API - Requirements for a system to be connected

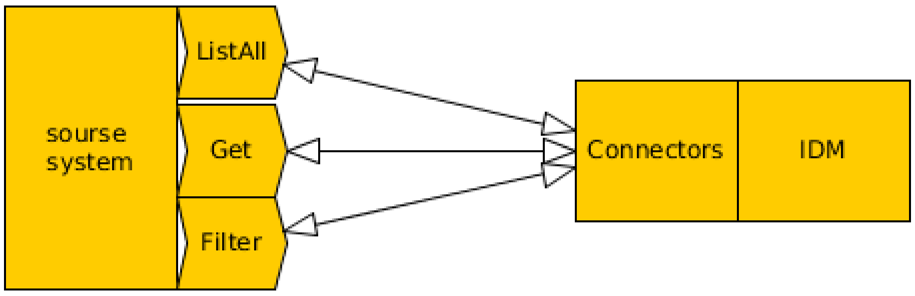
This tutorial is intended as a guide for an administrator/architect planning to connect a system to CzechIdM. In order to connect a system to CzechIdM, there are some minimal requirements on the system's API.

CzechIdM supports many connectors and many others can be developed. Thus one can connect virtually any system to CzechIdM. In general, if a system's API supports operations described below, it can be connected to CzechIdM.

A source system or managed system?

The requirements naturally differ for source systems (synchronization to CzechIdM) and managed systems (data provisioning from CzechIdM). These are the operations that a system's API must support in order to be able to connect it:

a) Source systems - Synchronization



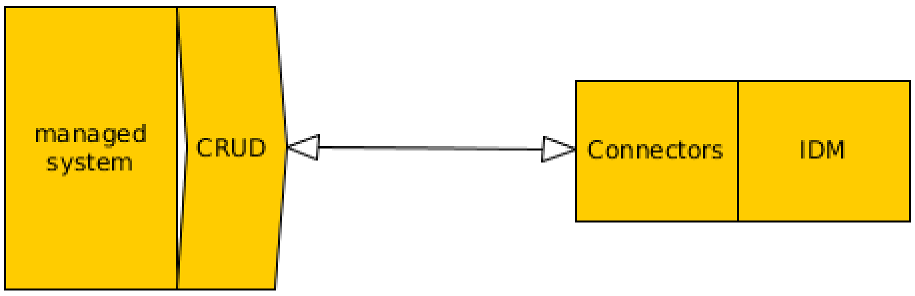
Clearly, for fetching data into CzechIdM, the connected system must be able to provide it. As a minimum, it must provide the list of all items and one item by its ID:

Operation	**Description**	**Operation parameters**	**Output**
List All	Operation should provide list of all objects in your system, at least their IDs		List of all objects, e.g. Identities
Get Object	When CzechIdM want to load one specific user	id - identifier of user needed	One specific object

The following operation is not a must, but it's useful when synchronizing the Czech IdM data with outside data, but only the data that has changed since the last synchronization run. It is typically based on the value of some special attribute like LastUpdateTime. It is especially useful in cases when there are many objects to be synchronized (over thousands) and/or the connected system is somewhat slow.

Operation	Description	Operation parameters	Output
Search objects	When one needs to synchronize object only specific objects, e.g. only those that have been changed since the last synchronization run.	List of objects	filter parameter - the token which is compared to some object attribute value - typically last update time

b) Managed systems - Provisioning



To be able to provide data to a connected system, its API must support the CRUD operations:

Operation	Description	Operation parameters	Output
Get Object	When CzechIdM wants to load one specific object	id of the object	One specific object
Delete Object	Delete object in a system	id of the object we want to delete	id of the deleted object
Update Object	After changing some attribute of an identity in CzechIdM, we want to provide the change into the managed system. If you also want to update the user's permissions in the managed system (e.g. user's AD Groups), the operation should support it too or provide an operation that does the job on its own	id of the user we want to update	id - identifier of the updated object
Create Object	Having created a new identity in CzechIdM, we want it in the managed system, too. This operation should work both with one input parameter (id) and without it	id of the object	id - if no input parameter has been given

If still in doubt, look at the example below. It might give you a better clue of what it should look like.

Example of REST

If you want to connect your system via REST, just prepare the endpoint as described and we can use our REST connector. It supports both provisioning and synchronization.

Endpoint	**Operation type**	**Corresponding operation**	**Input Parameters**	**Description**	**Output**
/users	Get	List All		This endpoints returns information about all identities in the system.	list of all users
/users/{id}	Delete	Delete Object	ID	This endpoint is used for deleting the existing users.	
/users/{id}	Get	Get Object	ID	This endpoint returns information about a user with a specific identifier.	one specific user
/users/{id}	Put	Create/Update Object	ID	This endpoint is used for updating existing users and creating new users with a known identifier in the system.	ID (TODO check)
/users	Post	Create Object		This endpoint is used for creating new users in the system.	ID (TODO check)

As you can see, the operations in this example fully correspond to the generic API description presented in the previous chapters.

Example of DB Table

If you want to connect one DB table to CzechIdM, your DBMS must support CRUD operations (which it does, naturally 🙂). So your main concern is to set the appropriate rights for the user that CzechIdM will use for the connection to DB.

SQL operation	**Corresponding operation**	**Input Parameters**	**Output**
SELECT * FROM <TABLE>	List All		list of all users
DELETE FROM <TABLE> WHERE <ID> = {userID}	Delete	userID	
SELECT * FROM <TABLE> WHERE <ID> = {userID}	Get Object	userID	one specific user
INSERT INTO <TABLE> VALUES «ID>={userID},…>	Create	userID	
UPDATE <TABLE> SET {attributes} WHERE <ID>={userID}	Update	attributes with values, userID	

<TABLE> - connected DB Table <ID> - the name of the column with the unique identifier - e.g. primary key

As you can see, the operations in this example fully correspond to the generic API description presented in the previous chapters.

A source system's requirements: identities, contracts, organizations

Each source system must contain some information based on the type of the connected system no matter the way in which the system is connected (REST, DB, CSV...). There three basic types of connected source systems: source of identities, source of contracts, and source of organizations.

Source of identities

An identity is a set of information describing one person. In general, this information should be independent of the user's position within the organization so it shouldn't contain information about e. g. superiors or work positions.

Must contain:

- ID - a unique identifier of the identity
- User name - a user name (or login) used to log in to CzechIdM

Can contain (often contains):

- Personal number
- First name
- Surname
- Phone
- Email
- Title before name
- Title after name

Cannot contain:

- Information about the user's position (belongs to contracts)
- Information about the user's superiors (belongs to contracts)

Source of contracts

A contract represents the relation of an identity to the company or organization. One identity can have multiple contracts and, as a result, different sets of permissions.

Must contain:

- Contract ID - a unique identifier of the contract
- Identity ID - an identifier of the identity to which to contract belongs

Can contain (often contains):

- Valid from - the start of the validity of the contract
- Valid till - the end of the validity of the contract
- Name of the position

- Department ID - the position within the organization, see below
- Department name
- Superior - either by Identity ID or Login

Source of organizations

IdM can use tree structures to represent the organizational structure of the company.

Must contain:

- Code - a unique identifier of the tree node

Can contain:

- Name of the tree node
- Superior element - can be null in which case it is considered to be the root element by default