OPSICONF 2026
OPSI-CLI
WORKSHOP

OPSI-CLI WORKSHOP

- 2 Stunden Workshop zu opsi-cli, dem OPSI Kommandozeilen-Werkzeug
- Vorstellung von Funktionen anhand von konkreten Beispielen
- Gegliedert in drei Themenbereiche:
  - Datenausgabe und -verarbeitung (Jan Schneider)
  - Arbeit mit OPSI-Paketen (Nils Dörrer)
  - Client-Management (Fabian Kalweit)
- Am Ende Zeit für allgemeine Fragen zu opsi-cli
- Zwischenfragen können jederzeit gestellt werden

OPSI-CLI

- Beginn der Entwicklung 2021
- Ziel: Ein komfortables, leistungsfähiges Kommandozeilen-Tool für interaktive Nutzung und Automatisierung
- Lauffähig unter Linux, Windows und macOS
- Verbindung zum OPSI-Server über HTTPS
- Der Funktionsumfang wird stetig erweitert

INSTALLATION UND UPGRADE

- opsi-cli ist auf jedem OPSI-Server vorhanden (opsi-utils-Paket)
- opsi-cli steht als OPSI-Paket zur Verfügung https://opsipackages.43.opsi.org/stable/
- Binary verfügbar unter https://tools.43.opsi.org/testing/

```
wget https://tools.43.opsi.org/testing/opsi-cli-linux-x64.run \
 -O opsi-cli
chmod +x opsi-cli
./opsi-cli --version
./opsi-cli self install
opsi-cli self upgrade --branch experimental
opsi-cli self installed-versions
```


DATENSÄTZE UND METADATEN

- Viele Funktionen von opsi-cli liefern tabellarische Daten zurück.
- Metadaten enthalten Informationen über die Attribute (Zeilen):
  - Name
  - Beschreibung
  - Datentyp
  - Identifier?
  - Standardmäßig ausgegeben?

OPSI-CLI DATASTORE

- Zugriff auf ausgewählte Objekte in der Datenbank über `opsi-cli datastore`
- Einfacher und komfortabler als direkte Verwendung der API über `opsi-cli jsonrpc`
- Einfache Ausgabe, Eingabe und Bearbeitung von Datensätzen
- Aktuell unterstützte Objekttypen:
  Clients, Config-States, Product-Client-States, Product-Property-States

OPSI-CLI DATASTORE CLIENT LIST

Beispiel: Auflisten von Clients mit `opsi-cli datastore client list`

```
opsi-cli datastore client list --where "id=nb*"
```


DATENSÄTZE FILTERN

- Die Angabe einer Filterbedingung mit `--where` ermöglicht die Auswahl bestimmter Datensätze.
- Dieser Filter ist Pflicht, um die ungewollt Verarbeitung sehr großer Datenmengen zu vermeiden.
- Über ein Attribut filtern: `--where "<attribut>=<wert>"`
- Verfügbare Operatoren: `=`, `>`, `<`, `>=`, `<=`
- Wildcards mit Stern (`*`) möglich: `--where "id=nb*"`
- Mehrere Filter werden UND-verknüpft: `--where "id=nb* --where "inventoryNumber=123*"`
- Kein Filter: `--where "<attribut>=*"`

VERFÜGBARE ATTRIBUTE

Welche Attribute stehen für die Ausgabe und Filterung zur Verfügung?

```
opsi-cli --list-attributes datastore client list
```

Auch andere Funktionen außerhalb von `opsi-cli datastore`, die tabellarische Daten liefern, haben diese Funktion.

ANPASSEN DER AUSGABEATTRIBUTE

```
# Datensätze mit Standard-Attributen anzeigen
opsi-cli datastore client list --where "id=nb*"
# Datensätze mit ausgewählten Attributen anzeigen
opsi-cli --attributes id,description datastore client list --where "id=nb*"
# Reihenfolge der Attribute kann geändert werden
opsi-cli --attributes description,id datastore client list --where "id=nb*"
# Datensätze mit allen Attributen anzeigen
opsi-cli --attributes all datastore client list --where "id=nb*"
```


AUSGABEFORMATE

Für alle tabellarischen Daten stehen verschiedene Ausgabeformate zur Verfügung (`--output-format`).
Die gängigen Ausgabeformate sind:

- table: Eine schön formatierte Tabelle, gut für interaktive Nutzung
- csv: Semikolon-getrennte Werte, für einfache Skripte oder Tabellenkalkulation
- json: JSON-Format, gut für komplexere Skripte
- pretty-json: Schön formatiertes JSON, gut für interaktive Nutzung
- key-value: Einfache Schlüssel-Wert-Paare, nützlich bei vielen Attributen

Ohne Angabe eines Ausgabeformates wird das Standard-Ausgabeformat verwendet, das sich je nach Funktion und Kontext anpasst.

BEISPIELE FÜR AUSGABEFORMATE

```
opsi-cli --output-format table datastore client list --where "id=nb*"
opsi-cli --output-format pretty-json datastore client list --where "id=nb*"
opsi-cli --output-format key-value datastore client list --where "id=nb*"
opsi-cli --output-format csv datastore client list --where "id=nb*"
```


EINFACHE SKRIPTE

Für einfache Skripte eignet sich oft das CSV-Format.

```
# Ausgabe von Daten im CSV-Format ohne Header zur Verwendung in Skripten
opsi-cli --output-format csv --no-header --attributes id,inventoryNumber,lastSeen \
  --sort-by lastSeen datastore client list --where "id=nb*"

for client_id in $(opsi-cli --output-format csv --no-header --attributes id \
  --sort-by lastSeen datastore client list --where "id=nb*"); do
    echo "Client ID: $client_id"
done
```


SETZEN VON ATTRIBUTEN

`opsi-cli datastore client update` ermöglicht das Setzen von Attributen.

Zum Testen von Änderungen ohne tatsächliche Ausführung kann `--dry-run` verwendet werden.

```
opsi-cli --dry-run datastore client update --where "id=nb*" \
  --set "inventoryNumber=new value" --set "description=new desc"
```


DATEN SPEICHERN UND WIEDER EINLESEN

`opsi-cli datastore client` **`apply`** ermöglicht das Anwenden von Änderungen.

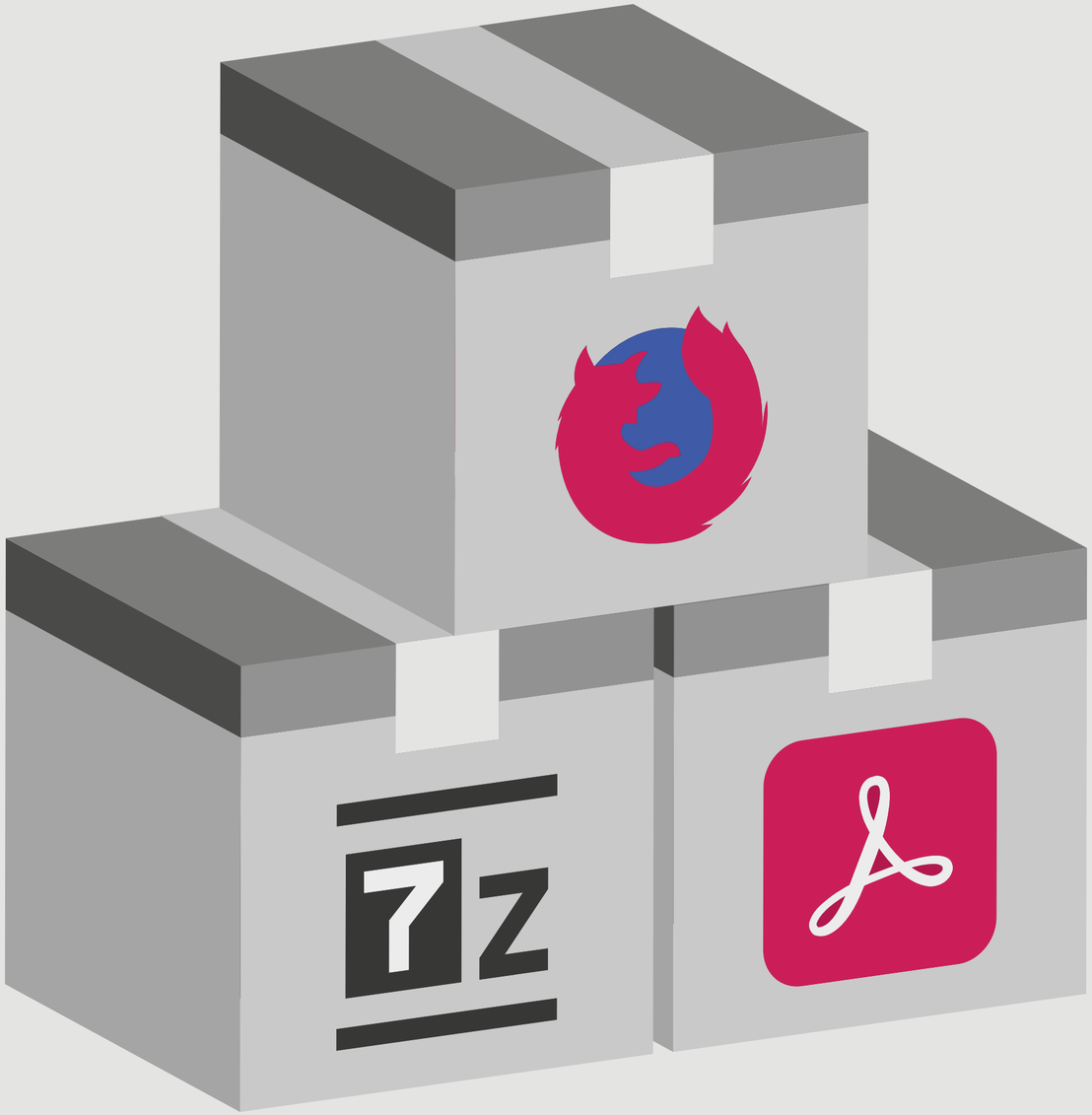
```
# In eine Datei schreiben
opsi-cli --output-format csv --attributes id,inventoryNumber --output-file clients.csv \
  datastore client list --where "id=nb*"
# Editieren der Datei
vim clients.csv
# Daten aus einer Datei lesen und aktualisieren mit "apply" (dry-run!)
opsi-cli --input-file clients.csv --dry-run datastore client apply
# In einem Schritt - Pipen von Daten
opsi-cli --output-format csv --attributes id,inventoryNumber \
   datastore client list --where "id=nb*" \
  | sed s'/0115/NEU_/' \
  | opsi-cli --input-file - --dry-run datastore client apply
```


EDITIEREN VON DATEN

`opsi-cli datastore client` **`edit`** ermöglicht das Editieren von Daten in einem Editor.

```
# Daten editieren mit "edit"
opsi-cli --dry-run datastore client edit --where "id=nb*"
# Editor und Datei-Format angeben
opsi-cli --dry-run --editor "code --wait" --edit-format key-value \
  datastore client edit --where "id=nb*"
```


ARBEIT MIT OPSI-PAKETEN



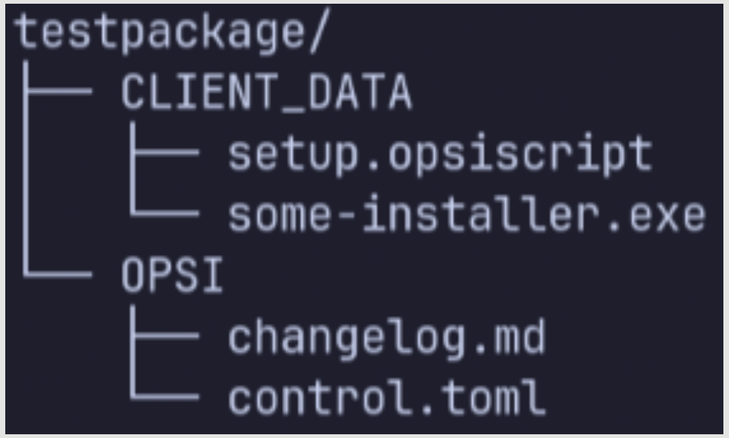
ARBEIT MIT OPSI-PAKETEN

- opsi-cli package: Alle Funktionen rund um opsi-Pakete
- Früher: verschiedene Werkzeuge wie `opsi-makepackage` und `opsi-package-manager`
- Jetzt: Alle Funktionen in einem Werkzeug, konsistente Bedienung und verfügbar unter Linux, Windows und macOS
- Viele neue Funktionen

PACKEN VON PAKETEN

- Erstellen eines opsi-Pakets aus CLIENT_DATA- und OPSI-Verzeichnis (früher `opsi-makepackage`)
- opsi-Paket ist komprimierte .opsi-Datei zum installieren oder weitergegeben

```
opsi-cli package make
```



EXTRAHIEREN VON PAKETEN

- Extrahieren eines opsi-Pakets in CLIENT_DATA- und OPSI-Verzeichnis (früher `opsi-package-manager -x`)
- Quelle ist lokale Datei oder eine URL

```
opsi-cli package extract <path/url>
```


DEMO: EXTRAHIEREN, EDITIEREN UND PACKEN EINES PAKETS

INSTALLIEREN

- Installation eines Pakets von lokaler Datei oder URL (früher `opsi-package-manager -i`)
- Installation auf den Clients anfordern, durch `--setup-where-installed`
- Installation mit neuer Produkt-ID (z.B. zum Testen) `--new-product-id <ID>`

```
opsi-cli package install <path/url>
```


AUFLISTEN

- Auflisten von Paketen, die auf Depot installiert sind
- Filtern nach Typ mit `--product-type` (localboot/netboot)
- Filtern nach Depot mit `--depots <depot-ID>`

```
opsi-cli package list <package-id-filter>
```


DEINSTALLIEREN

- Deinstallation eines Pakets von Depot (früher `opsi-package-manager -r`)
- Option `--depots <depot-IDs>` steuert, von welchen Depots deinstalliert wird
- Option `--purge` steuert, ob Installationsstände und ProductPropertyStates gelöscht werden

```
opsi-cli package uninstall <package-id>
```


DEMO: INSTALLIEREN, AUFLISTEN UND DEINSTALLIEREN EINES PAKETS

META-INFORMATIONEN BEARBEITEN

- Bearbeiten der control.toml eines Pakets
- Beispiel: Setzen der Produktversion oder Anpassen von Abhängigkeiten

```
opsi-cli package meta-edit set-product-version <version>
```


DEMO: UPDATE-SKRIPT FÜR EIN PAKET

OPSI-PAKETE AUS DEPOT-DATEN GENERIEREN

- OPSI-Pakete können direkt von einem Depot generiert werden
- Mit `--depot` kann angegeben werden, von welchem Depot extrahiert wird
- Nützlich, wenn Paketquellen nicht mehr verfügbar sind

```
opsi-cli package fetch <product-id>
```

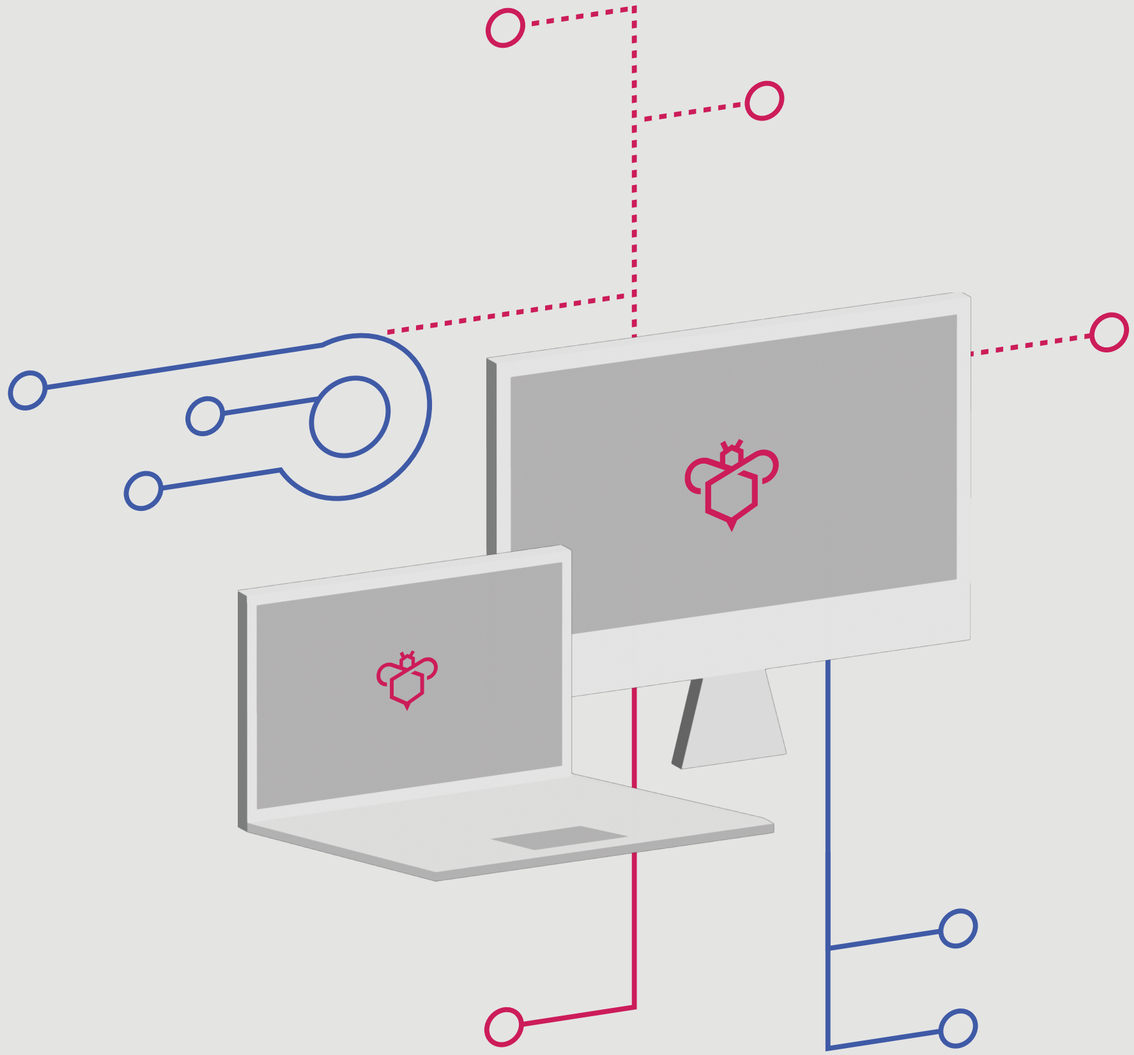

DEMO: EXTRAHIEREN VON INFORMATIONEN UND PAKET-ARCHIV

CONTROL.TOML

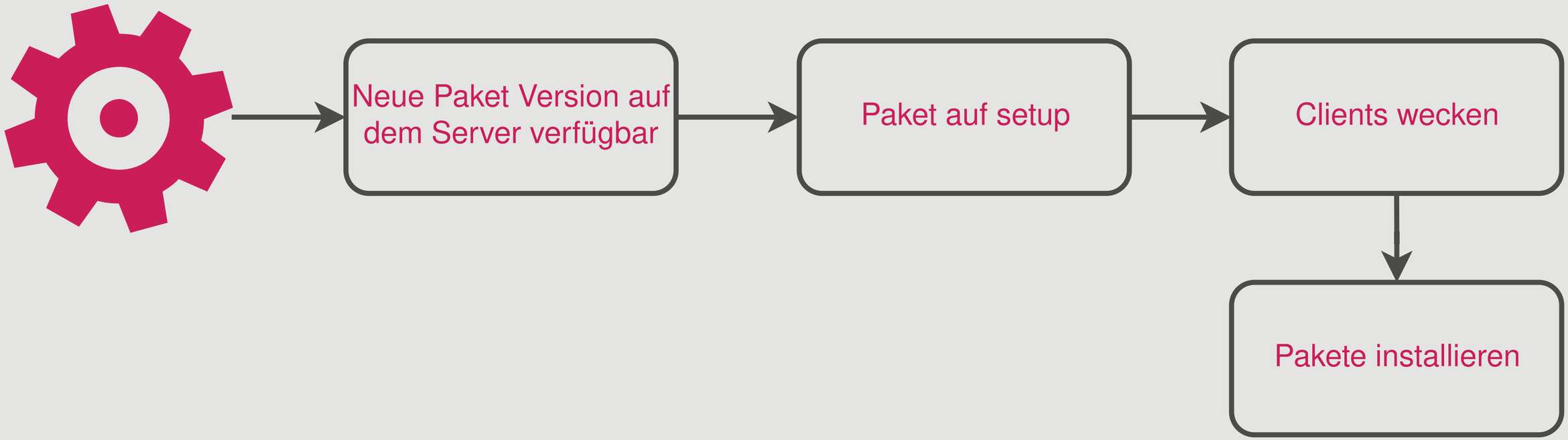
- Neues Format für control-files → toml
- control-files automatisch umgewandelt beim Packen
- Weitere Pflege: bevorzugt über control.toml
- Manuelles Umwandeln von control-files zu control.toml

```
opsi-cli package control-to-toml
```


CLIENT-MANAGEMENT



NEUE PAKETVERSION VERFÜGBAR



SET-ACTION-REQUEST

```
Usage: opsi-cli [GLOBAL OPTIONS] client-action [OPTIONS] COMMAND [ARGS]...

Manage opsi client actions.
* This command supports --dry-run: actions will be simulated and not performed.

 OPTIONS
 --version                          Show the version and exit.
 --clients                  TEXT    Comma-separated list of client IDs or 'all' for all clients.
 --client-groups            TEXT    Comma-separated list of client groups.
 --clients-from-depots      TEXT    Select clients from specified depots (comma-separated list).
 --exclude-clients          TEXT    Comma-separated list of client IDs to exclude.
 --exclude-client-groups    TEXT    Comma-separated list of client groups to exclude.
 --where-action-request     TEXT    Limit to clients with specified actionRequests set for any product (comma-separated list).
 --only-online                      Limit to clients connected to the messagebus.
 --ip-addresses             TEXT    Comma-separated list of IP addresses or networks.
 --exclude-ip-addresses     TEXT    Comma-separated list of IP addresses or networks to exclude.
 --help                             Show this message and exit.

 Commands
 execute                            Execute a command on selected clients
 process-actions                    Process action requests for selected clients
 reboot                             Reboot selected clients
 set-action-request                 Set action requests for opsi clients
 shutdown                           Shutdown selected clients
 trigger-event                      Trigger an event for selected clients
 wakeup                             Wake up selected clients
```


TRIGGER-EVENT

CLIENT-ACTION OPTIONS

--version	Show the version and exit.
--clients	list of client IDs (see --input-separator) or 'all' for all clients.
--client-groups	list of client groups (see --input-separator).
--clients-from-depots	Select clients from specified depots (see --input-separator).
--exclude-clients	list of client IDs to exclude (see --input-separator).
--exclude-client-groups	list of client groups to exclude (see --input-separator).
--where-action-request	Limit to clients with specified actionRequests set for any product (see --input-separator).
--only-online	Limit to clients connected to the messagebus.
--ip-addresses	list of IP addresses or networks (see --input-separator).
--exclude-ip-addresses	list of IP addresses or networks to exclude (see --input-separator).

TRIGGER-EVENT OPTIONS

--event	The type of event to trigger
--wakeup	Wakeup clients if not online (instead of event trigger)
--wakeup-timeout	Number of seconds to wait for client to wake up (0 = do not wait)

```
Usage: opsi-cli [GLOBAL OPTIONS] client-action [CLIENT-ACTION OPTIONS] trigger-event [OPTIONS]

opsi-cli client-action trigger-event command
* This command supports --dry-run: actions will be simulated and not performed.
```


LÖSUNG MIT OPSI-CLI

- Cronjob mit opsi-cli Befehl anlegen

```
opsi-cli client-action set-action-request --where-outdated
opsi-cli client-action trigger-event --wakeup
```

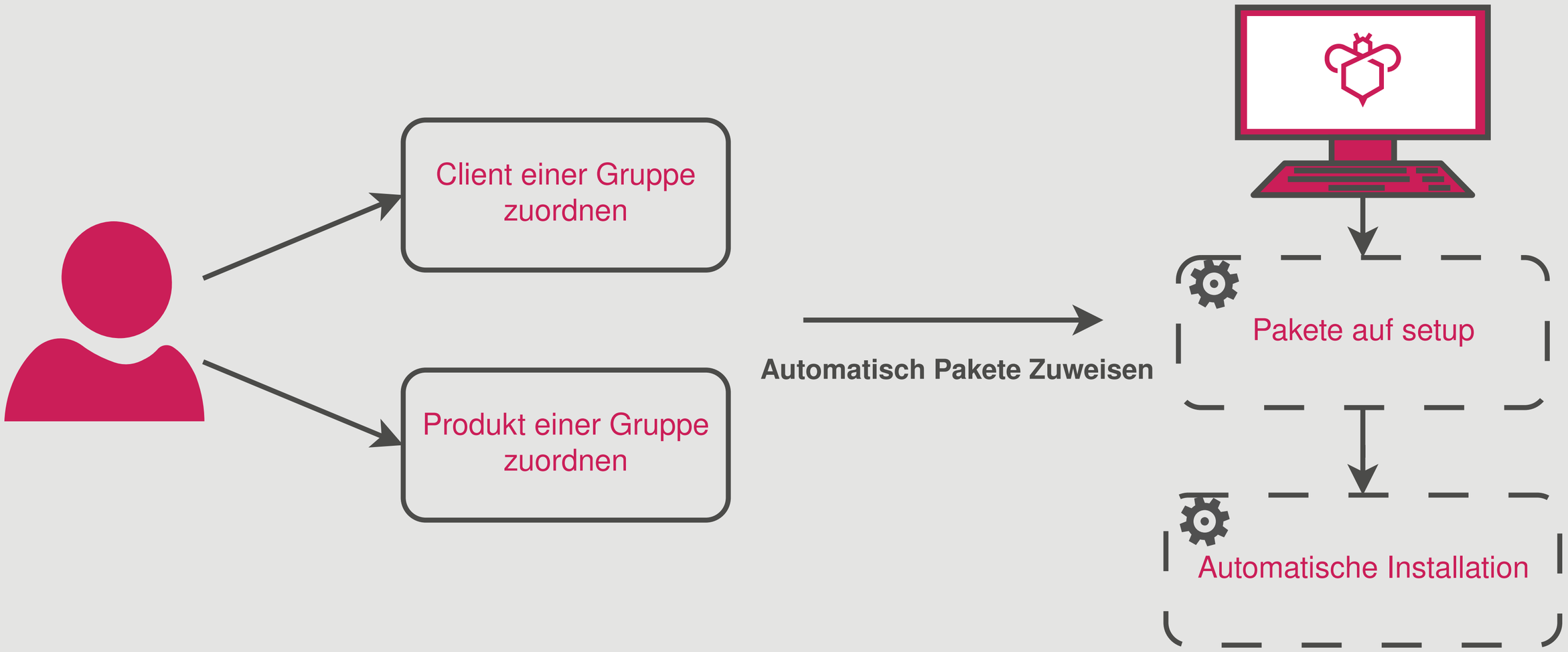
- Produkt Gruppe angeben

```
opsi-cli client-action set-action-request --product-groups <auto-update-products> --where-outdated
```

- Client Gruppe angeben

```
opsi-cli client-action --client-groups <my-client-group> set-action-request --product-groups <auto-update-products> --where-outdated
```


PRODUKTE NACH GRUPPE ZUORDNEN



LÖSUNG MIT OPSI-CLI

```
Usage: opsi-cli [GLOBAL OPTIONS] client-action [OPTIONS] COMMAND [ARGS]...

Manage opsi client actions.
* This command supports --dry-run: actions will be simulated and not performed.

 OPTIONS
 --version                          Show the version and exit.
 --clients                  TEXT    Comma-separated list of client IDs or 'all' for all clients.
 --client-groups            TEXT    Comma-separated list of client groups.
 --clients-from-depots      TEXT    Select clients from specified depots (comma-separated list).
 --exclude-clients          TEXT    Comma-separated list of client IDs to exclude.
 --exclude-client-groups    TEXT    Comma-separated list of client groups to exclude.
 --where-action-request     TEXT    Limit to clients with specified actionRequests set for any product (comma-separated list).
 --only-online                      Limit to clients connected to the messagebus.
 --ip-addresses             TEXT    Comma-separated list of IP addresses or networks.
 --exclude-ip-addresses     TEXT    Comma-separated list of IP addresses or networks to exclude.
 --help                             Show this message and exit.

 Commands
 execute                            Execute a command on selected clients
 process-actions                    Process action requests for selected clients
 reboot                             Reboot selected clients
 set-action-request                 Set action requests for opsi clients
 shutdown                           Shutdown selected clients
 trigger-event                      Trigger an event for selected clients
 wakeup                             Wake up selected clients
```

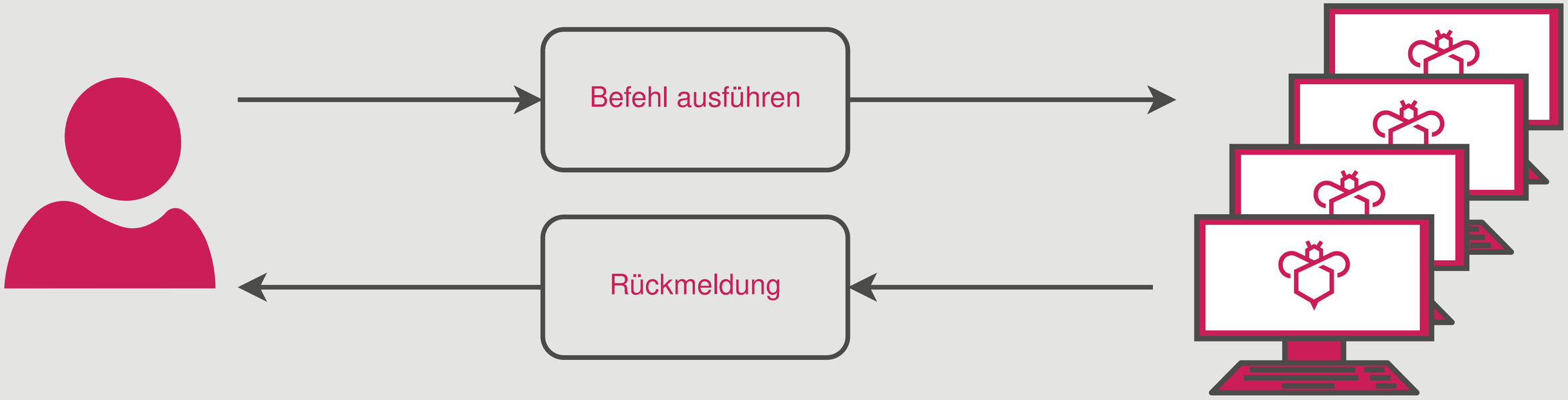

LÖSUNG MIT OPSI-CLI

- Produktgruppe anlegen und Produkte zuordnen
- Cronjob mit opsi-cli Befehl anlegen

```
opsi-cli client-action --client-groups <my-client-group>  set-action-request --product-groups <my-products>  --where-not-installed
```

- Client einer Hostgruppe zuordnen
- Beim nächsten Lauf des Cronjobs werden die Produkte für den Client zum Installieren vorgemerkt
- Die Installation wird beim nächsten opsi-client Event gestartet

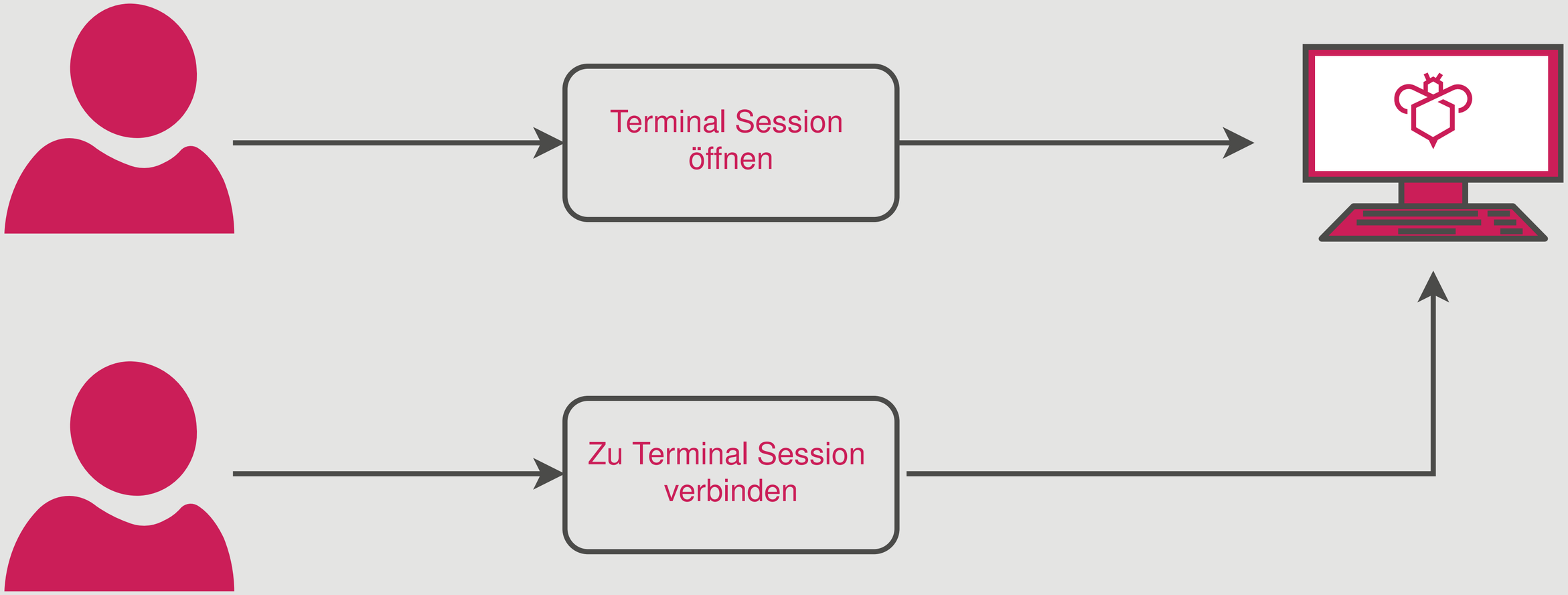
ADHOC ADMINISTRATION



UPDATE BEFEHL AN LINUX SERVER SENDEN

```
opsi-cli client-action --client-groups <linux-servers> execute "apt update && apt upgrade -y"
```


TERMINAL ZU CLIENT ÖFFNEN



TERMINAL ZU CLIENT ÖFFNEN

opsi cli terminal help

TERMINAL ZU CLIENT ÖFFNEN

- Terminal zu einem Client öffnen, um direkt Befehle auszuführen oder Logs zu checken

```
opsi-cli terminal <client-id>
```

- Aufschalten in bestehende Terminal-Session eines Clients

```
opsi-cli -l5 terminal <client-id>
opsi-cli terminal --terminal-id <terminal-id> <client-id>
```


WICHTIGE LINKS

- https://uib.de/
- https://opsi.org
- https://forum.opsi.org/
- https://docs.opsi.org/
- https://opsipackages.43.opsi.org/stable/
- https://tools.43.opsi.org/stable/

VIELEN DANK!

EXTRA: PAKET-INFORMATIONEN

- Paket-Archive können analysiert werden, ohne sie (ganz) zu extrahieren
- Es wird nur OPSI ausgepackt um die control.toml zu parsen

```
opsi-cli package info <path/url>
```


EXTRA: ADHOC WAN CLIENT?

```
opsi-cli client-action --clients wk-00001.acme.corp,wk-00002.acme.corp,wk-00003.acme.corp execute --timeout 2
  powershell -NoProfile -Command "
  \$in=\$false; \$found=\$false;
  Get-Content 'C:\Program Files (x86)\opsi.org\opsi-client-agent\opsiclientd\opsiclientd.conf' |
    ForEach-Object {
    if (\$_ -match '^\[.*\]') { \$in = \$_ -eq '[event_timer]' }
    elseif (\$in -and \$_ -match 'active\s*=\s*true') { \$found=\$true } };
  if (\$found) { 'TRUE' } else { 'FALSE' }"
```


EXTRA: ADHOC WAN CLIENT?

```
opsi-cli client-action --clients wk-00001.acme.corp,wk-00002.acme.corp,wk-00003.acme.corp execute --timeout 2
  powershell -NoProfile -Command "
  \$in=\$false; \$found=\$false;
  Get-Content 'C:\Program Files (x86)\opsi.org\opsi-client-agent\opsiclientd\opsiclientd.conf' |
    ForEach-Object {
    if (\$_ -match '^\[.*\]') { \$in = \$_ -eq '[event_timer]' }
    elseif (\$in -and \$_ -match 'active\s*=\s*true') { \$found=\$true } };
  if (\$found) { 'TRUE' } else { 'FALSE' }"
```

```
wk-00001.acme.corp      | FALSE
wk-00002.acme.corp      | FALSE
wk-00003.acme.corp      | Failed to start process
```


JSONRPC WAN CLIENT?

```
opsi-cli jsonrpc execute hostControlSafe_opsiclientdRpc getConfigValue '["event_timer","active"]' '["wk-00001.acme.corp","wk-00002.acme.corp","wk-0
```


JSONRPC WAN CLIENT?

```
opsi-cli jsonrpc execute hostControlSafe_opsiclientdRpc getConfigValue '["event_timer","active"]' '["wk-00001.acme.corp","wk-00002.acme.corp","wk-0
```


```
{
  "wk-00003.acme.corp": {
    "result": null,
    "error": "Host currently not connected to messagebus"
  },
  "wk-00002.acme.corp": {
    "result": false,
    "error": null
  },
  "wk-00003.acme.corp": {
    "result": false,
    "error": null
  }
}
```