Systems - Groups synchronization workflow

workflow, ad, ldap, roles, groups, synchronization, member, memberOf

Current stable version of this workflow can be found in extras module. Please **is highly recommended** use version from extras instead of core version.

This tutorial is intended as a guide to modify workflow for synchronization groups from Active Directory. After modifying this workflow, synchronization of groups can:

- create automatic roles by:
    - organization structure
    - attributes
- create role catalog
- roles assigned to catalog
- provisioning of membership of identities to another system
- resolve membership - users already have assigned groups in another system

For management of membership there is currently a few special chars, which are unsupported. In name of roles, there cannot be: " ' \

Before you start

In this tutorial we will be using Eclipse and Activiti framework.

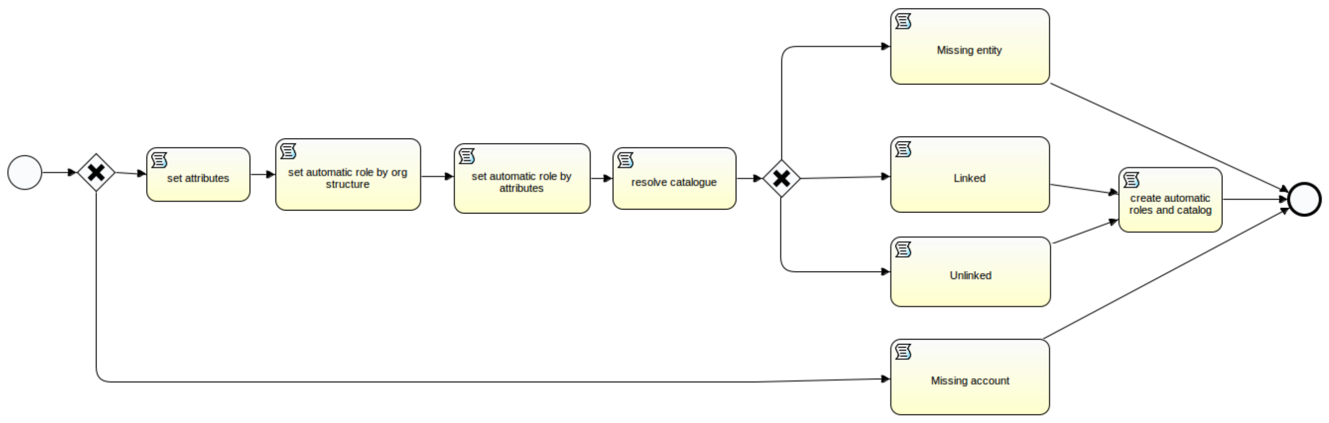
Workflow

For synchronization groups from AD, we have prepared workflow "syncRoleLdap.bpmn20.xml"

The workflow has modified these actions:

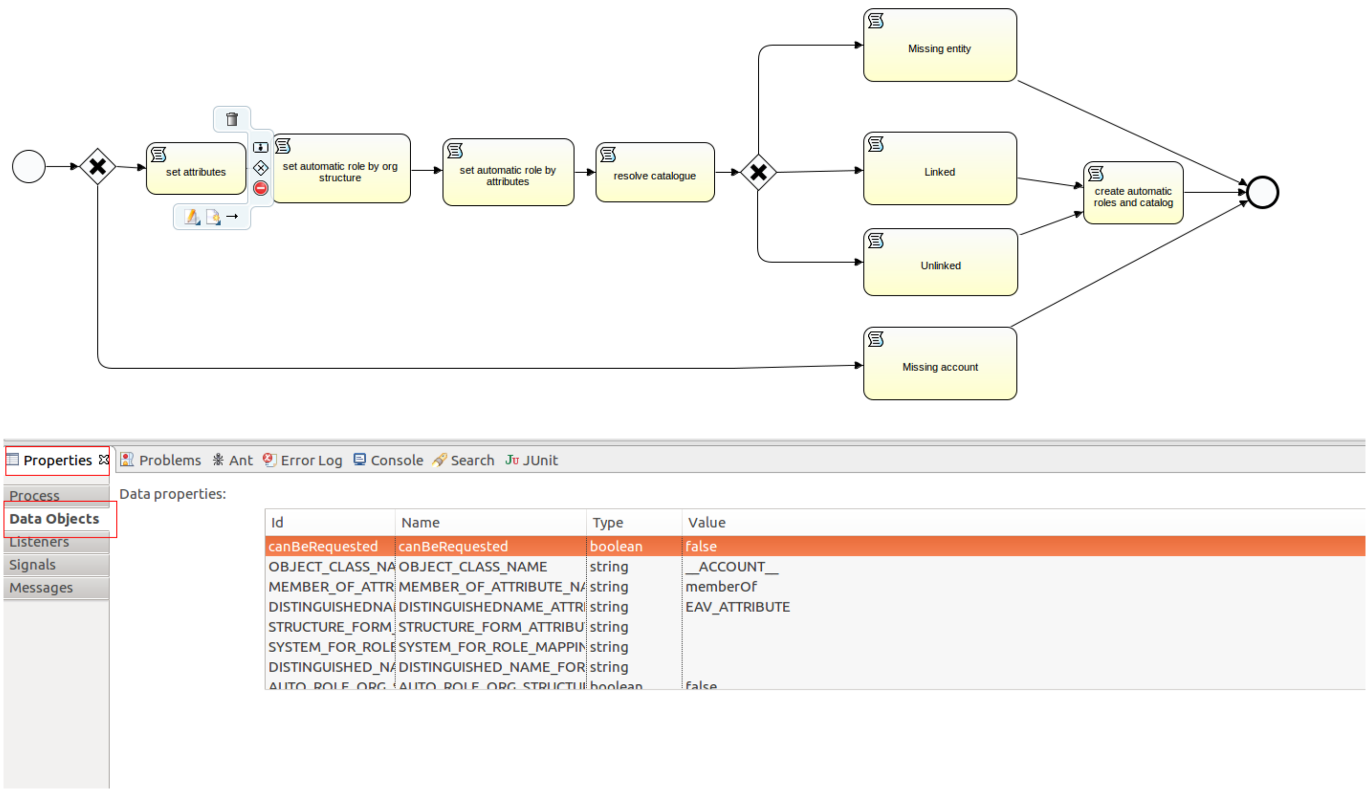
- Linked/update entity
- Missing entity/create entity
- Unlinked/create link and update entity.

Workflow is divided into 9 activities, but only the first four activities are mainly to be modified.


set attributes
set automatic role by org structure
set automatic role by attributes
resolve catalogue
Missing entity
Linked
Unlinked
Missing account
create automatic roles and catalog

Data Objects

First of all is crucial to set data objects in **Data Objects** tab.



Data properties:

Id	Name	Type	Value
canBeRequested	canBeRequested	boolean	false
OBJECT_CLASS_NA	OBJECT_CLASS_NAME	string	__ACCOUNT__
MEMBER_OF_ATTR	MEMBER_OF_ATTRIBUTE_NA	string	memberOf
DISTINGUISHEDNA	DISTINGUISHEDNAME_ATTRI	string	EAV_ATTRIBUTE
STRUCTURE_FORM_	STRUCTURE_FORM_ATTRIBU	string	
SYSTEM_FOR_ROLE	SYSTEM_FOR_ROLE_MAPPIN	string	
DISTINGUISHED_NA	DISTINGUISHED_NAME_FOR	string	
AUTO_ROLE_ORG_	AUTO_ROLE_ORG_STRUCTU	boolean	false

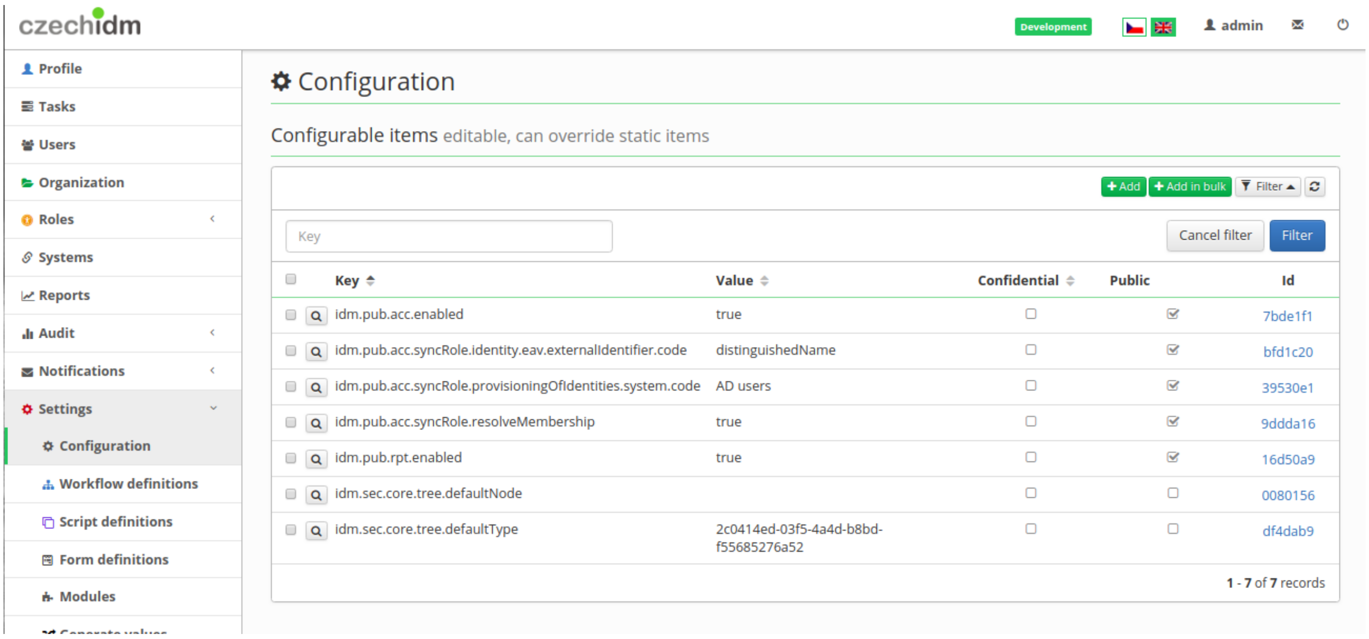
- STRUCTURE_FORM_ATTRIBUTE - code of eav of organization tree node. It is used to add an automatic role by organization structure. And every organization structure should have an eav with some value to determine, which role will have an automatic role on which tree node.
- CODE_OF_TYPE_STRUCTURE - code of structure, this define which organization tree will be used for searching/creating automatic roles for organization structure. Default value means default organization tree.

- AUTO_ROLE_ORG_STRUCTURE_ENABLED - if false, automatic roles by attribute methods are skipped.

- AUTO_ROLE_ATTRIBUTES_ENABLED - if false, automatic roles by organization structure methods are skipped.

Set aplication properties

Workflow was updated and now you can just add properties to application and you do not have to anyhow change workflow file. Application property are specified in **Settings** agenda in **Configuration** tab (like you can see on picture bellow).



Configuration

Configurable items editable, can override static items

Key	Value	Confidential	Public	Id
idm.pub.acc.enabled	true	☐	✓	7bde1f1
idm.pub.acc.syncRole.identity.eav.externalIdentifier.code	distinguishedName	☐	✓	bfd1c20
idm.pub.acc.syncRole.provisioningOfIdentities.system.code	AD users	☐	✓	39530e1
idm.pub.acc.syncRole.resolveMembership	true	☐	✓	9ddda16
idm.pub.rpt.enabled	true	☐	✓	16d50a9
idm.sec.core.tree.defaultNode		☐	☐	0080156
idm.sec.core.tree.defaultType	2c0414ed-03f5-4a4d-b8bd-f55685276a52	☐	☐	df4dab9

1 - 7 of 7 records

With button **Add** you can add any property described below and configure workflow.

Following properties are used, when the workflow is used for creating roles that manage group membership of accounts in a connected system:

- **idm.pub.acc.syncRole.provisioningOfIdentities.system.code** - (default: null, **_mandatory_**) - it is code (name) of the system, which is used for provisioning the identities to the system. E.g. "AD users". It is mandatory attribute, otherwise workflow will not be working.
- **idm.pub.acc.syncRole.system.mapping.objectClassName** - (default: __ACCOUNT__) - this is important to provisioning member attribute of identity. It is an object class name of identity schema. It should stay "__ACCOUNT__"
- **idm.pub.acc.syncRole.system.mapping.attributeMemberOf** - (default: ldapGroups) - it is the name of an attribute in a mapping of identity provisioning. It is usually ldapGroups (recommended) or memberOf. This attribute will be added to role's mapping with transformation script (which will be set later).
- **idm.pub.acc.syncRole.system.mapping.attributeRoleIdentificator** - (default: distinguishedName) - the name of an attribute in the connector, which holds the distinguished name of a role object.

Managing group membership of account - more special options for the roles:

- **idm.pub.acc.syncRole.roleSystem.forwardManagement.value** - (default: false) - every role, which manages group membership on the connected system, has the option forward

account management. This property can set this option.
- **idm.pub.acc.syncRole.roleSystem.update.manageforwardManagement** - (default: false) - this property will manage forward account management option even when updating existing roles by the synchronization.
- **idm.pub.acc.syncRole.roles.nameOfRoles.doNotSentValueOnExclusion** - (default: null) - every role, which manages group membership on the connected system, has the option to skip the value if it's assigned on an excluded contract (see the tutorial about this). Add to this property names of new roles separated with comma, which should have this option set (i.e., they should be skipped when the contract is excluded). (Does not work with roles, which have comma in name.)
- **idm.pub.acc.syncRole.roles.update.nameOfRoles.manageSentValueOnExclusion** - (default: false) - this property will set the option skip the value if it's assigned on an excluded contract even when updating existing roles by the synchronization. The option will be set to all roles in the property `idm.pub.acc.syncRole.roles.nameOfRoles.doNotSentValueOnExclusion` and unset to all other AD roles.

The workflow enables loading group membership from the system. That means, if the group in AD have some members and you want to assign roles to identities IdM based on that, you can use this workflow to do it. Typically, you would do it only as an **initial** loading. Necessary properties:

- **idm.pub.acc.syncRole.identity.eav.externalIdentifier.code** - (default: null, **_mandatory for resolving group membership_**) - code of EAV with a distinguished name of identities. It is used when creating a new role for a new group, or when loading group membership for existing roles. All identities managed by IdM in AD have to have the EAV containing their current **distinguishedName**, otherwise resolving membership will not work. Load the value to the EAV (recommended name: **distinguishedName**) when you run reconciliation of identities. Fix Me! create a tutorial. Then set the value "distinguishedName" to this property. **Once you load the group membership, remove this property, so the membership won't be loaded from AD for new roles** (IdM should be the authority for group membership in the standard production use).
- **idm.pub.acc.syncRole.update.resolveMembership** - (true/false, default: false) - with this property you can turn on resolving memberships of roles even in other situations than creating role. Recommended way: Turn it on for initial loading of group membership, turn it off afterwards. Note that resolving membership of newly created roles is done independently of this property, see the note in the property `idm.pub.acc.syncRole.identity.eav.externalIdentifier.code` above.
- **idm.pub.acc.syncRole.roles.attributeNameOfMembership** - (default: member) - it is name of attribute of role in source system, which holds identificators of identities. The default value is typical for AD.

Settings of the created role - properties connected to the requesting of roles and the role approval:

- **idm.pub.acc.syncRole.role.canBeRequested** - (true/false, default: false) - sets to all roles, if the role can be requested by common users (not superadmin)
- **idm.pub.acc.syncRole.role.update.manageCanBeRequested** - (true/false, default: false) - enable/disable setting can-be-requested role attribute
- **idm.pub.acc.syncRole.roles.create.priorityOfRoles** - (default: null, values: 1,2,3,4) - this property will set criticality (priority) of roles, which affects the approval process. **Only on create.**

- **idm.pub.acc.syncRole.roles.create.garanteeOfRoles** - (default: null) - fill in name of role, which will become the Role authorizer of all created roles. **Only on create.**

The workflow can create the folders in the role catalogue. It can be either one folder, or the tree structure of folders based on the DNs of the roles:

- **idm.pub.acc.syncRole.roleCatalog.ResolveCatalog** - (true/false, default: true) - this property enables creating of new folders in the role catalogue. Set it to false if you don't want to create and add roles to the role catalogue at all.
- **idm.pub.acc.syncRole.roles.allToOneCatalog** - (default: null) - use this property, if you want to add all roles in one folder. Set the name of the folder to this property. If a folder with the same code already exists, the workflow will create a new one (it won't reuse already existing folder).
- **idm.pub.acc.syncRole.roleCatalog.catalogueTreeInOneCatalog** - (default: null) - use this property, if you want to create and add roles in a tree structure of folders. Set the name of the root folder to this property. The workflow will create a tree of folders under this root folder. If this property will be changed, new catalog folder will be created. Name of folders in the role catalogue can be changed in IdM.

Since Extras version 1.8.0 you can use two new options which will help with following use case: I have more then 1 AD system connected as group source. Now the workflow has "global" configuration via application properties so I am not able to run scheduled synchronization and put group from one AD to catalog "one" and from second AD to catalog "two" and similar issue is with mapped systems. This changes are backward compatible because if you don't set these new properties the WF behavior is same as in previous version. If you set this property then the new behavior will be turned on.

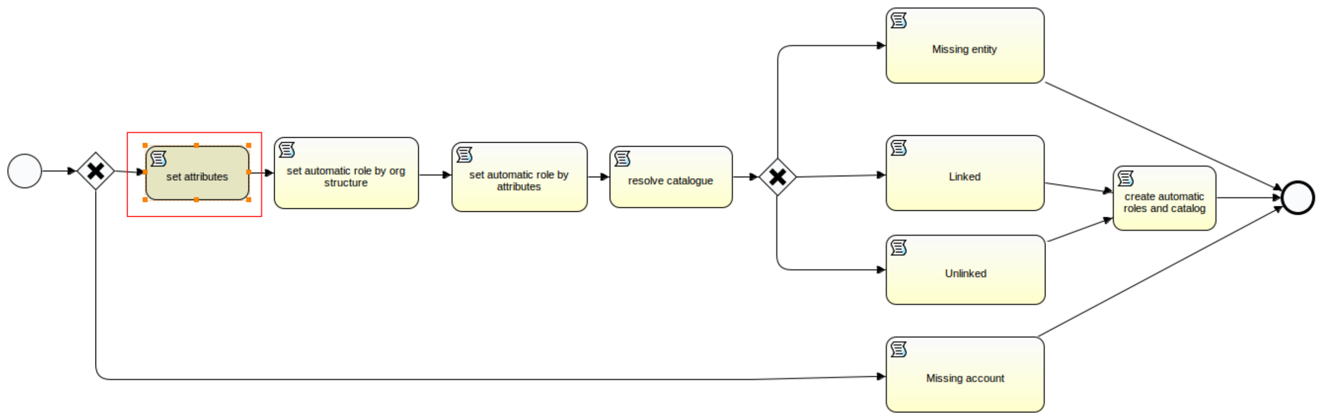
- **idm.pub.acc.syncRole.roles.catalogByCodeList** - UUID of code list for catalogs for each system. Item in code list has UUID of source system and as value they have code of role catalog
- **idm.pub.acc.syncRole.provisioningOfIdentities.codeList** - UUID of code list for mapped sysmtes for each system. Item in code list has UUID value of AD group system which is used for synchronization and as value UUID of AD system which is used for user provisioning

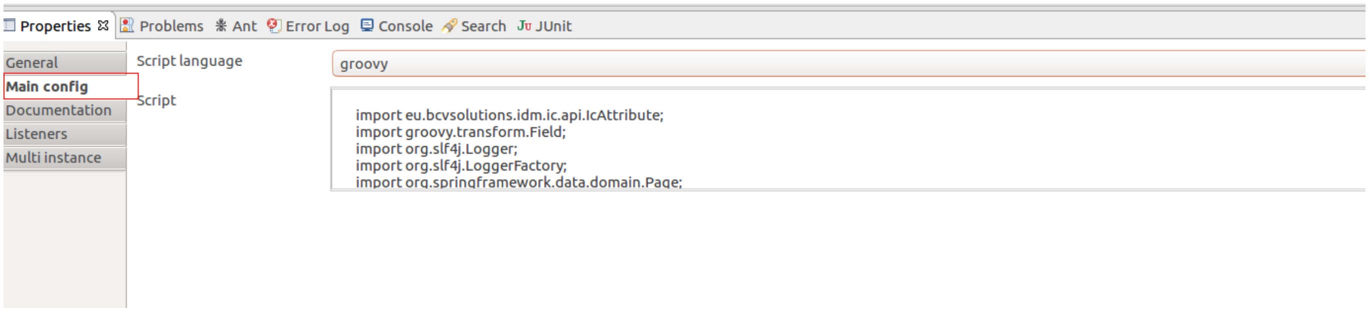
Set attributes activity

This activity is to get some attributes from icAttributes as name and distinguished name. You can get another if they will be needed in the following activities. And also there is setted attribute:

```
execution.setVariable("TRANSFORMATION_SCRIPT", "\""+distinguishedName+"\"");
```

value of this attribute will be used as transformation script. (It will be used for provisioning of member attribute of identities).


set attributes
set automatic role by org structure
set automatic role by attributes
resolve catalogue
Missing entity
Linked
Unlinked
Missing account
create automatic roles and catalog



Script language: groovy

Script:

```
import eu.bcvsolutions.idm.ic.api.IcAttribute;
import groovy.transform.Field;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.data.domain.Page;
```

Set automatic role by org structure

In this script, it is crucial to determine for a role to find tree node (organization) to create an automatic role by organization structure and set it to a variable like:

```
execution.setVariable("organizationTreeNode", pomList.get(0));
```

This script can be just slightly changed to work perfectly fine. In the default state, it is searching for a tree node, which has a name of the role in eav. In a previous activity, we can get even another attribute and with substring, it should be enough.

Set automatic role by attributes

This part is to create attributes for an automatic role. There can be more attributes, but only for identity/contract attributes or identity/contract eavs. You should create attributes (by default algorithm) put them into the field and lastly into "automaticAttributes" variable.

Resolve catalogue

This part is about Roles catalog. There are two methods. The first method creates catalog based on the Distinguished name of the role. The rightmost value of OU is a root of the catalog and then other OUs from the right are represented as folders and roles are assigned in the leftmost OU/catalog.

The other method is for removing folders of a catalog, if a role is transferred to another catalog, find the old one and set it in "catalogue" variable, and it will be erased.

This should be all, but if you made bigger changes in methods you should probably test your new workflow. After modifying the workflow, don't forget to update this workflow in IdM.