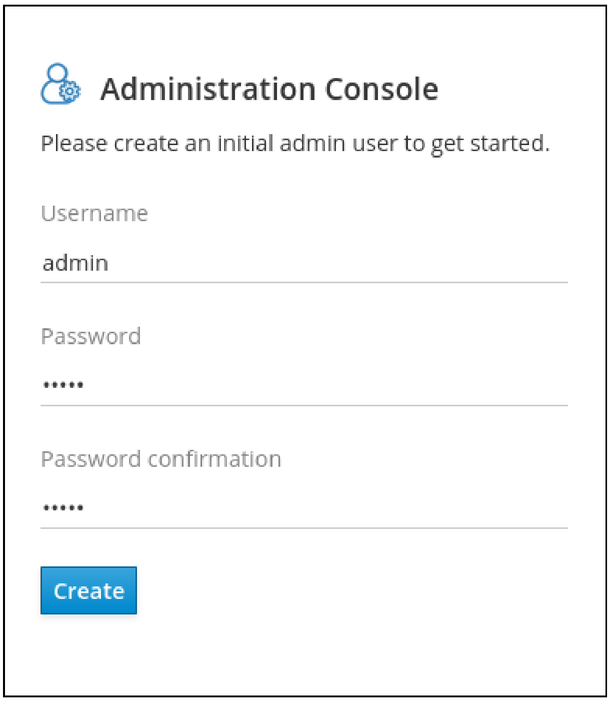
```
<drivers>
    <driver name="postgresql" module="org.postgresql">
        <xa-datasource-class>org.postgresql.xa.PGXADataSource</xa-
datasource-class>
    </driver>
    <driver name="h2" module="com.h2database.h2">
        <xa-datasource-class>org.h2.jdbcx.JdbcDataSource</xa-
datasource-class>
    </driver>
</drivers>
```

    b. In `datasource`, update the connection and security details for `KeycloakDS` datasource by replacing the existing datasource template with the following code:

```
<subsystem xmlns="urn:jboss:domain:datasources:5.0">
    <datasources>
      ...
        <datasource jndi-name="java:jboss/datasources/KeycloakDS" pool-name="KeycloakDS" enabled="true" use-java-context="true">
            <connection-url>jdbc:postgresql://localhost:5432/keycloak</connection-url>
            <driver>postgresql</driver>
            <security>
                <user-name>one</user-name>
                <password>one</password>
            </security>
        </datasource>
      ...
    </datasources>
</subsystem>
```

If you're not using the default keycloak configuration make sure to update the database connection string to point to the correct running postgre instance.

6. Navigate to `\one20\dep\keycloak\bin` and execute the `standalone.bat` file.
7. In your browser, go to http://localhost:8080/auth/.
8. Create an initial admin user with the credentials `admin/admin`.



Administration Console

Please create an initial admin user to get started.

Username: admin

Password: •••••

Password confirmation: •••••

You can also use a shell script for this step, as shown below:

a. Create a Keycloak admin user via `add-user-keycloak.bat` located in the `<keycloak>\bin` folder:

Command prompt

```
add-user-keycloak.bat -r master -u <username> -p <password>
```

- `username`, `password`: you can use the standard credentials `admin` / `admin`

b. Restart the Keycloak service to load the new user.

Log in to **Administration Console**.

9. Click **Add realm**.
    a. On the **Add realm** screen, click **Select file** and upload the following ataccamaone.json file.
    b. To finish importing, click **Create**.

Audit Module Clients and Roles

In case you would like to use the **Audit Module** you need to also import the **Audit Module** clients and roles (`audit-token-client`, `audit-webapp-public-client` and `AUDIT_admin`)

a. In your browser, go to **Keycloak** (http://localhost:8080/auth/).
b. Log in to **Administration Console**.
c. Select **Import** under **Manage** section.
d. Click **Select file**.
e. Select the JSON files for import: AUDIT_clients.json and AUDIT_admin_role.json
f. Click **Import**.

Keycloak Realm Settings

Make sure to set the **Require SSL** parameter in Keycloak according to your needs. Make sure to set the parameter properly especially in case your Keycloak installation has a public IP (e.g., is installed on a cloud):

a. Navigate to Keycloak admin console.
b. In the left menu, select **Realm Settings**.
c. In the **Login** tab, set the **Require SSL** parameter:
    - **External requests** (default): Keycloak requires SSL for interactions with external IP addresses. If you do not have SSL/HTTPS configured on the server or you try to access Keycloak over HTTP from a non-private IP address you will get an error.
    - **None:** unsecured connection. Keycloak does not require SSL. Should not be used in production environments.

10. Download Ataccama theme page template (ataccamaone-13.0.0-SNAPSHOT.zip).
11. Browse into the following location `\one20\dep\keycloak\themes`.
12. Extract the archive content into this folder (`one20\dep\keycloak\themes\ataccamaone`).
13. In you browser navigate to http://localhost:8080/auth/.
14. Log into **Administration Console**.
15. In **Ataccamaone** Realm navigate to **Themes**.
16. Select **ataccamaone** option in all drop down menus

If **ataccamaone** is not available as an option something has gone wrong with the installation.

17. Click **Save**.

Configuring Keycloak as a Service

1. Copy the `<keycloak>\docs\contrib\scripts\service` folder into `<keycloak>\bin`.
2. To configure Keycloak server startup, add the code below to `<keycloak>\bin\standalone.conf.bat` right before the`: JAVA_OPTS_SET` line:

\bin\standalone.conf.bat

```
rem # Ataccama's configuration
set "JAVA_OPTS=%JAVA_OPTS% -Dkeycloak.profile.feature.token_exchange=enabled
-Djboss.socket.binding.port-offset=3 -Djava.net.preferIPv4Stack=true
-Djboss.tx.node.id=atakeycloak"
set "NOPAUSE=true"
```

- `Djboss.socket.binding.port-offset`: use this option to change the offset for all ports.
    - For example, in the default configuration, Keycloak uses the following ports: HTTP:8080, HTTPS:8443, MNGMT_HTTP:9990, MNGMT_HTTPS:9993, AJP:8009. Setting the `port-offset=3` shifts all ports by 3. Hence, the actual ports used will be HTTP:8083, HTTPS:8446, MNGMT_HTTP:9993, MNGMT_HTTPS:9996, and AJP:8012.
- `Djboss.tx.node.id`: fill in with a unique value for the server

3. Run the following command as an administrator:

Command prompt

```
<keycloak>\bin\service\service.bat install
```

4. The service is now available under Windows Services with the Wildfly name.

If the service takes too long to restart or fails to start/stop, kill the `java.exe` process and restart Java.

2.8.3 Configuring Keycloak Clients

Configure clients under the **Ataccamaone** realm. Ataccama applications use different clients; for the RDM web application you will need to configure the following clients:

- rdm-admin-client
- rdm-token-client
- rdm-webapp-public-client

To see clients in Keycloak:

1. Sign in as an admin
2. Open the Keycloak administration console.
3. Make sure **Ataccamaone** realm is selected at the top of the left navigation bar.

4. From the left navigation bar, click **Clients**.
5. From the list of clients, click the client ID to open its configuration page.
6. Depending on the client's **Access Type** setting, different configuration fields are available and filled in.
7. Edit all filled-in fields that contain URLs, e.g., **Valid Redirect URIs, Base URL,** and **Admin URL.** Change the http://localhost:<port> port to the actual client location.
8. Click **Save.**

Complete steps 3-7 for all relevant RDM clients.

Edit Client Configuration Files

To define the configuration for your Keycloak clients, add or edit the `KeycloakDeploymentContributor` element in the runtime configuration file. The settings in the `KeycloakDeploymentContributor` should correspond to the Keycloak settings for client, which are defined in `application.properties`.

Keycloak is case-sensitive. Make sure to use lowercase if referring to the Keycloak server URL via hostname.

See Encrypting Passwords for information on how to encrypt passwords.

Name	**Mandatory**	**Description**
`ataccama.authentication.keycloak.server-url`	Yes	Keycloak server URL. Ending with /auth.
`ataccama.authentication.keycloak.realm`	Yes	Keycloak realm.
`ataccama.authentication.keycloak.admin.client-id`	Yes	Administration Keycloak client ID.
`ataccama.authentication.keycloak.admin.secret`	Yes	Administration Keycloak client secret.

Name	Mandatory	Description
`ataccama.authentication.keycloak.token.client-id`	Yes	Token Keycloak client ID.
`ataccama.authentication.keycloak.token.secret`	Yes	Token Keycloak client secret.
`ataccama.authentication.keycloak.token.issuer`	Yes	Token Keycloak issuer.
`ataccama.authentication.keycloak.public.client-id`	Yes	Keycloak public client ID for web application browsing.
`ataccama.client.connection.keycloak.http.enabled`	Yes	*Security header settings (see below)*
`ataccama.client.connection.keycloak.http.tls.enabled`	Yes	*Security header settings (see below)*

2.8.4 Web Application Security

You can configure RDM webapp security by adding response headers (security headers) to HTTP responses from the web application.

> We recommend setting security headers in case your web application is exposed to potential security attacks.

The security headers are configured in `application.properties`:

Name	Default value
`one.security.header.content-security-policy.connect-src`	'self' ${ataccama.authentication.keycloak.server-url}
`one.security.header.content-security-policy.script-src`	* 'unsafe-inline' 'unsafe-eval'
`one.security.header.content-security-policy.img-src`	'self' data:

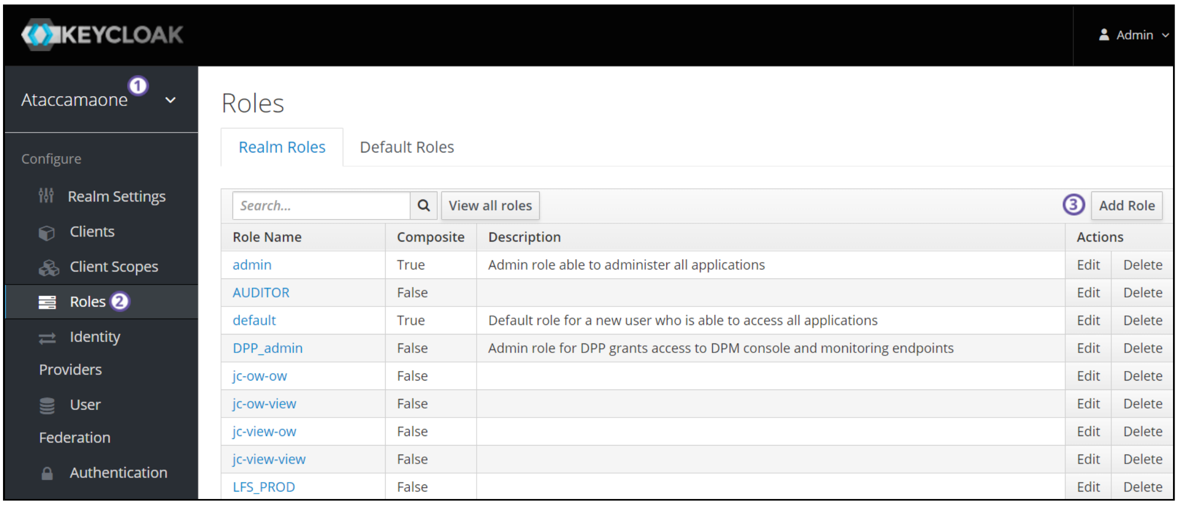
They are commented by default: uncomment to enable.

2.8.5 Mapping Roles and Users

Mapping of users and roles must be done in Keycloak. To do this, go to http://localhost:8080/auth/ and follow the steps below:

Creating Roles

1. Make sure Ataccamaone realm is selected at the top of the left navigation bar.
2. From the left navigation bar, click **Roles**.
3. In the **Realm Roles** tab, click **Add Role**:

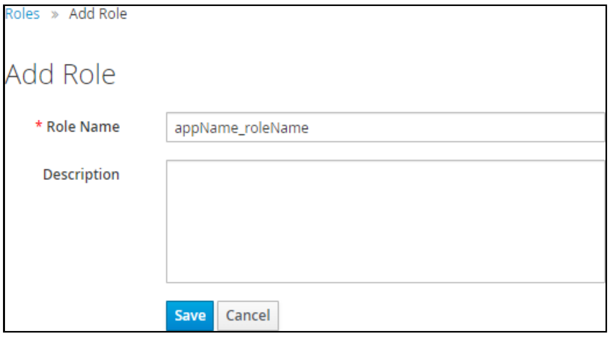


Role Name	Composite	Description	Actions	
admin	True	Admin role able to administer all applications	Edit	Delete
AUDITOR	False		Edit	Delete
default	True	Default role for a new user who is able to access all applications	Edit	Delete
DPP_admin	False	Admin role for DPP grants access to DPM console and monitoring endpoints	Edit	Delete
jc-ow-ow	False		Edit	Delete
jc-ow-view	False		Edit	Delete
jc-view-ow	False		Edit	Delete
jc-view-view	False		Edit	Delete
LFS_PROD	False		Edit	Delete

4. Fill in **Role Name** and **Description.**

> Once a role is created in Keycloak, it is not possible to rename it.

5. Click **Save.**



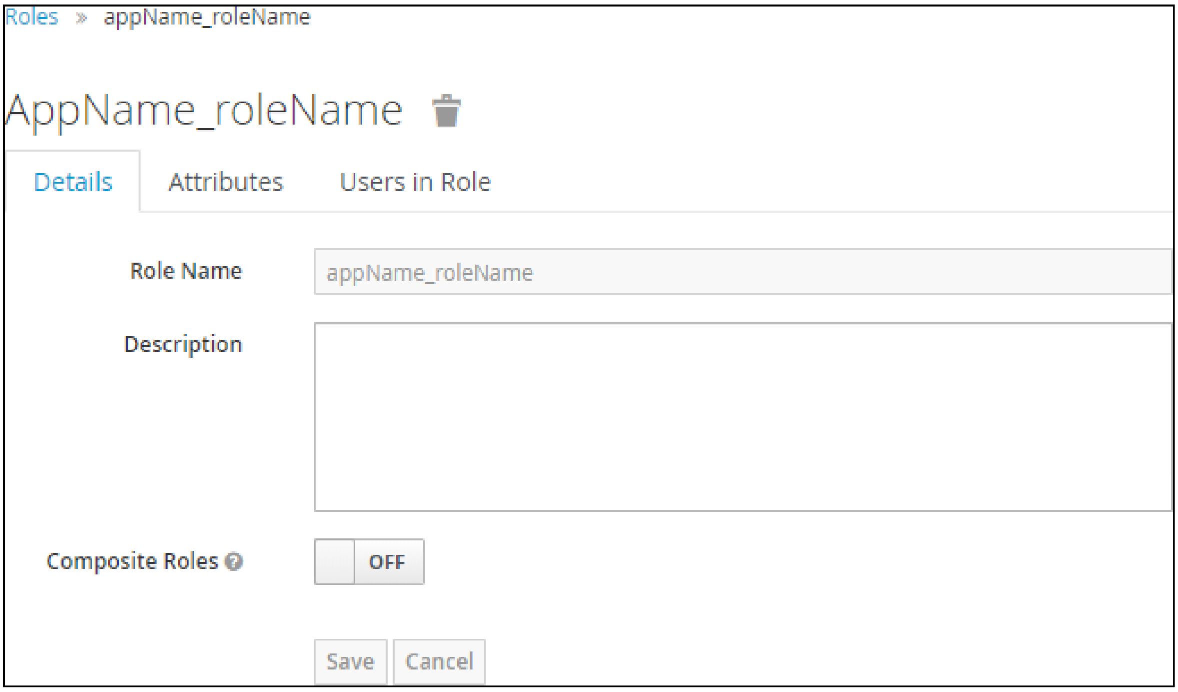
* Role Name: appName_roleName

Description:

6. The role appears in the roles list.

Editing Roles

1. Navigate to the **Roles** screen > **Realm Roles** tab.
2. From the list of roles, click a role name to open its configuration page. There you can view and edit role **Details, Attributes** and view list of **Users in Role**.



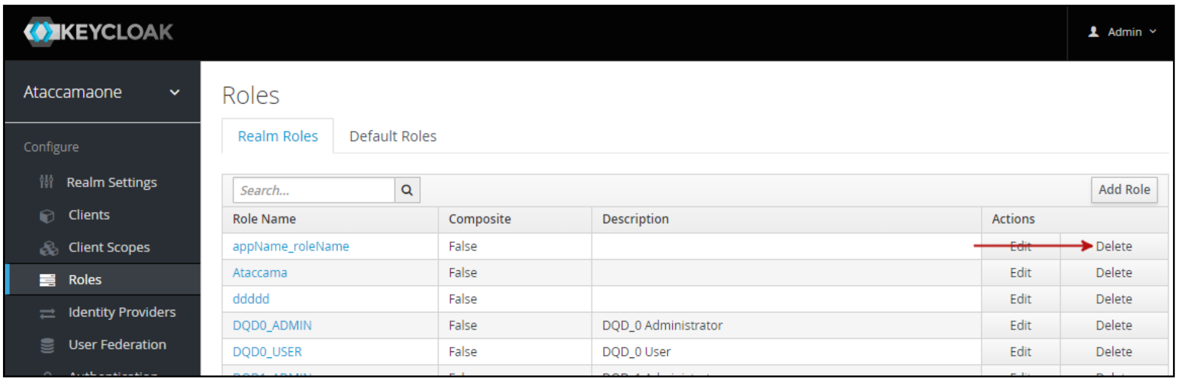
Role Name: appName_roleName

Description:

Composite Roles: OFF

Removing Roles

1. Navigate to the **Roles** screen > **Realm Roles** tab.
2. Click **Delete** for the role and confirm.



Role Name	Composite	Description	Actions	
appName_roleName	False		Edit	Delete
Ataccama	False		Edit	Delete
ddddd	False		Edit	Delete
DQD0_ADMIN	False	DQD_0 Administrator	Edit	Delete
DQD0_USER	False	DQD_0 User	Edit	Delete

Role Names Prefixes

As Keycloak can simultaneously manage roles and users for web applications of multiple Ataccama products, roles in Keycloak are automatically mapped to a specific Ataccama web application using the role prefix defined for the application. By default, the permission settings in RDM Web Application will only look at roles with the prefix `RDM_` .

Roles without prefix are intended to be composite roles which comprise of prefixed roles (it describes what roles should apply in each module). For example `admin` is a composition of `MMM_admin` , `RDM_admin` and others. For more information on composite roles, see https://www.keycloak.org/docs/latest/server_admin/.

Creating Role Names with Prefixes

The role name in Keycloak is created as <rolePrefix>_<roleName>, where:

- rolePrefix: a role name prefix defined in the `appName` element in the web application configuration file

  > Make sure to start the role name with the `<appName>_` prefix. Otherwise, the application will not recognize the role.

- roleName: a role name defined in the web application.

  > Use kebab-case (all lowercase with dashes as separators of words), no diacritics.

Managing Users

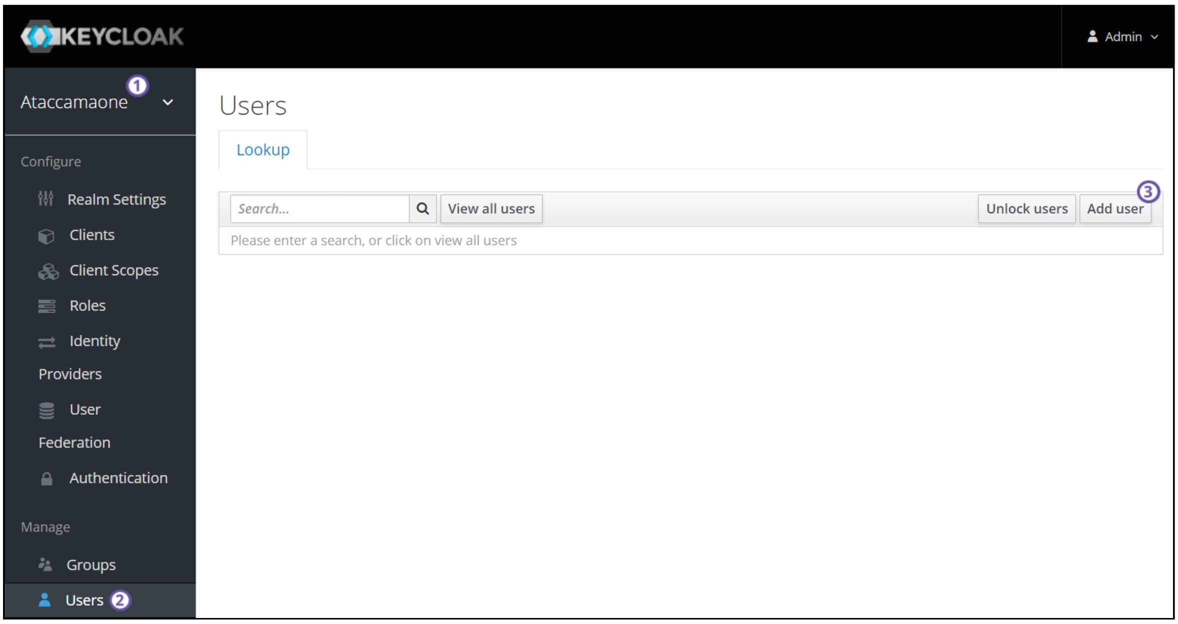
> Read the official Keycloak documentation for more information and general use guidelines: https://www.keycloak.org/docs/4.6/server_admin/index.html#user-management.

Creating Users

To create users in Keycloak:

1. Make sure Ataccamaone realm is selected at the top of the left navigation bar.
2. Navigate to the **Users** screen.

3. Click **Add User.**

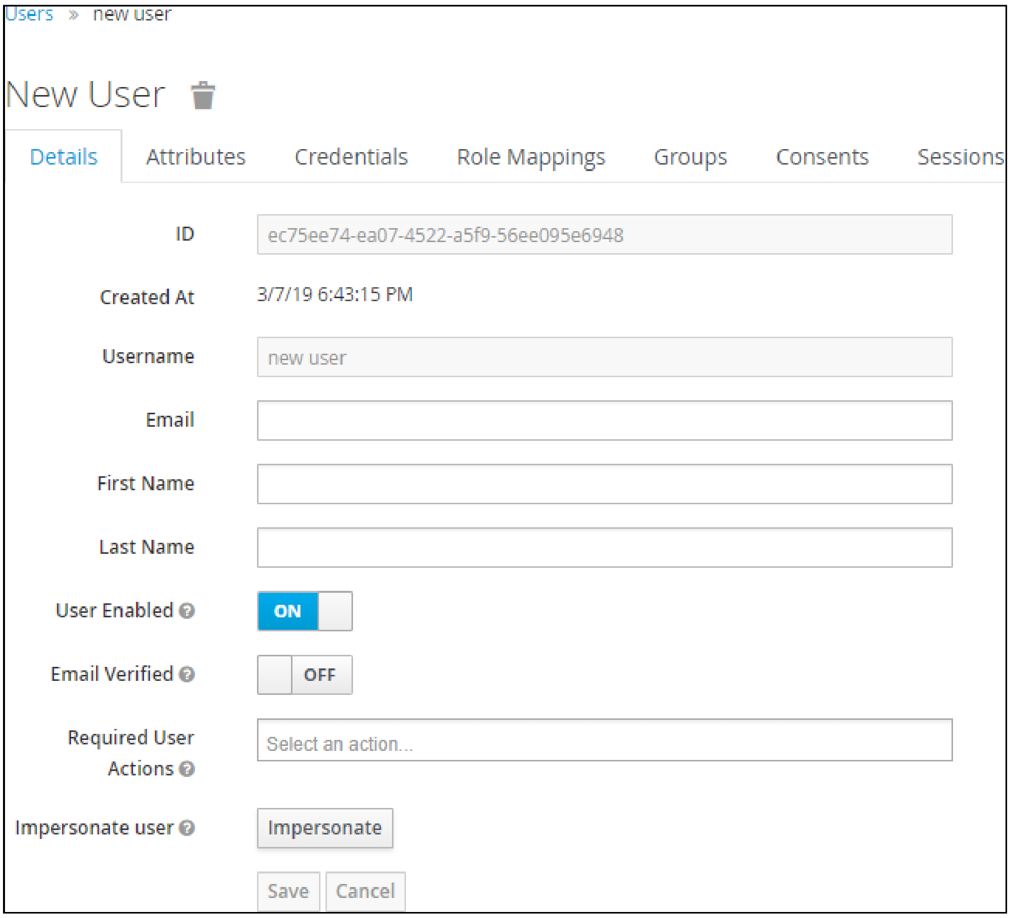


Please enter a search, or click on view all users

4. Fill in user details.
5. Click **Save.**
6. The user appears in the **Users** list.

Editing Users

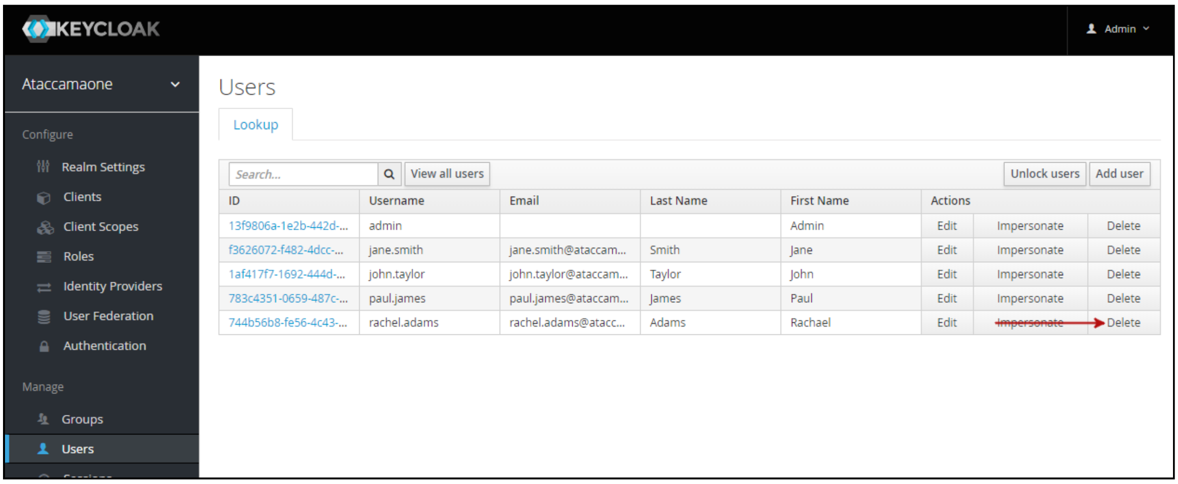
1. Navigate to the **Users** screen.
2. From the list of users, click user ID to open its configuration page. There you can view and edit user **Details, Attributes, Credentials** and view its **Groups** and **Role Mappings**.



ID: ec75ee74-ea07-4522-a5f9-56ee095e6948
Created At: 3/7/19 6:43:15 PM
Username: new user
Email:
First Name:
Last Name:
User Enabled: ON
Email Verified: OFF
Required User Actions: Select an action...
Impersonate user: Impersonate

Removing Users

1. Navigate to the **Users** screen.
2. Click **Delete** for the user and confirm.

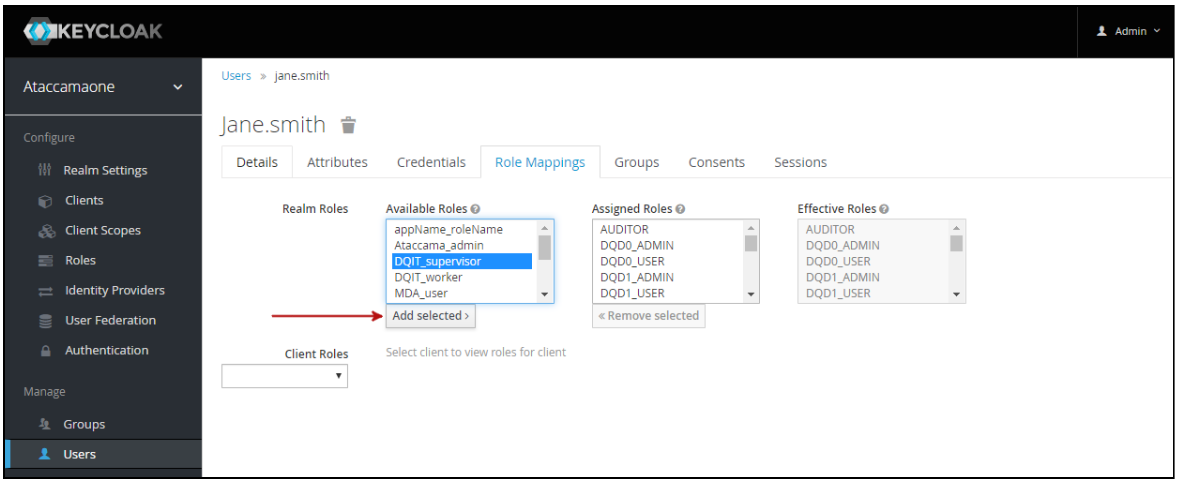


ID	Username	Email	Last Name	First Name	Actions		
13f9806a-1e2b-442d-...	admin			Admin	Edit	Impersonate	Delete
f3626072-f482-4dcc-...	jane.smith	jane.smith@ataccam...	Smith	Jane	Edit	Impersonate	Delete
1af417f7-1692-444d-...	john.taylor	john.taylor@ataccam...	Taylor	John	Edit	Impersonate	Delete
783c4351-0659-487c-...	paul.james	paul.james@ataccam...	James	Paul	Edit	Impersonate	Delete
744b56b8-fe56-4c43-...	rachel.adams	rachel.adams@atacc...	Adams	Rachael	Edit	Impersonate	Delete

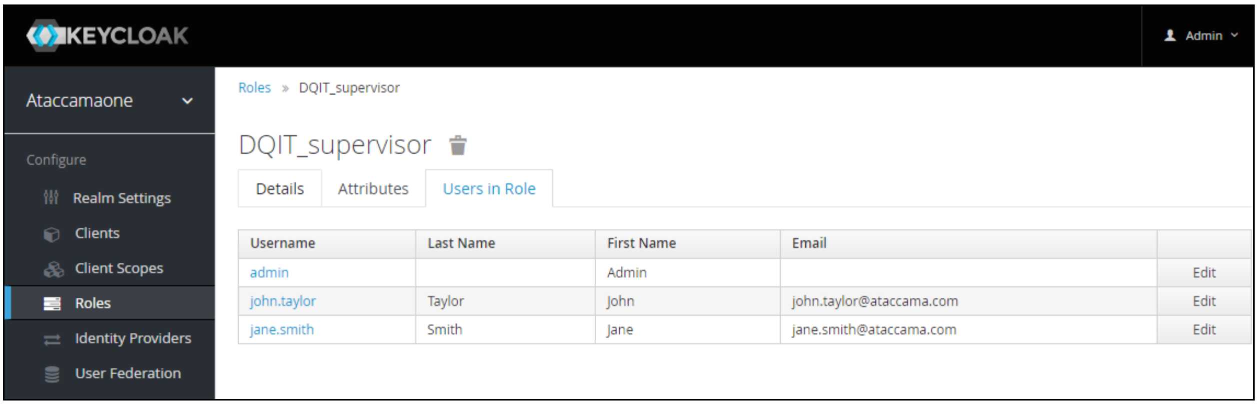
Mapping Roles to Users

To map roles to users in Keycloak:

1. Navigate to the **Users** screen
2. From the list of users, click user ID to open its configuration page.
3. In the **Role Mappings** tab, select from **Available Roles** and click **Add selected**

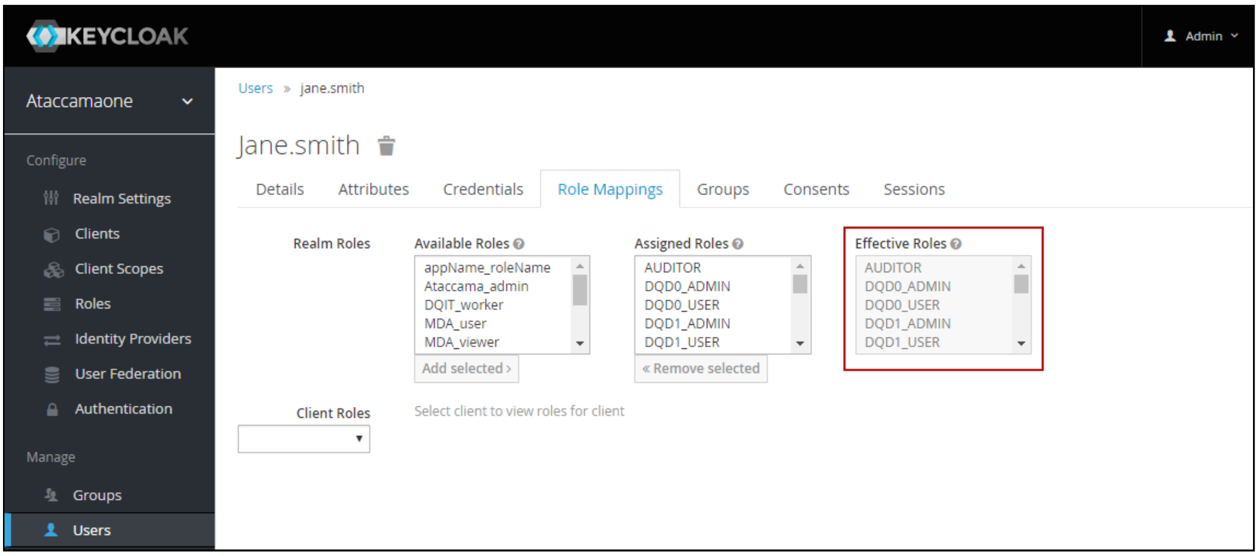


You can view all users with a role in the role configuration page, **Users in Role** tab.



Username	Last Name	First Name	Email	
admin		Admin		Edit
john.taylor	Taylor	John	john.taylor@ataccama.com	Edit
jane.smith	Smith	Jane	jane.smith@ataccama.com	Edit

You can view all roles assigned to a user in the user configuration page, **Role Mappings** tab.



Make sure that after each role change (manual or gained from group roles) all sessions of that particular user should be logout by **Keycloak Administrator** (**Log out all sessions**) using the **Sessions** tab from **Keycloak Administration Console**.

See Setting Permissions in RDM for detailed information on RDM permissions.

Due to configuration changes in Admin Console it no longer allows access to the role `RDM_admin`, but to the role defined in `ataccama.one.rdm.system-group-name` , which is where access to the RDM web application permissions tab is defined. This means the role defined in this property has access to both permissions tab and Admin Console and there is currently no role that has access to just the permissions tab, as previous `RDM_permissions` user did.

2.9 RDM Synchronization

RDM can be synchronized with external systems in several ways:

1. Synchronization via text files, with a possibility to transfer files to and from a remote server.
2. Synchronization with external databases (in both directions).
3. Writing data to and reading data from selected RDM tables via web services.

In addition, it is possible to configure an on-publish service (based on a ONE plan), which will take place each time a user tries to publish data in the web application.

The following articles provide instructions for configuring tasks and web service definitions for all these synchronization types:

2.9.1 Adding Databases Synchronized with RDM

RDM can synchronize reference data in its repository with any other defined databases. This article explains how to add such a database to your RDM project and define a data model for it, i.e., specify tables in that database that will be synchronized. The synchronization task itself is described in Configuring RDM Synchronization with External Databases.

Step 1 Define a New Connected System

1. Define a database connection in the IDE: switch to the File Explorer view, right-click **Databases > New Database Connection**. See Connecting to a Database for more information.
2. Switch back to the Model Explorer.
3. Expand **Connected Systems > Databases.**
4. Right-click **Databases > New system.**
5. Fill in External Database Attributes.
6. Click **OK** to save changes.

External Database Attributes

Attribute	Mandatory	Description
System name	Y	Name of the external database.
Database connection	Y	Name of a previously defined external database connection.
Description	N	Description of the database.
Tables	Y	Allows adding external database tables (can be added via XMI or database metadata import). See the next step.
Relationships	N	Allows adding relationships between external database tables (can be added via XMI or database metadata import). See the next step.

Step 2 Build the External Database Data Model

After a defining a database to synchronize with, it is necessary to build and define the data model for it. At least one table should be defined. As with the RDM data model , this can be done in three ways:

- Manually creating the model by creating tables and relationships between them.
- Importing database metadata.
- Importing the model from a modeling tool.

See Creating the RDM Data Model for more information.

External Database Table Attributes

Attribute	Mandatory	Description
Table name	Y	Name of the table in the external database.
Synchronize with RDM table	Y	Name of the corresponding table in the RDM model.
Synchronize periodicity	N	Specifies the frequency of synchronization. (For documentation purposes only).
Description	N	Description of the table.
Input columns	Y	Allows defining external database attributes: • **Name** - name of the column in the external database table. • **Type** - data type of the column (see Domain Attributes for data type descriptions). • **Mapped rdm column name** - name of the counterpart column in the RDM table, with which the column in the external database table will be synchronized. • **Database type** - database data type of the column. • **Required** - specifies whether records can have this column empty.

Attribute	Mandatory	Description
RDM Key	Y	Attributes (ideally primary keys) that allow determining corresponding records in the RDM table and the external database table: • **RDM table column** - key in the RDM table. • **System table column** - key in the external database table.

External Database Relationship Attributes

Attribute	Mandatory	Description
Name	Y	Name of the relationship.
Parent table	Y	Name of the parent table.
Child table	Y	Name of the child table.
Foreign key	Y	Defines the relationship between the parent and child system table: • **Parent column** - name of the column defining the relationship in the parent table • **Child column** - name of the column defining the relationship in the child table

Make sure you add database connections to the Runtime Configuration of your ONE Runtime Server Development Guide to have them available to the server. You can always export this configuration as a valid runtime configuration XML via **File > Export** > *search for* **Runtime Configuration**. See Exporting and Importing Runtime Configuration for more information.

2.9.2 Adding Remote Servers for File Transfers

RDM can synchronize its repository with external systems via text file transfers to and from remote servers. See Configuring RDM Synchronization via Text Files to learn how to configure jobs for these tasks. This article explains how to add a remote server to the RDM configuration.

1. Define a server connection in ONE Desktop: switch to the File Explorer view, right-click **Servers > New Server**.
2. Switch back to the Model Explorer.
3. Expand **Connected Systems > Servers**.
4. Right-click **Servers > New system**.
5. Fill in the fields for server definition:

Field	Mandatory	Description
System name	Y	Name of a previously defined server connection. To define a server connection, switch to the File Explorer view, right-click **Servers > New Server**. See Creating a New Server Connection for more information.
Known hosts file	Y	Path to the `known_hosts` file, which stores public keys of previously accessed SSH servers. The file is usually located in `[user_home_directory]/.ssh/` for both Windows and Linux.
Description	N	Description of the server.

6. Click **OK** to save changes.

A remote server is now added and available in the RDM Model Project.

> Make sure you add server specification to the Runtime Configuration of your ONE Runtime Server Development Guide to have them available to the server. You can always export this configuration as a valid runtime configuration XML via **File > Export** > *search for* **Runtime Configuration**. See Exporting and Importing Runtime Configuration for more information.

2.9.3 Configuring RDM Synchronization via Text Files

RDM can import and export data text files. It is also possible to set up file transfers between RDM and a remote server in order to download files for import, or upload exported files.

Read more about import and export tasks using text files in the following articles:

Configuring Tasks for Importing Data from Text Files

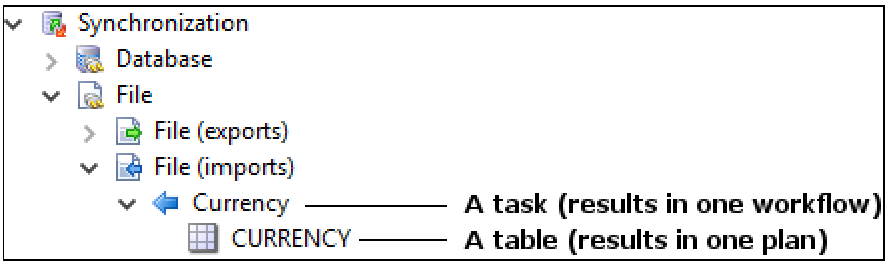
The following steps specify how to configure a task for importing data to RDM tables from text files. Additionally, files can be downloaded from a remote server, one by one in separate tasks, or as an archive in one task. The configuration results in a set of plan files and one workflow `.ewf` file executing all relevant plans and performing additional tasks like downloading from a remote server, unpacking, and deleting files after importing.

Step 1 Understand Import Task Anatomy

A text file import task has the following hierarchical structure:

- **Task** = one workflow file
    - **Table** = one plan file

Several tables can be updated in one task.



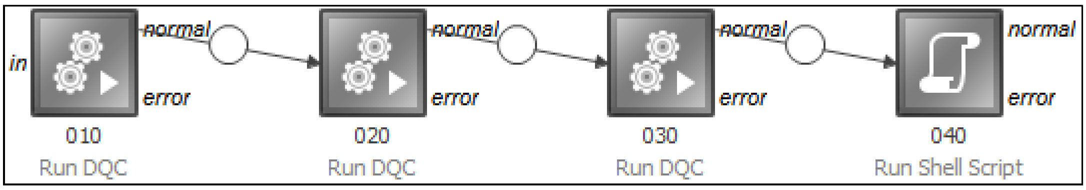
Import Task Anatomy

Step 2 Determine Your Use Case

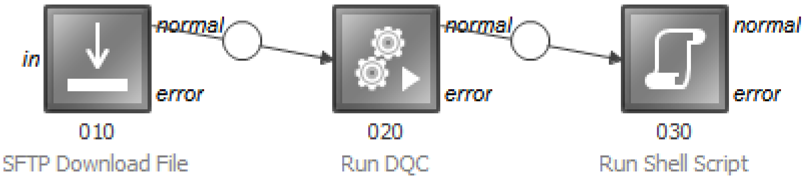
The import functionality in RDM supports a few optional features, like support for unzipping archives and transferring files from a remote server. Combinations of the available options result in the following possible use cases:

1. Manually place CSV or TXT files to the local server (containing the RDM Runtime and ONE Runtime Server Development Guide) and import data to RDM (one or multiple tables per

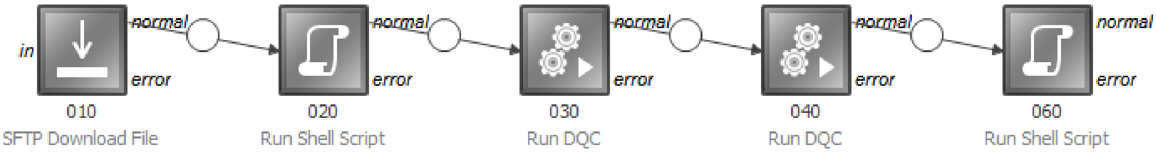
task), plus optionally delete the file after the import is finished.



2. Download one text file from a remote server to the local server and import data to one RDM table in one task, plus optionally delete the file after the import is finished.



3. Download an archive containing one or several text files from a remote server to the local server, unpack it, and import data to one or several RDM tables in one task, plus optionally delete the file after the import is finished.



The output of any configuration is a workflow file consisting of several tasks like downloading an archive, unzipping it, and running ONE plans (one plan per table), and deleting all files after the import is finished.

Step 3 Define a Synchronization Task in the RDM Project

1. Expand **Synchronization** > **File** > **File (imports)**
2. Right-click **FIle (imports)** and select **New synchronization task.**
3. Fill in the fields:

Field	Required for Use Cases	Description
Name	ALL	Task name.

Field	Required for Use Cases	Description
Server name	2,3	Name of a previously defined server connection. To define a server connection, switch to the File Explorer view, right-click **Servers > New Server**. See Creating a New Server Connection for more information.
Local Server Downloads Directory	2,3 + 1*	Directory on the local server (containing the RDM Runtime and ONE Runtime Server Development Guide) to which the file with data is downloaded from the remote server (use cases 2 and 3). * Directory which is cleaned up after a successful import if **Delete file after import** is checked (use case 1). The path is relative to your RDM project folder and should point to a descendant of the Files folder, e.g., `Files/data/FTs/exports`.
Remove Server Source Directory	2,3	Directory on the remote server containing the file to download to the local server.

Field	Required for Use Cases	Description
File Name	2,3	File (text or archive) to download from the remote server.
Compression	3	Specifies the compression type (if any).
Delete file after import	N/A	If checked, the text files and archive are deleted after a successful import.
Authentication Strategy	ALL	Determines the authentication configuration for the RDM Importer step in the generated import plan. See RDM App Variables for information on authentication strategies.
Tables	ALL	The list of tables that are updated with the task.

4. Click **OK** to save changes and close the window – a new synchronization task is created under **Synchronization > File > File (imports),** listing the affected tables and the **Synchronization task (generated**) node.
5. Save changes to your project and reload metadata.
6. Right-click the newly created task and select **Generate workflow and plan(s)** – a workflow file is generated to **Files > workflows** and all dependent plans are generated to **Files > plans > synchronization > FTs > imports**.

> The plan naming convention is `[task_name]_[table_name]_import.plan`, e.g., `newtask_PRODUCTS_import.plan`.

> The workflow naming convention is `[task_name]_import.ewf`, e.g., `newtask_import.ewf`.

You can also generate plans separately for each selected table: right-click the node with the table under the task and select **Generate plan.**

7. The first part of the configuration is done. Now configure the generated plans.

Step 4 Configure Plans for Importing Data

When a plan for importing data from a text file is generated for the first time, it contains two steps:

- Rdm Importer - actually imports data to the RDM Repository.
- Text File Writer - writes errors sent by Rdm Importer to a text file.

The plan must be further configured:

1. Right-click a node with a table under an import task and select **Open Plan.**
2. Connect Rdm Importer with Text File Writer.
3. Add an input step containing data to be imported (Text File Reader, Jdbc Reader, etc.) and connect it to the *in* endpoint of Rdm Importer
4. (Optional: if you are passing credentials manually) Add an input step containing parameters for the Rdm Importer step and connect it to the *parameters* endpoint of Rdm Importer. The data source should have one row of data containing a valid username and password.

Example Parameters File

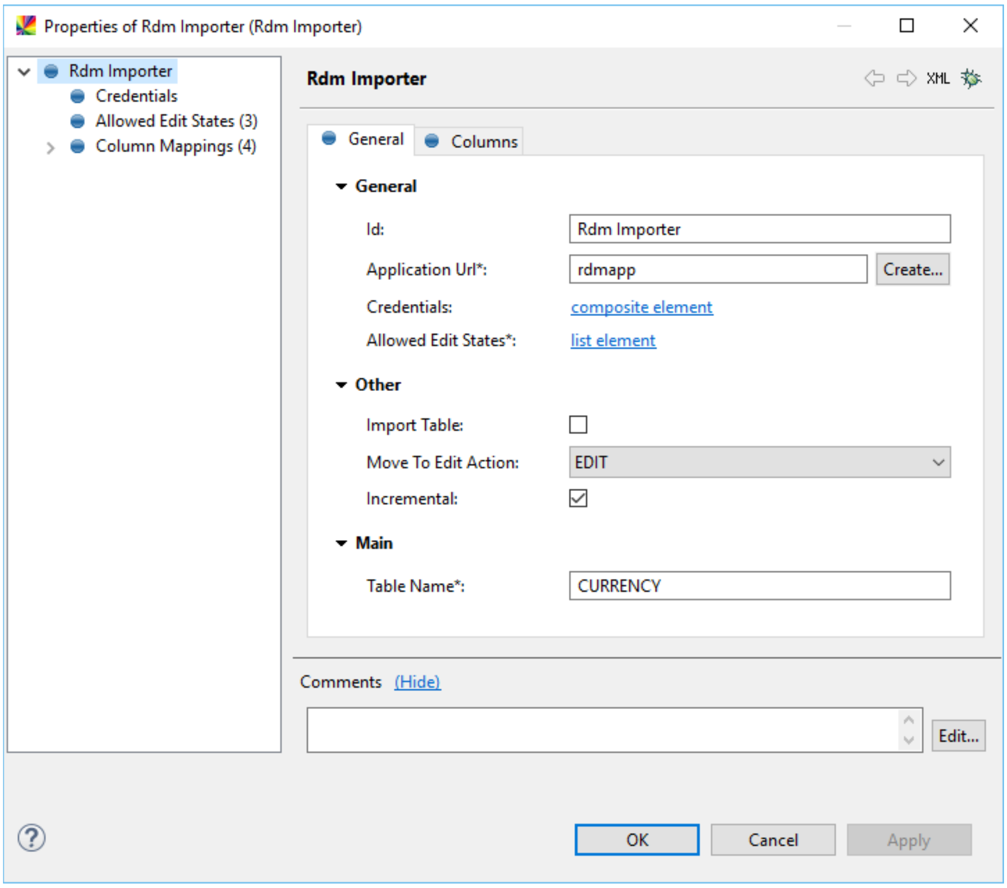
```
username;password
betty;5ecurePassword
```

Pro Tip

You can add additional steps before Rdm Importer to filter or transform data. See the screenshot of the configured plan below.

5. Double-click the Rdm Importer step to start configuring it in the **General** tab:
    a. In the **Move to Edit Action** field, select the state in which the records will be published.
    b. Check **Incremental** (recommended) to turn on the incremental mode: new records are imported, existing records not present in the input file are untouched, matched records are updated.

   c. Check **Import Table** for the data to be imported to the corresponding table in the *Import* mode. This allows checking the imported data before moving it to the *Edit* mode.



Id: Rdm Importer

Application Url*: rdmapp

Credentials: composite element

Allowed Edit States*: list element

Import Table: ☐

Move To Edit Action: EDIT

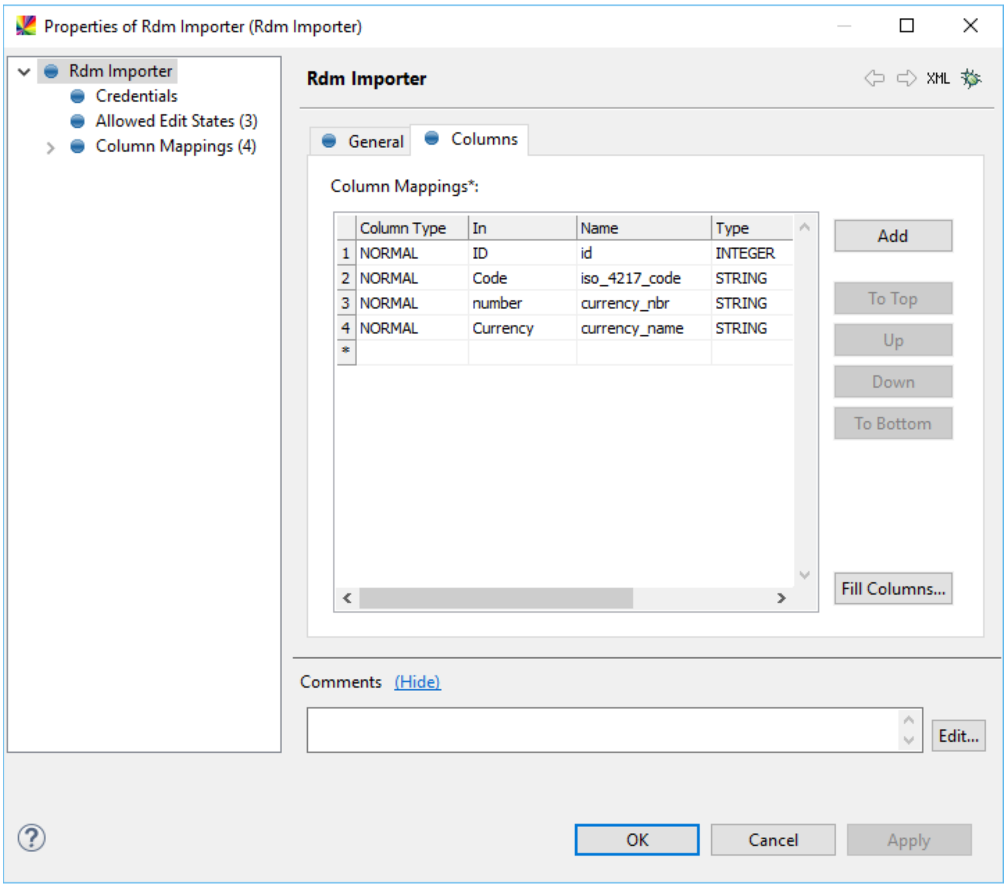
Incremental: ☑

Table Name*: CURRENCY

RDM Importer General Tab

6. Switch to the **Columns** tab, map columns from the data source (**In** column) to the columns in the **Name** column (attribute names as modeled for the table). Select the appropriate **Column Type** for each column:
    - NORMAL - a regular RDM column containing reference data (columns of this type are modeled under the **RDM Logical Model > Tables** node).
    - INPUT - a column from the counterpart external database table (see Adding Databases Synchronized with RDM for more information).
    - CHANGE_TYPE - the edited state of the record (new, edited, deleted).
    - FROM - date and time when the record was published.
    - TO - date and time when the record was moved to the **History** table (a change was made to the record and published on this date).
    - PRIMARY_KEY - the RDM-generated id of the record.

- GROUP_PRIMARY_KEY - the RDM-generated group id of the record (used to group the same records with different business dates).
- HCN - history change number; the ordinal number of the publishing action during which the record was published.
- USERNAME - the user that has made the change to the record.

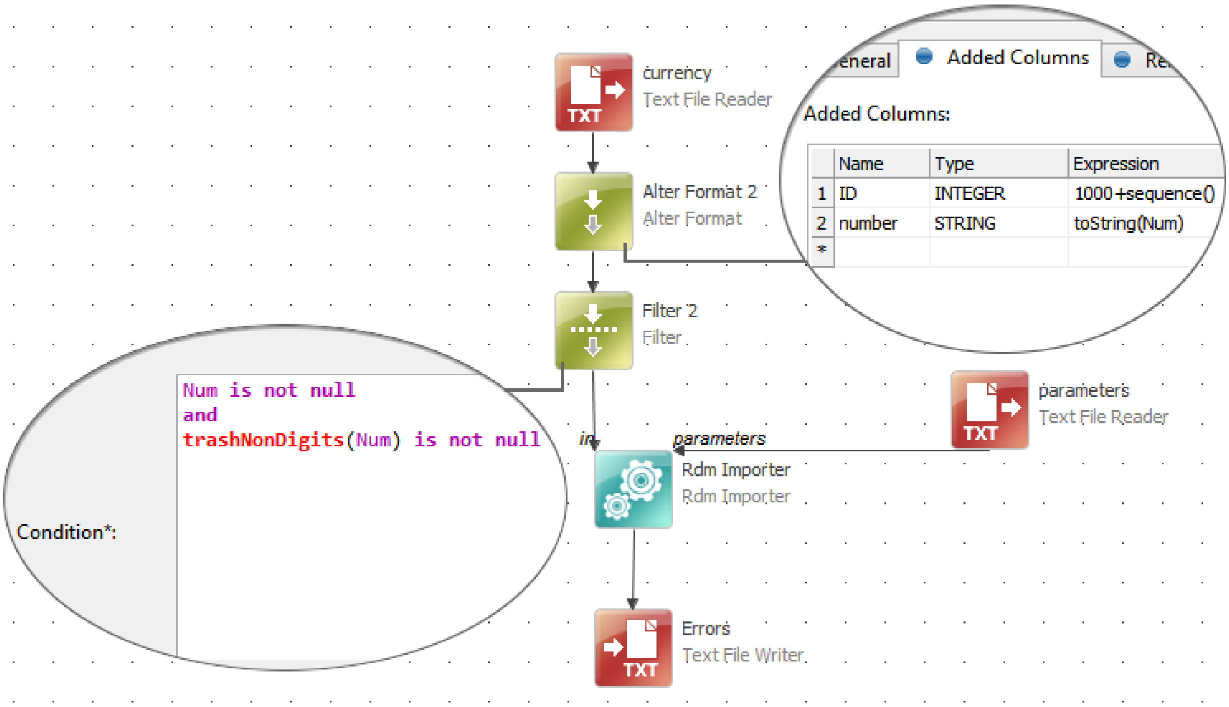


	Column Type	In	Name	Type
1	NORMAL	ID	id	INTEGER
2	NORMAL	Code	iso_4217_code	STRING
3	NORMAL	number	currency_nbr	STRING
4	NORMAL	Currency	currency_name	STRING
*				

RDM Importer Columns Tab

7. (Optional: if you are passing credentials manually) From the step navigation tree, select **Credentials** and configure the credentials:
   - In the **User** field, enter the name of the attribute containing the username.
   - In the **Password** field, enter the attribute containing user password.
8. Click **OK** to save changes and close the window.

The plan is ready to be tested and deployed. The plan below adds two columns in the Alter Format step and filters the `Num` column before sending data to the Rdm Importer step, which sends data to the RDM Repository.



Added Columns:

	Name	Type	Expression
1	ID	INTEGER	1000+sequence()
2	number	STRING	toString(Num)
*			

Condition*:

```
Num is not null
and
trashNonDigits(Num) is not null
```

A Configured Plan

Step 5 Schedule the Import Task

The workflow generated into **Files > workflows** can be scheduled for regular runs by the scheduler. See Scheduling RDM Tasks to learn how.

Configuring Tasks for Exporting Data to Text Files

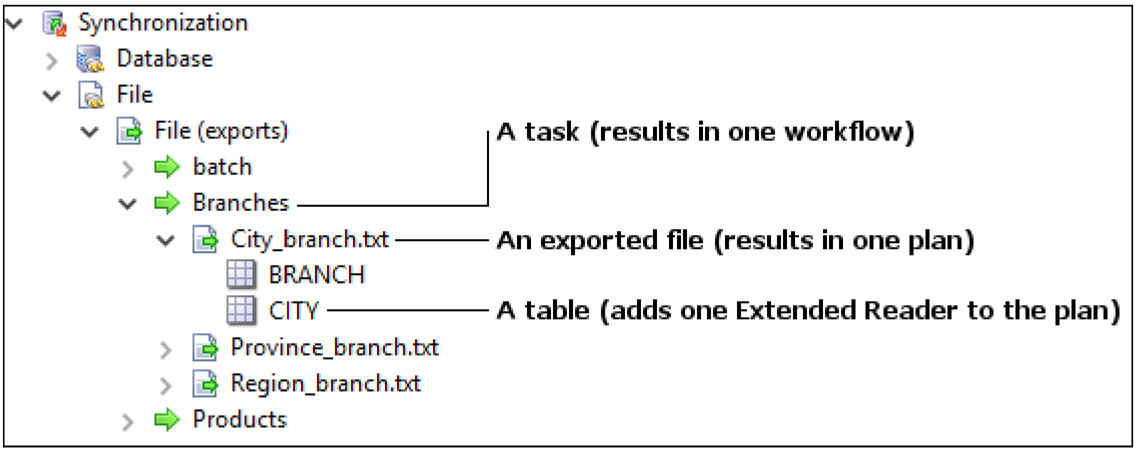
The following steps specify how to configure a task for exporting data from RDM tables to text files. Additionally, files can be transferred to a remote server, one by one or as an archive. The configuration results in a set of plan files and one workflow `.ewf` file executing all relevant plans and performing additional tasks like compressing and uploading files to a remote server.

Step 1 Understand Export Task Anatomy

A text file export task has the following hierarchical structure:

- **Task** = one workflow file
  - **Export** = one plan file and one exported text file
    - **Table** = one Extended Reader step in the plan

A task can include several exports, which in turn can export data from several tables at once.



A task (results in one workflow)

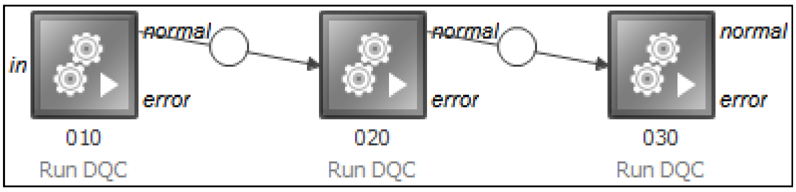
An exported file (results in one plan)

A table (adds one Extended Reader to the plan)

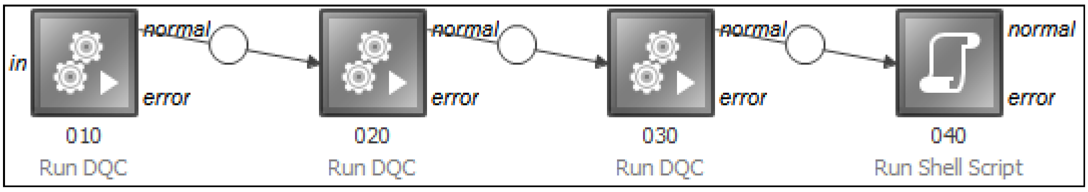
Step 2 Determine Your Use Case

The export functionality in RDM supports a few optional features, like compressing exported text files into an archive, or sending exported files to the remote server. Combinations of the available options result in the following possible use cases:

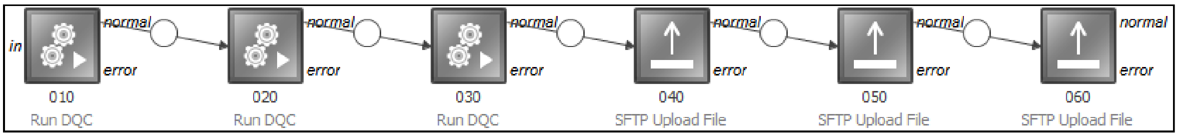
1. Export Files from RDM to the local server (containing the RDM Runtime and ONE Runtime Server Development Guide).



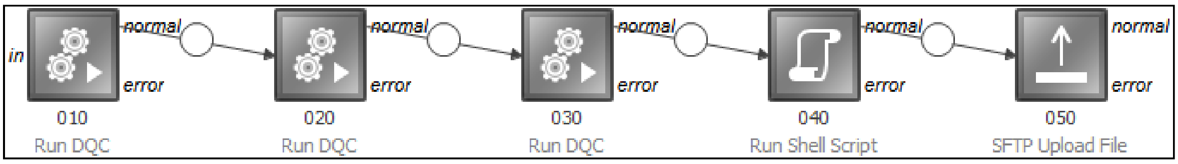
2. (1) + compress the exported files into an archive (ZIP and GZIP formats are supported).



3. (1) + send the exported files to a remote server.



4. (2) + send the archive to a remote server.



Step 3 Define a Synchronization Task in the RDM Project

1. Expand **Synchronization > File > File (exports).**
2. Right-click **File (exports)** and select **New synchronization task.**
3. Fill in the fields according to your use case:

Field	Required for Use Cases	Description
Name	ALL	Task name.
Server name	3, 4	Name of a previously defined server connection. To define a server connection, switch to the File Explorer view, right-click **Servers > New Server**. See Creating a New Server Connection for more information.
Remote Server Target Directory	3, 4	Directory on the remote server to which the file with data is uploaded from the local server (containing the RDM Runtime and ONE Runtime Server Development Guide).
Local Server Archive Directory	2, 4	Directory on the local server (containing the RDM Runtime and ONE Runtime Server Development Guide), to which the archive is saved (if **Compression** is used). The path is relative to your RDM project folder and should point to a descendant of the Files folder, e.g., `Files/data/FTs/exports`.

Field	Required for Use Cases	Description
Archive Name	2, 4	File (text or archive) to download from the remote server.
Compression	2, 4	Specifies the compression type (if any).
Exports	ALL	The list of sub-tasks that export data from one or several tables into one text file – a plan is generated for each export.

4. Under Exports, click **Add** and then double-click the row number to enter export definition details.
5. Fill in the fields according to your use case:

Field	Required for Use Cases	Description
File name	ALL	RDM exports data to a file with this name (TXT and CSV files supported).
Field separator	ALL	The symbol that separates columns in the text file.
Line separator	ALL	The symbol that marks the end of the line: `\r` (MacOS 9 and older) , `\n` - (Linux and MacOs X), or `\r\n` (Windows).
Write header	N/A	If checked, column headers are written to the text file together with data.
Target directory	3	Directory on the remote server to which the file is sent.

Field	Required for Use Cases	Description
File path	ALL	Directory on the local server where to save the text file. The path is relative to your RDM project folder and should point to a descendant of the Files folder, e.g., `Files/data/FTs/exports`.
Authentication Strategy	ALL	Determines the authentication configuration for the RDM Extended Reader step in the generated export plan. See RDM App Variables for information on authentication strategies.
Generate additional steps	N/A	If false (default), only the RDM Extended Reader step is generated or updated according to the logical model within the generated plan (other steps stay untouched). If true, a set of default steps is generated or updated automatically into the plan.
Tables > Name	ALL	The list of tables that whose data is exported within the task.
Tables > History	N/A	If checked, data from the ALL_HISTORY mode is downloaded together with currently published. Also, a `_hist` suffix is added to the file name. See RDM Data Viewing Modes for more information on the ALL_HISTORY table.

6. Click **Apply** to save the current configuration and then click the back arrow in top right corner.
7. Repeat steps 4-5 until done with all exports.

8. Click **OK** to close the windows.
9. Save changes to your project and reload metadata.
10. Right-click the newly created task and select **Generate workflow and plan(s)** – a workflow file is generated to **Files > workflows** and all dependent plans are generated to **Files > plans > synchronization > FTs > exports**.

    The workflow naming convention for is `[task_name]_export.ewf`, e.g., `newtask_export.ewf`.

    The plan naming convention is `[task_name]_[file_name]_export.plan`, e.g., `newtask_products.txt_export.plan`

    You can also generate plans separately for each exported file: right-click the node with the filename under the task and select **Generate plan.**

Step 4 Configure Plans for Exporting Data

When a plan for exporting data to a text file is generated for the first time, it contains the following steps:

- Rdm Extended Reader times the number of exported tables – reads data from Rdm,
- Text File Writer - writes exported data to a text file.
- Multiplicator - optional step for the case of multiple Rdm Extended Readers: makes input parameters available to all Extended Readers.
- Sort - optional step for sorting data

The plan must be further configured:

1. Right-click a node with the filename under an export task and select **Open Plan.**
2. Add an input step (e.g., a Text File Reader or Jdbc Reader) containing parameters for Rdm Extended Reader. The data source should have one row of data containing the following data (the column names are up to you – they will be mapped to configuration fields in the Rdm Reader steps):
    - **username** (Optional: if you are passing credentials manually) - a valid RDM user with read rights to the table
    - **password** (Optional: if you are passing credentials manually) - user password

- **timestamp** - (1) the date as of which to export data; (2) in case **incremental timestamp** is set, the end of the period for which to export data from the ALL_HISTORY mode.
- **incremental timestamp** (optional) - the beginning of the period for which to export data from the ALL_HISTORY mode.

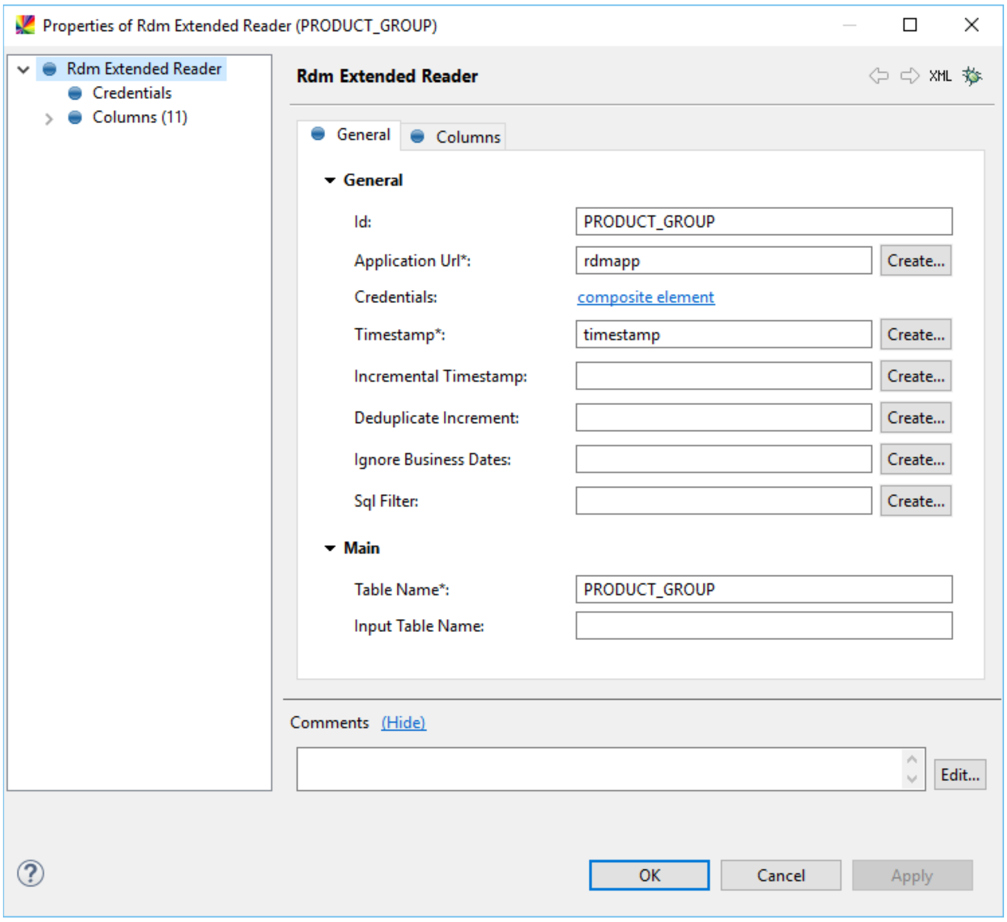
Example Parameters File

```
username;password;timestamp;timestamp_increment
betty;5ecurePassword;;2016-01-01 00:00:00
```

Leaving the `timestamp` attribute empty is treated as "now."

The date format in the example is the recommended date format.

3. Connect the input step with parameters to Multiplicator.
4. For each RDM Extended Reader step, double-click the step to start configuration in the **General** tab: map the parameters from step 2 to the step configuration corresponding fields.



Id: PRODUCT_GROUP

Application Url*: rdmapp

Credentials: composite element

Timestamp*: timestamp

Incremental Timestamp:

Deduplicate Increment:

Ignore Business Dates:

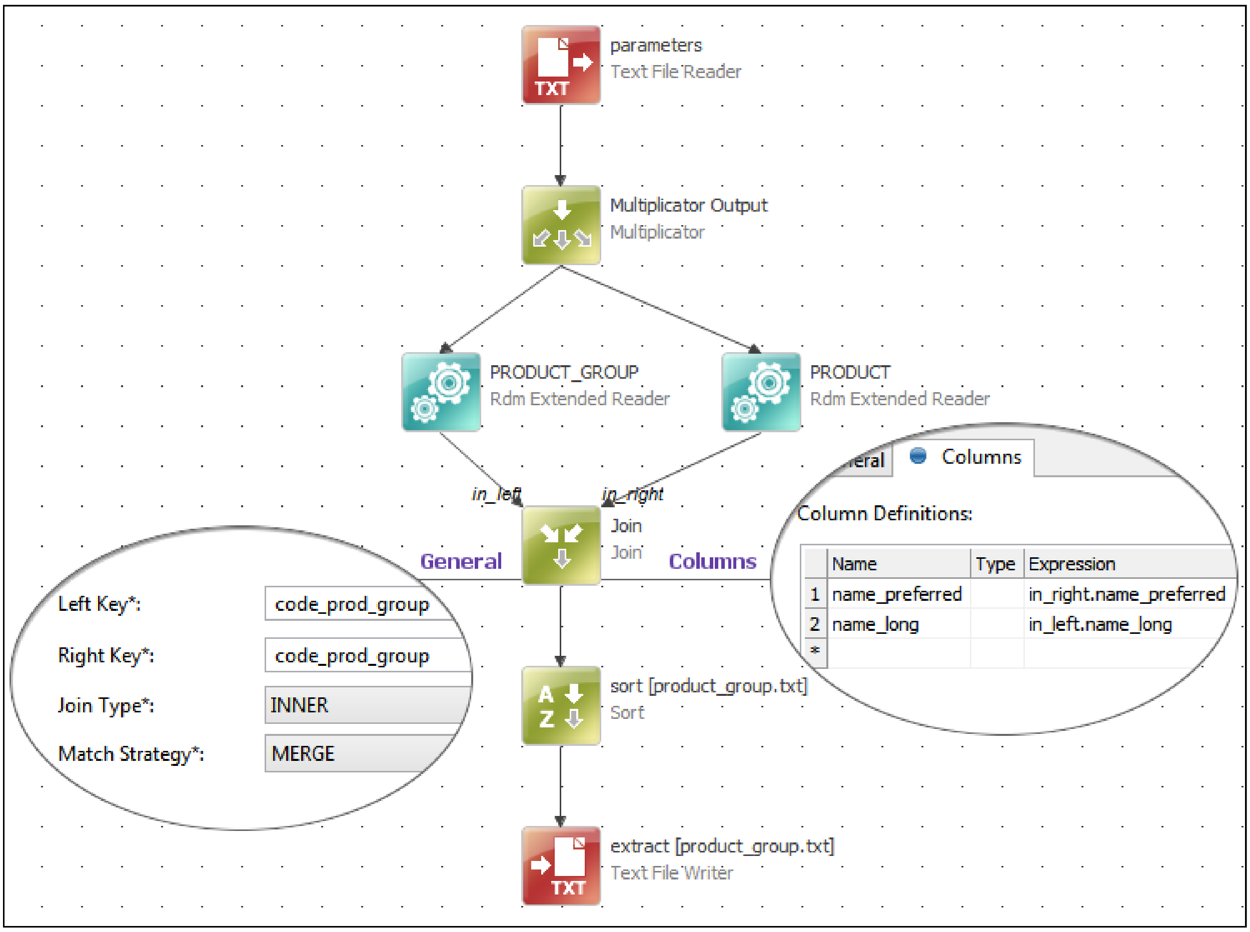
Sql Filter:

Table Name*: PRODUCT_GROUP

Input Table Name:

5. (Optional: if you are passing credentials manually) From the step navigation tree, select **Credentials** and configure the credentials:
    - In the **User** field, enter the name of the attribute containing the username.
    - In the **Password** field, enter the attribute containing user password.
6. In the case of multiple Rdm Extended Reader steps, join and transform their data flows as you desire (e.g., by using the Join steps) and connect them to the Sort step (example plan is shown below).
7. (Optional) Configure the Sort step to have the data sorted.
8. Double-click Text File Writer, switch to the **Columns** tab and specify the columns that should be written the text file or tick **Write All Columns.**

The plan is ready to be tested and deployed. In the example plan below, we are joining data from two RDM tables: PRODUCT and PRODUCT_GROUP, with the goal of extracting a list of all products into the `name_long` column and the product groups they belong to into the `name_preferred` column. We join the data flows by the the `code_prod_group` column present in both tables.



Left Key*: code_prod_group

Right Key*: code_prod_group

Join Type*: INNER

Match Strategy*: MERGE

Column Definitions:

	Name	Type	Expression
1	name_preferred		in_right.name_preferred
2	name_long		in_left.name_long
*			

Step 5 Schedule the Export Task

The workflow generated into **Files > workflows** can be scheduled for regular runs by the scheduler. See Scheduling RDM Tasks to learn how.

2.9.4 Configuring RDM Synchronization via Web Services

RDM can import and export data via SOAP over HTTP web service calls.

Read more about preparing web service configurations for these tasks:

Preparing Web Service Configurations for Reading Data

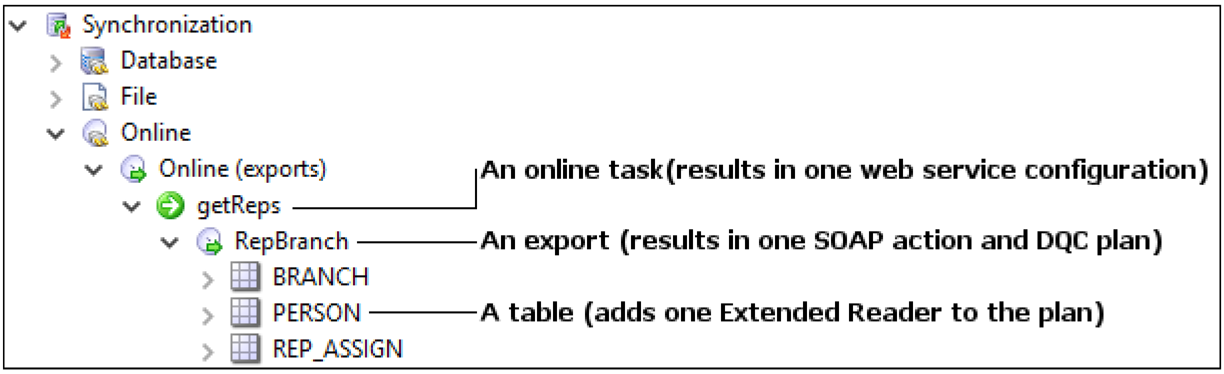
The following steps specify how to prepare a web service configuration for reading (exporting) data from RDM tables. The configuration results in a set of plan files and one `.online` file containing the web service configuration.

Step 1 Understand the Online Export Task Anatomy

Defining a web service configuration for reading (exporting) data has the following hierarchical structure of constituents:

- **Online Task** = `.online` file = a service with a specific URL and predefined endpoint.
    - **Export** = ONE plan and SOAP action.
        - **Table** - one Extended Reader step in the plan.

You can make several tables available for reading data from the same URL location. Each table collection like that will have its own SOAP action. In this case, you will need to configure the generated plan for each task to define which attributes from each table you want to have available or calculate a custom attribute. See Preparing Web Service Configurations for Reading Data below.


Synchronization
Database
File
Online
Online (exports)
getReps
RepBranch
BRANCH
PERSON
REP_ASSIGN
An online task(results in one web service configuration)
An export (results in one SOAP action and DQC plan)
A table (adds one Extended Reader to the plan)

Online Export Task Anatomy

Step 2 Prepare Web Service Definition in the RDM Project

1. Expand **Synchronization** > **Online** > **Online (exports).**
2. Right-click **Online (exports)** and select **New Online task.**
3. Specify the service **Name**.
4. Select a **Method**:
    - **JSON over HTTP (GET)** - request with export input parameters in the URL, response in JSON format
    - **JSON over HTTP (POST)** - request with export input parameters in JSON format, response in JSON format
    - **SOAP over HTTP** - standard SOAP format
    - **TXT/CSV over HTTP** - request with export input parameters in the URL, response in CSV format
5. Under **Exports** specify the name.
6. Double-click the number next to the export specification to open its details.

7. Configure **Authentication Strategy**. See RDM App Variables for information on authentication strategies.
8. Under **Tables** specify which tables are available for consumption:
   a. In the **Name** field, select an existing table.
   b. Check **History** if you want historical records to be available for consumption (via the `timestamp_increment` web service input element).
   c. Specify which columns should be available for consumption: either check **Use All Columns** or select columns manually under **Columns**.

   > Note that when **Use All Columns** is checked, adding and deleting columns in the table in question is reflected the plans and `.online` files generated in the last step below.

9. (Optional) Under **Additional output columns**, list custom calculated columns, e.g., calculate age from the date of birth column. You will need to specify the algorithms populating such columns in the generated plan. See Preparing Web Service Configurations for Reading Data below.
10. Click **Apply** to save the configuration – a new configuration appears under **Synchronization > Online > Online (exports).**
11. Click the back arrow in the top right corner of the configuration dialog to go back and add another export under **Exports.**
12. Repeat steps 4-9 until done with all exports.
13. For each online task, right-click the task and select **Generate online file and plan(s)** – all plans and the `.online` file is generated.
    a. Plans are generated into **Files > plans > synchronization > onlines > exports**.

    > The file naming convention is `[task_name]_[export_name]_export.plan`, e.g., `newtask_PRODUCTS_export.plan`.

    b. The `.online` file is generated to **Files > onlineServices.**

    > The file naming convention is `[task_name]_export.online`, e.g., `newtask_export.online`.

Configuring a Plan for Reading Data

As has been noted above, each export under a data-reading task results in a plan file. Typically, generated files do not require modification: the configuration is ready for testing and deployment. However, if you would like to get a custom column set returned, you will need to modify the generated plan.

There are two main cases when you need to do this:

- You want to calculate a special value to a custom column. For example, your data has a date of birth column, but you would (also) like to have age returned.
- You want to receive data from different tables and combine them into a custom report. For example, you want to combine the information from the BRANCH table, which contains basic information about bank branches, with the information from the REGION and CITY tables in order to get full information about the branch location.

Preparing Web Service Configurations for Writing Data

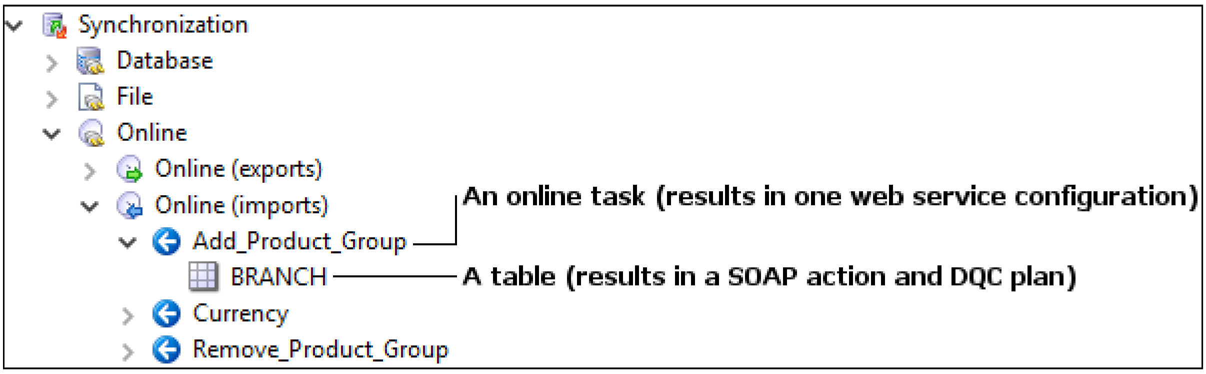
The following steps specify how to prepare a web service configuration for writing (importing) data to RDM tables. The configuration results in a set of plan files and one `.online` file containing the web service configuration.

Step 1 Understand the Online Import Task Anatomy

Defining a web service configuration for writing (importing) data has the following hierarchical structure of constituents:

- **Online Task** = `.online` file = a service with a specific URL location.
    - **Table** = ONE plan and SOAP action.

Several tables can be made available for writing from the same URL location, but each will have its own SOAP action.



Step 2 Prepare Web Service Definition in the RDM Project

1. Expand **Synchronization > Online > Online (imports).**
2. Right-click **Online (imports)** and select **New Online task.**
3. Specify the service name **Name**.
4. Configure **Authentication Strategy**. See RDM App Variables for information on authentication strategies.
5. Under **Tables** specify which tables can be updated by the service:
    - **Name** - select an existing table.
    - **Incremental** - determines the import mode:
        - *unchecked* = FULL mode (matched records are updated, records not present in the request are deleted).
        - *checked* = Incremental mode (matched records are updated, records not present in the request are not touched).

    > A unique key must be set up to match and compare existing and incoming records. See RDM Tables.

6. Click **OK** to save the configuration – a new configuration appears under **Synchronization > Online > Online (imports)** listing the affected tables.
7. Right-click the newly created task and select **Generate online file and plan(s)** – all plans and the `.online` file is generated.
    a. Plans are generated to **Files > plans > synchronization > onlines > imports.**

    > The file naming convention is `[task_name]_[table_name]_import.plan`, e.g., `newtask_PRODUCTS_import.plan`.

    b. The `.online` file is generated into **Files > onlineServices**.

    > The file naming convention is `[task_name]_import.online`, e.g., `newtask_import.online`.

Typically, generated files do not require modification, the configuration is ready for testing and deployment.

2.9.5 Configuring RDM Synchronization with External Databases

RDM can synchronize reference data in its repository with any other external databases. This article explains how to configure a workflow to do that. After you configure the synchronization and deploy the configuration, you will be able to monitor synchronization tasks in the web application: see Monitoring RDM Synchronization.

Prerequisites

- At least one connected system defined under **Connected Systems > Databases**. See Adding Databases Synchronized with RDM.

Step 1 Define a Synchronization Task

1. Expand **Synchronization > Database.**
2. Right-click **Database** and select **New synchronization task.**
3. Fill in the attributes:

Name	Required	Description
Name	Y	Name of the synchronization task.
Use Reverse Synchronization	N/A	If checked, synchronization with an external system is set up and a plan with the *system_input* suffix will be used in the synchronization workflow. See Configuring RDM Synchronization with External Databases below. If you change this setting after generating the workflow file, you need to manually delete the workflow file and regenerate it.

Name	Required	Description
Authentication Strategy	Y	Determines the authentication configuration for the RDM steps in the generated synchronization plans. See RDM App Variables for information on authentication strategies.
Tables	Y	Definition of tables that are synchronized in this task: • **Name** - table name in one of the connected systems • **System name** - name of the database in which the table is located Several tables from different databases can be synchronized in one task.

4. Click **OK** to save changes.

Step 2 Generate Plans and Workflows

Each synchronization task consists of three or four ONE plans and one `.ewf` (workflow) file, which links the plans:

- `[task_name]_input.plan` - imports data from RDM and writes them to TXT files (one per synchronized table).
- `[task_name]_synchronization.plan` - extracts data from the TXT files with imported data and writes it to consuming systems.
- `[task_name]_system_input.plan` - imports data from consuming systems to RDM.

> This plan is included into the workflow only if **Use Inputs** option is checked in the **App Variables** node. See RDM App Variables for more information.

- `[task_name]_output.plan` - reports to RDM about the outcome of the synchronization
- `[task_name].ewf` - a workflow file that puts the plan files above in execution order. Sample file contents are provided below.

To generate files, right-click a synchronization task and select **Generate workflow and plans** – a workflow file is generated to **Files > workflows** and all dependent plans are generated to **Files > plans > synchronization > databases.**

> You can easily open all plans and the workflow by right-clicking a synchronization task and selecting **Open workflow** or a particular plan.

Step 3 Configure the Input Plan

After generating `[task_name]_input.plan` for the first time, you will need to configure certain steps in it. Other plans will not need any configuration unless you chose to pass credentials manually when configuring the task (or it is the setting in RDM App Variables).

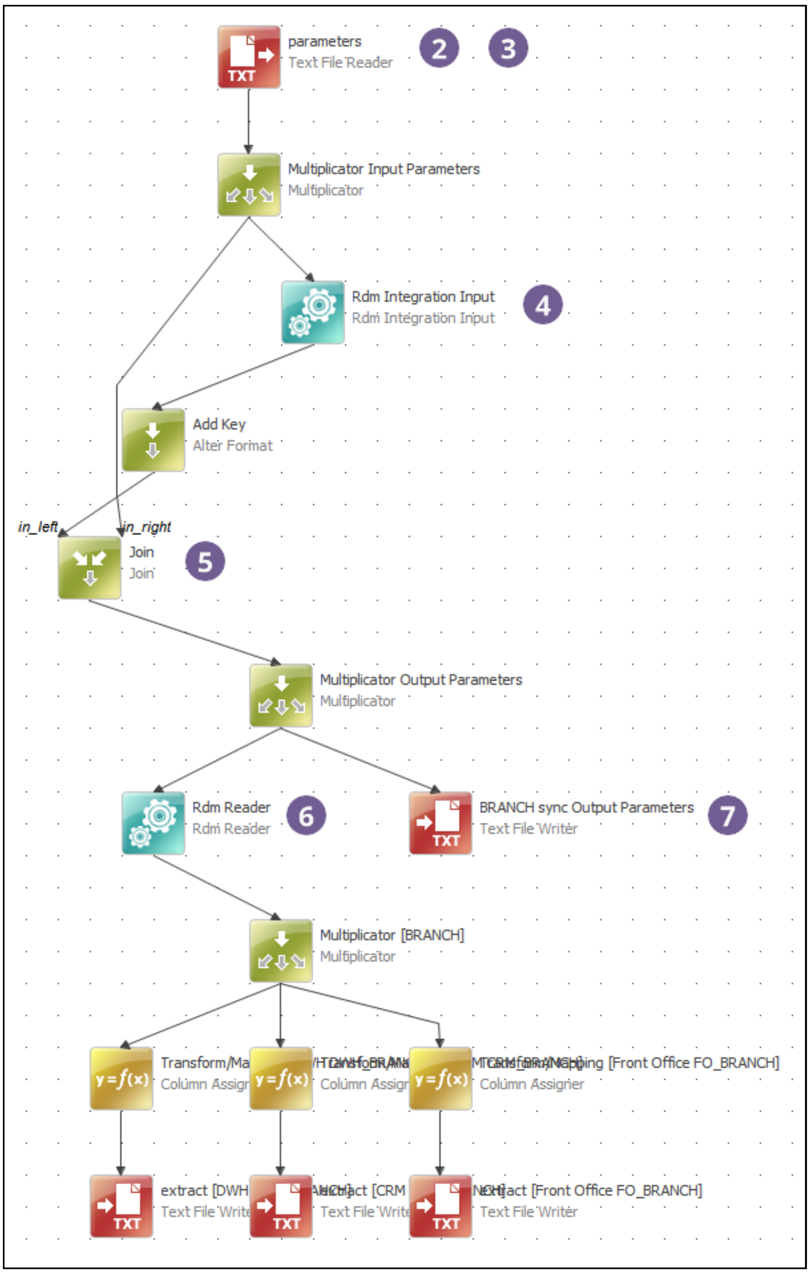
1. Right-click a synchronization task and select **Open input plan.**
2. Add an input step (e.g., JDBC Reader or Text File Reader) with a data source containing one row of data with the following attributes (attribute names are arbitrary, only their meaning is important). We will refer to this data source as *parameters* in the following steps:
    a. **Username** (Optional: if you are passing credentials manually) - a valid RDM user with read rights to the table.
    b. **Password** (Optional: if you are passing credentials manually) - user password.
    c. **Type** - type of synchronization to perform:
        i. **FULL** - exports all published data as seen on the date specified in the timestamp.
        ii. **INCREMENT** - exports only the changes since the last synchronization until the date specified in the timestamp.
    d. **Timestamp** - data are downloaded in the published state as of this date.

Example Parameters File

```
username;password;type;timestamp
betty;5ecurePassword;INCREMENT;2016-01-01 00:00:00
```

3. Configure the added input step.
    a. Switch to the **Shadow Columns** tab and add one column with the following properties:
        i. **Name** - key

        ii. **Type** - INTEGER
        iii. **Default Expression** - 1
4. Configure the RDM Integration Input step:
    1. Map attributes from *parameters* to the **Timestamp**, **Process Mode** attributes in the step.
    2. (Optional: if you are passing credentials manually) From the step navigation tree, select **Credentials** and configure the credentials:
        - In the **User** field, enter the name of the attribute containing the username.
        - In the **Password** field, enter the attribute containing user password.
5. Configure the Join step:
    - In the **General** tab, map the *key* column added in step 2 into the **Right Key** field.
6. Configure the Rdm Reader step:
    - Map the user attribute from *parameters* to the **User** and **Password** parameters.
7. Configure the **[task_name] Output Parameters** Text File Writer step:
    1. Open the step and switch to the **Columns** tab.
    2. Correct columns to those that you use in the *parameters* step.



Step 4 Configure Authentication (Optional)

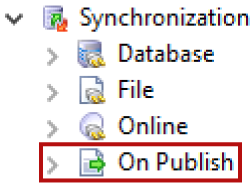
If the **Global Authentication Strategy** in RDM App Variables or local **Authentication Strategy** for the synchronization task is set to **Pass Credentials Manually,** you will need to make that all RDM steps are connected to a data source (e.g., a text file containing attributes with the username and password used to authenticate into the RDM web application), similarly to the configuration of the Input Plan above.

Step 5 Schedule the Synchronization Task

The workflow task generated in step Configuring RDM Synchronization with External Databases above can be scheduled for regular runs by the Task Scheduler. See Scheduling RDM Tasks to learn how.

2.9.6 Configuring an On Publish Action

RDM lets you configure an action happening after any publish event and notify any system integrated with RDM. The action is exposed as a SOAP web service action with a ONE plan behind, which you can configure in any way to your needs.



Step 1 Configure a Server Connection

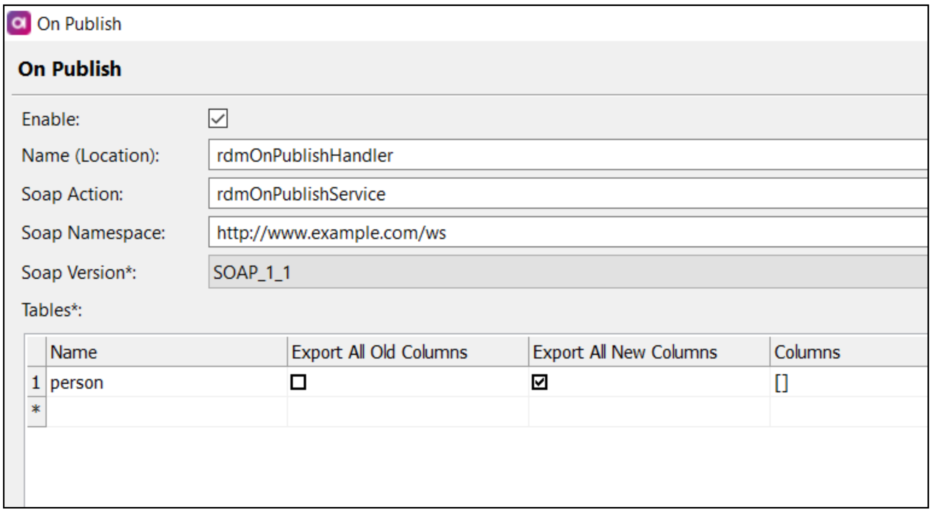
Create a generic server connection with URL corresponding to the location of the RDM runtime server. If the server is secured, provide the username and password with sufficient permissions.

For detailed instructions on creating a server connection, see Creating a New Server Connection. For more information on securing the server, see Server Security.

Step 2 Configure the Event

1. From the Model Explorer, expand the **Synchronization > On Publish** node.
2. Double-click **On Publish.**
3. Perform general service configuration:
    - Most attributes are pre-filled with default values. You can leave them like that or change them according to your needs.
    - Check the **Enable** checkbox.
4. Under **Tables,** add tables that will be tracked by the service.
5. Configure which columns should be available for the service.
    - Enable **Export All New Values** - the service will carry the new (published) values of all columns.
    - Enable **Export All Old Columns** - the service will carry the old values of all columns.
    - Select which columns you want the service to carry.

6. When done with the configuration, right-click the **On Publish** node and select **Generate plan and online file.** The files are named according to the name given to the service during the configuration.
    - The plan is generated to **Files > plans > synchronization > onPublish.**
    - The `.online` file is generated to **Files > onlineServices.**



On Publish

Enable: ☑
Name (Location): rdmOnPublishHandler
Soap Action: rdmOnPublishService
Soap Namespace: http://www.example.com/ws
Soap Version*: SOAP_1_1
Tables*:

	Name	Export All Old Columns	Export All New Columns	Columns
1	person	☐	☑	[]
*				

Step 3 Configure the Plan

When the plan is generated for the first time, it contains the following steps:

- Integration Input times the number of tracked tables - a generic input step reading data from RDM.
- Alter Format times the number of tracked tables - adds and removes several technical RDM columns.

Connect Integration Inputs and Alter Formats and configure the plan according to your needs.

Best Practices

When configuring the plan it is important to understand that If the plan triggered by the publishing action fails, publishing in the web application will fail as well and the data will not be published to the RDM storage.

In relation to this, there are some best practices for configuring the plan:

- Write to a highly available location.
- Do no write to several different locations in parallel streams: if writing to one location is successful and unsuccessful to another, RDM and consuming systems will be in an inconsistent state.

Example Configurations

Text File

1. Export changed records to a text file.
2. Check the filesystem regularly with the Wait For File task and trigger a synchronization when the file appears.

Database

1. Write to a dedicated event table in the DB.
2. Check the table regularly with workflow tasks like Wait For SQL Value or Wait For SQL Row and trigger a synchronization based on an expected value in some column.

Web Service

Resend the data to another service in the format of your convenience: XML via SOAP or JSON via HTTP.

Step 4 Deploy the Configuration

If you have configured an on-publish action for the first time, it is necessary to perform the following deployment steps:

1. Generate and deploy a new configuration package for the web application: How to Deploy an RDM Web App Configuration.
2. Generate the runtime configuration containing the RDM server connection definition: Importing and Exporting Runtime Configuration.
3. If your Server Configuration does not have the Online Services Component configured, configure it.
4. Deploy the plan and the `.online` file to the server running RDM.

Updating the Configuration

If you decide to update the configuration in future (add or remove tables, change the plan, etc.), all you need to do is replaced the existing plan and `.online` file on the server and reload the service configuration. For detailed instructions on reloading the service configuration, see OnlineCtl.

2.9.7 Configuring Plans for Exporting Data from RDM Hierarchies

RDM lets you export data from a configured hierarchy. The configuration results in a plan for each hierarchy.

Step 1 Generate the Plan

1. Expand **RDM Logical Model > Hierarchies.**

2. Right-click one of the existing hierarchies and click **Generate export plan.**

> The plan naming convention is `[hierarchy_name]_export.plan`.

Step 2 Configure the Plan

When a plan for exporting data from a hierarchy is generated for the first time, it contains the following steps:

- Multiplicator for making input parameters available to all Extended Readers.
- Rdm Extended Reader times the number of levels in the hierarchy for exporting data from RDM.
- Complex XML Writer for writing data into an XML file with.

Complete the plan:

1. Right-click the hierarchy for which the plan has been generated in the previous step and click **Open export plan.**
2. Add an input step (e.g., a Text File Reader or Jdbc Reader) containing parameters for Rdm Extended Reader. The data source should have one row of data containing the following data (the column names are up to you – they will be mapped to configuration fields in the Rdm Reader steps):
    - **username** (Optional: if you are passing credentials manually) - a valid RDM user with read rights to the table
    - **password** (Optional: if you are passing credentials manually) - user password
    - **timestamp** - the date as of which to export data

Example Parameters File

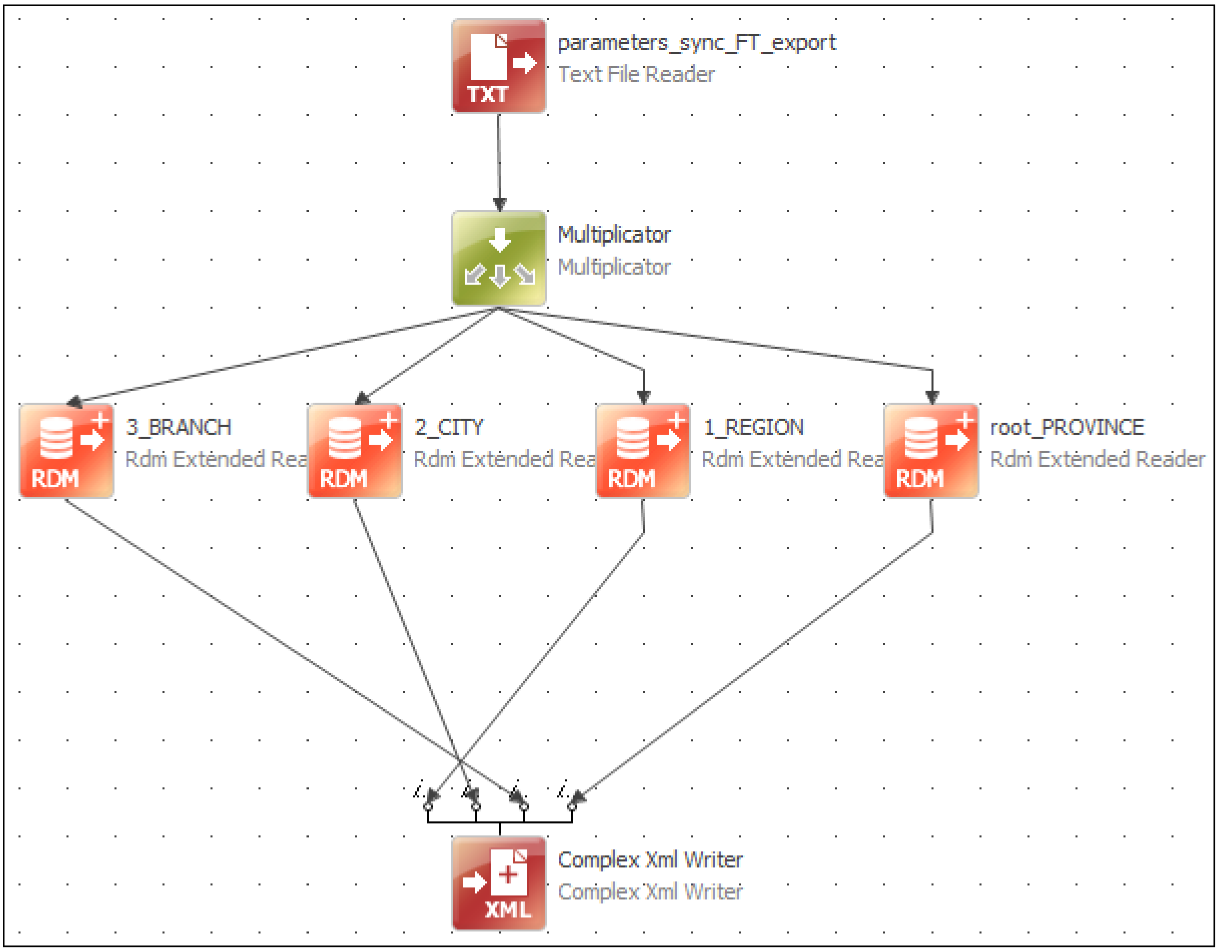
```
username;password;timestamp
betty;5ecurePassword;2016-01-01 00:00:00
```

> Leaving the `timestamp` attribute empty is treated as "now."

> The date format in the example is the recommended date format.

3. Connect the input step with parameters to Multiplicator.
4. For each RDM Extended Reader step, double-click the step to start configuration in the **General** tab: map the timestamp column from your parameters input to the **Timestamp** attribute of the step.
5. (Optional: if you are passing credentials manually) From the step navigation tree, select **Credentials** and configure the credentials:
    - In the **User** field, enter the name of the attribute containing the username.
    - In the **Password** field, enter the attribute containing user password.

The plan is ready to be tested and deployed.



Notes

Special characters are escaped according to XML standards, e.g., `&` is exported as `&amp;`

2.10 Configuring Initial Loads to RDM Tables

When the RDM logical model is created and deployed for the first time, tables created in the web application are empty. To fill an empty table with data, you can either perform a bulk import from a text file in the web application (see Importing and Exporting Data in RDM) or configure a plan, which can load data from any number of sources, transform and join them in any required way, and then populate the chosen RDM table.

It is also possible to generate a plan that will import records together with their editing history – a separate plan is generated for this case.

To configure an initial load plan for a table, follow the steps below:

1. Make sure the **Initial load** or **Initial load - history** checkbox or both are checked for the table that needs to be populated: expand **RDM Logical Model > Tables** and then double click **[your table].** See RDM Tables for screenshots and full reference.

   > When **Initial load - history** is checked, a separate plan with the `_history` suffix is generated.

2. Right-click the table and select **Generate initial load plan(s)** – plans are generated to **Files > plans > batchLoadPlans**.

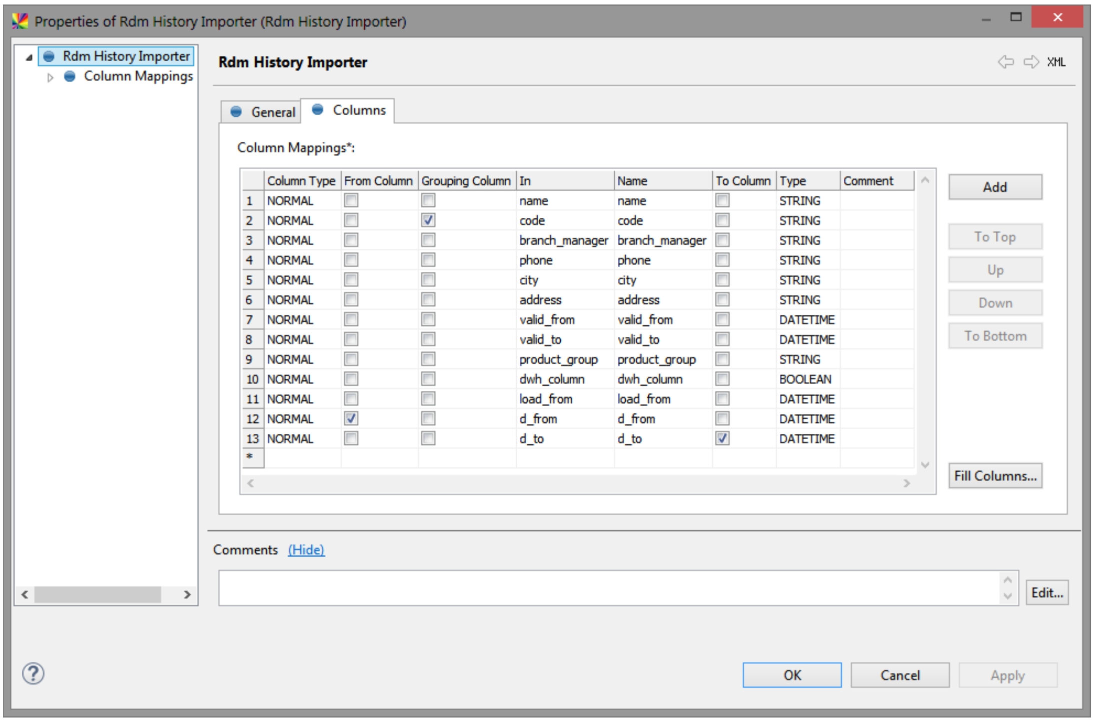
   > To generate plans for all tables, right-click the **Tables** node and select **Generate.**

3. Right-click the table and select **Open initial load plan** or **Open initial load history plan.**
4. Configure your plan:
   a. If you are configuring a **simple load plan (no history)**, then the configuration is very similar or the same to the one described in Configuring Tasks for Importing Data from Text Files.
   b. If you are configuring a **load plan with history** (suffixed with `_history`, the general logic is the same, but a different RDM step is used – Rdm History Importer – which has a slightly different **Columns** tab configuration:
      i. Two columns are required and automatically generated here: `d_from` and `d_to` - these define the beginning and end of validity for a particular edit of any given

record. These columns have **From Column** and **To Column** marked, respectively. Make sure your source data has columns containing this data.

ii. You need to mark one or several columns as **Grouping Column** – these will act like a primary key.

Below is a sample configuration:



Rdm History Importer

Column Mappings*:

	Column Type	From Column	Grouping Column	In	Name	To Column	Type	Comment
1	NORMAL	☐	☐	name	name	☐	STRING	
2	NORMAL	☐	☑	code	code	☐	STRING	
3	NORMAL	☐	☐	branch_manager	branch_manager	☐	STRING	
4	NORMAL	☐	☐	phone	phone	☐	STRING	
5	NORMAL	☐	☐	city	city	☐	STRING	
6	NORMAL	☐	☐	address	address	☐	STRING	
7	NORMAL	☐	☐	valid_from	valid_from	☐	DATETIME	
8	NORMAL	☐	☐	valid_to	valid_to	☐	DATETIME	
9	NORMAL	☐	☐	product_group	product_group	☐	STRING	
10	NORMAL	☐	☐	dwh_column	dwh_column	☐	BOOLEAN	
11	NORMAL	☐	☐	load_from	load_from	☐	DATETIME	
12	NORMAL	☑	☐	d_from	d_from	☐	DATETIME	
13	NORMAL	☐	☐	d_to	d_to	☑	DATETIME	
*								

Sample Rdm History Importer Configuration

5. Save your project and reload metadata.

2.11 Configuring RDM Workflows

Workflows in RDM allow you to track changes to records made to a record in the web application and make sure all necessary parties approve or further edit the record before publishing it. It is possible to configure email notifications notified relevant parties that they need to take action to move the record in the editing workflow.

Workflows are defined separately for each table, which allows to create custom workflows for every individual table.

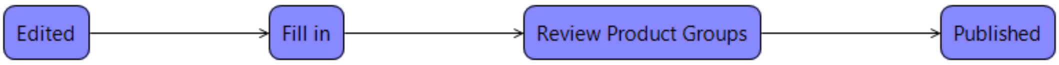
Workflows are configured in the **Workflows and Notifications** node of the RDM project. To be able to see workflows in the application, make sure the Enable checkbox in the **Workflows and Notifications** section is checked.

2.11.1 How Workflows in RDM Work

In the context of RDM, a workflow is a process, which starts with making changes to the data and ends with publishing them. These two points are represented in RDM with two record states:

- *Edited* - a state after a record has been created, modified or deleted.
- *Published* - a state after the record has been published.

Each record goes through these two states. For further control and additional input, you can model custom workflows for each table per particular action (create, edit, delete). Such workflows place additional states between the basic ones mentioned above. During each state, all assigned users approve the changes made, fill in the necessary attributes, and advance the record to its next state.



Example Workflow

In the RDM model project, record states are modeled as **Statuses**. Read more in the next section.

The first step is *Edited* and last step is *Published*. Other statuses are named as *Custom* steps.

See Moving Records Through RDM Workflows in RDM Web Application User Guide for web application perspective on workflows.

2.11.2 Statuses

Statuses are reusable abstract steps that compose workflows and reflect the state of the record as discussed above. Statuses themselves do not carry any particular function apart from being containers for steps in different workflows (statuses are defined just by label).

To see how statuses are reusable, think of a status called *Fill in*. Such a status can be used in different workflows for the same table or even in different tables:

- In workflow 1, it might mean filling column A and B by a user with the *Supervisor* role.
- In workflow 2 in a different table, it might mean filling column C and D by a user with either the *Developer* or *Administrator* role.
- Furthermore, in workflow 1, the *Fill in* step might be followed by the *Approve* step while in workflow 2, this will be the last necessary step before publishing the record.

So the general purpose of the *Fill in* status is to be used in a workflow, which, among other things, requires its participants to fill additional information in of the workflow steps. The actual columns to be filled and other conditions are configured inside the step to which the status is mapped.

Of course, you might use a completely different set of statuses for each workflow, but it will mean unnecessary double work.

How to Create a New Status

1. Right-click **Statuses > New status...**
2. Fill in **Label** - name of the status (will be shown in the RDM web application in the **[State]** column and **Workflow state** field of record details).
3. Click **OK** to save changes.

Now the status can be used in workflows.

2.11.3 How to Configure a Workflow for a Table

Create Statuses

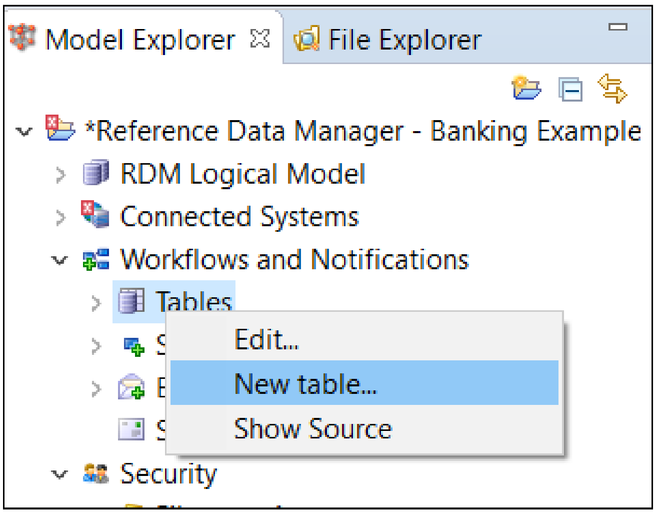
Make sure you have created all the statuses you will need for the workflow. You will need as many statuses as the number of approval stages the record will pass before it is ready to be published + two technical statuses representing:

1. A record in the *Edited* state (the first state before the record enter the workflow)
2. A record in the *Published* state (the step into which the record will exit the workflow)

Select a Table

Workflows are defined separately for each table, therefore you need to select one of the tables defined in the **RDM Logical Model > Tables** node:

1. Go to **Workflows and Notifications > Tables** (right-click) **> New table...**
2. Select (Ctrl+Space) a table in the **Table name** attribute in the **General** tab.



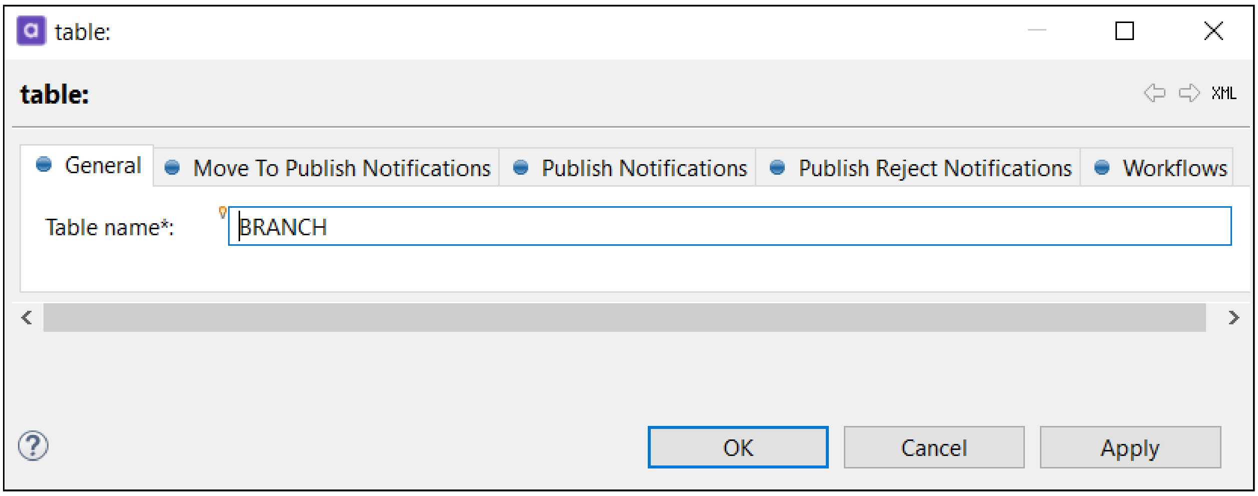


Table name*: BRANCH

Create a New Workflow

1. Switch to the **Workflows** tab.
2. Click **Add** (1) to add a new workflow, then double-click the number on its left (2) to enter its details.

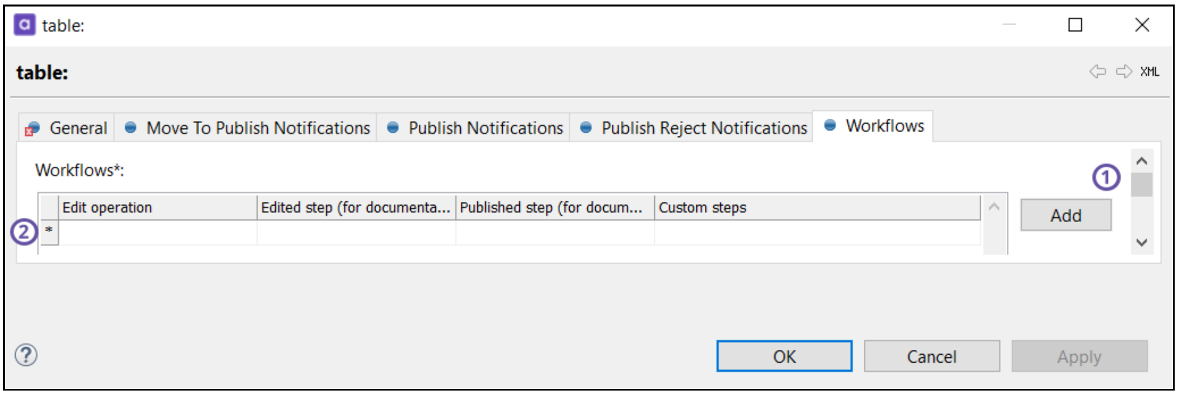


table:
table:
XML
General
Move To Publish Notifications
Publish Notifications
Publish Reject Notifications
Workflows
Workflows*:
①
Edit operation
Edited step (for documenta...
Published step (for docum...
Custom steps
②
*
Add
OK
Cancel
Apply

3. Fill out **Edit operation** field to define the type of action which will trigger the workflow; select from:
    - **Create**
    - **Delete**
    - **Update**

Add Approval Steps

The *Edited* and *Published* steps are pre-configured and will appear in the special fields when you are creating record workflows, to define the actual steps of your custom workflow:

1. Add a step and double-click the number on its left to enter its details. Use Ctrl+space to select from available statuses.

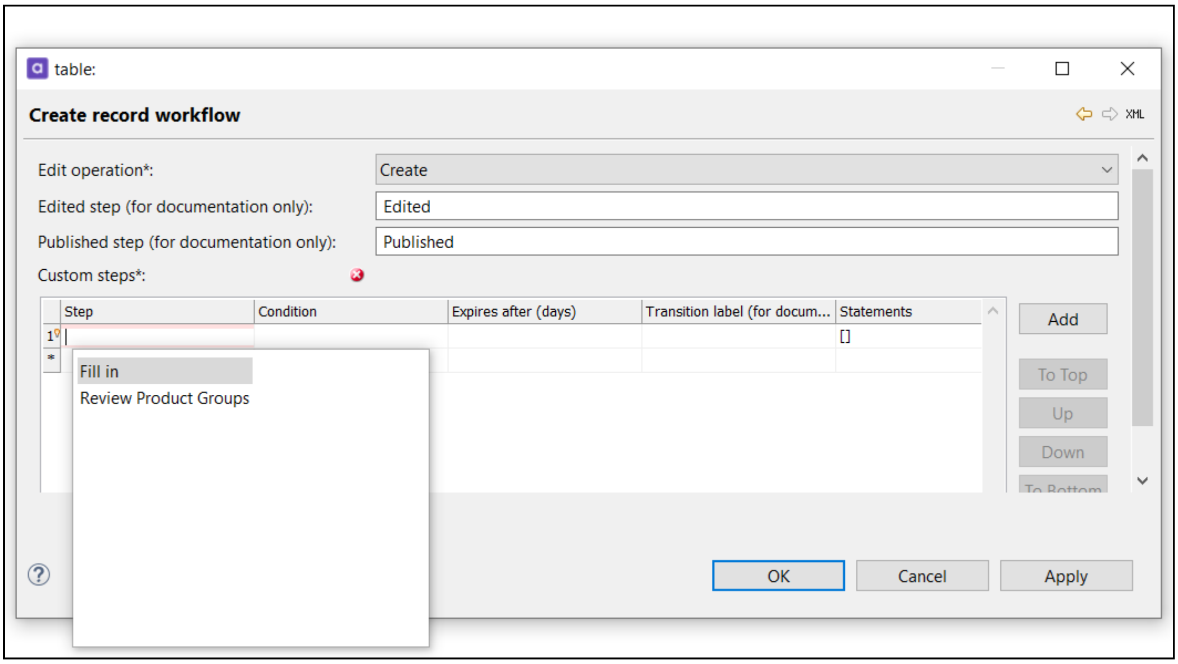


table:

Create record workflow

Edit operation*: Create

Edited step (for documentation only): Edited

Published step (for documentation only): Published

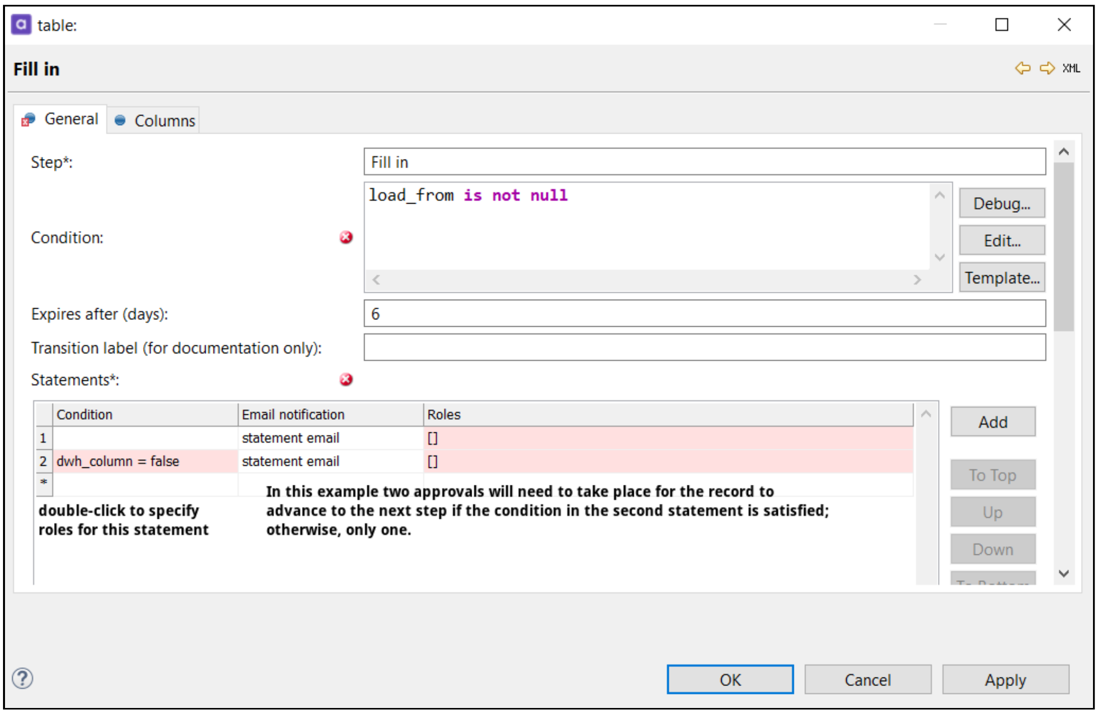
Custom steps*:

	Step	Condition	Expires after (days)	Transition label (for docum...	Statements
1					[]
*					

Fill in
Review Product Groups

2. Define step logic in the **General** tab:
    a. **General step characteristics and conditions:**

Name	Required	Description
Condition	N	The condition that the record must satisfy to enter the step. If the record does not satisfy the condition, it will be automatically moved to the next step. The condition uses Ataccama ONE expression syntax. See Expressions in the built-in Help. Besides using standard business attributes in the condition, you can also use usernames in conditions, e.g., you can let some users skip all or some of the workflow steps by defining a condition as follows: `meta.username <> 'admin'`.
Expires after (days)	N	After the specified number of days, the workflow automatically moves to the next step.
Transition label	N	Label of the transition used in the workflow diagram (see .Configuring RDM Workflows v11.2.0#How to Open the Workflow Diagram below). Will be filled after consequent steps are added, see .Configuring RDM Workflows v11.2.0#Connect the Steps and Add Transition Labels below.



Step*: Fill in

Condition:

```
load_from is not null
```

Expires after (days): 6

Transition label (for documentation only):

Statements*:

	Condition	Email notification	Roles
1		statement email	[]
2	dwh_column = false	statement email	[]
*			

double-click to specify roles for this statement

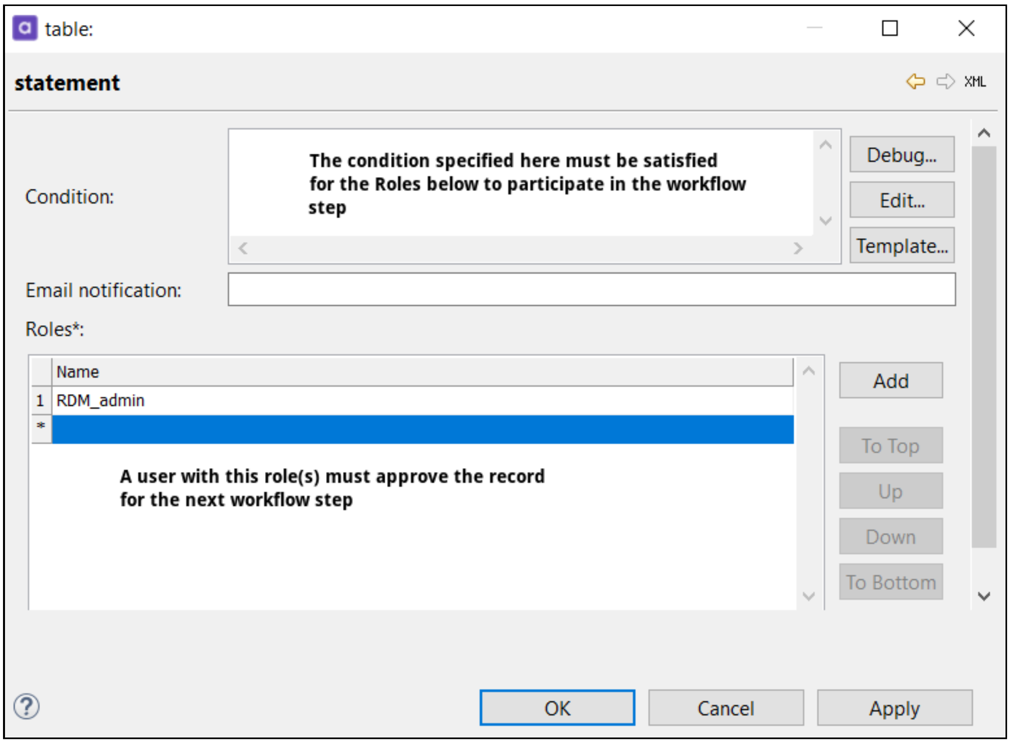
In this example two approvals will need to take place for the record to advance to the next step if the condition in the second statement is satisfied; otherwise, only one.

b. **Statements**

Name	Required	Description
Condition	N	Only records satisfying this condition will be available for approval to users with roles attribute. The condition uses Ataccama syntax. See Commonly Used Functions for the most common ONE functions.
Email notification	N	A previously defined email template to be sent when a record arrives to the current step.
Roles	N	Allows adding roles required to approve the record and move it to the next workflow step (users with these roles will receive an email notification).

Name	Required	Description
Name	N	A user with at least one of the roles listed in this column has to approve the change to move the record to the next workflow step.

The number of configured statements corresponds to the number of approvals which will need to take place to move a record to the next step if all conditions are satisfied.



The condition specified here must be satisfied for the Roles below to participate in the workflow step

A user with this role(s) must approve the record for the next workflow step

c. **Notifications**

Name	Required	Description
Email notification after finishing the step	N	A previously defined email template to be sent when a record moves to the next step. • **Email notification** - email template to be sent. • **Roles** - roles to which the email will be sent.
Reject email notification	N	A previously defined email template to be sent when a record is rejected in the current step (and moved to the previous step). • **Email notification** - email template to sent. • **Roles** - roles to which the email will be sent.

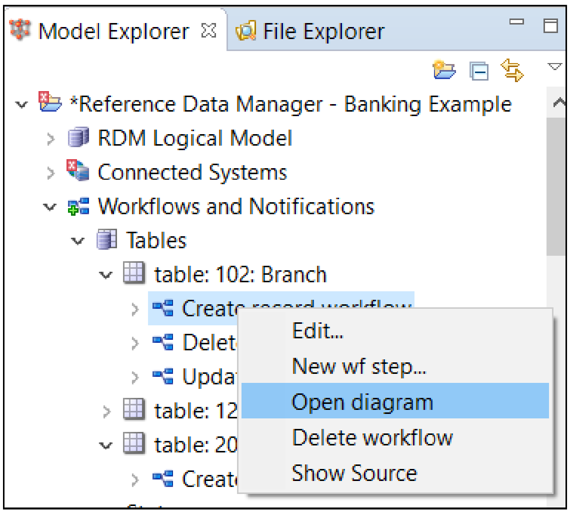
3. Switch to the **Columns** tab, specify which columns can be modified in this workflow step by step participants, apply changes, and go back to define the next workflow step:
4. Repeat this step after you have configured all approval steps for this workflow.

Connect the Steps and Add Transition Labels

Go back to the created steps and connect them with each other by filling in the **Transition label** attribute (optional). You can then control the workflow order by opening its diagram. See below.

2.11.4 How to Open the Workflow Diagram

The workflow diagram allows viewing the graphical representation of the workflow and getting a complete overview of the workflow.



Opening the Workflow Diagram

2.11.5 Creating RDM Email Templates

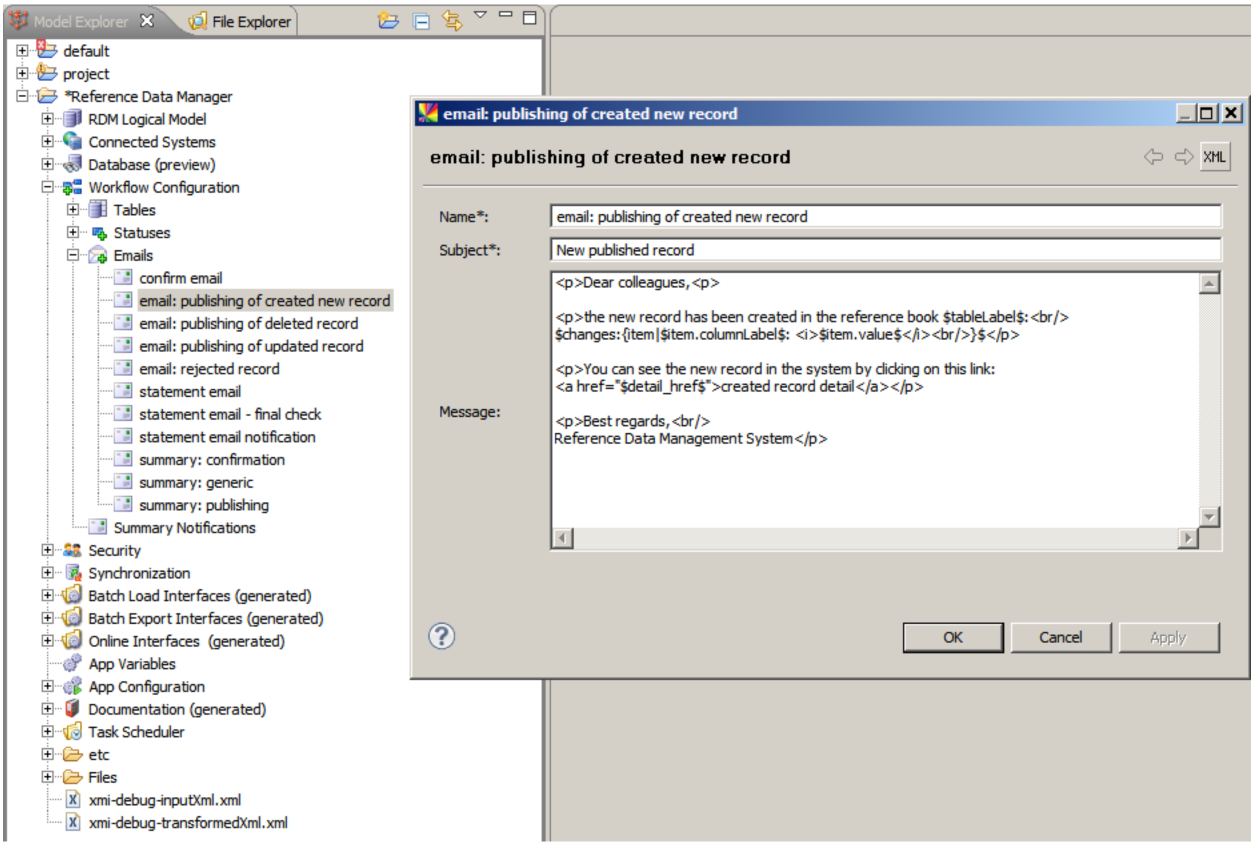
Email templates in RDM are used for table-specific record change notifications , workflows, and summary notifications. Email templates are defined in the **Workflow Configuration > Emails** sub-node.

How to Create an Email Template

1. Right-click **Emails > New email.**
2. Fill in the attributes:

Name	Required	Description
Name	Y	Name of the email template used in the configuration process
Subject	Y	Subject of the email displayed to the recipient
Message	N	Body of the email message in the HTML format. Message examples are provided in Creating RDM Email Templates below and can be used as is. For in-depth explanation of supported variables, see Creating RDM Email Templates further below.

3. Click **OK** to save changes.



Name*: email: publishing of created new record

Subject*: New published record

Message:

```
<p>Dear colleagues,<p>

<p>the new record has been created in the reference book $tableLabel$:<br/>
$changes:{item|$item.columnLabel$: <i>$item.value$</i><br/>}$</p>

<p>You can see the new record in the system by clicking on this link:
<a href="$detail_href$">created record detail</a></p>

<p>Best regards,<br/>
Reference Data Management System</p>
```

Email Template Example

Email Template Examples

Regular Notifications

Email Template	Corresponding Email Example
Sample Email Template - New Record Rejected	Dear colleagues, A record has been rejected in the reference book 102: Branch: Name: *Guelph test 4* Code: *guelph02* Branch Manager: *John Smith* Phone: *(516) 324-7777* City: *GUELPH* Address: *49 Kent Street* Valid From: *2013-11-01 00:00:00.0* You can see the new record in the system by clicking on this link: rejected record detail Best regards, Reference Data Management System

Sample Email Template - New Record Rejected

```
<p>Dear colleagues,<p>

<p>A record has been rejected in the
reference book $tableLabel$:<br/>
$columns:{item|$item.columnLabel$:
<i>$item.value$</i><br/>}$
username: $username$</p>

<p>You can see the rejected record in the
system by clicking on this link:
<a href="$detail_href$">rejected record
detail</a></p>
<p>Best regards,<br/>
Reference Data Management System</p>
```

Email Template | **Corresponding Email Example**

Sample Email Template - New Record Created

```
<p>Dear colleagues,<p>

<p>A new record has been created in the
reference book $tableLabel$:<br/>
$changes:{item|$item.columnLabel$:
<i>$item.value$</i><br/>}$</p>

<p>You can see the new record in the system
by clicking on this link:
<a href="$detail_href$">created record
detail</a></p>

<p>Best regards,<br/>
Reference Data Management System</p>
```

Dear colleagues,

A new record has been created in the reference book 102: Branch:
Name: *Guelph test 4*
Code: *guelph02*
Branch Manager: *John Smith*
Phone: *(516) 324-7777*
City: *GUELPH*
Address: *49 Kent Street*
Valid From: *2013-11-01 00:00:00.0*

You can see the new record in the system by clicking on this link: created record detail

Best regards,
Reference Data Management System

Sample Email Template - Record Edited

```
<p>Dear colleagues,</p>

<p>A record has been updated in the
reference book $tableLabel$:<br/>
$changes:{item|old value:
<i>$item.oldValue$</i> -> new value:
<i>$item.value$</i> for column
<i>$item.columnLabel$</i><br/>}$</i></p>

<p>The changes are now available in the
system by clicking on this link:
<a href="$detail_href$">updated record
detail</a></p>

<p>Best regards,<br/>
Reference Data Management System</p>
```

Dear colleagues,

A record has been updated in the reference book 102: Branch:
old value: *Augustin Noble* -> new value: *Vanessa Lewinsky* for column *Branch Manager*

The changes are now available in the system by clicking on this link: updated record detail

Best regards,
Reference Data Management System

Summary Notifications

Email Template	Corresponding Email Example
Sample Summary Notification (code below)	Dear colleagues, The following tables have been updated: • **TABLE1** - the number of changed records: 3 • NEW: 1 • CHANGED: 2 • **TABLE1** - the number of changed records: 2 • DELETED: 2 Best regards, Reference Data Management System

Sample Summary Notification

```
<p>Dear colleagues,<p>
<p>The following tables have been updated:<br/>
   <ul>
      $root.objects.tableName:{item|
<li><b>$item.object$</b> - the number of changed
records: $item.count$</li>
        <ul>
           $item.objects.state:{item1|
<li>$item1.object$: $item1.count$</li>}$
        </ul>
     </br>}$
   </ul>
</p>

<p>Best regards,<br/>
Reference Data Management System</p>
```

Message Variables

Email messages can contain information about affected tables and records.

Regular Notifications

Regular notifications (sent after each change) support the following variables:

Name	Description
Table label	Label of the table
Item	An enveloping variable to include change details (see examples below); *item1* can be used instead, x
Item.columnLabel	Column label of the specified table column.
Item.oldValue	Value of the item before the change.
Item.value	Value of the item after the change.
Username	User that made the change.

Name	Description
Change	A variable containing the list of changes.
Columns	A variable containing the list of columns.
Detail_href	The Hyperlink that leads to the record.
Environment	Application environment (such as DEV or PROD) as set in RDM Application Properties.

Summary Notifications

Summary Notifications support the following variables divided into two types:

1. **objects**:
    - **tableName** - `$root.objects.tableName:{information inside}$` - technical (database) name of the table; must be used at the beginning of the list of changes.
    - **state** - `$item.objects.state:{state information inside}$` - provides the record state of the table (NEW, CHANGED, DELETED) and is accessible within the **tableName** object
2. **attributes** - attributes of declared objects
    - **object** - either name or record state of the declared variable
    - **count** - number of affected records per state or overall for the table

Usage:

`$root.objects.tableName:{table_variable|$table_variable.object$}$` - writes names of affected tables

`$root.objects.tableName:{table_variable|$table_variable.count$}$` - writes counts of changes in affected tables

`$table_variable.objects.state:{count_variable|$count_variable.object$}$` - writes names of record states

`$table_variable.objects.state:{count_variable|$count_variable.count$}$` - writes counts of all record states

Syntax:

$** `object` **:{** `object_variable_declaration` **|$ `object_variable_call` **.** `attribute` **}$**

- **object** - either `$root.objects.tableName` or accessible within it `table_variable.objects.state`
- **object variable declaration** - variable name of the object (can be anything)
- **|** - a separator between object and object variable declaration and object references
- **object variable call** - calls a declared variable
- **attribute** - either **object** or **count** (explanation above)

2.11.6 Configuring RDM Email Notifications

RDM can be configured to send email notifications triggered by user actions. One kind of notifications is set up together with record change approval workflows (see Configuring RDM Workflows).

Two Types of Notifications

This article describes two other notification types:

- Workflow-independent email notifications sent out after each action like sending to publish, publishing, or rejecting publishing - Configuring RDM Email Notifications
- Summary emails notifications sent when an action causes the sending of notification messages exceeding the configured email number threshold - Configuring RDM Email Notifications.

> All email notifications are configured using previously defined email templates and (optionally) roles.

Notifications

Email notifications for actions performed on a given table are set up in the following three tabs of a given table (**Workflow Configuration > Tables > [table]**):

- **Confirmation notifications tab** - allows to assign email templates to be sent when a record has been sent to publish.
- **Publish notifications tab** - allows to assign email templates to be sent when a record has been published.
- **Publish Reject notifications tab** - allows to assign email templates to be sent when a record has been rejected.

Each of these tabs has the following sections:

- **Notification - create record** - allows to assign email templates and roles when the record has been created.
- **Notification - update record** - allows to assign email templates and roles when the record has been updated.
- **Notification - delete record** - allows to assign email templates and roles when the record has been deleted.

Each section has the following attributes:

- **Email** - email template to be used for a given notification.
- **Roles** - users with these roles will receive a given notification.

To add a new table, locate **Workflow Configuration > Tables**, right-click **Tables > New table...**

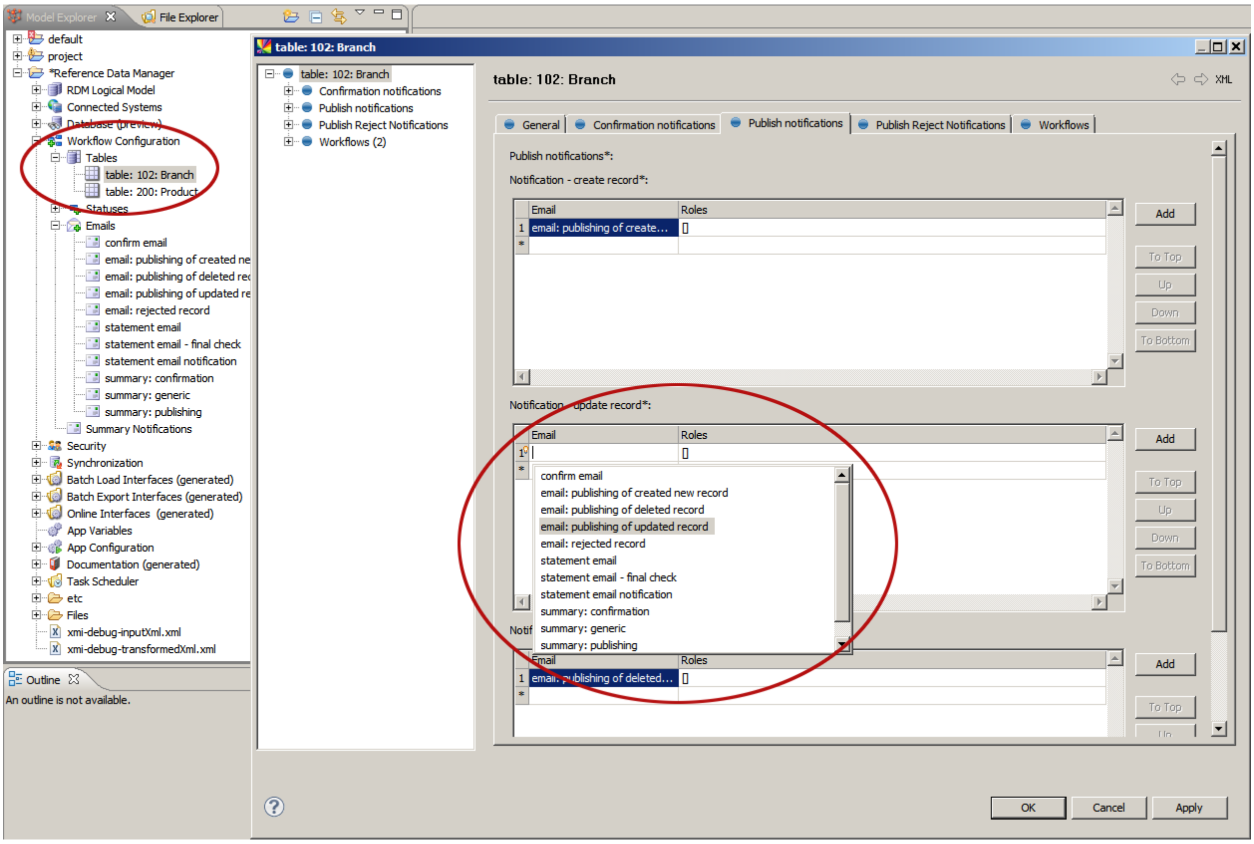


table: 102: Branch

Publish notifications*:

Notification - create record*:

	Email	Roles
1	email: publishing of create...	
*		

Notification - update record*:

	Email	Roles
1		
*		

confirm email
email: publishing of created new record
email: publishing of deleted record
email: publishing of updated record
email: rejected record
statement email
statement email - final check
statement email notification
summary: confirmation
summary: generic
summary: publishing

	Email	Roles
1	email: publishing of deleted...	
*		

Email Notification Configuration Example

Summary Notifications

When email notifications have been configured for a table, RDM will send one email for every change per each performed action. For example, if someone simultaneously publishes or approves 20 records in the workflow (to save their time) and email notifications triggered by step completion are set up, workflow participants will receive 20 emails. To avoid this, summary notifications can be set up in the **Summary Notifications** node.

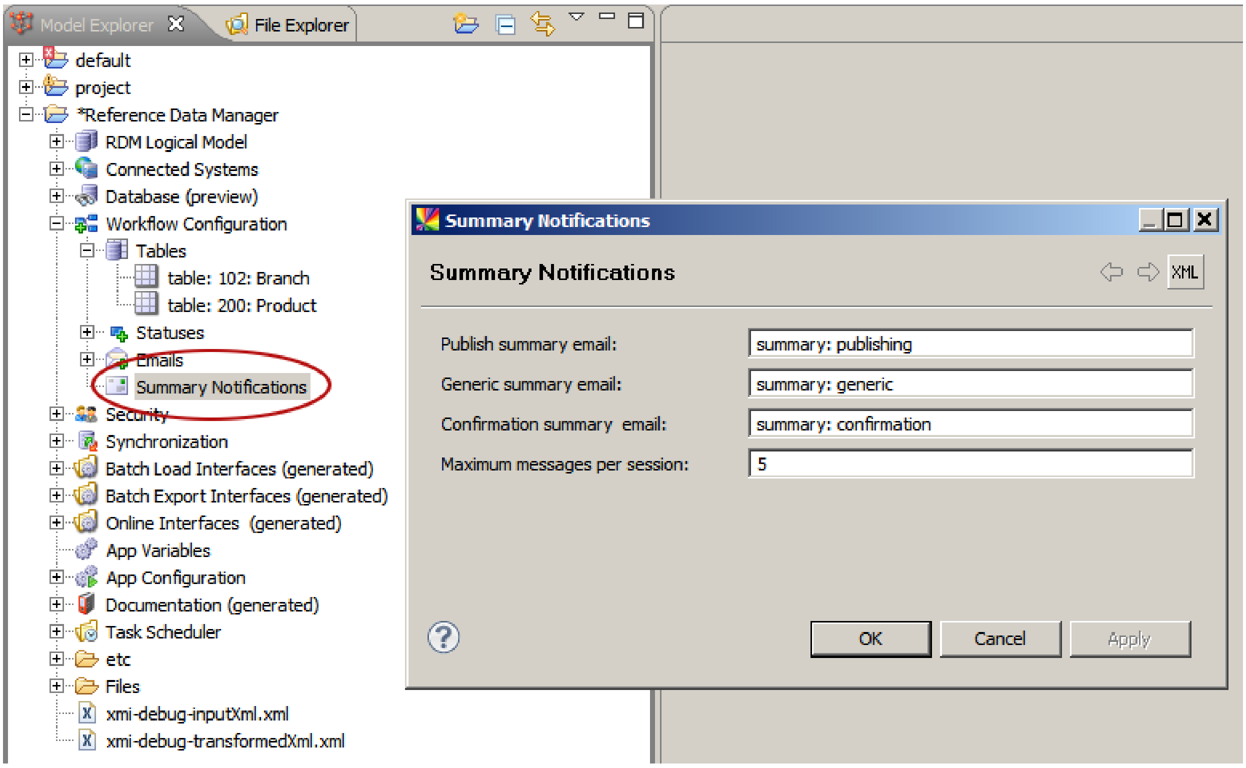
The **Summary Notifications** node allows setting a threshold value for the number of emails sent per action. If more records are published/moved in the workflow/sent to publish simultaneously, one "mass email" will be sent, notifying the recipient of the action taken on multiple records.

> Summary Notifications settings are applied to all tables created under **Workflow Configuration > Tables** that have any kind of email notifications set up.

The attributes of summary notifications are the following:

Name	Required	Description
Publish summary email	N	An email template for a mass publish action.
Generic summary email	N	An email template used when if one or both of the other summary emails are not configured.
Confirmation email summary email	N	An email template for a mass move to publish action.
Maximum messages per session	Yes if any of the above is defined	Defines the threshold for the "mass email" trigger.

> **Example**: if **Maximum messages per session** is set to 3, then if an RDM user performs an action which causes 4 or more changes, a corresponding mass email will be sent.



Summary Notifications

Publish summary email: summary: publishing
Generic summary email: summary: generic
Confirmation summary email: summary: confirmation
Maximum messages per session: 5

Summary Notifications Configuration Example

2.12 Scheduling RDM Tasks

Data exporting/synchronizing workflows can be easily scheduled by the native ONE Scheduler. Task schedulers are generated based on the model definition.

Scheduler is an ONE Runtime Server Development Guide component that allows scheduling and running generic jobs. Since the scheduler is an online server component, the Online Server must be running in order to start scheduled jobs.

Schedules for RDM tasks are defined in the **Task Scheduler** node of the RDM project.

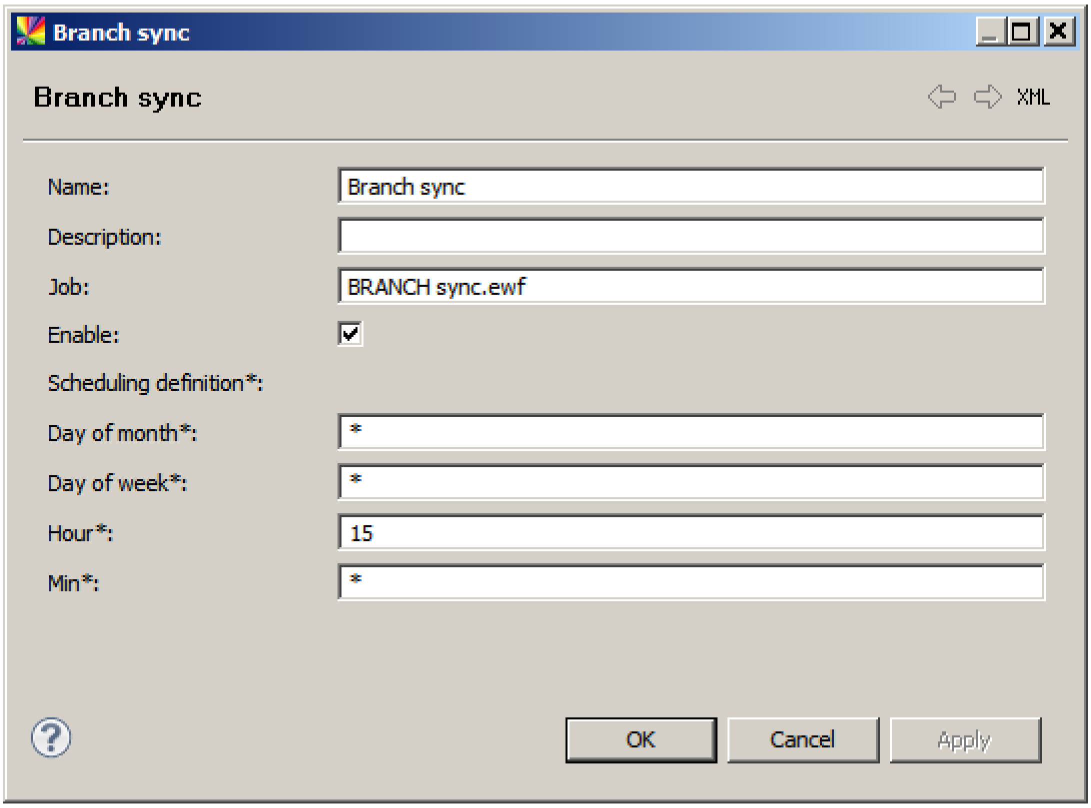
2.12.1 How to Set up a Scheduled Task

1. Right-click **Task Scheduler > New task...**
2. Fill in the attributes
3. Click **OK** to save changes
4. Right-click **[specific task name] > Generate schedule** – schedule definition is generated into **Files > schedulers**. Sample file contents are provided below.

To (re)generate all schedule definitions, click **Task Scheduler** and select **Generate schedules.**

Scheduler Task Example (.sch file)

```
<?xml version='1.0' encoding='UTF-8'?>
<scheduleDefinition>
    <description>test1</description>
    <enabled>true</enabled>
    <job class="com.ataccama.adt.scheduler.job.WorkflowJob">
        <workflow>RDM:BRANCH sync.ewf</workflow>
    </job>
    <scheduling>* 15 * *</scheduling>
</scheduleDefinition>
```



Name:	Branch sync
Description:	
Job:	BRANCH sync.ewf
Enable:	☑
Scheduling definition*:	
Day of month*:	*
Day of week*:	*
Hour*:	15
Min*:	*

Scheduler Task Definition

2.12.2 Scheduled Task Attributes

Name	Required	Attribute
Name	Y	Name of the scheduled task.
Description	N	Description of what the task does.
Job	Y	Name of the job to be scheduled (selected from previously defined).
Enable	Y	If checked, the `.sch` file is generated.
Scheduling definition	Y	Definition of the schedule: • **Day of month** - day of the month when the job should be run (e.g., 16 means on the 16th day of every month) • **Day of week** - day of the week when the job should be run (e.g., 4 means every Thursday) • **Hour** - hour of the day when the job should be run (e.g., 17 means at 5pm) • **Min** - minute of the hour when the job should be run (e.g., 30 means at 30 minutes of the specified hour)

An asterisk * used for the definitions above means "every," e.g., when used in the **Hour** and **Day of week** attributes, the task will be run every hour or every day.

See Scheduling Definition for details and options of the configuration.

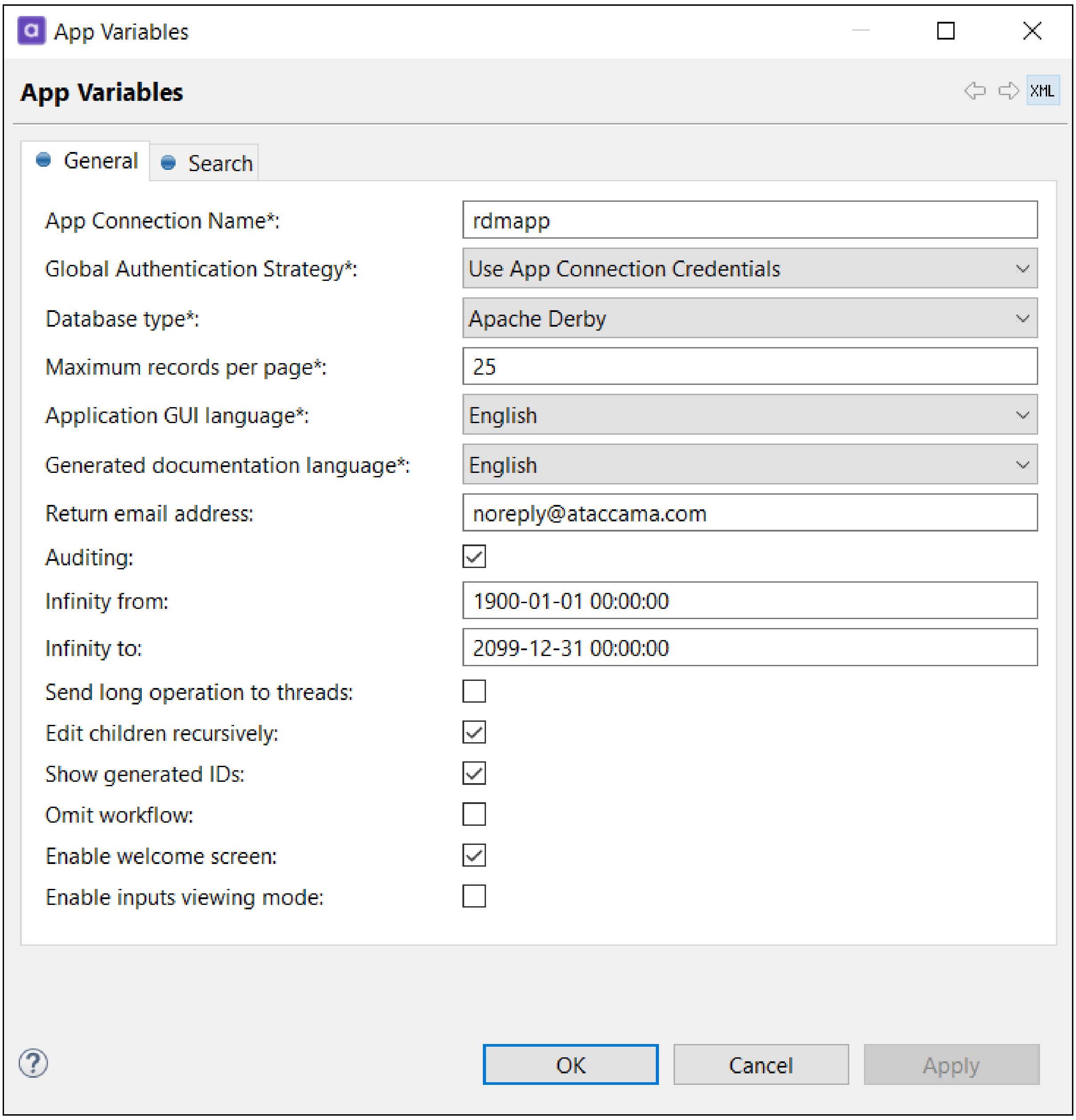
2.13 RDM App Variables

The **App Variables** node allows setting global settings for the RDM web application and repository:

Name	Required	Description
App Connection Name	Y	The name of the server resource where RDM is running. The URL from this connection will be inherited by various RDM steps used in generated synchronization, batch load, and batch export plans. See Adding the RDM Server Resource for instructions on configuring a server connection for RDM.
Global Authentication Strategy	Y	Determines the default authentication strategy for load and synchronization plans: • **Use App Connection Credentials** (default)- the plans will be pre-configured to use the username nad password used in the RDM server connection. • **Pass Credentials Manually** - the plans will be pre-configured to expect an external data resource (e.g., a text file) containing the username and password. You can override these settings in local table and synchronization configurations.
Database type	Y	The database type used for the RDM repository.
Maximum records per page	Y	Specifies the maximum number of records shown on one page in the RDM web application as a default setting for all users.

Name	Required	Description
Application GUI language	Y	Specifies the language for properties used in the RDM web application as a default setting for all users. Available languages: English, Czech, Russian, and German.
Generated documentation language	Y	Specifies the language for generated documentation.
Return email address	N	The email address displayed in the *From* line of received notification email messages.
Auditing	N	True/false, specifies whether the RDM web application provides audit logs for defined operations. If set to true, the `logging.xml` file will be created upon generating web application configuration. See Setting up RDM Auditing for more information.
Infinity from	N	Minimum possible value for a FROM business date. See Business Date Columns.
Infinity to	N	Maximum possible value for a TO business date. See Business Date Columns.
Send long operations to threads	N	When set to true, the RDM web application will perform long-lasting operations (e.g., publishing thousands of records) in the background, and the user will be able to close the browser tab with RDM without canceling the operation. This feature is not supported on WebSphere application servers and should be set to *false.*

Name	Required	Description
Edit children recursively	N	When set to true, the "Edit children recursively" option is set to true by default in the webapp.
Show generated IDs	N	When set to true, columns with generated primary/group IDs are displayed in tables by default (users can override this option in the web application).
Omit Workflow	N	When set to true, no workflow takes place. After saving, a record is immediately moved to the Publish section.
Enable Welcome Screen	N	When set to true, welcome screen with the entity list after every log in.
Enable Inputs Viewing Mode	N	When set to true, user is able to see INPUTS perspective in webapp to import and merge records from external databases to RDM.



App Connection Name*: rdmapp
Global Authentication Strategy*: Use App Connection Credentials
Database type*: Apache Derby
Maximum records per page*: 25
Application GUI language*: English
Generated documentation language*: English
Return email address: noreply@ataccama.com
Auditing: ☑
Infinity from: 1900-01-01 00:00:00
Infinity to: 2099-12-31 00:00:00
Send long operation to threads: ☐
Edit children recursively: ☑
Show generated IDs: ☑
Omit workflow: ☐
Enable welcome screen: ☑
Enable inputs viewing mode: ☐

2.13.1 Adding the RDM Server Resource

As part of the RDM project configuration, it is necessary to a remote server resource pointing to the URL where RDM is running. This definition can be reused for automatic authentication to the application when running synchronization or initial load plans. The URL of the server is also used to construct the links to changed records in case email notifications have been configured.

To create a server resource:

1. In ONE Desktop, switch to the File Explorer view.

2. Right-click **Servers > New Server** and configure the connection:
   - The URL should correspond to the URL at which the RDM web application is accessible.
   - The user should be someone with at least viewing rights.

   > Generally, for the sake of convenience, we recommend to create a separate "technical" user with read-write rights to all tables.
   >
   > The drawback of this approach is that SOAP services will be all available to anyone who the URL without having to pass any credentials.

3. Click **Finish** to save your changes. The server is configured.
4. Use the server connection name in RDM App Variables.

For the detailed instructions on configuring server connections in ONE Desktop, see Creating a New Server Connection.

2.13.2 Configuring RDM Stored Filters

When using RDM, the filter can load pre-configured queries for ease of access in addition to its ability of saving the filters directly in the web application.

1. In the **App Variables** node, switch to the **Search** tab.
2. To create a new search definition, click **Add** in the search panel.
3. Click the row number of the new search filter to open the editing panel.
4. By selecting **Enable**, your search definitions will appear in the My Saved Filters section of the filters.
5. Enter the Name which will appear in the filter.
6. Click **Ctrl+Space** to get a list of entities and select the one you wish to configure the filter for.
7. Select **Join by OR** if you wish to use an OR operator. The default setting uses the AND operator.
8. If the basic filtering is not enough, it is possible to write an advanced condition in the **Advanced SQL Condition** column.

   > Advanced filtering also allows to refer to default columns, like **[State]** or **[Valid]**, using the same syntax as in the webapp advanced filters, eg. `"[ State ]" = 'Edited'`.

9. Next, under **Columns**, add columns that will be available to users in this search definition. **Ctrl+Space** to get a list of available columns.

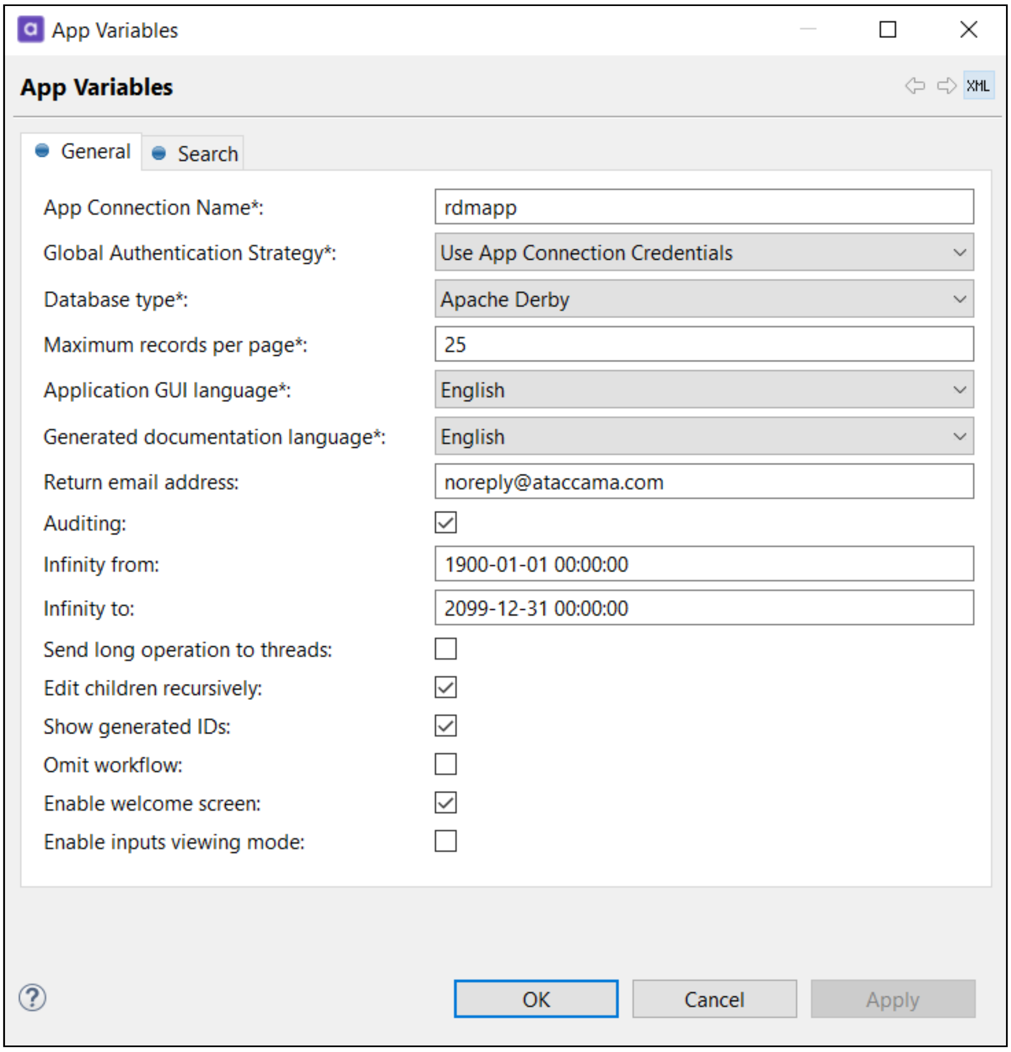
10. Double click the **Default Operators** column to get a list of all available operators.
11. Enter the **Default Value** you wish to filter by.
12. Finally, if you wish to make searching case sensitive, check the **Case Sensitive** box.
13. Click **Apply**.
14. Click the back arrow and repeat steps 2-13 to add another search definition.

2.14 Setting up RDM Auditing

RDM allows auditing user actions in the web application and logging them to a text file.

Prerequisite

Before setting up RDM Auditing, ensure **Auditing** is enabled in RDM App Variables.



App Connection Name*: rdmapp
Global Authentication Strategy*: Use App Connection Credentials
Database type*: Apache Derby
Maximum records per page*: 25
Application GUI language*: English
Generated documentation language*: English
Return email address: noreply@ataccama.com
Auditing: ☑
Infinity from: 1900-01-01 00:00:00
Infinity to: 2099-12-31 00:00:00
Send long operation to threads: ☐
Edit children recursively: ☑
Show generated IDs: ☑
Omit workflow: ☐
Enable welcome screen: ☑
Enable inputs viewing mode: ☐

2.14.1 Auditing Node

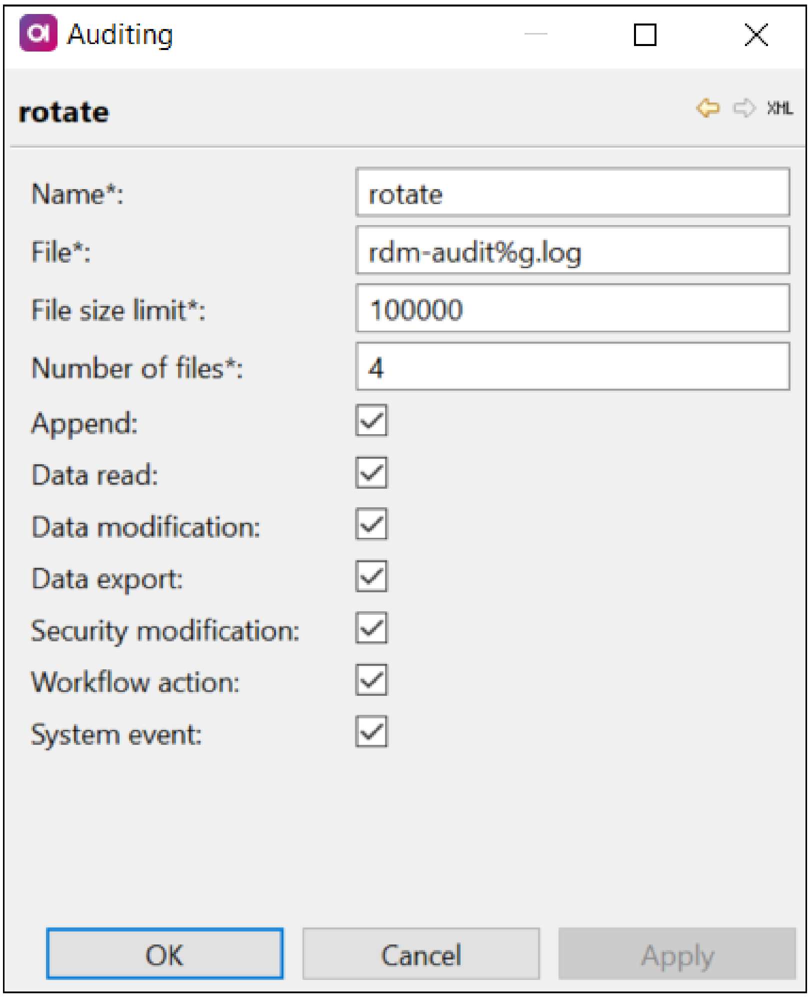
Auditing in configured in the **Auditing** node with the following configuration fields:

Name	Required	Description
File	Y	Logging file name pattern; both absolute and relative paths can be used (the path is relative to the folder, from which the web server is started). The following special characters can be used in the file name pattern: • `/` - local pathname separator • `%t` - system temporary directory • `%h` - value of the "user.home" system property • `%g` - generation number to distinguish rotated logs • `%u` - a unique number to resolve naming conflicts • `%%` - translates to a single percent sign "%"
File Size Limit	Y	The maximum file size in bytes before the log is rotated.
Number of files	Y	Total number of log files to keep.
Append	N	If true, the log records are appended to the log file.
Data read	N	A user opens a table or record details, refreshes contents of the table by applying a filter, etc.
Data modification	N	A user creates, edits, deletes, sends to publish, or publishes a record.
Data export	N	A user exports data (bulk export or dump).
Security modification	N	Permissions are changed.
Workflow action	N	A record goes through a workflow state.

Name	Required	Description
System event	N	User login or logout.

See the full list of auditable actions per category below.

2.14.2 Configuration Example



rotate

Name*: rotate
File*: rdm-audit%g.log
File size limit*: 100000
Number of files*: 4
Append: ☑
Data read: ☑
Data modification: ☑
Data export: ☑
Security modification: ☑
Workflow action: ☑
System event: ☑

Sample Auditing Configuration

2.14.3 Full List of Auditable Actions

Data Read

Action Code	Description
READ_TABLES_CHANGES	a user reads the **Change Log**

Action Code	Description
READ_ENTITY_DESCRIPTION	a user clicks the **Description** button
FIND_ROWS	a user opens a table
FIND_ROW_DETAIL	a user opens record details

Data Read Examples

```
/* action logged when a user opens a table to see the content of the table (the
same information is logged whenever user refreshes the content of the table, e.g.,
applies a filter, etc.): */
Aug 24, 2015 5:06:28 PM [Data read][FIND_ROWS]
INFO: {"ATTRIBUTES":{"Filter":{},"Count":30,"Offset":0,"EntityName":"PRODUCTS"},"R
EMOTE_ADDR":"172.16.10.116","OPERATION":"FIND_ROWS","DATE":"Mon Aug 24 17:06:28
CEST 2015","TYPE":"Data read","USER":"alice"}

/* action logged when a user clicks on the "Description" link */
Aug 24, 2015 5:06:27 PM [Data read][READ_ENTITY_DESCRIPTION]
INFO: {"ATTRIBUTES":{"EntityName":"PRODUCTS"},"REMOTE_ADDR":"172.16.10.116","OPERA
TION":"READ_ENTITY_DESCRIPTION","DATE":"Mon Aug 24 17:06:27 CEST 2015","TYPE":"Dat
a read","USER":"alice"}

/*action logged when a user opens a record detail */
Aug 24, 2015 5:19:54 PM [Data read][FIND_ROW_DETAIL]
INFO: {"ATTRIBUTES":{"RowId":2,"EntityName":"PRODUCTS"},"REMOTE_ADDR":"172.16.10.1
16","OPERATION":"FIND_ROW_DETAIL","DATE":"Mon Aug 24 17:19:54 CEST 2015","TYPE":"D
ata read","USER":"alice"}
```

Data Modification

Action Code	Description
MODIFY_TABLES_REJECT_ROWS	a user rejects record publishing.
MODIFY_MOVE_WORKFLOW_EXPIRED_ROWS	a record moves to a different workflow state after staying untouched for longer than configured maximum number of days.
MODIFY_IMPORT_TABLES	a user imports data via **Load from dump.**
MODIFY_TABLES_CONFIRM_ROWS	a user publishes changes to a record.
MODIFY_RETURN_TO_EDIT	a user clicks **Return to edit** on a record with unpublished changes.

Action Code	Description
MODIFY_MOVE_TO_CONFIRMATION	a user clicks **Move to publish** on a record with unpublished changes.
MODIFY_DELETE_ROWS	a user deletes a record.
MODIFY_UNDO	a user clicks **Undo** on a record with unpublished changes.
MODIFY_CREATE_ROW	a user creates a record.
MODIFY_EDIT_ROW	a user edits a record.
IMPORT_DATA	a user imports data via **Bulk Import** or import plan using the Rdm Importer step.

Data Modification Examples

```
/* action logged when a user edits a record */
Aug 24, 2015 5:26:12 PM [Data modification][MODIFY_EDIT_ROW]
INFO: {"ATTRIBUTES":{"Recursive":false,"RowId":2,"EntityName":"PRODUCT_TYPE"},"REM
OTE_ADDR":"172.16.10.116","OPERATION":"MODIFY_EDIT_ROW","DATE":"Mon Aug 24
17:26:12 CEST 2015","TYPE":"Data modification","USER":"alice"}

/* action logged when a user clicks Undo (e.g., after editing a record) */
Aug 24, 2015 5:22:47 PM [Data modification][MODIFY_UNDO]
INFO: {"ATTRIBUTES":{"Filter":{"IDS_IN":"(7)"},"EntityName":"PRODUCT_TYPE"},"REMOT
E_ADDR":"172.16.10.116","OPERATION":"MODIFY_UNDO","DATE":"Mon Aug 24 17:22:47 CEST
2015","TYPE":"Data modification","USER":"alice"}

/* action logged when a user moves a record into the confirmation state (moves to
publish) */
Aug 24, 2015 5:24:47 PM [Data modification][MODIFY_MOVE_TO_CONFIRMATION]
INFO: {"ATTRIBUTES":{"Filter":{"IDS_IN":"(3)"},"EntityName":"PRODUCT_TYPE"},"REMOT
E_ADDR":"172.16.10.116","OPERATION":"MODIFY_MOVE_TO_CONFIRMATION","DATE":"Mon Aug
24 17:24:47 CEST 2015","TYPE":"Data modification","USER":"alice"}
```

Data Export

Action Code	Description
EXPORT_TAGS	a user creates a dump
EXPORT_ENTITY	a user exports table data via **Bulk Export**

Action Code	Description
RDM_SYNCHRONIZE_EXPORT	a synchronization plan is run

Data Export Examples

```
/* action logged when a user exports data from a table */
Aug 24, 2015 5:21:09 PM [Data export][EXPORT_ENTITY]
INFO: {"ATTRIBUTES":{"Filter":{},"ExportType":"LABELS","EntityName":"PRODUCTS"},"R
EMOTE_ADDR":"172.16.10.116","OPERATION":"EXPORT_ENTITY","DATE":"Mon Aug 24
17:21:09 CEST 2015","TYPE":"Data export","USER":"alice"}
```

Security Modification

Action Code	Description
SEC_DELETE_ROLES	a role is deleted.
SEC_CREATE_ROLE	a role is created.
SEC_ASSIGN_ROLES_TO_USER	a role is assigned to a user.
SEC_REMOVE_ROLES_FROM_USER	a role is removed from a user.
SEC_ASSIGN_ROLE_TO_ENTITY	a role is given rights to a table.
SEC_REMOVE_ROLE_FROM_ENTITY	a role has rights to a table removed.
SEC_ASSIGN_ROLE_TO_COLUMN	a role is given rights to a column.
SEC_REMOVE_ROLE_FROM_COLUMN	a role has rights to a column removed.

Security Modification Examples

```
/* logged when a user assigns view (read only) permission to role RDM_ADMIN on
table PRODUCT_TYPE */
Aug 24, 2015 5:27:53 PM [Security modification][SEC_ASSIGN_ROLE_TO_ENTITY]
INFO: {"ATTRIBUTES":{"RoleType":"V","Role":["RDM_ADMIN"],"EntityName":"PRODUCT_TYP
E"},"REMOTE_ADDR":"172.16.10.116","OPERATION":"SEC_ASSIGN_ROLE_TO_ENTITY","DATE":
"Mon Aug 24 17:27:53 CEST 2015","TYPE":"Security modification","USER":"alice"}

/* logged when a user assigns view(read only) permission to role RDM_ADMIN on
column DESCRIPTION on table PRODUCT_TYPE */
Aug 24, 2015 5:28:05 PM [Security modification][SEC_ASSIGN_ROLE_TO_COLUMN]
INFO: {"ATTRIBUTES":{"Columns":["DESCRIPTION"],"RoleType":"V","Role":"RDM_ADMIN","
EntityName":"PRODUCT_TYPE"},"REMOTE_ADDR":"172.16.10.116","OPERATION":
"SEC_ASSIGN_ROLE_TO_COLUMN","DATE":"Mon Aug 24 17:32:41 CEST 2015","TYPE":"Securit
y modification","USER":"alice"}
```

Workflow Action

Action Code	Description
WF_STATE_CHANGE	a record is moved to a different workflow state

Workflow Action Example

```
/* logged when a user moves a record within a workflow */
Aug 25, 2015 9:14:18 AM [Workflow action][WF_STATE_CHANGE]
INFO: {"ATTRIBUTES":{"WfAction":"APPROVE","WfSystemFields":{"load_from":"2015-05-2
5 00:00:00.0","dwh_column":"true"},"Comment":{},"RowId":196,"EntityName":"BRANCH"}
,"REMOTE_ADDR":"127.0.0.1","OPERATION":"WF_STATE_CHANGE","DATE":"Tue Aug 25
09:14:18 CEST 2015","TYPE":"Workflow action","USER":"alice"}
```

System Event

Action Code	Description
USER_LOGON	a user logs in.
USER_LOGOUT	a user logs out.

System Event Examples

```
/* logged when a user is logged in */
Aug 24, 2015 5:02:22 PM [System event][USER_LOGON]
INFO: {"REMOTE_ADDR":"172.16.10.116","OPERATION":"USER_LOGON","DATE":"Mon Aug 24
17:02:22 CEST 2015","TYPE":"System event","USER":"alice"}

/* logged when the user logged out */
Aug 24, 2015 5:05:43 PM [System event][USER_LOGOUT]
INFO: {"REMOTE_ADDR":"172.16.10.116","OPERATION":"USER_LOGOUT","DATE":"Mon Aug 24
17:05:43 CEST 2015","TYPE":"System event","USER":"alice"}
```

2.15 How to Deploy an RDM Web App Configuration

After defining the RDM model for the first time or making changes to the existing one, the model configuration needs to be deployed to the RDM web application.

> For compatibility reasons, the RDM web application is initially loaded with a default empty configuration that needs to be replaced by following the steps below.

2.15.1 Quick Facts

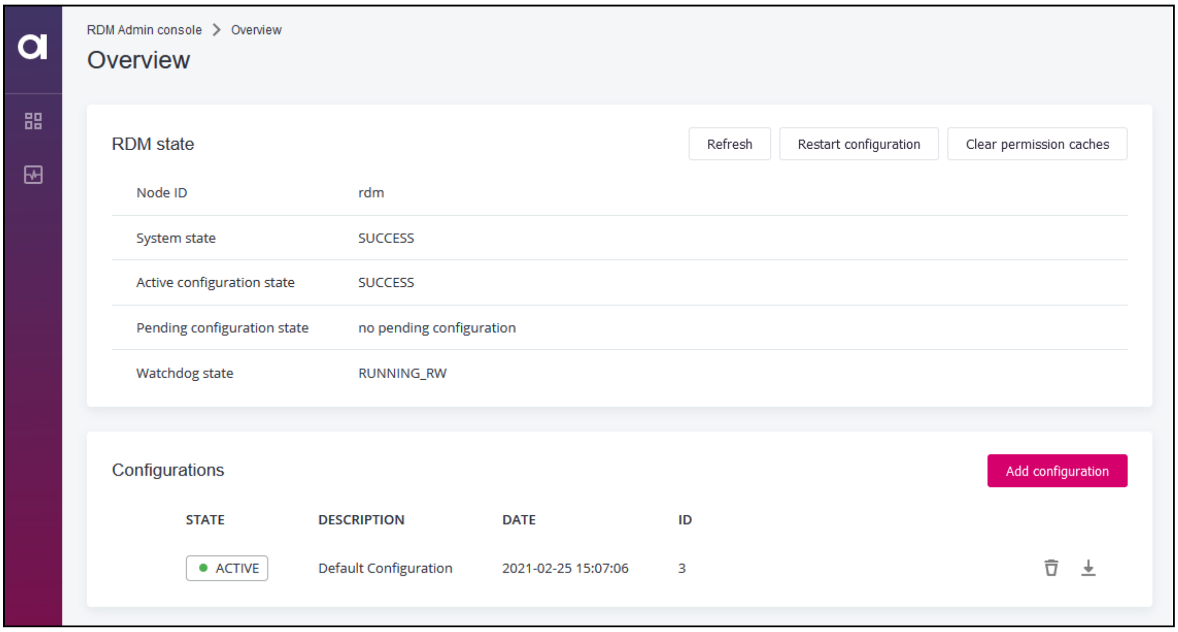
1. When a new configuration is being deployed, the RDM web application is not available.
2. When a new configuration is being deployed, the RDM web application performs all configured validations on all data in all tables.
3. When a new configuration is being deployed, the RDM web application checks the structure of the RDM Repository and compares it to the model currently being deployed.
4. Generally unavailability time depends on the following factors:
    - Application server hardware and memory
    - Database server hardware and memory
    - Network capacity and availability
    - The number of records in tables
    - The number of columns
    - The number of relationships and their depth (parent > child 1 > child 2 >...)
    - The number of mn-references
    - The number and complexity of validations (SQL validations are generally faster than online (ONE plans/components) validations)

2.15.2 The Procedure

1. In ONE Desktop, right-click the **App Configuration** node and select **Generate...**
2. Keep the **Default location** option selected (default location is the `etc` folder) and click **Generate**. A `configurationFiles.zip` file has been generated.
3. Open your browser.
4. Navigate to the **RDM Admin console** (`http://{rdm_url}:8060/admin`).

   > Do not delete the active configuration – it is needed when deploying a new configuration in the following steps. Note that even after the new configuration is made active, there is no need to delete the old one.
   >
   > Likewise, do not delete the configuration even if the configuration state changes to `FAILURE`. Instead, resolve the issue in the configuration manually or generate a new one and upload the configuration again.

5. Click **Add configuration**.

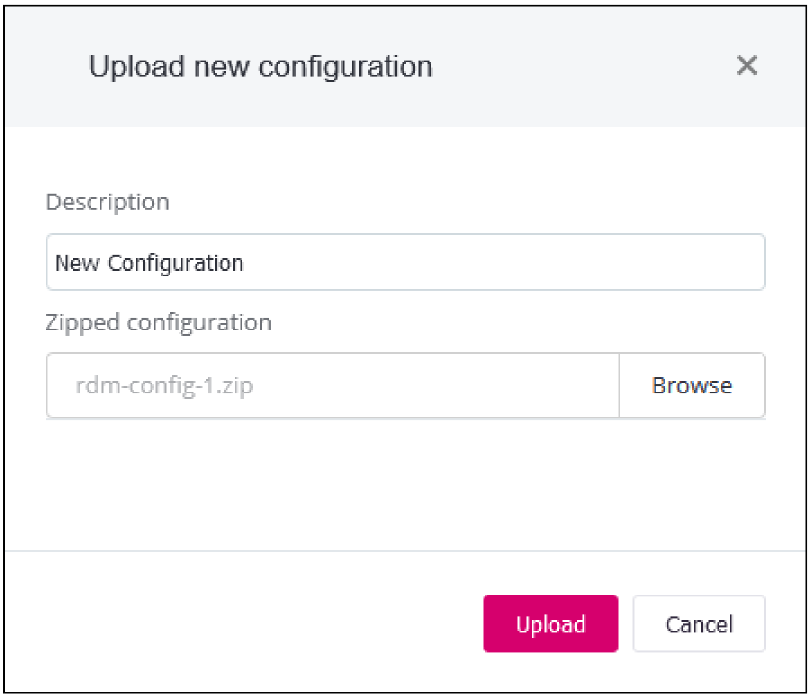


Node ID	rdm
System state	SUCCESS
Active configuration state	SUCCESS
Pending configuration state	no pending configuration
Watchdog state	RUNNING_RW

STATE	DESCRIPTION	DATE	ID
● ACTIVE	Default Configuration	2021-02-25 15:07:06	3

6. In the **Upload new Configuration** dialog:
   a. Enter Description.
   b. Select the `configurationFiles.zip` archive.

c. Click **Upload**.

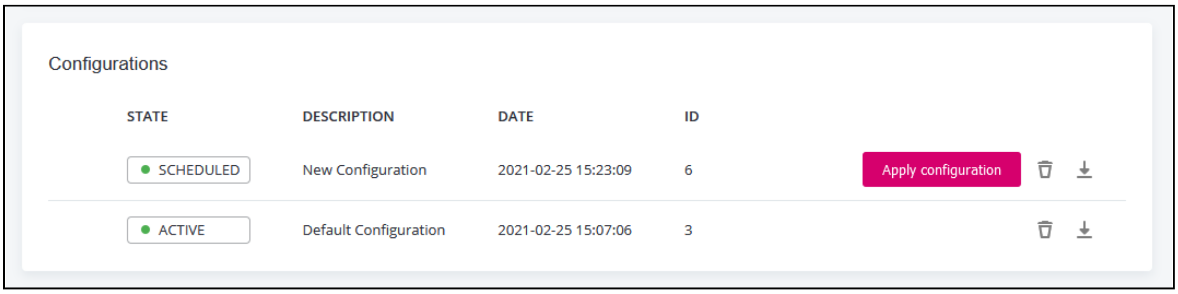


Upload new configuration

Description: New Configuration

Zipped configuration: rdm-config-1.zip

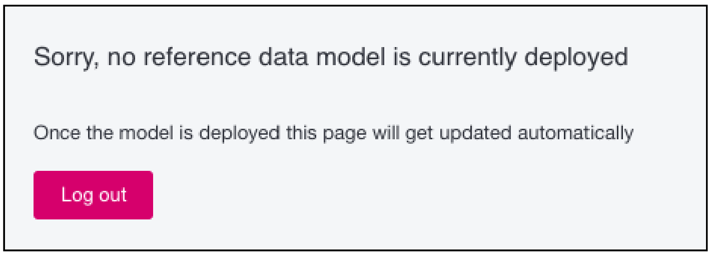
7. Click **Apply configuration** and confirm your selection in the pop-up window.



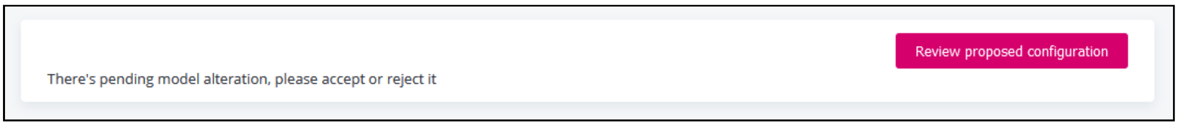
Configurations

STATE	DESCRIPTION	DATE	ID
SCHEDULED	New Configuration	2021-02-25 15:23:09	6
ACTIVE	Default Configuration	2021-02-25 15:07:06	3

Configuration is restarted and the old (active) and new (scheduled) configurations are compared.

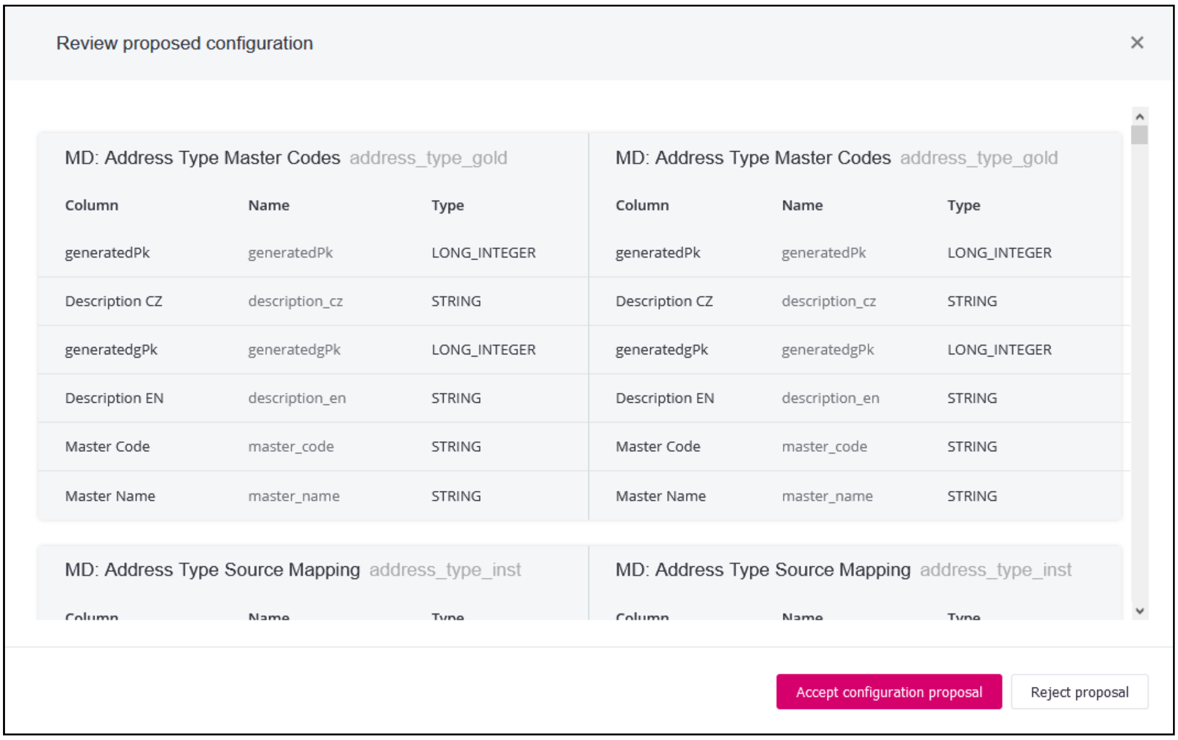
> Note that if you try to access the RDM application at this time, an error message will appear:
>
> 
>
> Sorry, no reference data model is currently deployed
>
> Once the model is deployed this page will get updated automatically
>
>

8. Click **Review proposed configuration** to view the comparison.



There's pending model alteration, please accept or reject it

9. In the pop-up window, review changes to the model and click **Accept configuration proposal**.



MD: Address Type Master Codes address_type_gold

Column	Name	Type
generatedPk	generatedPk	LONG_INTEGER
Description CZ	description_cz	STRING
generatedgPk	generatedgPk	LONG_INTEGER
Description EN	description_en	STRING
Master Code	master_code	STRING
Master Name	master_name	STRING

MD: Address Type Master Codes address_type_gold

Column	Name	Type
generatedPk	generatedPk	LONG_INTEGER
Description CZ	description_cz	STRING
generatedgPk	generatedgPk	LONG_INTEGER
Description EN	description_en	STRING
Master Code	master_code	STRING
Master Name	master_name	STRING

MD: Address Type Source Mapping address_type_inst

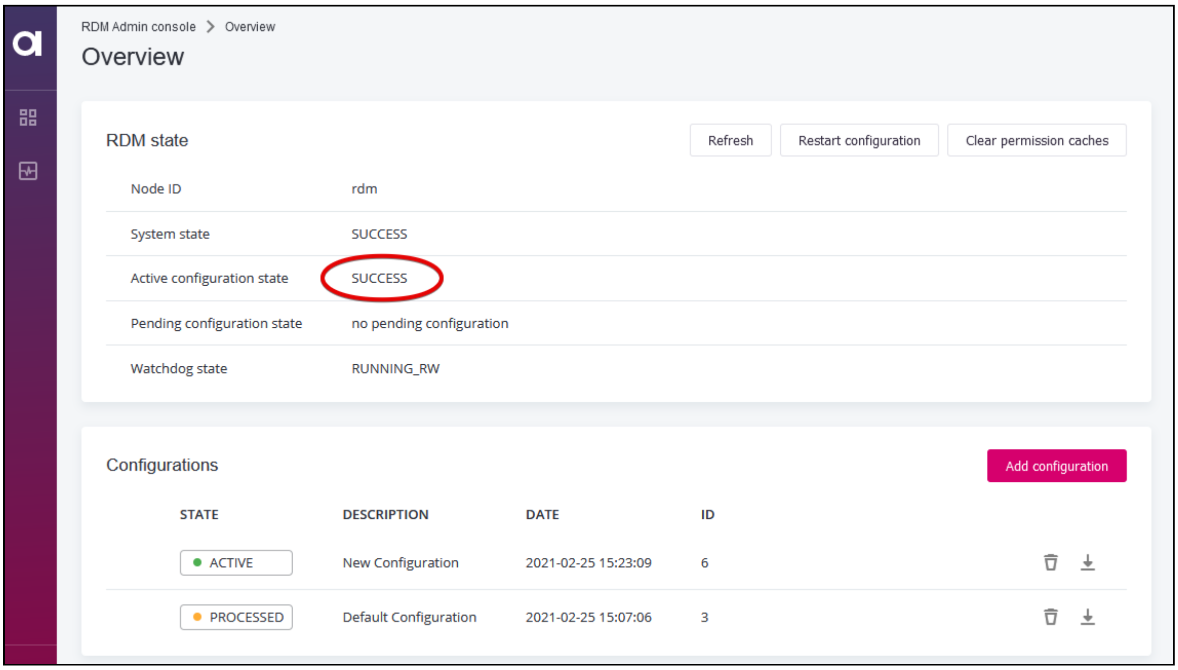
Column Name Type

MD: Address Type Source Mapping address_type_inst

Column Name Type

> If the new configuration is in any way unsuitable, the **Reject proposal** button is the last chance to safely delete it and start the process anew. Once accepted and made active, we do not advise deleting a configuration, even if the configuration state changes to `FAILURE`. For example, this can occur if the configuration contains an SQL validation referencing a non-existing column.

10. Back on the **Overview** screen, the **Active configuration state** section will show **Success**. This means the application is ready to be used again.



Overview

RDM state

Node ID	rdm
System state	SUCCESS
Active configuration state	SUCCESS
Pending configuration state	no pending configuration
Watchdog state	RUNNING_RW

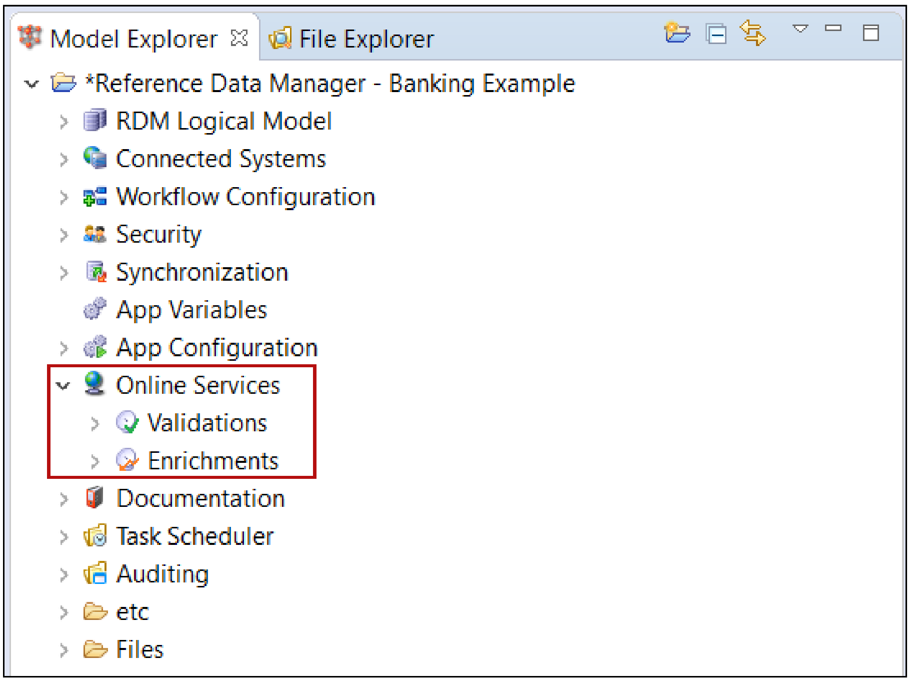
Configurations

STATE	DESCRIPTION	DATE	ID
● ACTIVE	New Configuration	2021-02-25 15:23:09	6
● PROCESSED	Default Configuration	2021-02-25 15:07:06	3

2.16 Configuring RDM Online Services

It is possible to create and configure web services that perform validation or enrichment of records when a user is editing record values in the application.

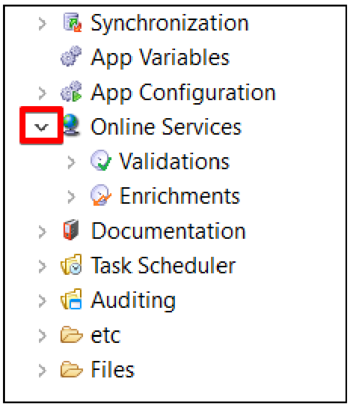
Web services are situated in the online service node and can be easily created from ONE plans and components. The resulting `.online` file and the underlying component, together with its constituent files (e.g., lookups), are placed to the server with ONE Desktop.



2.16.1 Step 1 Configure the Service

The first step to configuring RDM online services is creating a validation or enrichment service. Differences in configuration of validation and enrichment services are indicated in the information boxes. To get started, open your ONE Desktop workspace.

1. Expand the Online Services node.



2. Double-click the Validations/Enrichments node to open the configuration.
3. Double-click a new row to add a new validations/enrichments service. The **Name** of the validation/enrichment given in this section will define the name of the `.online` file.
4. Double-click and select **edit element** in the **Services** column to configure web services for online validation or enrichment.

5. Enter the **Name** in the appropriate column. The name given here will be the name of the service and the component. The component will have the name of the online file name and the entered service name in the `.comp` file.
6. The columns labeled **Server, Namespace** and **Soap Version** will be generated automatically, but can be altered.
7. Double-click the row number to open the service configuration dialog.

> For **Validations** it is possible to specify bulk validation by selecting the **Bulk Validation** checkbox. Enabling this option will send records in batches for validation; if it is not selected, records will be sent one by one.

8. In the **Tables** field, it is necessary to add the table/s for validation. Select **Ctrl+Space** for a list of available tables.
9. Double-click and select **edit element** to open the column definition dialog.

> For **Validations**, check **Message** if the column will show a validation message.
>
> For **Enrichments**, it is necessary to check which columns will be **Input Columns** and which will be **Output Columns**. Input columns will be the enrichment inputs, while the Output columns are the columns which will be enriched. It is possible to set one column as both the input and output column, although this is a less likely use-case.

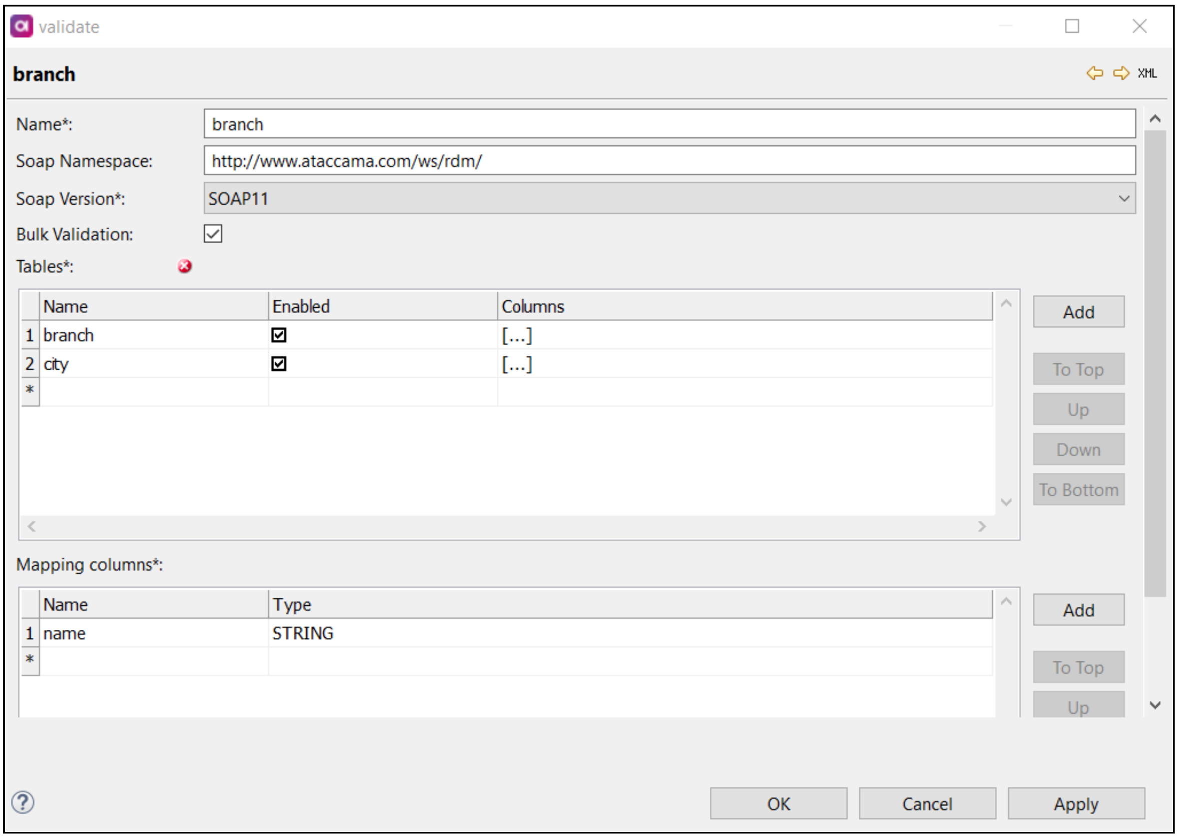
10. Select **Ctrl+Space** for a list of available columns.
11. Repeat steps 8-10 for as many tables and columns as necessary.
12. Click **Apply** and **Ok**.

Mapping Definition

If you are creating a service for only one table, it is not necessary to fill in the **Mapping Columns** or **Mapped column** attribute. Mapping definition is necessary if you are creating a service for more than one table. Mapping columns are columns which will be used in the component for the validation logic definition.

To get started, open your IDE workspace.

1. Navigate to the service configuration dialog.



branch

Name*: branch

Soap Namespace: http://www.ataccama.com/ws/rdm/

Soap Version*: SOAP11

Bulk Validation: ☑

Tables*:

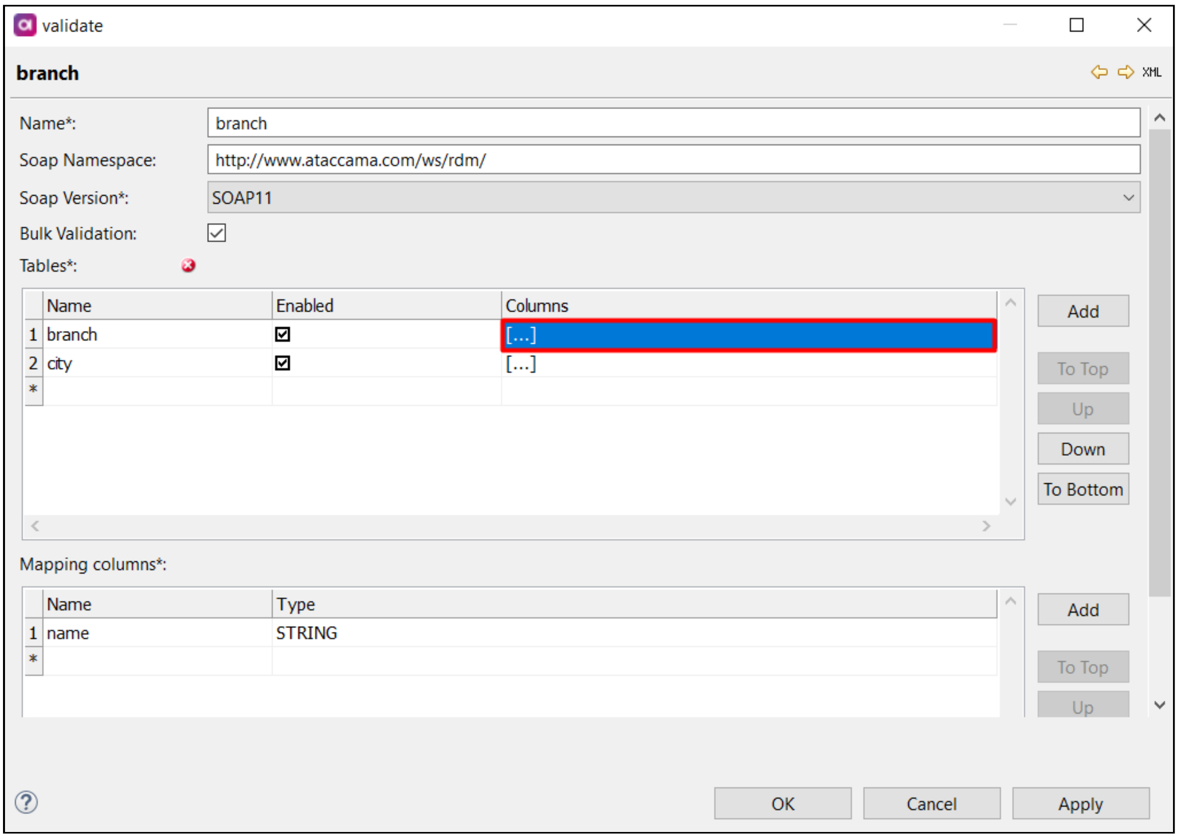
	Name	Enabled	Columns
1	branch	☑	[...]
2	city	☑	[...]
*			

Mapping columns*:

	Name	Type
1	name	STRING
*		

2. In the **Mapping Columns** field, double-click a new row to create a new mapping column.
3. Give your column a **Name** and define the **Type** of column it will be.
4. Repeat steps 2 and 3 for as many columns as you require.

5. Open the column definition dialog in the **Tables** field.



branch

Name*: branch
Soap Namespace: http://www.ataccama.com/ws/rdm/
Soap Version*: SOAP11
Bulk Validation: ☑
Tables*:

	Name	Enabled	Columns
1	branch	☑	[...]
2	city	☑	[...]
*			

Mapping columns*:

	Name	Type
1	name	STRING
*		

6. In the **Mapped Column** field, select **Ctrl+Space** to see a list of available mapping columns.
7. Select a mapping column and click **Apply**.
8. Repeat steps 5-7 for the defined tables and columns.

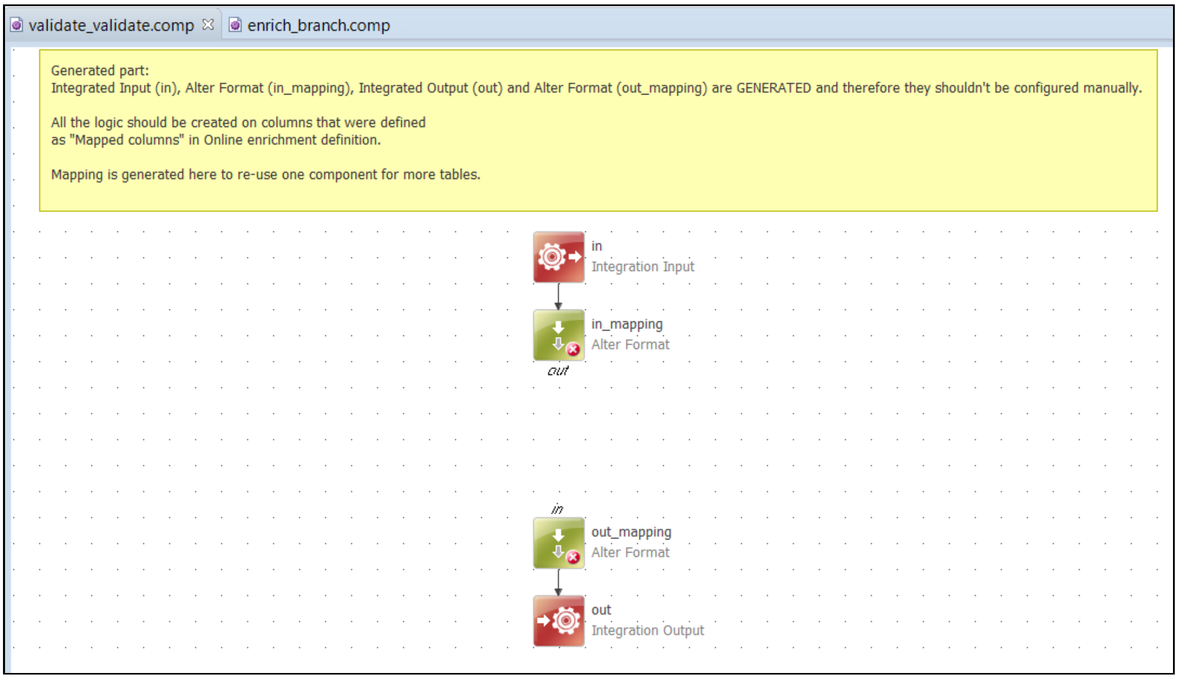
If two or more defined tables have columns with the same name, then mapping and all the checkboxes must be defined in the same way for them.

2.16.2 Step 2 Configure the Component

Once you have completed step 1, it is necessary to complete defining the logic for the component.

1. In the IDE workspace, right-click the **Online Services** node and select **Generate All Plans and Components...**
2. Navigate to **Files>Plans>Components>Validators/Enrichers** to find all of the automatically generated components.
3. Double-click a component file to open the logic definition.

4. The component will have 4 automatically generated steps: Integration Input **in**, Integration Output **out**, Alter Format **in_mapping** and Alter Format **out_mapping.** For more information on what is generated, see below.



Generated part:
Integrated Input (in), Alter Format (in_mapping), Integrated Output (out) and Alter Format (out_mapping) are GENERATED and therefore they shouldn't be configured manually.

All the logic should be created on columns that were defined
as "Mapped columns" in Online enrichment definition.

Mapping is generated here to re-use one component for more tables.

5. Place the custom, non-generated logic in between these steps.
6. On the main project file, right-click and select **Generate** to regenerate the entire project.

Generated Component Steps

This section describes the generated component steps in more detail.

Integration Input **in**:

Validations: input for all table columns.

Enrichments: input for all table columns which have checked **Input Column.**

Alter Format **in_mapping**:

Validations: For one table validations, all table columns (with the prefix *validation_*) that have checked **Message** are added. For multiple table validations, all table columns are removed as they are mapped to mapping columns but mapping columns (with the prefix *validation_*) that have checked **Message** are added.

Enrichments: For single table enrichments, all table columns that have checked **Output Column** and that have NOT checked **Input Column** are added. For multiple table enrichments, all table columns that have checked **Input Column** are removed, but mapping columns that have checked **Output Column** but not **Input Column** are included.

Alter Format **out_mapping**:

Validations: For one table validations, all table columns are removed. For multiple table validations, all mapping columns and mapping columns (with the prefix *validation_*) that have checked **Message** are removed because they are mapped to each relevant table column (with the prefix *validation_*).

Enrichments: For single table enrichments, all table columns that have checked **Input Column** and that have NOT checked **Output Column** are removed. For multiple table enrichments, all mapping columns are removed and those that have checked **Output Column** are mapped to each relevant table column.

Integration Input **out**:

This step contains every column that was not deleted.

For further information on deploying online services, see Online Services Component.

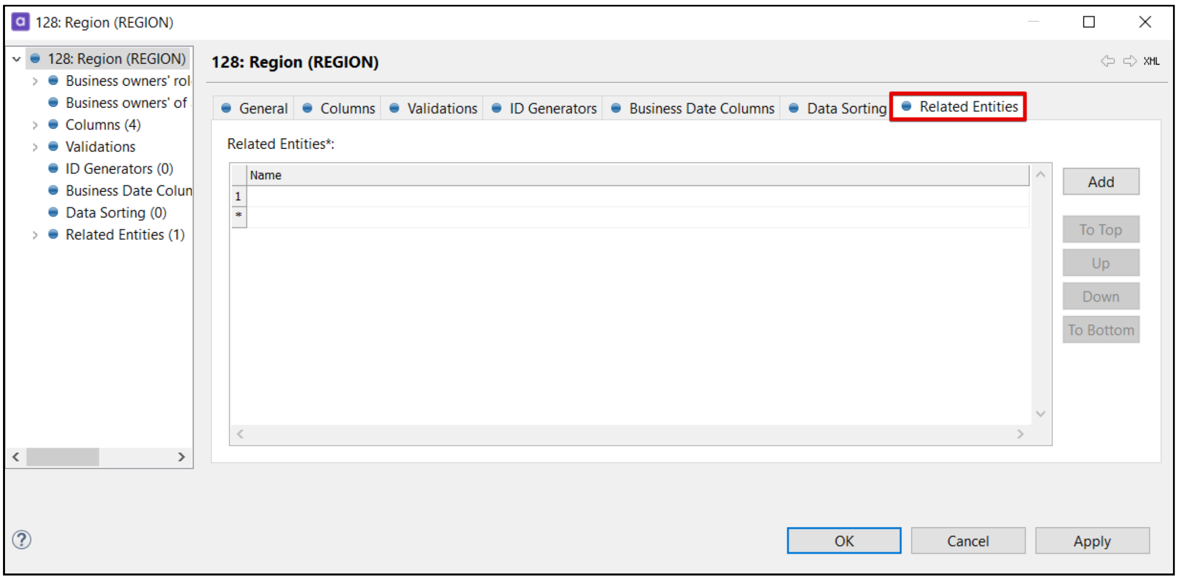
2.17 Configuring RDM Related Entities

RDM related entities allow you to quickly switch between a table and another entity (table, view or dataset) which is related to it.

To configure RDM Related Entities:

1. Open your project in ONE Desktop.
2. Go to **RDM Logical Model > Tables**.
3. Double-click the table you wish to join related entities.

4. Open the **Related Entities** tab.



5. Double-click a new row
6. **Ctrl+Space** to get a list of related entities.
7. Select the entity you wish to link to your table.
8. Repeat steps 5-7 for as many entities as you wish.
9. Click **Apply** and then **Ok**.

2.18 Enabling WatchDog High Availability

This info should remain internal until we decide otherwise. See ONE-45785 - Remove RDM Watchdog from official documentation **DONE** .

WatchDog High Availability is available on all builds but disabled by default.

To enable WatchDog HA, you need to add the property `ataccama.one.rdm.static-config.use-watchdog=true` into `application.properties` and set different `server.port` values for the individual instances (the values of `ataccama.one.rdm.application.url` and `ataccama.one.rdm.server.url` also need to be adjusted accordingly).

For example:

application.properties for Instance 1

```
server.port=8060
ataccama.one.rdm.application.url=localhost:8060
ataccama.one.rdm.server.url=http://localhost:8061
ataccama.one.rdm.static-config.use-watchdog=true
```

application.properties for Instance 2

```
server.port=8062
ataccama.one.rdm.application.url=localhost:8062
ataccama.one.rdm.server.url=http://localhost:8063
ataccama.one.rdm.static-config.use-watchdog=true
```

With RDM WatchDog HA enabled you can run multiple RDM instances over the same databases — at any point, only a single instance will run in RW mode, with the remaining instance(s) in RO mode. RDM internally provides switching of the instances, meaning RO modes are periodically checking if a RW instance exists, and the first instance to detect that this is not the case will switch to RW mode. It is necessary to refresh the Web App for the RO instance in order to unlock the RW functionalities.

Stopping an active RW instance is achievable only if the instance is deleted from the Write Permission Domain Table (`wdtbl`) table in the RDM database. The application and metrics for determining if the RW instance has failed are defined during project implementation, and will vary (typically the external application defined during project implementation will carry out RDM node availability checks and decide that a RDM node needs to be stopped, which it will achieve by updating the `wdtbl` table).

2.19 RDM REST API

The RDM REST API lets you read the data and metadata from your RDM instance using REST interfaces.

Requests are made to the following base URL: `https://rdm.<domain>/api/rest/` (or `localhost:8060` for self-managed, on-premise deployments). To get a list of all available endpoints and easily test them through Swagger UI, go to `https://rdm.<domain>/swagger-ui/api-docs.html`.

2.19.1 Authentication

Authentication is based on the Bearer scheme, which means that you need an access token obtained from Keycloak to send any requests. To configure authentication, set up a service account client in Keycloak and assign the necessary roles to it, then use it to retrieve the access token.

> By default, these settings are enabled for the `rdm-token-client` in Keycloak.

The specific roles you assign will depend on your RDM configuration. Typically, these roles are provided in the following properties (listed here with their default values):

- `ataccama.one.rdm.app-login-role=RDM` - The role required to access RDM and read other roles from Keycloak. The service account client must have this role assigned.
- `ataccama.one.rdm.group-regex-filter=RDM.*` - Only the roles matching this regular expression are mapped to RDM so you can use them in RDM to define access to entities and attributes.
- `ataccama.one.rdm.user-regex-filter=(^(?!service-account-).*)|service-account-.*rdm.*` - Only users matching this regular expression are loaded to RDM. If the default value is used, the property filters out all technical users not related to RDM (that is, whose name doesn't contain the string `rdm`). Given that the username is derived from the client name, the client you want to use to access REST APIs must also contain `rdm` in its name.

For any custom Keycloak client you want to use with the RDM REST API (using OAuth 2.0), you need to enable the `Service Accounts Enabled` option in Keycloak. This creates the corresponding technical user (for example, for the `rdm-token-client`, this would be `service-account-rdm-token-client`). The technical user must also pass the filter set in `user-regex-filter` in order for the roles and users to be mapped correctly between RDM and Keycloak.

For more detailed instructions about the configuration procedure, see Authenticating API Requests.

2.19.2 Get application status

To get information about the application status, use the following request: `GET /status`.

This call has no path or query string parameters and the response is a string, with one of the following values expected. You can continue using the API if the response returned is `SUCCESS`.

Application status	Description
STARTING	The application is starting.

Application status	Description
WAITING_FOR_ACTION	The application is waiting for some user action. You need to review and approve or reject a model proposal before proceeding.
PROCESSING_MODEL	The application is processing or updating the model.
STOPPING_MODEL	The application is stopping the model.
FAILURE	The application failed to start or deploy the model.
SUCCESS	The application is running and the model is successfully deployed. This is the only state in which the application is responding to requests.

2.19.3 List data models

To get a list of all configured data models in the application, send a `GET` request to the `/models` endpoint.

Request	Description
`GET /models`	Returns all available data models and the total number of models.
`GET /models?state=active`	Filters data models by their state. Valid values: `scheduled`, `approved`, `active`, `processed`. If no models are found for the state, the request returns an empty array (`data`).

For both requests, the response includes the number of available models (`count`), as well as the following information about each model: the model name and identifier, the date and time when the model was uploaded, the model state.

GET /models response body

```
{
    "count": 2,
    "data": [
        {
            "id": 2,
            "name": "TEST",
            "date": "2021-03-11T13:32:37.638Z",
            "state": "ACTIVE"
        },
        {
            "id": 1,
            "name": "DEFAULT",
            "date": "2021-03-11T13:31:06.094Z",
            "state": "PROCESSED"
        }
    ]
}
```

2.19.4 Get records with basic filtering

There are two requests you can use to retrieve table records with basic filtering applied.

Request	Description
`GET /entity/{entityName}`	Returns all records from the table (`entityName`) matching the applied search criteria.
`GET /entity/{entityName}/{dataStage}`	Returns all records from the table (`entityName`) matching the applied search criteria in the specified data viewing mode (`dataStage`). The `dataStage` parameter allows the following values: `EDITED`, `PUBLISHED` (alternative: `CONFIRMED`), `HISTORY`, `ALL_HISTORY`, `INPUTS`, `CART`. For more details about viewing modes, see RDM Data Viewing Modes.

To narrow down the results, provide basic filters as query string parameters in the form of key-value pairs. If no filter is defined, both requests return all table records. You can also paginate results as needed.

Query string parameter	Data type	Required	Description
`key=value`	String	No	A basic filter consisting of one or more key-value pairs. You can add as many as needed. One key-value pair corresponds to one search condition, with the key referring to the column (attribute) name by which records are filtered and the value representing the search query. The search operator for each pair is equal (=) and the logical operator used to join multiple conditions is AND. The following system columns can be queried as well: • `generatedpk` • `generatedgpk` • `username` • `ac_state` • `ac_edit_state` • `ac_date_from` • `ac_date_to` When filtering by date or datetime, use the ISO8601 format: `2022-02-20`, `2022-02-09T00:00Z`, or `2022-12-03T10:15:30+01:00`.
_count	Integer	No	Used for pagination. Defines the number of items listed on one page.

Query string parameter	Data type	Required	Description
_offset	Integer	No	Used for pagination. Defines how many items are skipped before returning results.

Examples

The following examples show how to construct a request with basic filtering:

- To read records from the `Branch` table and filter the results based on the `Branch manager` and `City` attributes:

  ```
  GET /entity/branch?branch_manager=john%20smith&code=denver
  ```

- To display records in the Edit mode from the `Country` table, with the record edit state set to `changed`. The page size is limited to 30, with no records skipped.

  ```
  GET /entity/country/edited?ac_edit_state=changed&_offset=0&_count=30
  ```

In both cases, the response contains the total number of records matching the search criteria (`count`), regardless of any pagination options applied, and the filtered records (`data`).

GET /entity/{entityName} response body

```
{
    "count": 6,
    "data": [
        {
            "generatedpk": "1",
            "generatedgpk": "2",
            "call_code": "+1",
            "name_official": "United States of America",
            "name_eng": "United States of America",
            "flag": "us",
            "name": "United States",
            "id": "1",
            "iso2": "US",
            "iso3": "USA"
        },
        {
            "generatedpk": "2",
            "generatedgpk": "3",
            "call_code": "+44",
            "name_official": "United Kingdom of Great Britain and Northern
Ireland",
            "name_eng": "United Kingdom of Great Britain and Northern Ireland",
            "flag": "uk",
            "name": "United Kingdom",
            "id": "2",
            "iso2": "GB",
            "iso3": "GBR"
        },
        {
            "generatedpk": "3",
            "generatedgpk": "4",
            "call_code": "+33",
            "name_official": "République française",
            "name_eng": "French Republic",
            "flag": "france",
            "name": "France",
            "id": "3",
            "iso2": "FR",
            "iso3": "FRA"
        },
        {
            "generatedpk": "4",
            "generatedgpk": "5",
            "call_code": "+49",
            "name_official": "Bundesrepublik Deutschland",
            "name_eng": "Federal Republic of Germany",
            "flag": "germany",
            "name": "Germany",
            "id": "4",
            "iso2": "DE",
            "iso3": "DEU"
        },
        {
            "generatedpk": "5",
            "generatedgpk": "6",
            "call_code": "+358",
            "name_official": "Suomen tasavalta",
            "name_eng": "Republic of Finland",
            "flag": "finland",
            "name": "Finland",
            "id": "5",
            "iso2": "FI",
            "iso3": "FIN"
        },
        {
            "generatedpk": "6",
            "generatedgpk": "7",
            "call_code": "+1",
            "name_official": "Canada",
            "name_eng": "Canada",
            "flag": "canada",
            "name": "Canada",
            "id": "6",
            "iso2": "CA",
            "iso3": "CAN"
        }
    ]
}
```

2.19.5 Get records with advanced filtering

In addition to simple search queries, you can also use the RDM REST API for more complex filtering. If no filter is defined, both requests return all table records. You can also paginate results as needed.

Request	**Description**
`POST /entity/{entityName}`	Returns all records from the table (`entityName`) matching the applied search criteria. Allows using advanced filtering options.

Request	Description
`POST /entity/{entityName}/{dataStage}`	Returns all records from the table (`entityName`) matching the applied search criteria in the specified data viewing mode (`dataStage`). Allows using advanced filtering options. The `dataStage` parameter allows the following values: `EDITED`, `PUBLISHED` (alternative: `CONFIRMED`), `HISTORY`, `ALL_HISTORY`, `IMPORT`, `INPUTS`, `CART`. For more details about viewing modes, see RDM Data Viewing Modes.

For both requests, the filter is provided in the request body in the JSON format and has the following structure:

- `filter`: Contains one required field, `conditions`, and two optional ones, `joinType` and `ordering`.
  - `joinType`: The logical operator between the conditions. Possible values: AND (default), OR.
  - `conditions`: An array of filtering options. Each condition must include `column` and `value` fields. If no additional fields are provided, default values are used instead.
    - `column`: The name of the column (attribute) by which records are filtered.

      The following system columns can be queried as well:

      - `generatedpk`
      - `generatedgpk`
      - `username`
      - `ac_state`
      - `ac_edit_state`
      - `ac_date_from`
      - `ac_date_to`
    - `value`: The search query that is used to filter records.

> When filtering by date or datetime, use the ISO8601 format: `2022-02-20`, `2022-02-09T00:00Z`, or `2022-12-03T10:15:30+01:00`.

        - `operator`: The search operator. The default value is `eq` (equal to). Supported options depend on the column data type. Possible values: `EQ`, `NEQ` (not equal), `LTE` (less than or equal), `GTE` (greater than or equal), `LT` (less than), `GT` (greater than), `CONTAINS`, `EXCEPT`, `BEGINS_WITH`, `ENDS_WITH`, `IS_EMPTY`, `IS_NOT_EMPTY`.
        - `caseSensitive`: Used only for string columns. Set this to `true` if you want the filtering value to be case-sensitive. Default value: `false`.
    - `ordering`: Defines how returned records are sorted. If not provided, records are ordered by `generatedpk` (record identifier in the table) in ascending order.
        - `column`: The name of the column by which records are sorted.
        - `descending`: The order in which records are sorted. Default value: `false`.
- `offset`: Used for pagination. Defines how many items are skipped before returning results.
- `count`: Used for pagination. Defines the number of items listed on one page.
- `modeSetup`: Used to define additional attributes for the viewing mode if `dataStage` is selected.
    - `<attribute>`: The following list contains the possible values for the `modeSetup` fields and their descriptions.

      modeSetup attribute values

Dataviewing mode	modeSetup attribute	Data type	Description
Published (alternative: Confirmed), Edited, History, All History	`businessDate`	Datetime	Filters records with the given business date. If null, no restrictions are applied.

Data viewing mode	modeSetup attribute	Data type	Description
Edited, All History	`editState`	String	Filters records based on their edit state. Possible values: `UNCHANGED`, `NEW`, `DELETED`, `CHANGED`.
Edited, All History	`usernames`	Array of strings	Filters records assigned to given users.
Edited	`workflows`	Array of strings	Filters records in the given workflow states.
Edited	`valid`	Boolean	Filters records based on whether they are marked as valid (`true`) or invalid (`false`). If null, all records are returned.
History	`historyDate`	Datetime	Filters records with the given date (historic snapshot).

Dataviewing mode	modeSetup attribute	Data type	Description
All History	`tags`	Array of strings	Filters records with the given tags.
All History	`from` and `to`	Datetime	Filters records existing in the period delimited by `from` and `to`.

Advanced filter JSON structure

```
{
    "filter": {
        "joinType": "or",
        "conditions": [
          {
            "column": "name",
            "operator": "eq",
            "value": "france",
            "caseSensitive": "true"
          },
          {
            "column": "name",
            "value": "United States"
          }
        ],
        "ordering": [
          {
            "column": "name",
            "descending": "false"
          }
        ]
    },
    "offset": "1",
    "count": "5",
    "modeSetup": {
        "historyDate": "2022-02-09T00:00Z"
    }
}
```

Examples

The following example shows how to query the `Country` table in the History viewing mode using complex filters.

Here, we are looking for records where the country name matches the values 'France' or 'United States'. Since the operator for the filtering values is not specified, the default option is used (`eq`). The case is ignored, and the results are ordered by the country name in ascending order. An additional filtering option is used to find records created before a particular date.

POST /entity/{entityName}/{dataStage} request body

```
{
    "filter": {
        "joinType": "or",
        "conditions": [
          {
            "column": "name",
            "value": "france"
          },
          {
            "column": "name",
            "value": "United States"
          }
        ],
        "ordering": [
          {
            "column": "name",
            "descending": "false"
          }
        ]
    },
    "offset": "1",
    "count": "5",
    "modeSetup": {
        "historyDate": "2022-02-09T00:00Z"
    }
}
```

The response contains the total number of records matching the search criteria (`count`), regardless of any pagination options applied, and the filtered records (`data`). The same applies for the request `POST /entity/{entityName}`.

In the previous request, we defined the paging so that the first result is skipped and a maximum of 5 records are returned.

POST /entity/{entityName}/{dataStage} response body

```
{
    "count": 2,
    "data": [
        {
            "generatedpk": "1",
            "generatedgpk": "2",
            "ac_date_from": "2021-03-11T13:37:26.000Z",
            "ac_date_to": null,
            "username": "admin",
            "call_code": "+1",
            "name_official": "United States of America",
            "name_eng": "United States of America",
            "flag": "us",
            "name": "United States",
            "id": "1",
            "iso2": "US",
            "iso3": "USA"
        }
    ]
}
```

2.19.6 Get record link

You can use the REST API to generate an encoded link to a particular RDM record. To do this, use the request `GET /link` and provide the required query string parameters.

Query string parameter	Data type	Required	Description
entityName	String	Yes	The name of the table where the record is located.
generatedpk	Integer	Yes	The generated primary key, that is, the record identifier. Corresponds to the `ID` column in the table.

Query string parameter	Data type	Required	Description
mode	String	No	The view mode in which the record will be displayed. Valid values: `PUBLISHED` (alternative: `CONFIRMED`), `EDIT`, `HISTORY`, `ALL_HISTORY`, `CART`, `INPUTS`. For more details about viewing modes, see RDM Data Viewing Modes.
hcn	Integer	No	The history change number of the record. The parameter has higher priority over `mode`; if the `hcn` parameter is defined, the mode is automatically set to `ALL_HISTORY`.

The response returns a string you can use to view the record. Keep in mind that users still need to have the correct viewing permissions for the entity and be logged in to RDM to access the record.

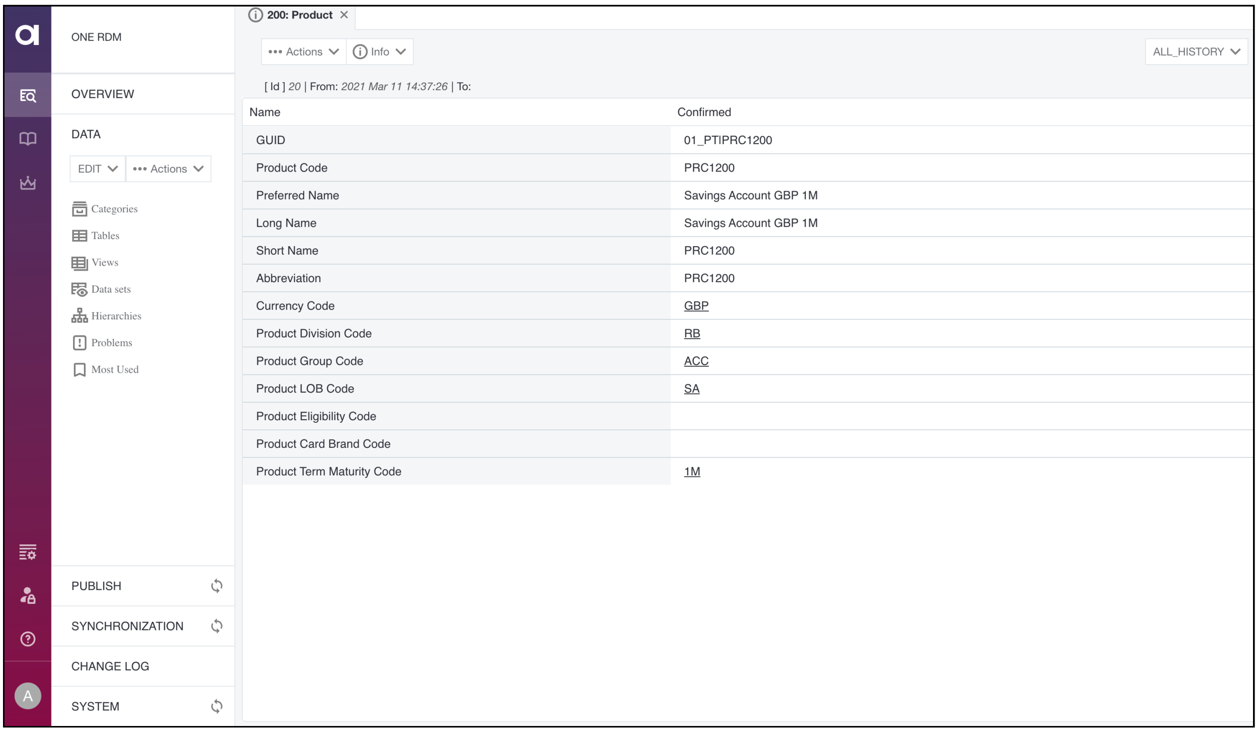
GET /link request example

```
GET https://rdm.<domain>/api/rest/link?generatedpk=2&entityname=product
```

GET /link response body

```
rdm.<domain>/
mailLink.html#encoded:4616471645162637D3B7224716263722A3B5B7224656471696C645162622
A3B722D6F6465622A3B722D6F64656
22A32214C4C4F584943545F4259522C2228636E622A313D7C222964622A3B55303D5C222471626C656
E416D65622A3220727F64657364722D7D7D5D7
```

Once you open the link in your browser, the record is displayed in the RDM web application as follows:



[Id] *20* | From: *2021 Mar 11 14:37:26* | To:

Name	Confirmed
GUID	01_PTIPRC1200
Product Code	PRC1200
Preferred Name	Savings Account GBP 1M
Long Name	Savings Account GBP 1M
Short Name	PRC1200
Abbreviation	PRC1200
Currency Code	GBP
Product Division Code	RB
Product Group Code	ACC
Product LOB Code	SA
Product Eligibility Code	
Product Card Brand Code	
Product Term Maturity Code	1M

3 RDM Reference Documentation

3.1 RDM Application Properties

This article is intended to serve as a reference point for RDM configuration. As such, it provides an overview of the available properties and, when applicable, refers users to more comprehensive sources. The properties described here are defined in the `rdm/etc/application.properties` file. For each property, you will find information about the required data type, its default value, and a short description. The Mandatory column specifies whether a property is required for the application to run and function as expected.

3.1.1 RDM Properties

Use the following properties to configure server settings for RDM, specify the type of the repository, and the path to the license.

Name	Data Type	Default Value	Mandatory	Description
`server.port`	Number	`8060`	Yes	The number of the port where the RDM application is running.
`server.servlet.context-path`	String	`/`	Yes	The context path to the RDM application server. It is `root` by default.
`ataccama.one.rdm.id`	String	`rdm`	No	The RDM application ID. If not set, `canonical_hostname:context_path` is used.

Name	Data Type	Default Value	Mandatory	Description
`ataccama.one.rdm.repository`	String	`database`	Yes	The type of RDM repository. If set to `database`, the configuration set in the database is used.
`ataccama.one.rdm.application.url`	String	`localhost:8060`	Yes	The URL of the RDM application.
`ataccama.one.rdm.license-folder`	String	`/home/user.name/`	Yes	The path to the folder that contains the license. By default, the application searches for the license in the home directory of the user.
`ataccama.one.rdm.server.url`	String	`http://localhost:8061`	Yes	The URL of the RDM (runtime) server.
`ataccama.one.rdm.environment`	String	`""`	No	The name of the environmmnent used (available values are `dev`, `prod`). Can be used as a variable in email templates.
`ataccama.one.rdm.max.config.file.size.kb`	Number	`1000`	No	The maximum size of the project configuration file that can be imported to RDM. Expressed in kilobytes.

3.1.2 RDM Data Connection

Use the following properties to configure the connection to the RDM storage database. See Encrypting Passwords for information on how to encrypt passwords.

Name	Data Type	Default Value	Mandatory	Description
`ataccama.one.rdm.datasource.rdm-data.url`	String	/	Yes	The URL of the RDM storage database.
`ataccama.one.rdm.datasource.rdm-data.jdbcUrl`	String	/	Yes	The JDBC URL of the RDM storage database.
`ataccama.one.rdm.datasource.rdm-data.username`	String	/	Yes	The username for the RDM storage database.
`ataccama.one.rdm.datasource.rdm-data.password`	String	/	Yes	The password for the RDM storage database.
`ataccama.one.rdm.datasource.rdm-data.driverClassName`	String	/	Yes	The driver class name for the RDM storage database.
`ataccama.one.rdm.datasource.rdm-data.continue-on-error`	Boolean	`true`	No	When set to `true`, if the initial attempt to connect to the database fails, repeated attempts will be made until connection is established.
`ataccama.one.rdm.datasource.rdm-data.waiting.timeout`	String	`120s`	No	Timeout for database connection attempts in seconds.
`ataccama.one.rdm.datasource.rdm-data.waiting.interval`	String	`10s`	No	Interval between connection attempts in seconds.

3.1.3 RDM Metadata Connection

The following properties configure the connection to the database where RDM metadata is stored. See Encrypting Passwords for information on how to encrypt passwords.

Name	Data Type	Default Value	Mandatory	Description
`ataccama.one.rdm.datasource.rdm-repo.url`	String	/	Yes	The URL of the RDM metadata database.
`ataccama.one.rdm.datasource.rdm-repo.jdbcUrl`	String	/	Yes	The JDBC URL of the RDM metadata database.
`ataccama.one.rdm.datasource.rdm-repo.username`	String	/	Yes	The username for the RDM metadata database.
`ataccama.one.rdm.datasource.rdm-repo.password`	String	/	Yes	The password for the RDM metadata database.
`ataccama.one.rdm.datasource.rdm-repo.driverClassName`	String	/	Yes	The driver class name for the RDM metadata database.
`ataccama.one.rdm.datasource.rdm-repo.continue-on-error`	Boolean	`true`	No	When set to `true`, if the initial attempt to connect to the database fails, repeated attempts will be made until connection is established.
`ataccama.one.rdm.datasource.rdm-repo.waiting.timeout`	String	`120s`	No	Timeout for database connection attempts in seconds.
`ataccama.one.rdm.datasource.rdm-repo.waiting.interval`	String	`10s`	No	Interval between connection attempts in seconds.

3.1.4 Keycloak

The following properties configure Keycloak. See Encrypting Passwords for information on how to encrypt passwords.

Name	Data Type	Default Value	Mandatory	Description
`ataccama.authentication.keycloak.server-url`	String	`http://localhost:8080/auth`	Yes	The URL of the server where Keycloak is running.
`ataccama.authentication.keycloak.realm`	String	`ataccamaone`	Yes	The name of the Keycloak realm.
`ataccama.authentication.keycloak.admin.client-id`	String	/	Yes	The client identifier used to verify the admin user's authorization token.
`ataccama.authentication.keycloak.admin.secret`	String	/	Yes	The secret key of the client identifier for the admin account. Secret keys can be generated using Keycloak. Used by `BASIC` and `SECRET_JWT` strategies.
`ataccama.authentication.keycloak.token.client-id`	String	/	Yes	The client identifier. Used to verify a user's authorization token and to log in a user.
`ataccama.authentication.keycloak.token.secret`	String	/	Yes	The secret key of the client. Secret keys can be generated using Keycloak. Used for `BASIC` and `SECRET_JWT` strategies.
`ataccama.authentication.keycloak.token.issuer`	String	/	Yes	Specifies the issuer of the JWT token. Typically, Keycloak uses the URL of the realm as the token issuer.
`ataccama.authentication.keycloak.public.client-id`	String	/	Yes	Keycloak public client ID for web application browsing.
`ataccama.client.connection.keycloak.http.enabled`	Boolean	`true`	Yes	Specifies whether the HTTP protocol is being used with Keycloak.

Name	Data Type	Default Value	Mandatory	Description
`ataccama.client.connection.keycloak.http.tls.enabled`	Boolean	`false`	Yes	Specifies whether the TLS protocol is being used with Keycloak.
`ataccama.one.rdm.user-synchronization-schedule`	String	/	No	The schedule for automatic synchronization of user or role mapping with Keycloak. The value is a cron expression that consists of six fields representing, in this order, second, minute, hour, day, month, weekday. For more information, see the official Spring documentation.

3.1.5 SSL

Use these properties to set up SSL.

Name	Data Type	Default Value	Mandatory	Description
`server.ssl.enabled`	Boolean	`true`	No	Enables SSL.
`server.ssl.key-store`	String	/	No	The full path to the keystore.
`server.ssl.key-store-password`	String	/	No	The password for decrypting the keystore. Used if the keystore is encrypted (recommended).
`server.ssl.key-password`	String	/	No	The password for the private key. Used if the private key is encrypted.
`server.ssl.trust-store`	String	/	No	The full path to the truststore.

Name	Data Type	Default Value	Mandatory	Description
`server.ssl.trust-store-password`	String	/	No	The password for the truststore. Used if the truststore is encrypted.

3.1.6 Web Application Links

The following properties specify the links to other applications.

Name	Data Type	Default Value	Mandatory	Description
`ataccama.one.rdm.mdm.link`	String	`http://localhost:8051/`	No	The URL for the MDM link.
`ataccama.one.rdm.catalog.link`	String	`http://localhost:8020`	No	The URL for the link to the catalog (MMM) application.

3.1.7 Logging

The following properties configure logging. Logging levels can also be set via `LOG_PATH`.

Name	Data Type	Default Value	Mandatory	Description
`ataccama.logging.plainTextConsoleAppender`	Boolean	`true`	No	If set to `true`, logs are outputted as plain text in the console.
`ataccama.logging.jsonConsoleAppender`	Boolean	`false`	No	If set to `true`, logs are outputted in JSON format in the console.
`ataccama.logging.plainTextFileAppender`	Boolean	`false`	No	If set to `true`, logs are outputted as plain text in the log file.

Name	Data Type	Default Value	Mandatory	Description
`ataccama.logging.jsonFileAppender`	Boolean	`true`	No	If set to `true`, logs are outputted in JSON format in the log file.
`logging.file.path`	String	`${ataccama.path.root}/log`	No	The location of the `log` folder. This value can be updated using the system property `LOG_PATH`, which can be used for further logging configuration.
`logging.level.root`	String	`INFO`	No	The root logging level. Available values are `INFO`, `WARN`, `ERROR`, `DEBUG`, `OFF`.
`logging.level.com.ataccama`	String	`INFO`	No	The logging level for `com.ataccama` packages. Available values are `INFO`, `WARN`, `ERROR`, `DEBUG`.
`logging.level.com.ataccama.rdm`	String	`INFO`	No	The logging level for RDM packages. Available values are `INFO`, `WARN`, `ERROR`, `DEBUG`.

3.1.8 Endpoints for Monitoring

The following properties configure endpoints for monitoring. For more information, see Configuring Monitoring.

Name	Data Type	Default Value	Mandatory	Description
`management.endpoints.enabled-by-default`	Boolean	`false`	No	Enables all actuator endpoints. If set to `false`, it is possible to configure individually which endpoints should be enabled.
`management.endpoint.info.enabled`	Boolean	`true`	No	Enables `/info` monitoring endpoint.
`management.endpoint.health.enabled`	Boolean	`true`	No	Enables `/health` monitoring endpoint.
`management.endpoint.prometheus.enabled`	Boolean	`true`	No	Enables `/prometheus` monitoring endpoint.
`management.endpoints.web.exposure.include`	String	`health,info,prometheus`	No	A comma-separated list of exposed actuator endpoints that should provide information about the application. These endpoints track the following: • `health` - The health status of the application. • `info` - Other information about the application. • `prometheus` - Provides all metrics from the application in a format that Prometheus can scrape.

Name	Data Type	Default Value	Mandatory	Description
`management.endpoint.health.show-details`	String	`always`	No	Specifies how much information is provided by the `health` monitoring endpoint. The following values are available: • `never` - Health details are never displayed to any user. • `when-authorized` - Only authorized users have access to health information. • `always` - All users can see health details.
`management.endpoint.health.show-components`	String	`always`	No	Specifies how much detail the `health` monitoring endpoint provides about the application components. You can also define which components are shown. The following values are available: • `never` - Component information is never displayed to any user. • `when-authorized` - Only authorized users have access to information about components. • `always` - All users can see component details.
`management.endpoint.health.status.order`	String	`down,out-of-service,reloading,unknown,up`	No	A comma-separated list that determines how the `/health` monitoring endpoint prioritizes application health statuses.

Name	Data Type	Default Value	Mandatory	Description
`management.info.git.mode`	String	`full`	No	Configures how much information the `/info` monitoring endpoint retrieves from Git about the application source code repository. To show all available information from the `git.properties` file, set the value to `full`. To display only basic information, such as the name of the branch, the commit identifier, and the time the commit was made, set the value to `simple`.
`management.endpoint.health.probes.enabled`	Boolean	`true`	No	Enables `/health/liveness` and `/health/readiness` endpoints.
`management.endpoint.health.group.liveness.include`	String	`diskSpace,ping`	No	Defines which components are covered by the liveness probe. These components are a subset of `/health` components.
`management.endpoint.health.group.readiness.include`	String	`db`	No	Defines which components are covered by the readiness probe. These components are a subset of `/health` components.
`ataccama.authentication.http.acl.endpoints.prometheus.endpoint-filter`	String	`/actuator/prometheus`	No	Enables ACL-based authentication on the selected endpoint. The same filter can be enabled on other endpoints.
`ataccama.authentication.http.acl.endpoints.prometheus.allowed-roles`	String	`ONE_PLATFORM_MONITORING`	No	Allows access to the endpoint defined in the `endpoint-filter` property for the selected user roles.

Name	Data Type	Default Value	Mandatory	Description
`management.metrics.web.server.auto-time-requests`	Boolean	`false`	No	Enables the timing metrics to all Spring endpoints.

3.1.9 Static Configuration

Use the following properties to set static configuration.

Name	Data Type	Default Value	Mandatory	Description
`ataccama.one.rdm.static-config.username-case-insensitive`	Boolean	`false`	Yes	If set to `true`, the username is case insensitive.
`ataccama.one.rdm.static-config.start-empty`	Boolean	`false`	Yes	If set to `true`, RDM starts with default empty configuration. Otherwise, it waits for user to upload configuration.

Name	Data Type	Default Value	Mandatory	Description
`ataccama.one.rdm.static-config.io-mnrefs-strategy`	Boolean	`DEFAULT`	No	Persistence strategy for storing MN reference values. The following values are available: • `DEFAULT` (backward compatible and recommended) - uses full quoting of MN reference values when storing values to the database. • `SIMPLE` - uses simplified quoting of the values when possible (for single-key MN reference relationships). This mode is not recommended unless it is required for a specific integration. Once the value is set, it cannot be changed otherwise the MN reference data would have to be reprocessed.
`ataccama.one.rdm.app-login-role`	String	`RDM`	Yes	The name of role that is required to successfully log in to RDM. When not defined, any user can access the application. In such a case a warning is reported to the backend log. If the defined role does not exist in Keycloak, an error is reported to the log and no user can log in to the application.
`ataccama.one.rdm.group-regex-filter`	String	`RDM.*`	Yes	Prefix for additional RDM roles with `app-login-role`. The roles are visible on the Permissions tab.

Name	**Data Type**	**Default Value**	**Mandatory**	**Description**
`ataccama.one.rdm.user-regex-filter`	String	`(^(?!service-account-).*)|service-account-.*rdm.*`	No	Filters RDM service accounts so that technical users not related to RDM are hidden in the web app. If a username matches this regular expression, it is loaded to RDM and shown on the Permissions tab. If set to `""` (empty string), all users are shown.
`ataccama.one.rdm.system-group-name`	String	`RDM_admin`	Yes	The system group name for RDM. Users with this role can perform system-related operations and have higher privileges than regular users (for example, they can see all tables).
`ataccama.one.rdm.permissions-group-name`	String	`""`	No	The name of the role with access to the Permissions tab in RDM. When empty, the `system-group-name` role is used to determine access to the permissions.
`spring.datasource.maxActive`	Number	`20`	No	The maximum number of active connections that can be allocated from the datasource pool at the same time.

Name	Data Type	Default Value	Mandatory	Description
`spring.datasource.maxIdle`	Number	`10`	No	The maximum number of connections that should be kept in the pool at all times. Idle connections are checked periodically (if enabled) and connections that have been idle for longer than `minEvictableIdleTimeMillis` will be released.
`spring.datasource.maxWait`	Number	`-1`	No	The maximum time interval that the pool waits (when there are no available connections) for a connection to be returned before throwing an exception. Expressed in `ms`. If set to `-1`, the waiting time is not limited.

3.1.10 Mail Settings

The following properties configure mail settings. If workflows are used in the project, these properties are mandatory.

Name	Data Type	Default Value	Mandatory	Description
`spring.mail.host`	String	/	No	The hostname of the mail server.
`spring.mail.port`	Number	/	No	The number of the the mail server port.
`spring.mail.username`	String	/	No	The username used to authenticate to the mail server.
`spring.mail.password`	String	/	No	The password used to authenticate to the mail server.
`spring.mail.default-encoding`	String	/	No	The default encoding of the emails.

Name	Data Type	Default Value	Mandatory	Description
`spring.mail.properties.mail.transport.protocol`	String	/	No	Specifies the protocol used to send emails.
`spring.mail.properties.mail.smtp.port`	Number	/	No	The number of the SMTP port.
`spring.mail.properties.mail.smtp.auth`	Boolean	/	No	Enables the SMTP authentication.
`spring.mail.properties.mail.smtp.starttls.enable`	Boolean	/	No	Enables STARTTLS for SMTP.
`spring.mail.properties.mail.smtp.starttls.required`	String	/	No	Specifies whether the STARTTLS is required for SMTP.

3.1.11 Client Security Headers

You can configure RDM Webapp security by adding response headers (security headers) to HTTP responses from the web application.

Name	Data Type	Default Value	Mandatory	Description
`one.security.header.content-security-policy.connect-src`	String	`self' ${ataccama.authentication.keycloak.server-url}`	No	Specifies allowed connections. We strongly recommend using the default value.
`one.security.header.content-security-policy.script-src`	String	`self' 'unsafe-eval' 'sha256-XI/joSm13E0tRqSDZUO5DZQUbuNxa2lnkOORub88i8U=' 'sha256-7qt6iyJjmGKP6A18nPa5hTNifcr+JTAgPsN9Qpn+QgM='`	No	Specifies allowed script sources. We strongly recommend using the default value.

Name	Data Type	Default Value	Mandatory	Description
`one.security.header.content-security-policy.img-src`	String	`self' data:`	No	Specifies allowed image sources. We strongly recommend using the default value.
`internal.encryption.keystore`	String	/	No	The path to the internal keystore.
`internal.encryption.keystore.password`	String	/	No	The password for the internal keystore.
`internal.encryption.keystore.passwordFile`	String	/	No	The path to the password file for the internal keystore.
`properties.encryption.keystore`	String	/	No	The path to the properties keystore.
`properties.encryption.keystore.password`	String	/	No	The password for the properties keystore.
`properties.encryption.keystore.passwordFile`	String	/	No	The path to the password file for the properties keystore.
`one.security.header.X-Frame-Options`	String	`deny`	No	Protects against clickjacking. If set to `deny`, iframes are disabled.
`one.security.header.X-Permitted-Cross-Domain-Policies`	String	`none`	No	Specifies if cross-domain requests from Flash and PDF documents are allowed.

Name	Data Type	Default Value	Mandatory	Description
`one.security.header.Referrer-Policy:`	String	`strict-origin`	No	Defines how much referrer information (sent with the Referer header) should be included with requests. If set to `strict-origin`, only the origin is sent.
`one.security.header.X-XSS-Protection`	String	`1; mode=block`	No	Protects against cross-site scripting attacks. If set to `block`, the page does not load when an attack is detected.
`one.security.header.X-Content-Type-Options`	String	`nosniff`	No	Protects against MIME sniffing.

3.1.12 Retry for Keycloak Connection to RDM

If the initial connection to Keycloak fails, the RDM webapp can try to connect repeatedly before startup.

Name	Data Type	Default Value	Mandatory	Description
`ataccama.one.rdm.retry.keycloak.max-attempts`	Number	`999999999`	No	Maximal number of connection attempts.
`ataccama.one.rdm.retry.keycloak.wait-duration`	String	`10s`	No	Interval between attempts in seconds.

4 RDM Backup and Recovery

4.1 RDM Backup, Restore

Majority of primary data is stored in a database - the only exceptions are input and output files, possibly some extra logic implemented on individual projects being located on the Runtime server .

Configuration and scripts are usually backed-up on project basis and it's restore is done using project specific deployment scripts e.g. via GIT

There is no out of the box functionality dealing with backup/restore, so please use the following as an input to your project-specific setup.

Related articles:

- MDM Backup and Recovery

4.1.1 Configuration source

The whole RDM configuration is done in ONE Desktop. Configuration is stored on a local drive and usually synced to a version control system like Git that covers backup/versioning/restore functionalities by default. Anyway. there are two parts of configuration deployed differently:

- RDM WebApp: Zip file is generated by Model Project in ONE Desktop and has to be manually deployed via UI. Deployed configuration is then stored in RDM's storage incl. historical versions
- Runtime server: plain xml files with no deployment options (except for mentioned Git approach)

4.1.2 Storage

DB backup (dump) and it's restore is enough for the basic usage see as an example Dumping and Restoring in MDM (not sure whether we have it for RDM; this is approach for PaaS (note: this one gonna internal; once reviewed, we will create a public guide where PaaS specifics will be removed.)

There is also a possibility to Dump and Load limited subset of data from RDM FE (data history, versions and technical tables are not taken into the account!!!).

Warning: additional data sources used in RDM project has to be covered on project basis

4.1.3 Filesystem

Do not backup the whole filesystem since the backup might be redundant and/or useless after the recovery. There is anything special in terms of backup/recovery process - simply backup/restore the whole folder keeping the artefacts mentioned below.

RDM app server (Runtime server)

Configuration (optional)

- Configuration should be managed in version control system and installation should be automated - you should be able to build the same environment from scratch without any backup. Same applies for application.properties

Data (optional)

- There is usually a workflow that's responsible for downloading/uploading input/output data to the application server. So initiate the appropriate WF after recovery is usually enough
- Data backup might me needed only in case
    - RDM produces export that's triggered based on some internal event (data load finished successfully) - this type of files should be
    - Rare: Data folder contains a primary data for lookups (that are not stored as part of the configuration)

Lookups (optional)

- Lookup should be available in raw form somewhere and you should be able to rebuild them.. or if needed (like you need to be able to recover in short time), you can backup the lookups.

Other parts

- EWF execution status => to mitigate this, switch to DB persistence

RDM webapp

The web server application.properties should be managed in version file systems (Git)

RDM configuration is in DB

4.2 Migration path

Migration have to be considered when

1. moving the solution from one DB server to the another
2. performing some significant structural changes in RDM or changing setting significantly (validations are redesigned)
3. combination of above mentioned

- **?** How to combine such migration when (RDM) upgrade is performed as well
  - AFIK it will be heavily dependent on the particular versions. Sometimes it is super easy, sometimes very complex
  - Upgrading RDM

Regardless the migration type, the above mentioned locations has to be considered and the content has to be transferred and/or upgraded! Please also consider known issues in release notes.

4.2.1 Configuration source

Upgrade has to be made via ONE Desktop otherwise solution won't start after the runtime upgrade and submitted to Git if relevant !!!

Note: last such upgrade was needed when transiting from v12.x and older to v13 , otherwise there is a chance that older configuration will work with newer version (but doesn't mean it will be valid for future releases)

4.2.2 Storage

New entities, columns are added automatically

- No objects are renamed (transformed to add and remove)
- No objects are deleted (removed entities, columns)

Some of the structures are transformed automatically unless it is written differently in upgrade guides (otherwise **RDM patch** has to be prepared)

If it is simple migration DB dump mentioned above can be used Dumping and Restoring in RDM Also consider the following constrains for DB dump way

- DB dump is the easiest option to migrate all data 1:1

- DB dump will be also used when we are migration within PaaS versions, but also there are significant changes in configuration (added entities and columns are okay)
  - v12 (VM based) -> v13. x.x (new helm cart)
- ❗ DB dump option works only for the same RDBS (e.g. Postgres ➡ Postgres)
  - how about different Postgres versions?

Migration between two different RDBMS is not supported by the tool and it has to be done fully manually.

4.2.3 Filesystem

RDM app server

MDM server (Runtime)

- re(deploy)

Configuration (optional)

- re(deply) from Git or other location
- configuration has to be upgraded first

Data (optional)

- Usually no copy is needed as there is usually some kind of orchestration to download/upload files before processing
- Data backup might me needed only in case
  - RDM produces export that's triggered based on some internal event (data load finished successfully) - this type of files should be
  - Rare: Data folder contains a primary data for lookups (that are not stored as part of the configuration)

Lookups (optional)

- Lookup should be available in raw form somewhere and you should be able to rebuild them.. or if needed (like you need to be able to recover in short time), you can backup the lookups.

Others

- there might be EWF execution status history that needs to be migrated (in case file persistence is used)
- there might be some other files stored on file system such as Streaming's dead letter queue

MDM webapp server

MDM webApp

- re(deploy)

Configuration

- application.properties (from git or re/deployed)