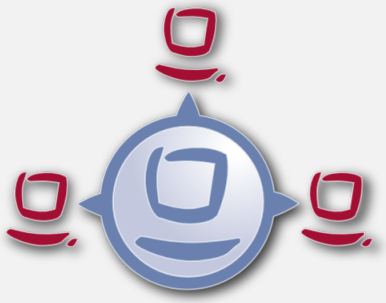

opsi Directory Connector

uib gmbh
Bonifaziusplatz 1b
55118 Mainz
Tel.:+49 6131 275610
www.uib.de
info@uib.de

Stand: 10. September 2019

Inhaltsverzeichnis

Kapitel 1

Copyright

Das Copyright an diesem Handbuch liegt bei der uib gmbh in Mainz.

Dieses Handuch ist veröffentlicht unter der creative commons Lizenz
Namensnennung - Weitergabe unter gleichen Bedingungen (by-sa).

Eine Beschreibung der Lizenz finden Sie hier:
https://creativecommons.org/licenses/by-sa/3.0/de/

Der rechtsverbindliche Text der Lizenz ist hier:
https://creativecommons.org/licenses/by-sa/3.0/de/legalcode

Die Software von opsi ist in weiten Teilen Open Source.
Nicht Open Source sind die Teile des Quellcodes, welche neue Erweiterungen enthalten die noch unter Kofinanzierung stehen, also noch nicht bezahlt sind.
siehe auch: http://uib.de/de/opsi-erweiterungen/erweiterungen/

Der restliche Quellcode ist veröffentlicht unter der AGPLv3:

Der rechtsverbindliche Text der AGPLv3 Lizenz ist hier:
http://www.gnu.org/licenses/agpl-3.0-standalone.html

Deutsche Infos zur AGPL: http://www.gnu.org/licenses/agpl-3.0.de.html

Für Lizenzen zur Nutzung von opsi im Zusammenhang mit Closed Source Software kontaktieren Sie bitte die uib gmbh.

Die Namen *opsi*, *opsi.org*, *open pc server integration* und das opsi-logo sind eingetragene Marken der uib gmbh.

Kapitel 2

opsi directory connector

2.1 Einführung

Der opsi Directory Connector ist ein Werkzeug um Daten aus einem Verzeichnisdienst in eine opsi-Installation zu überführen. Dadurch wird mehrfacher Pflegeaufwand in unterschiedlichen Systemen vermieden.

2.2 Vorbedingungen für die opsi Erweiterung *opsi directory connector*

Dieses Modul ist momentan eine kofinanzierte opsi Erweiterung.
Es sind eine Reihe von Vorbedingungen nötig, um dieses Modul einsetzen zu können. Das bedeutet, dass Sie zum Einsatz eine Freischaltdatei benötigen. Diese Freischaltung erhalten Sie, wenn Sie die Erweiterung kaufen. Zu Evaluierungszwecken stellen wir Ihnen auch eine zeitlich befristete Freischaltung kostenlos zur Verfügung (→ mail an info@uib.de).

2.2.1 Allgemeine Anforderungen

Der Quell-Verzeichnisdienst muss das LDAP-Protokoll implementieren.

Das Ziel-Opsi-System sollte mindestens opsi 4.0.7 verwenden. Ältere Versionen können funktionieren, wurden aber nicht getestet.

Die Maschine, auf welcher der Connector laufen soll, muss über das Netzwerk Zugriff auf den Directory- und opsi-Server haben. Es ist möglich alle Komponenten auf der gleichen Maschine zu betreiben, aber es wird davon ausgegangen, dass jeweils getrennte Maschinen verwendet werden.

2.2.2 Hardware-Anforderungen

Diese Anforderungen richten sich an eine einfache Verwendung in einer kleinen Umgebung mit bis zu 500 Clients. Diese Anforderungen fallen in großen Umgebungen gegebenenfalls größer aus, weshalb Anpassungen notwendig sein können.

- 256 MB freier Arbeitsspeicher
- Netzwerkverbindungen

2.2.3 Software-Anforderungen

Es wird nur die Installation und der Betrieb des Connectors unter Linux unterstützt. Eine Unterstützung für Windows ist nicht geplant.

Der Connector verwendet Python 3, welches mindestens in Version 3.2 vorliegen muss.

Durch die Verwendung standardisierter Protokolle zur Kommunikation werden keine zusätzlichen opsi- oder Verzeichnisdienst-spezifischen Komponenten benötigt.

2.3 Installation

Zur Installation fügen Sie bitte das opsi-Repository wie im Dokument *Getting Started* beschrieben hinzu.

Anschließend verwenden Sie den Paket-Manager des Betriebssystems um das Paket `opsi-directory-connector` zu installieren.

Auf einer Debian-basierten Maschine kann die Installation wie folgt durchgeführt werden:

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

Anmerkung
CentOS und RedHat stellen in Version 6 und 7 kein Python 3 als Teil ihrer Kern-Repositories bereit, weshalb die Installation auf diesen Maschinen von uns nicht unterstützt wird.

2.4 Konfiguration

Der Connector kann über eine Vielzahl an Einstellungsmöglichkeiten an verschiedenste Umgebungen angepasst werden.

Die Konfiguration geschieht über eine Konfigurationsdatei im JSON-Format, welche gültiges JSON enthalten muss. Zur Angabe von boolschen Werten verwenden Sie bitte `true` oder `false`. Text muss mit doppelten Anführungszeichen eingegeben werden, beispielsweise `"das ist Text"`.

Eine Beispiel-Konfiguration wird unter `/etc/opsi/opsidirectoryconnector.example.conf` bereitgestellt. Diese Datei kann als eine Vorlage für eigene Konfigurationen verwendet werden.

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

2.4.1 Directory-Einstellungen

Diese Einstellungen werden benötigt, um eine Verbindung zum Verzeichnisdienst herzustellen und den Suchbereich auf bestimmte Bereiche und Objekte einzugrenzen.

```
{
    "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
```

```
        }
    },
    ...
}
```

Unter `address` muss angegeben werden unter welcher Adresse der Server angesprochen wird. `user` und `password` werden für die Authentifikation an Selbigem verwendet. Sofern für `passwordFile` ein Wert angegeben wird, wird dieser als Pfad zu einer Datei, welche das Passwort enthält, interpretiert. Der Inhalt dieser Datei wird als Passwort verwendet werden. Dadurch muss das Passwort nicht im Klartext in der Konfigurationsdatei vorgehalten werden. Das so ausgelesene Passwort wird eventuell gesetzte Werte für `password` überschreiben.

Tipp
Wir empfehlen die Verwendung eines gesonderten Benutzerkontos.

Anmerkung
Je nach verwendeter Directory-Software und dessen Konfiguration können zur Anmeldung verschiedene Formate eines Benutzernamens zum Tragen kommen.
Neben *Down-Level Logon Name* im Stile von `DOMAIN\\username` kann das Format auch *User Principal Name* im Stile von `user@domain` oder ein *Distinguished Name* (DN) wie `uid=opsiconnect,cn=users,dc=test,dc=intranet` sein.

Über `search_base` wird angegeben ab welchem Punkt nach passenden Element gesucht wird. Über `search_query_computers` kann der für die Suche nach Clients verwendete Filter konfiguriert werden.

Über den optionalen Parameter `identifying_attribute` wird ab Version 23 festgelegt welches Attribut verwendet werden soll um einen Client eindeutig zu identifizieren. Als Standard wird hier `dn` verwendet. Eine häufige Alternative zu `dn` ist der Wert `distinguishedName`, diese Variante kommt oftmals in Microsoft Active Directory zum Einsatz.

Der Parameter `connection_options` beinhaltet zusätzliche Optionen zur Konfiguration der Verbindung. Mit `start_tls` kann gesteuert werden, ob eine gesicherte Verbindung verwendet werden soll.

Ist der optionale Parameter `paged_search_limit` vorhanden und als Wert eine Ganzzahl angegeben, so werden zum Auslesen der Elemente aus dem Directory mehrere Abfragen verwendet. Wieviele Elemente eine Antwort maximal enthält wird über den gesetzten Wert gesteuert. Dieses Verhalten wird seit Version 20 unterstützt.

Anmerkung
Weitere Verbindungs-Optionen werden auf Nachfrage implementiert.

Seit Version 14 ist es möglich über den Aufrufparameter `--check-directory` die Verbindungseinstellungen zum Verzeichnis zu prüfen, ohne dass eine Verbindung zum opsi-Server hergestellt wird.

2.4.2 Verbindung zu Univention Corporate Server

Für eine Verbindung zu Univention Corporate Server (UCS) muss für die Verbindung als Benutzername ein vollständiger *Distinguished Name* verwendet werden. Dieser hat die Form `uid=<username>,cn=users,dc=company,dc=mydomain`.

Unter UCS ist LDAP über die Ports 7389 (ungesichert) bzw. 7636 (SSL-gesichert) erreichbar. Ist auf dem Server ebenfalls Samba installiert und als AD-kompatibler Domaincontroller eingerichtet, so lauscht dieser auf den Ports 389 (ungesichert) bzw. 636 (SSL-gesichert). Für die Verwendung der SSL-gesicherten Ports muss die Verbindungseinstellung `start_tls` auf `true` gesetzt werden.

Die beiden möglichen Verbindungen unterscheiden sich auch in der Art der Anmeldung. Bei LDAP kommt `uid=...` zum Tragen, wohingegen bei Samba mittels `dn=...` gearbeitet wird.

In der Regel wird man nach Rechner-Objekten im Container `computers` suchen. Der folgende Befehl gibt den dazu passenden Wert für `search_base` aus:

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

Für die Suche nach Windows-Clients kann `(objectClass=univentionWindows)` als Wert für `search_query_computers` angegeben werden.

Wie ein Benutzer mit nur lesendem Zugriff angelegt werden kann, ist im Univention-Wiki zu finden: Cool Solution - LDAP search user

2.4.3 Verhaltens-Einstellungen

Die Einstellungen steuern das Verhalten des Connectors.

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

Wird `write_changes_to_opsi` auf `false` gesetzt werden keine Daten nach opsi geschrieben. Mit dieser Einstellung ist es möglich die Verbindungseinstellungen zu überprüfen, bevor sie angewendet werden.

Per `root_dir_in_opsi` wird angegeben welche Gruppe in opsi als Wurzelgruppe verwerndet werden soll. Es muss von Ihnen sichergestellt werden, dass diese Gruppe existiert.

Anmerkung
Die Gruppe *clientdirectory* wird im Configed als *DIRECTORY* angezeigt. Sollen also Clients oder Gruppen direkt unterhalb von *DIRECTORY* erscheinen, so muss als Wert für `root_dir_in_opsi` der Wert `clientdirectory` eingetragen werden.

Wird `update_existing_clients` auf `false` gesetzt, so werden bereits in opsi existierende Clients nicht verändert. Wird dieser Wert auf `true` gesetzt, so werden möglicherweise manuell gesetzte Daten mit den Werten aus dem Directory überschrieben.

Falls `prefer_location_from_directory` auf `true` gesetzt, werden Clients in opsi an die Position verschoben, welche sie im Directory haben. Für das Deaktivieren dieses Verhalten, muss dieser Wert auf `false` gesetzt werden.

Die Gruppenbehandlung kann seit Version 31 über den optionalen Schlüssel `group_handlng` gesteuert werden. Der Default ist hierbei `cn`. Dabei werden Gruppen aus dem DN eines Computers abgeleitet und entsprechend als Teil des opsi-Directory angelegt. Ein Client ist dabei nur Mitglied einer Gruppe.

Wird das Gruppenhandling auf `ucsatschool` gesetzt, so wird das Verhalten auf die Verwendung in UCS@School-Umgebungen angepasst. Dabei wird der opsi-directory-connector automatisch nach Schulen suchen und für diese die Räume ermitteln, welche dann nach opsi synchronisiert werden. Für jede ermittelte Schule wird in opsi eine Gruppe angelegt. Um dem Gruppen von UCS@School zu folgen, bei welchen ein Rechner in mehr als einem Raum zu finden sein kann, werden die Gruppen dabei nicht Gruppe innerhalb des opsi-Directory angelegt, sondern als normale Gruppe, so dass ein Client auch in opsi in mehreren Gruppen sein kann.

2.4.4 Mappings

Mit einem derart flexiblen System wie ein Verzeichnisdienst benötigt der Connector Informationen darüber welche Attribute im Directory auf welche Attribute in opsi angewendet werden sollen.

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

Es gibt ein Mapping für Client-Attribute. Der Schlüssel des Mappings ist das Attribut in opsi und der Wert ist das Attribut aus dem Verzeichnisdienst. Ist der Wert (in der Zuordnung) leer, so wird keine Zuordnung vorgenommen.

Anmerkung
Sollte der aus dem Verzeichnis ausgelesene Wert für die ID des Clients nicht als FQDN erkennbar sein, so wird ein enstprechender FQDN erstellt. Der Domain-Teil hierfür wird aus den DC-Werten des Elements gebildet.

Tipp
Unter Univention Corporate Server (UCS) kann bei `hardwareAddress` der Wert `macAddress` angegeben werden, wenn die Verbindung über LDAP (Port 7389 oder 7636) hergestellt wird.

2.4.5 Manuelle Zuordnung von Gruppennamen

Gruppennamen werden in der Regel ohne große Anpassungen übernommen. Allerdings kann es dabei vorkommen, dass Gruppennamen verwendet werden sollen, welche in opsi ungültig sind.

Für diese Sonderfälle kann eine manuelle Zuordnung von Gruppennamen vorgenommen werden, welche es erlaubt auch diese Fälle zu behandeln.

Zur Einrichtung wird in `mapping` ein Eintrag `group_name` angelegt, in welchem eine Zuordnung der Directory-Seite zur opsi-Seite vorgenommen wird. Für Gruppen, welche in dieser Zuordnung nicht vorkommen, wird der Namen nicht angepasst. Die Gruppennamen werden immer in Kleinbuchstaben verarbeitet, weshalb die Einträge hier in Kleinbuchstaben erfolgen müssen. Möglich ist dies ab Version 23.

Das folgende Beispiel behandelt die aus dem Directory stammende Gruppe `_server` in opsi als `server`.

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

Warnung
Bei unbedachtem Einsatz kann die manuelle Zuordnung unerwünschte Seiteneffekte haben. Deshalb sollte diese Zuordnungsmöglichkeit nur in Ausnahmefällen eingesetzt werden.

2.4.6 opsi-Verbindungs-Einstellungen

Hierüber wird gesteuert wie der Connector sich zu opsi verbindet.

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

Unter `address` ist die Adresse des opsi-Servers einzutragen. Vergessen Sie nicht die Angabe des Ports!

Anmerkung
Ein Proxy für die Verbindung kann über die Umgebungsvariable *HTTPS_PROXY* gesetzt werden.

Mittels `username` und `password` wird geregelt welche Zugangsdaten zur Authentifizierung am opsi-Server verwendet werden. Sofern für `passwordFile` ein Wert angegeben wird, wird dieser als Pfad zu einer Datei, welche das Passwort enthält, interpretiert. Der Inhalt dieser Datei wird als Passwort verwendet werden. Dadurch muss das Passwort nicht im Klartext in der Konfigurationsdatei vorgehalten werden. Das so ausgelesene Passwort wird eventuell gesetzte Werte für `password` überschreiben.

Tipp
Wir empfehlen die Verwendung eines gesonderten Benutzers. Die Anlage zusätzlicher Benutzer ist im Dokument *Getting Started* beschrieben.

Ist der Parameter `exit_on_error` auf `true` gestellt, so führt ein Problem bei der Aktualisierung der Daten in opsi - das kann bspw. auch durch die Übermittlung von für opsi ungültige Werte geschehen - zu einem Abbruch. Steht dies auf `false`, so werden Fehler geloggt, aber der Lauf wird nicht beendet.

Unter `connection_options` werden Optionen für die Verbindung zum opsi-Server festgelegt. Mittels `verify_certificate` wird die Überprüfung des Server-Zertifikats gesteuert. Für selbstsignierte Zertifikate kann dieser Wert auf `false` gesetzt werden.

Seit Version 14 ist es möglich über den Aufrufparameter `--check-opsi` die Verbindung zum opsi-Server zu testen, ohne dass eine Verbindung zum Verzeichnisdienst hergestellt wird.

2.5 Den Connector ausführen

Nach der Installation existiert ein Binary `opsidirectoryconnector` auf dem System.

Dieses muss einen Parameter `--config` zusammen mit dem Pfad zur Konfigurationsdatei übergeben bekommen.

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

Anmerkung
Der ausführende Benutzer benötigt keinen Zugriff auf das opsi-System, da der zugreifende Benutzer in der Konfigurationsdatei hinterlegt ist.

2.5.1 Beispiel: wiederkehrende Verarbeitung mit systemd

Der Connector macht aktuell bei der Ausführung eine Synchronisationslauf, aber die Chancen stehen gut, dass eine ständige Synchronisation erfolgt.

Es ist einfach, die Ausführung wiederkehrender Läufe zu automatisieren.

Wir werden hierbei systemd verwenden. Im Gegensatz zu cronjobs wird systemd verhindern, dass überlappende Läufe stattfinden, weshalb systemd eine gute Wahl ist.

Das folgende Beispiel wird den Connector so einrichten, dass er fünf Minuten nach dem Start der Maschine ausgeführt wird und danach jede Stunde.

Unter `/etc/systemd/system/`, dem Verzeichnis für benutzerdefinierte Units, müssen die zwei folgenden Dateien abgelegt werden. Eine Datei ist der Timer, welche unseren Job wiederkehrend aufruft und die Andere ist für den Job selbst.

Bitte füllen Sie die Datei `opsi-directory-connector.timer` mit dem folgenden Inhalt:

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

[Timer]
OnBootSec=5min
OnUnitActiveSec=1hour

[Install]
WantedBy=timers.target
```

Und dies muss nach `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
```

Um den Timer zu aktivieren und ihn sofort zu starten, können die folgenden Befehle verwendet werden:

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

Falls der Timer nicht gestartet wird, wird er erst nach dem nächsten Neustart der Maschine ausgeführt werden.

2.5.2 Beispiel: wiederkehrende Verarbeitung als Cronjob

Es ist einfach, die Ausführung wiederkehrender Läufe über einen Crobjob zu automatisieren.

Bitte beachten Sie, dass überlappende Läufe stattfinden können, weshalb der Synchronisationsintervall am besten größer gewählt werden sollte. Zur Vermeidung dieses Problems wird die Verwendung von **systemd** anstatt **cron** empfohlen!

Zur Bearbeitung der Cronjob-Datei wird in der Regel `crontab -e` aufgerufen. Für eine zu jeder Stunde stattfindenden Synchronisation kann dort folgendes als Cronjob hinterlegt werden.

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