# System

system

The CzechIdM system determines the behavior towards connected end systems.

**The system allows:**

- to define configuration and **connection** of an end system (via connector).
- to create / generate a scheme of an end system.
- to create mapping for **provisioning** and **sync** .
- to create mapping attributes. These allow you to define the behavior of attributes (and their values) both towards the end system and IDM.
- to create and start synchronization configurations.
- to manage queues and archive **provisioning** .

## Creating a copy of a system

The system in CzechIdM can be duplicated. The duplicate contains the entire system configuration:

- **Configuration of the system connection**, including secret values.
- System **scheme**.
- **Mapping** system.
- **Mapped attributes**.
- **Synchronization** configuration.

The system name must be unique, so a unique name is generated when duplicating. For example, if you duplicate a system named **LDAP**, then the resulting system is **Copy-of-LDAP**. If this system already exists, then a number postfix is added (**Copy-of-LDAP1**) and etc.

A newly created duplicate is always set as an **inactive** system (with provisioning queue available). Likewise, all possible sync settings are set to **inactive** (with provisioning queue available).

# End systems connected to CzechIdM

Systems get connected to IdM through synchronization and provisioning operations.

## Steps to connect a new system:

1. a new system named **Name-of-your-choice** must be created
2. you choose a suitable database connector for it
3. you need to set the connector up

4. create a system scheme: either manually, or it is returned by the connector. In doing so, you get all the attributes available in the system
5. then, map all these attributes. This serves two basic purposes:
6. makes the scheme attributes accessible to the IdM system-users
7. defines which value the attribute is to be mapped into in the IdM system (identity, group, extended attribute, hidden attribute).
8. for optimalization purposes, an attribute can be cached. New attribute mapping is marked as 'cached' by default.
9. in order to create an account for this user in the end system,

- you must link a new role to the system
- and assign this role to a user

# Synchronization

sync, identity

The basic task of synchronization is to ensure the correct state of the data on the end system (typically, users accounts) and in IdM. The correct state is defined both by the data in IdM (account management) and by the IdM configuration itself. This is usually taken care of by setting provisioning and synchronization on a given system.

## Entities that support sync

| Name | Entity name (DTO) | More details |
|---|---|---|
| Identity | IdmIdentityDto | synchronization |
| Contractual relationship | IdmIdentityContractDto | relation-sync |
| Time slices of contractual relationship | IdmContractSliceDto | contract-slice-sync |
| Tree | IdmTreeNodeDto | tree-sync |
| Role | IdmRoleDto | role-sync |
| Role catalogue | IdmRoleCatalogueDto | |

### A typical synchronization process runs as follows:

1. **Finding changed accounts** on the end system.
2. **Iteration of changed accounts** and evaluation of the situation for each account (accounts in IdM).
3. **Action performance** for the situation found (e.g. creation of an identity in IdM).
4. **Running subsequent operations** (e.g. provisioning of updating an account on the end system).

### During synchronization, identities can be evaluated for different situations they find themselves in, namely:

- (non)-existent links
- (non)-existent entities

- (non)-existent accounts

More on this topic later.

# Synchronization/provisioning strategies

An attribute mapping strategy defines how attributes, and particularly their values, will be handled during provisioning and synchronization.

Here's a list of strategies to consider:

- **SET (set as IdM calculates)**: the default strategy which sends the acquired value to the end system as IdM calculates it. If the value is the same on the end system, the attribute isn't sent.
- **WRITE-IF-NULL (Set only if the value on the system is null)** - Before the final comparison whether a value is non-existent, **transformation from the system** is applied to the attribute value from the end system. The transformation is defined in the mapping of the attribute.
- This strategy is applied both to account **editing** and its **creation**.
- **CREATE (Set only when creating)**
- **MERGE:** alters the logic of calculating attributes and their values. If there are more attributes overloading the same default attribute, all the values are merged into one. This resulting value (list) is sent to the IdM system as a "wish" of how the value should look like on the end system.
- **AUTHORITATIVE_MERGE (Authoritative merge)**: alters the logic of calculating attributes and their values. The original values of the attribute on the end system are not taken into account (i.e. if some other system or administrator had put a different value to the attribute, it would be ignored by IdM).

The **WRITE-IF-NULL** strategy has a lower priority than **SET**, **MERGE** and **AUTHORITATIVE_MERGE**. Therefore, if we define one attribute with the **SET** strategy and another one with the **WRITE-IF-NULL** strategy in the default mapping on the system, then the value of the **WRITE-IF-NULL** attribute is never used. In this case, it is advisable to define the attribute with only the **WRITE-IF-NULL** strategy in the default mapping. The other attribute with the **SET** strategy will have to be defined in the role as an overload of the default attribute.

The **CREATE** strategy has a lower priority than **WRITE-IF-NULL**, **SET**, **MERGE** and **AUTHORITATIVE-MERGE**.

A situation where there are two overloaded attributes for the same default attribute, one having the **AUTHORITATIVE_MERGE** strategy and the other the **SET** or **MERGE** strategy, will not be valid. **Such a situation is collisional and will result in an exception during provisioning!**

Both the MERGE and AUTHORITATIVE_MERGE strategies can be linked only to a multi-value scheme attribute!

## Additional characteristics of strategies

Aside from a strategy, every attribute can also have the indications **Always send** and **Send only if IdM value exists**. These indications can be combined with all the above strategies.

- **Always send** - ensures that the attribute will be sent to the end system even when it has been assessed that its value hasn't changed.

* **Send only if IdM value exists** - ensures that the attribute will be sent only when the calculated IdM value has some value (if it is a String type value, it cannot be blank). This setting ensures that the IdM system doesn't delete attribute values on the end system when it can't calculate them.

## Password for confidential storage

In previous IdM versions (>7.6.0), confidential storage may store passwords for remote connector servers for systems that don’t exist. If you encounter difficulties with stored passwords you can remove these values by a database query. From version 7.6.0 onwards, all values from the confidential storage are deleted once a system has been removed.