WinRM + AD Connector

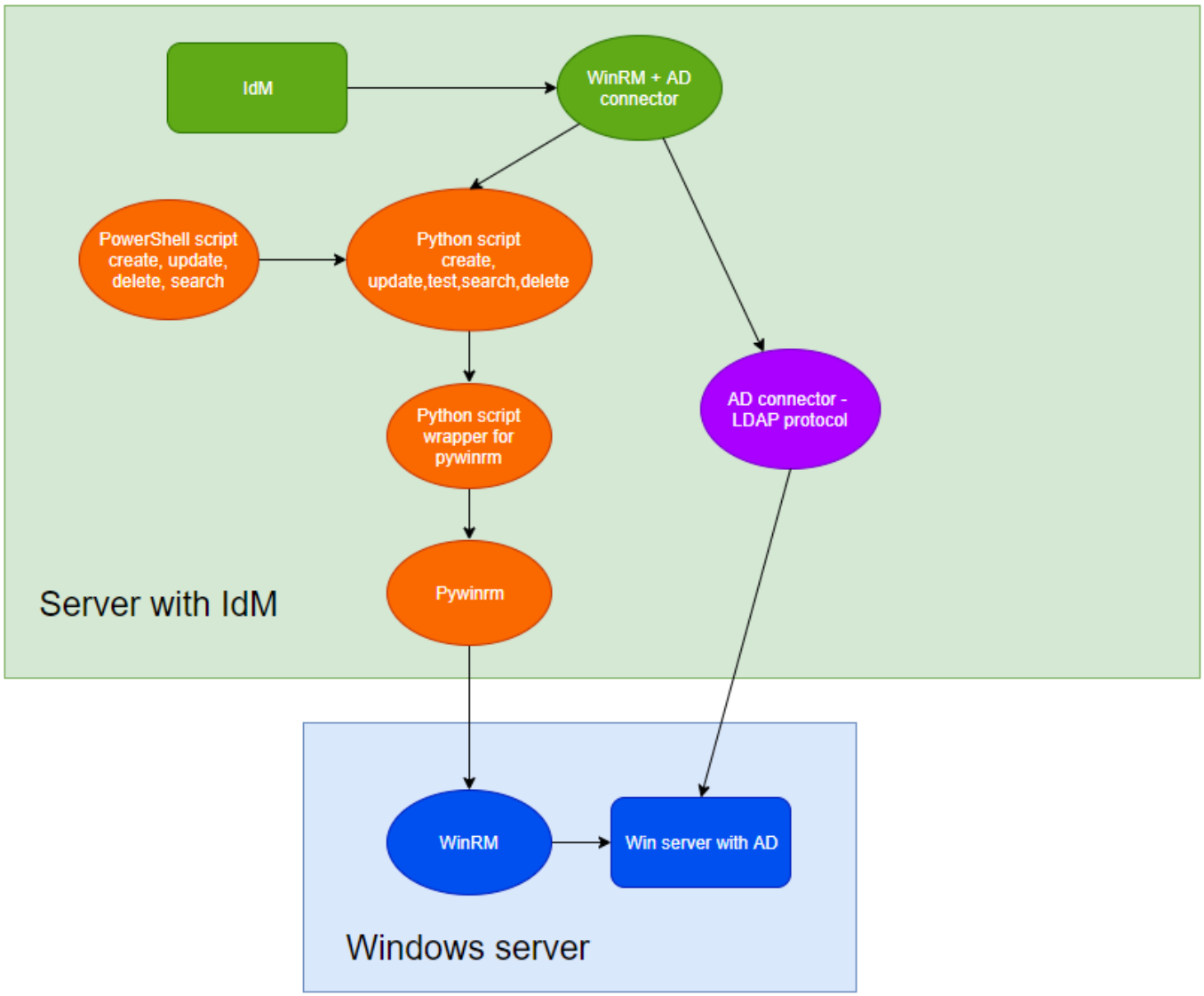
This connector is combining WinRM and AD connector into one. The main advantage of this is you can execute operation by AD connector or by WinRM connector or with both together in specified order.

General information

Typical use cases for this combined connector are:

- Management of home directories - User is created via AD connector and home directory is created by WinRM Connector (powershell)
- Management of o365
- Management of Exchange
- Management of OpenLims via special client which is on the windows servers and is executed from powershelll
- Basically you use this to connect to system which can be controlled via powershell and is dependent on AD.

Schema:



When you use this connector then in IdM you will has only one system and every user who is managed via this system will have only one account. For example if you want to manage home directories together with AD then user will have only one account and so when you create user, directory will be created to.

Theoretically you can use WinRM connector for home directories and AD connector for user management separately. You will have two system in IdM and user will have two accounts. But then you will have no control over the order of execution. And when you need to set some ACL permissions to the home directory the user must be created before.

When you want to execute some operation via both connectors and the first connector execution will failed then the execution by the second connector is not executed. You will see error in provisioning in IdM. In case where the second execution will fail you will see error in IdM again. Then when retry provisioning will kick in, IdM perform search to the end system again that mean if you want for example assign role in AD to user and then execute powershell for Exchange and the powershell execution will fail for some reason. Retry provisioning will know that the role is already assigned so nothing will happen via AD connector and only powershell will be executed.

WinRM Connector

This part is only what is supported by WinRM connector. AD connector has the same functionality as if you use the standalone version.

Windows Remote Management (WinRM) connector can be used to connect to basically to any system which can be managed via powershell commands or some specialized client which can be called from powershell.

Connector is based on Connid CMD connector. We made fork of CMD connector version 0.4-SNAPSHOT.

We implemented some features which were missing.

- It contains more configuration fields for connecting to WinRM, which is the main purpose of this connector.
- Password for WinRM user is GuardedString in connector but we send is as plain text in to bash script. (This behavior is same in CMD connector for __PASSWORD__ attribute)
- If script return exit code other then 0 exceptions is thrown.
- Item In folder scripts/NameOfSystem you can find python scripts for each supported operation method:
  - Create
  - Update
  - Delete
  - Test
  - Search

Where "NameOfSystem" is one of these following values Exchange, OpenLims, o365, homeDir (More systems will maybe come in future). If you want use this connector for another system you can just implement scripts yourself. As a template you can use existing python + ps scripts.

Powershell scripts are in subfolders. It's not only "normal" powershell script which contains the commands which we want to execute, but it must handle exceptions and in the case of search scripts the response should be in json format, so we can parse in connector a forward it to IdM. The risk of not catching exceptions can be that IdM will show operation as successful but it failed or the other way around.

All of these scripts logging into connector server log

Then in folder "scripts" you can find python script, which is wrapper for pywinrm client - https://github.com/diyan/pywinrm which is used for connecting and executing PS scripts in windows server. You need to install first. In the link above there is a tutorial.

It's better to run it in connector server instead of directly adding dependency to your application(IdM). The reason for this is simple - better security. You can choose user with some limited permissions which will be used as the owner of connector server and then give him access to run only the scripts which you want.

It supports basic, ntlm, kerberos and credssp authentication schema for WinRM

It supports HTTP and HTTPS communication. HTTPS communication can be a little bit tricky to

configure. You need the right certificate which is used in WinRM listener on Win server and then import crt to the trust store on machine where this connector is running or you can edit file winrm_wrapper.py to change the path to .pem certificate which is needed for HTTPS connection.

```
p = winrm.protocol.Protocol(endpoint=endpoint,
                            transport=authentication,
                            username=user,
                            password=password,
                            ca_trust_path='/opt/connid-connector-
server/certs/winrm_ca.pem')
```

Schema generation

Connector is supporting basic schema generation. You will get these attributes:

- __NAME__
- __UID__
- __PASSWORD__

You need to create other attributes manually based on the system which you want to connect and you needs.

Requirements

Here are some requirements needed in order to use winrm connector.

- Windows server with activated winrm service
- User account (local when using basic authentication scheme, domain otherwise) with correct permissions for connecting wia winrm and executing powershell scripts
- Correctly set up network access and firewall rules to allow winrm communication from IdM server to desired server
- Allowed winrm ports - 5985 for HTTP and 5986 for HTTPS (by default)
- Write powershell scripts, which will be performing desired operations on MS server (CREATE, UPDATE, ...)
- Write python scripts that transform data from ConId API to powershell script (examples in GIT repository)

Provisioning

For objectClass GROUPS provisioning is not supported in current version.

For objectClass ACCOUNT, the connector is supporting these operations: CREATE, UPDATE, DELETE, SEARCH.

Synchronization

For ACCOUNT you need to use Reconciliation, normal synchronization is not supported in current version.

Supported operations

Object	Operations
__ACCOUNT__	CREATE, UPDATE, DELETE, SEARCH
__GROUP__	NONE

Managing users groups

When you use this connector for some system where you need to manage groups for users (OpenLims). Attribute for roles must be called "roles" is schema definition. Then it's work in the same way as roles for AD. That's mean you need to create role in IdM which will have assigned this system and in mapping override attribute "roles" with value which the system accept. Strategy should be Merge or Authoritative merge

Scripts

python

Python scripts should start with these two lines:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
```

The second line is important because in python 2.x default encoding is ASCII so if don't specify the encoding in python file then we will have problems with using diacritics. Then if we need to load powershell script into python and replace some params, It's recommended to open with encoding.

```
import codecs
f = codecs.open(os.environ["script"], encoding='utf-8', mode='r')
#os.environ["script"] is path to script which is send from IdM configuration
command = f.read()
command = command.replace("$firstName", winrm_wrapper.getParam("firstName"))
```

For getting parameter from environment you can use method in winrm_wrapper which will return value or empty string if the variable is not in environment. It will return value as unicode with utf-8 encoding

We are using encoding otherwise you will have problem with diacritics in powershell when you want to encode the powershell script before sending it via WinRM.

Installation

For using WinRM part of this connector you need to install a few things which is needed, otherwise you can skip these steps.

- Install python, tested version is 2.7
- Install pip for managing Python packages - for linux use package managers based on you distribution and install package python-pip. If you are using windows pip will be installed together with python if you use official installator.
- Install pywinrm and dependencies. You can follow official guide https://github.com/diyan/pywinrm Just don't forget to install additional packages if you want use Kerberos or CredSSP authentication. If you are using windows you need to execute only the commands for pip, you don't need to install other system dependencies.

Better way to install python packages through `pip` is to **not** install them system-wide. Create user for the connector server (see later on this page) and install packages only for this user. This ensures stability (system-wide updates do not change versions, thus cannot break your connector) and isolation from the rest of the OS (`pip` does not accidentally break OS-provided libraries). To install what you need, just issue:

```
su - connector-server
pip install --user pywinrm

#those only if you need them
pip install --user pywinrm[kerberos]
pip install --user pywinrm[credssp]
```

Now we have prepared the tool which is used by our connector. Next you need to install java connector server. Connector server is not mandatory but as we wrote in the first section it's recommended to use it.

Configure log rotation for connector server log file

You can download whole bundle with prepared and tested connector server https://git.bcvsolutions.eu/modules/connector-server/tree/develop

It's private at this time

Or you can follow this guide and prepare the connector server yourself if you want. This connector is tested in java connector server 1.4.5.1 https://connid.atlassian.net/wiki/spaces/BASE/pages/360458/Downloads#Downloads-JavaConnectorServer and with connector-framework 1.4.3.0

Next you will need to add these libraries into lib folder of the connector server:

- jackson-annotations-2.9.8
- jackson-core-2.9.8
- jackson-databind-2.9.8

You will probably need to add these libs into classpath in ConnectorServer.sh or ConnectorServer.bat it depends on your OS.

Next it's good to do some more configuration as setting new password for connector server and create new user under which the connector server will be started.

For setting new password for you remote connector use

```
./bin/ConnectorServer.sh -setKey -key yourKey -properties
conf/connectorserver.properties
```

If you want to be able to run connector server as a service follow next steps

```
# create user which we run the connector server
useradd connector-server

#create file
/etc/systemd/system/java-connector-server.service

# content of the file, change path according where you have your connector
server
[Unit]
Description=Java Connector Server Service
[Service]
User=connector-server
WorkingDirectory=/opt/connid-connector-server
ExecStart=/bin/bash /opt/connid-connector-server/bin/ConnectorServer.sh -run
-properties /opt/connid-connector-server/conf/connectorserver.properties
SuccessExitStatus=143
[Install]
WantedBy=multi-user.target

# Reload and enable deamon
systemctl daemon-reload
systemctl enable java-connector-server

# Use this to start/stop/status
systemctl start java-connector-server
systemctl stop java-connector-server
systemctl status java-connector-server
```

Now you can put winrm-ad-connector-1.0.1.jar to the bundles folder inside connector server and you can start it.

Next thing which you need to do is configure WinRM on windows server or check if WinRM is

accessible. You can follow steps from out tutorial

Configuration

In configuration you have the option to configure AD connector and WinRM connector. So follow WinRM configuration below and AD configuration.

Connector has few settings which need to be configured before you used it.

If your connector server is running on Windows then you need to enter "python " before the actual path to script. E.g. "python C:\scripts\homeDir\testDir.py"

Create script

Path to Python create script

Powershell create script

Path to powershell create script which will be loaded into python and executed on Windows

Update script

Path to Python update script

Powershell update script

Path to powershell update script which will be loaded into python and executed on Windows

Search script

Path to Python search script

Powershell search script

Path to powershell search script which will be loaded into python and executed on Windows

Delete script

Path to Python delete script

Powershell delete script

Path to powershell delete script which will be loaded into python and executed on Windows

Test script

Path to Python test script

Endpoint

URL to the endpoint, where is WinRM accessible. Usually https://HOST:5986/wsman for HTTPS and http://HOST:5985/wsman for HTTP

Authentication schema

One from supported values - basic, ntlm, kerberos, credssp

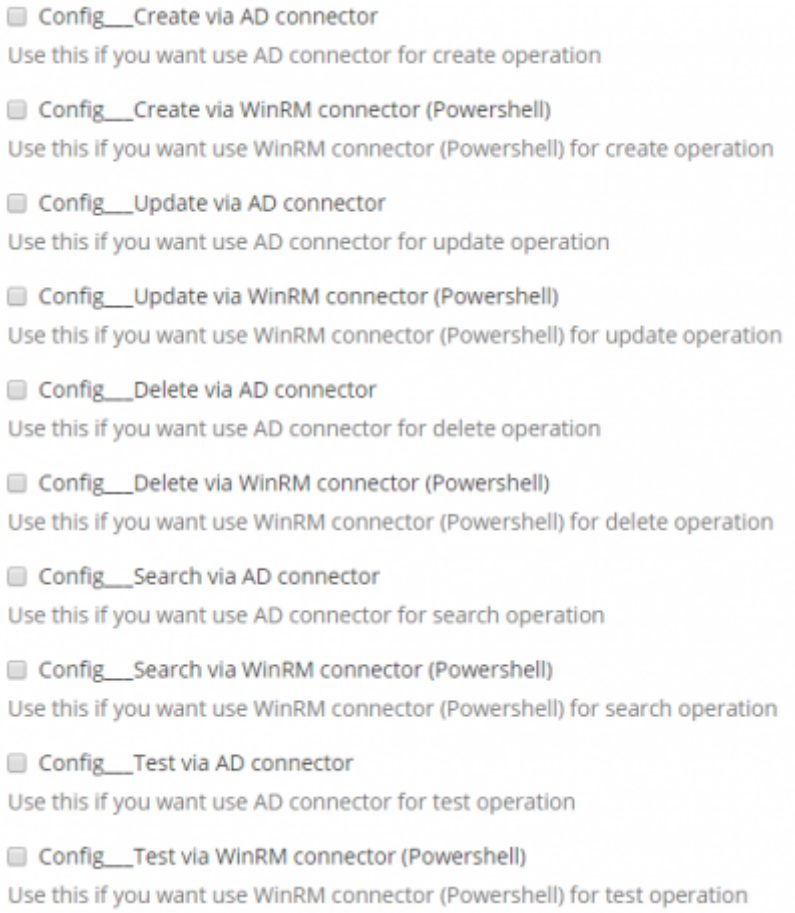
User

Username for user which will be used for authentication to WinRM

Password

Password for this user

Then there are some other options which can be configured. You can configure which connector will be used for which operation. For example you can use AD + WinRM for create and only WinRM for delete, etc.



You can configure the order of connectors. Default behavior is that AD connector is first.