



# opsi Directory Connector



# Contents

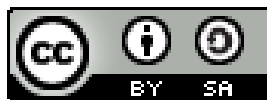
<b>1</b>	<b>Copyright</b>	<b>1</b>
<b>2</b>	<b><i>opsi directory connector</i></b>	<b>2</b>
2.1	Introduction . . . . .	2
2.2	Prerequisites for the opsi extension <i>opsi directory connector</i> . . . . .	2
2.2.1	General Requirements . . . . .	2
2.2.2	Hardware Requirements . . . . .	2
2.2.3	Software Requirements . . . . .	3
2.3	Installation . . . . .	3
2.4	Configuration . . . . .	3
2.4.1	Directory settings . . . . .	3
2.4.2	Configure connection for UCS . . . . .	4
2.4.3	Behaviour settings . . . . .	5
2.4.4	Attribute-Mappings . . . . .	5
2.4.5	Manual assignment of group names . . . . .	6
2.4.6	opsi connection settings . . . . .	6
2.5	Running the connector . . . . .	7
2.5.1	Example: recurring runs with systemd . . . . .	7
2.5.2	Example: recurring runs as cronjob . . . . .	8

# Chapter 1

## Copyright

The Copyright of this manual is held by uib gmbh in Mainz, Germany.

This manual is published under the creative commons license  
*Attribution - ShareAlike* (by-sa).



A German description can be found here:

<https://creativecommons.org/licenses/by-sa/3.0/de/>

The legally binding German license can be found here:

<https://creativecommons.org/licenses/by-sa/3.0/de/legalcode>

The English description can be found here: <https://creativecommons.org/licenses/by-sa/3.0/>

The English license can be found here: <https://creativecommons.org/licenses/by-sa/3.0/legalcode>

Most parts of the opsi software are open source.

The parts of opsi that are not open source are still under a co-funded development. Information about these parts can be found here: [opsi cofunding projects](#)

All the open source code is published under the AGPLv3.



The legally binding AGPLv3 license can be found here: <http://www.gnu.org/licenses/agpl-3.0-standalone.html>

Some information around the AGPL: <http://www.gnu.org/licenses/agpl-3.0.en.html>

For licenses to use opsi in the context of closed software please contact the uib gmbh.

The names *opsi*, *opsi.org*, *open pc server integration* and the opsi logo are registered trade marks of uib gmbh.

## Chapter 2

# *opsi directory connector*

### 2.1 Introduction

The opsi directory connector is a tool to transfer data from a directory service to an opsi installation. This avoids the need of maintaining data in two separate systems.

### 2.2 Prerequisites for the opsi extension *opsi directory connector*

This module is currently a [co-funded opsi extension](#).

Some preconditions are required, in order to use this module. That means that you need a suitable modules file to unlock this extension. You can get this file by purchasing the extension module. For evaluation purposes you can get a temporary modules file without charge. ( → mail us at [info@uib.de](mailto:info@uib.de)).

#### 2.2.1 General Requirements

The source directory service must implement the LDAP protocol. Samba 4 or Active Directory are examples for directory services that support the LDAP protocol.

The target opsi server should run at least opsi 4.0.7. Prior versions may work but have not been tested.

The machine running the connector must have access to the ones running the directory and opsi over the network.

It is possible to have everything running on the same server but this manual will assume that different machines are used.

#### 2.2.2 Hardware Requirements

These requirements are intended for a basic use in a small-sized environment with up to 500 clients. The requirements may change running the connector in larger environments so please be aware that adjustments may be necessary.

- 256 MB of free RAM
- Network connection

## 2.2.3 Software Requirements

The installation and operation is only supported on Linux. Support for Windows is not planned.

The connector uses Python 3 which has to be available in at least version 3.2.

The connector uses standardized protocols and therefore does not require any further opsi- or directory-components to be installed.

## 2.3 Installation

To install the connector please add the opsi repository as described in the *Getting Started* document.

Then use your operating system's package manager to install the package `opsi-directory-connector`.

On an Debian-based machine installation can be done like this:

```
apt-get install opsi-directory-connector
```

---

### Note

CentOS and RedHat version 6 and 7 do not have Python 3 as part of their core repositories. Therefore we currently can not support installation on these operating systems.

---

## 2.4 Configuration

The connector features a variety of settings to fit into different environments.

The configuration is made in a JSON-formatted file and must contain valid JSON! To specify any boolean value please use `true` or `false`. Text has to be entered in double quotes like `"this is text"`.

An example configuration will be provided at `/etc/opsi/opsidirectoryconnector.example.conf`. This file can be used as a template for your own configurations.

```
cp /etc/opsi/opsidirectoryconnector.example.conf /etc/opsi/opsidirectoryconnector-custom.conf
```

### 2.4.1 Directory settings

These are the settings required to access the directory and limit the search scopes to specific areas and entries.

```
{
  "directory": {
    "address": "ldap://192.168.12.34",
    "user": "DOMAIN\\opsiconnector",
    "password": "insertpasswordhere",
    "passwordFile": "",
    "search_base": "dc=testcompy,dc=local",
    "search_query_computers": "(objectClass=computer)",
    "identifying_attribute": "dn",
    "connection_options": {
      "start_tls": true,
      "paged_search_limit": 768
    }
  },
  ...
}
```

Under `address` you specify the directory-server. `user` and `password` are required for authentication at the directory. If `passwordFile` is set the value will be interpreted as path to a file that contains the password. The content of the file will be used as password. This allows for not having the password written in plaintext in the file. This will override the value set for `password` if the file can be read.

---

**Tip**

We recommend using a dedicated user account.

---

---

**Note**

Depending on the used directory software and its configuration the format for the username can be different. Besides *Down-Level Logon Name* in the format of `DOMAIN\username` it is possible that the format uses *User Principal Name* with the format of `user@domain` or a *Distinguished Name (DN)* like `uid=opsiconnect,cn=users,dc=test,dc=intranet`.

---

With `search_base` you specify the location from where on the connector looks for matching entries. Through `search_query_computers` the filter used for finding clients can be configured.

Through the optional parameter `identifying_attribute` it is possible to set the attribute used for the unique identification of a client. This is possible since version 23. The default is the usage of `dn`. A common alternative to `dn` is the value `distinguishedName` which is often used in Microsoft Active Directory.

The parameter `connection_options` contains additional options to configure the connection. With `start_tls` it is possible to control if a secure connection should be used.

If the optional parameter `paged_search_limit` is present and its value is an integer multiple queries are made to the directory in order to read its elements. How many elements a response contains is limited through the given value. This is supported since version 20.

---

**Note**

Additional connection options will be implemented on demand.

---

Since version 14 it is possible to test the connection to the directory through the parameter `--check-directory` without connecting to the opsi server.

## 2.4.2 Configure connection for UCS

For a connection to Univention Corporate Server a full *Distinguished Name* has to be used as username. This has the form `uid=<username>,cn=users,dc=company,dc=mydomain`.

On UCS LDAP is reachable through ports 7389 (unsecured) resp. 7636 (secured via SSL). If Samba is installed on the Server and used as AD-compatible domain controller then it is listening on ports 389 (unsecured) resp. 636 (secured via SSL). To make use of the secured ports set the connection option `start_tls` to `true`.

The different connections also change the DN used for authentication. LDAP uses `uid=...` where Samba works with `dn=...`

Usually clients are found in the container `computers`. The following command shows a matching value for `search_base`:

```
echo "cn=computers,$(ucr get ldap/base)"
```

To search for Windows clients you can set `search_query_computers` to `(objectClass=univentionWindows)`.

How you can create a user with read only access is described in the Univention wiki: [Cool Solution - LDAP search user](#)

### 2.4.3 Behaviour settings

These settings defines the behaviour of the connector.

```
{
  ...
  "behaviour": {
    "write_changes_to_opsi": true,
    "root_dir_in_opsi": "clientdirectory",
    "update_existing_clients": true,
    "prefer_location_from_directory": true
  },
  ...
}
```

If `write_changes_to_opsi` is set to `false` no data will be written to opsi. This can be used to check settings before applying them.

Via `root_dir_in_opsi` you define what group should be used as the root in opsi. You need to make sure that this group exists.

---

#### Note

The group `clientdirectory` is shown as `DIRECTORY` in configed. If clients or groups are to appear directly below `DIRECTORY` the value for `root_dir_in_opsi` has to be `clientdirectory`.

---

If `update_existing_clients` is set to `false` clients already existing in opsi will not be altered. If this is set to `true` clients may have any manually set data overridden with the values from the directory.

If `prefer_location_from_directory` is set to `true` clients will be moved in opsi to the same location they have in the directory. If you want to disable this set it to `false`.

The handling of groups can be configured through the optional key `group_handling` since version 31. The default is `cn`. With this setting the groups are extracted from the DN of a computer and created as part of the opsi directory. A client can only be member of one group.

If `group_handling` is set to `ucsatschool` the handling will be adapted for use in [UCS@School](#) environments. opsi-directory-connector will search for schools and find their rooms automatically. For every school a group will be created in opsi. In [UCS@School](#) a computer can be member of multiple rooms. To follow this behaviour the groups in opsi will be created as groups and not as part of the opsi directory.

### 2.4.4 Attribute-Mappings

With a system as flexible as a directory service the connector must be given information about what attributes in the directory match these of the corresponding opsi objects.

```
{
  ...
  "mapping": {
    "client": {
      "id": "name",
      "description": "description",
      "notes": "",
      "hardwareAddress": "",
      "ipAddress": "",
      "inventoryNumber": "",
      "oneTimePassword": ""
    }
  },
  ...
}
```

There is a mapping for client attributes. The key of the mapping is the attribute in opsi and the value is the attribute from the directory. If the value (in the mapping) is empty no mapping will be done.

---

**Note**

If the value read from the directory for the client ID does not seem to be an FQDN an FQDN will be created. The domain part for this will be created from the DC of the read element.

---

**Tip**

On UCS the value for `hardwareAddress` can be set to `macAddress` if the connection is made through LDAP (ports 7389 or 7636).

---

## 2.4.5 Manual assignment of group names

Group names are usually used without any major adjustments. But this may lead to cases where names should be used that are invalid in opsi.

For this special cases a manual assignment of group names can be helpful.

To configure this an entry `group_name` has to be created in `mapping`. This holds the mapping from the directory to opsi. Names that are not present in this mapping aren't changed. The group names are always processed in lowercase. This can be configured since version 23.

The following example handles the group `_server` originating from the directory as `server` in opsi.

```
{
  ...
  "mapping": {
    "client": {
      ...
    },
    "group_name": {
      "_server": "server"
    }
  },
  ...
}
```

**Warning**

Please be careful with this feature as it may introduce undesired side effects. It should only be used for special cases!

---

## 2.4.6 opsi connection settings

This specifies how the connector accesses opsi.

```
{
  ...
  "opsi": {
    "address": "https://localhost:4447",
    "username": "syncuser",
    "password": "secret",
    "exit_on_error": false,
    "passwordFile": "",
  }
}
```



```
    "connection_options": {
      "verify_certificate": true
    }
  }
}
```

Set `address` to the address of your opsi server. Please include the port.

---

**Note**

To use a proxy for the connection use the environment variable `HTTPS_PROXY`.

---

`username` and `password` should be set accordingly to authenticate at the opsi server. If `passwordFile` is set the value will be interpreted as path to a file that contains the password. The content of the file will be used as password. This allows for not having the password written in plaintext in the file. This will override the value set for `password` if the file can be read.

---

**Tip**

We recommended setting up a dedicated user for this task. Refer to the document *Getting Started* on how to do this.

---

If the parameter `exit_on_error` is `true` then any problem that appears when updating data in opsi opsi - this could be triggered by submitting values that are invalid in opsi - results in a break. If this is `false` then problems will be logged but the run will not be stopped.

With `connection_options` the options for connecting to opsi can be set. `verify_certificate` configures the verification of the server certificate. For selfsigned certificates this can be set to `false`.

Since version 14 it is possible to test the connection to the opsi server through the paramter `--check-opsi` without connecting to the directory.

## 2.5 Running the connector

After installation a binary called `opsidirectoryconnector` will be present on the system.

It is required to pass an argument `--config` together with the path to the configuration.

```
opsidirectoryconnector --config /etc/opsi/opsidirectoryconnector-custom.conf
```

---

**Note**

The user running the binary does not require any access to opsi as this is all specified in the configuration.

---

### 2.5.1 Example: recurring runs with systemd

The connector currently does one synchronisation run when executed but the chances are good that you want to have a constant synchronisation of data.

It is easy to automate the execution of the connector to have recurring runs.

We will use `systemd` for this. In contrast to `cronjobs` `systemd` will avoid overlapping runs and is therefore a good choice.

The following example will set up the connector so that it is run five minutes after the machine was booted and from then on every hour.

In the directory `/etc/systemd/system/`, this is the directory for user-defined units, you need to place the two following files. One for the timer that makes the job recurring and one for the job itself.

Please put this inside `opsi-directory-connector.timer`:

---

```
[Unit]
Description=Start the opsi-directory-connector in regular intervals

[Timer]
OnBootSec=5min
OnUnitActiveSec=1hour

[Install]
WantedBy=timers.target
```

And this is the content of `opsi-directory-connector.service`:

```
[Unit]
Description=Sync clients from AD to opsi.
Wants=network.target

[Service]
Type=oneshot
ExecStart=/usr/bin/opsidirectoryconnector --config /etc/opsi/opsidirectoryconnector-custom.conf
```

To enable the timer and start it right away use the following commands:

```
systemctl enable opsi-directory-connector.timer
systemctl start opsi-directory-connector.timer
```

If the timer does not get started it will be first run after the next reboot of the machine.

## 2.5.2 Example: recurring runs as cronjob

It is easy to automate recurring runs through a cronjob.

Please be aware that overlapping runs may happen with cron and therefore the interval should be higher. To avoid this problem it is recommended to use **systemd** instead of **cron**

The cronjob file can usually be edited through `crontab -e`. For an synchronisation that happens every hour there can be used the following:

```
0 * * * * /usr/bin/opsidirectoryconnector --config /etc/opsi/opsidirectoryconnector-custom.conf
```