WinRM + AD Connector

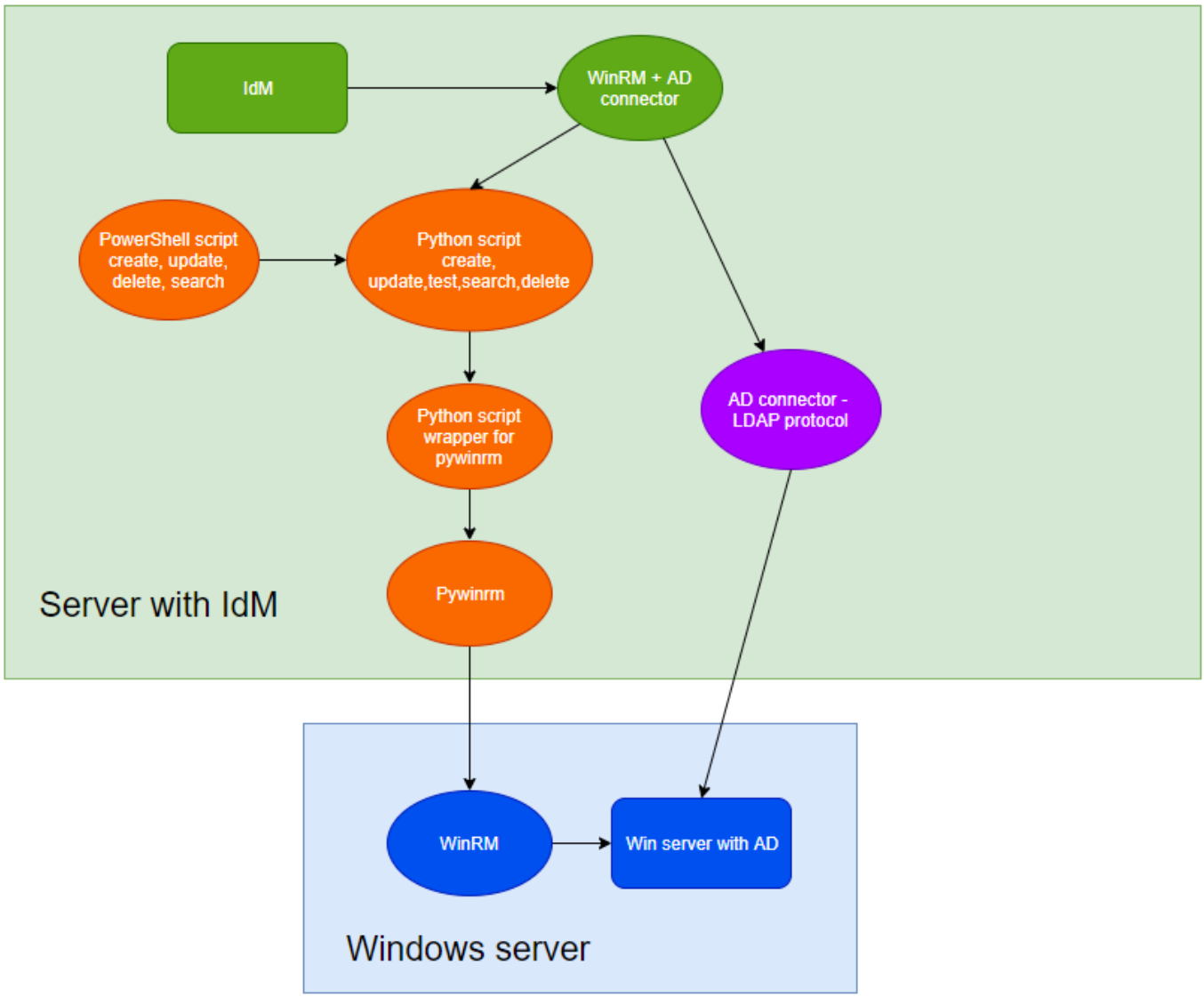
This connector is combining WinRM and AD connector into one. The main advantage of this is you can execute operation by AD connector or by WinRM connector or with both together in specified order.

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). Owner of home directory can be set only locally.
- 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 have 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 fails then the execution by the second connector will not be executed. You will see error in provisioning in IdM. In case the second execution fails you will see error in IdM again. Then when retry provisioning kicks in, IdM performs search to the end system again, that means if you want for example assign role in AD to user and then execute powershell for Exchange and the powershell execution fails 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. All log messages are shown after powershell script is executed and the control is returned into connector. So it can see that the log is frozen if the powershell script will run some time. Disadvantage is, if your powershell script froze for real you will not see any log. This can happen for example if you execute some command which will wait for user input, but you can prevent this one by using special parametr

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. To use Kerberos, you need to have properly-configured `/etc/krb5.conf` in your system. (see https://doc.czechidm.com/doc-admin-guide/1.3/connector-server.html#_configuring_kerberos_support)

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. Store the crt to the on the machine where this connector is running and for: **WinRM < 1.0.5** Edit 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')
```

WinRM >= 1.0.5 there is configuration field called "WinRM___CA trust path" - Path to certificate which will be used in HTTPS communication. E.g /path/to/file/crt.pem Be sure you are using up to date winrm_wrapper.py otherwise this new config property don't work and you will be forced to use previous solution.

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 via 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 ConnId API to PowerShell script (examples in GIT repository https://github.com/bcvsolutions/winrm-ad-connector)

Version and compatibility

- 1.0.0 - IdM 9.x and above
- 1.0.1 - IdM 9.x and above
- 1.0.2 - IdM 9.x and above

Cross domain feature available:

- 1.0.3 - IdM LTS 9.7.x with Extras module 1.8.1
- 1.0.4 - IdM 10.3.0 and above
- 1.0.5 - IdM 10.3.0 and above
- 1.0.6 - IdM > 10.3.0 < 11.2.0
- 1.0.7 - IdM 11.2.0 and above = CzechIdM supports cross domain. No need for extras module
- 1.0.15 - IdM ??? and above = Cross domain provisioning works from AD connector itself, works for large numbers of groups

If you want to use IdM 10.0.0, 10.1.0 or 10.2.0 and be able to use cross domain functions you need to update IdM to 10.3.0, if you don't need cross domain feature and don't want to update IdM you can try to use connector 1.0.2

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

For more information about how to write scripts, follow How to write scripts for WinRM + AD Connector

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.

When connector server is running on Windows all powershell scripts need to you Write-Output function instead of Write-Host

In windows system python was running in encoding cp1252 and I had to use command below to reconfigure to utf8. This command is available from python version 3.7

```
sys.stdout.reconfigure(encoding='utf-8')
```

Update operation is a little bit trickier. In this script it'll be needed implementation of both create and update operations. Because of some reason there could be user, which already has account on AD, but does not have home directory. And if you need some additional attributes to create a home directory (e.g. for each department different folder), in update operation an attribute will not be normally send if value of the attribute was not changed.

For search and delete operations IdM only sends uid. So in this scripts you cannot use any other attributes. For example someone would want to rename home directory in delete script and leave it there for period of time as backup. But in this situation you can only add to home directory's name some static text

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 versions are 3.6 and 3.9
- 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] python-krb5ticket
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 is strongly

recommended.

1. Follow this howto to install remote connector server as a service.
2. Put `winrm-ad-connector-1.0.1.jar` to the `bundles` folder inside connector server installation and restart the connector server.
3. Put WinRM server's CA certificate in this file inside connector server installation - `certs/winrm\_ca.pem`. It is necessary to supply CA certificate, not the server's certificate. For certificates, use PEM format.
4. Put CA certificate to AD servers in the Java truststore created in this file in the connector server - `conf/truststore.jks`. (How to get CA to AD server? Internal guide 

move to wiki)

1. Configure WinRM on windows server or check if WinRM is accessible. You can follow steps from our 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

CA trust path

Path to certificate which will be used in HTTPS communication. E.g /path/to/file/crt.pem

Ignore CA validation

If you want to connect to WinRM without CA validation - Don't use in production, only for testing!

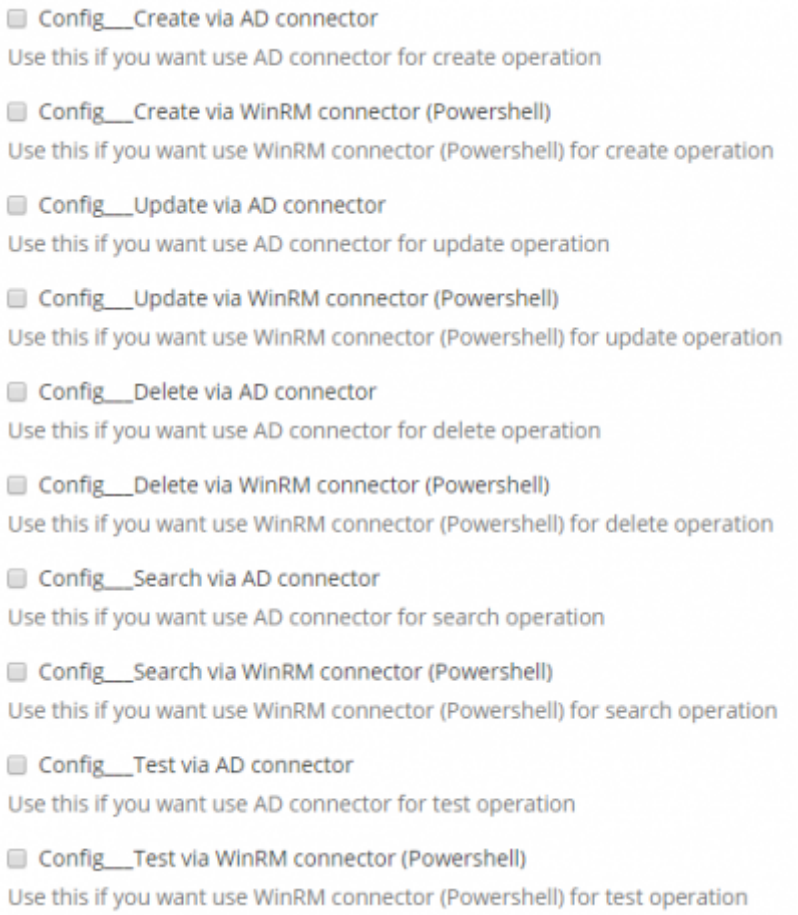
Support AD CrossDomain Provisioning

AD connector will be used to handle cross domain groups. If this is selected, the Use only WinRM for Groups checkbox must be unselected, otherwise the AD connector won't get the necessary data to do anything.

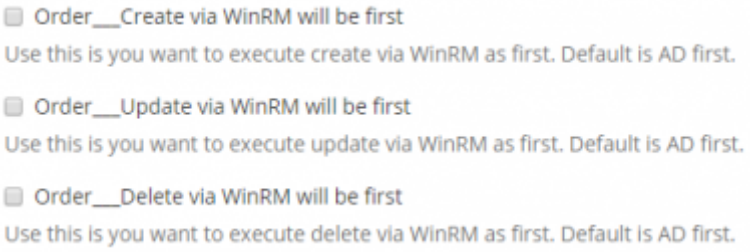
Prohibit WinRMCrossDomainProvisioning

WinRM won't be used for cross domains - this should be used alongside Support ADCrossDomainProvisioning.

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.



Note that when you check both the AD and WinRM options for the Create / Update / Delete operation, the same type of operation is called in both connectors. For example, if one connector creates an object, the other connector tries to create it as well, then the operation ends up with an error because the object already exists (but the first connector does not rollback!).

Cross domain configuration

Principal must be in the form of [domain]\[name] (ZOO\Administrator) - using Distinguished Name from AD works for AD but not for winrm

Adding or removing too many groups at once is not supported if using WinRM to manage groups. Lenght limitation of environment variable on Windows is 8 191 characters so if combination of all added and removed group's DNs plus comas, doublequotas and the script itself is longer than that cmd.exe just ignores it. The result is no script is being run and no error or exception ocures. It is relatively safe to add or remove about 50-100 groups at once.

TODO: Using AD to manage groups should fix this issue - once it is properly tested, write here that when using supportASCrossDomainProvisioning, the connector can add or remove any number of groups

IdM 11.2.0 has support for cross domain. You need connector version 1.0.7

Do the following configuration in IdM:

- Additional___Use only WinRM for groups - true
- Config___Update via WinRM connector (Powershell) - true
- Config___Create via WinRM connector (Powershell) - true

TODO: fix version number when it is released

Idm 14.?.? has support for cross domain via AD. You need connector 1.0.15

Use the following configuration:

- Config___Update via AD connector - true
- Config___Create via AD connector - true
- **Additional___Use only WinRM for groups - false**
- **prohibitWinRMCrossDomainProvisioning - true**
- **supportADCrossDomainProvisioning - true**

Scripts can be found on GitHub

Send attributes only to WinRM

In some cases, when you are using AD and WinRM for same operation, you want to use some attributes only in WinRM (powershell).

The reason is that is some attribute for script and AD has no clue about this attribute and the AD part will fail.

To achieve this, you can specify, which attributes should be send only to WinRM.

- Go to system detail - Configuration
- Go to tab Additional connector configuration
- Click on Manage attributes
- Add new attribute with code attributesForWinRM
    - Attribute is Short text and multivalued
- Save it and go back to Additional connector configuration
- Fill attribute names which should be send only to WinRM
- Each name on it's own lane
- Names should be the ones which are in schema.

Interesting articles about WinRM

- https://www.bloggingforlogging.com/2018/01/24/demystifying-winrm/