openITCOCKPIT Agent

- Was ist der openITCOCKPIT Agent?
- Benutzung
- Sicherheitskonzept
    - "Basic-Auth" Verfahren
    - Zertifikat-Authentifizierung
- Installation
- Konfiguration
    - Was ist der Pull / Push Mode?
    - Gesamte Konfigurationsmöglichkeiten
        - 1. Als Parameter zum Programmstart
        - 2. In den Konfigurationsdateien
    - Führe ein PowerShell Script als customcheck aus
    - Nachträglicher Wechsel zwischen Pull- und Push-Mode
        - Der Pull-Mode wurde konfiguriert, es sind bereits Services angelegt und im Monitoring
        - Der Push-Mode wurde konfiguriert, es sind bereits Services angelegt und im Monitoring
- Zugriff auf den Webserver des Agent
    - Konfiguration per API abrufen / aktualisieren
    - Konfiguration per API über einen anderen Host
        - Erstellung eines eigenen Zertifikats durch die openITCOCKPIT Agent CA
    - Zugriff auf den verschlüsselten Agent Webserver
- openITCOCKPIT Integration - Nice to know :)
    - Zertifikatauthentifizierung
- Bekannte Probleme
    - C - Kompatibilität bei Linux Systemen
    - Fehler bei den Checks "Network device IO", "Processes" oder "Windows services"
    - Disk Usage checks auf Windows sind nicht verfügbar

Was ist der openITCOCKPIT Agent?

Auf Basis der Programmiersprache Python wurde der Agent als universell einsetzbare Lösung entwickelt, um grundlegende Informationen des Systems auslesen zu können.

Dabei geht es um die Nutzung der CPU (gesamt und pro CPU-Kern), den Zeitpunkt des letzten Bootvorgangs zur Berechnung der Systemlaufzeit, Informationen zu der Nutzung von RAM und swap, angemeldete Benutzer, Festplattenverbrauch, Systemload, einige Sensoren und um den Status angeschlossener und virtueller Netzwerkgeräte.

Es werden ebenfalls Informationen über laufende Prozesse gesammelt, wie z.B. die Prozess ID, den Namen, Daten über RAM und CPU Verbrauch, sowie dessen Festplattenaktivitäten.

Im Standard werden in einem Intervall von 30 Sekunden vom System neue Werte abgefragt. Da der Agent keine historischen Daten liefern muss, sondern nur den aktuellen Zustand des Systems, werden veraltete Informationen regelmäßig mit Neuen überschrieben und nicht gespeichert.

Um eigene Checks oder das Ausführen bestehender Nagios Plugins zu ermöglichen, kann über eine Konfigurationsdatei festgelegt werden, welche Kommandos (Befehl oder Pfad zu einer ausführbaren Datei) in bestimmten zeitlichen Abständen, mit festgelegten Timeouts ausgeführt werden sollen.

Über diese Methode können auch einfach Nagios Plugins in den Agent integriert werden.

Über Startparameter oder eine Konfigurationsdatei, kann die Standardkonfiguration z.B. für den Intervall der Checks angepasst werden.

Für den externen Zugriff auf diese Daten startet der Agent einen lokalen Webserver (konfigurierbar mit IP-Adresse und Port) und stellt alle aktuellen, gesammelten Daten, formatiert nach JSON Standard, zur Verfügung.

Zusätzlich stehen Konfigurationsoptionen zur Verfügung, die es erlauben, die gesammelten Daten automatisch an eine openITCOCKPIT Instanz zu senden.

Es stehen für Windows, macOS und die gängigen Linux Systeme, wie Debian, Ubuntu oder Arch Pakete zur Verfügung.

Download hier: https://openitcockpit.io/agent

Benutzung

Melden Sie sich zunächst beim Webinterface Ihres openITCOCKPIT 4 an.

Wenn Sie bereits einen Host für den Computer konfiguriert haben, auf dem der Agent ausgeführt wird, navigieren Sie zu *Administration openITCOCKPIT Agent Agent Configuration*. Wählen Sie den Host aus, auf dem der Agent installiert ist, und befolgen Sie die dortigen Anweisungen.

Wenn Sie noch keinen Host für den Computer konfiguriert haben, auf dem der Agent ausgeführt wird, navigieren Sie zu *Monitoring Hosts* und klicken Sie auf **+ New**.

Klicken Sie nach der Hostkonfiguration auf **Create host and setup agent**, um den Host zu speichern und direkt mit der integrierten Agentenkonfiguration fortzufahren.

Sicherheitskonzept

Der Webserver des Agenten kann unter Verwendung eigener oder automatisch generierter Webserver-Zertifikate eine HTTPS Verbindung anstelle von HTTP anbieten.

Die automatische Generierung von Zertifikaten kann nur über eine konfigurierte Schnittstelle zu einem openITCOCKPIT Server durchgeführt werden.

Um die durch den Agenten abgefragten Systeminformationen zu schützen und Missbrauch der Konfigurationsschnittstelle über den Webserver zu vermeiden, sind zwei unabhängig voneinander funktionierende Authentifizierungsmechanismen in den Agenten integriert.

"Basic-Auth" Verfahren

Für den Zugriff auf den Webserver kann eine Kombination aus Benutzername und Passwort konfiguriert werden.

Diese wird beim Zugriff auf die Webseite mit dem sogenannten „Basic-Auth“ Verfahren abgefragt.

Zertifikat-Authentifizierung

Der openITCOCKPIT Server wird als CA (Zertifizierungsstelle), also für die Generierung der benötigten Zertifikate verwendet.

Für den Zugriff auf den Webserver des Agent wird ein Client-Zertifikat einer konfigurierten CA für die Anfrage benötigt. Anfragen, ohne ein von der selben CA ausgestelltes Zertifikat, werden abgewiesen.

Wurde ein Zertifikat für einen Agent generiert, erwartet der Server auch die Verwendung dessen, egal, ob sich der Agent im Pull- oder Push-Modus befindet.

Pull-Modus

Das Zertifikat (einschließlich Updates) wird von openITCOCKPIT an den Agenten übertragen.

Während der Konfiguration des Agenten im openITCOCKPIT muss dazu die Option "Try autossl mode" aktiviert werden.

Push-Modus

Das Zertifikat (einschließlich Aktualisierungen) wird vom Agenten angefordert. Eine entsprechende Anfrage wird regelmäßig an den openITCOCKPIT Server gesendet.

Dieser Anfrage muss aus Sicherheitsgründen im openITCOCKPIT durch einen Benutzer manuell vertraut werden, um das Zertifikat zu erzeugen.

Benötigte Erweiterung der Standardkonfiguration im **Push**-Modus:

config.cnf

```
[oitc]

# The UUID of the Host.
# You can find this information in the openITCOCKPIT interface
# Example: 402357e4-dc34-4f5b-a86d-e59cfbb3ffe7
hostuuid =

# Address of your openITCOCKPIT Server
# Example: https://openitcockpit.io/receiver
url =

# API-Key of your openITCOCKPIT Server
apikey =
```

Installation

Es stehen für Windows, macOS und die gängigen Linux Systeme, wie Debian, Ubuntu oder Arch Pakete zur Verfügung.

Download hier: https://openitcockpit.io/agent

Konfiguration

Was ist der Pull / Push Mode?

Soll der Agent im **Pull**-Modus konfiguriert werden, muss er über das Netzwerk erreichbar sein.

openITCOCKPIT wird versuchen, über die IP-Adresse des Hosts eine Verbindung zum Agenten herzustellen.

Die Prüfergebnisse werden in einem minütlichen Prüfintervall abgerufen.

Soll der Agent im **Push**-Modus konfiguriert werden, muss er eine Verbindung zum openITCOCKPIT-Server in Ihrem Netzwerk herstellen.

Der Agent sendet die Prüfergebnisse in einem bestimmten Intervall an den openITCOCKPIT-Server.

Benötigte Erweiterung der Standardkonfiguration:

config.cnf

```
[oitc]

# Enable Push Mode
enabled = true

# The UUID of the Host.
# You can find this information in the openITCOCKPIT interface
# Example: 402357e4-dc34-4f5b-a86d-e59cfbb3ffe7
hostuuid =

# Address of your openITCOCKPIT Server
# Example: https://openitcockpit.io/receiver
url =

# API-Key of your openITCOCKPIT Server
apikey =
```

Gesamte Konfigurationsmöglichkeiten

Nach der Installation kann der Agent konfiguriert werden.

Die Konfigurationsdatei des Agenten (INI-Format) ist abhängig vom System an verschiedenen Orten gespeichert. Es gelten folgende Dateipfade:

Windows	`C:\Program Files\openitcockpit-agent\config.cnf` `C:\Program Files\openitcockpit-agent\customchecks.cnf`
Linux	`/etc/openitcockpit-agent/config.cnf` `/etc/openitcockpit-agent/customchecks.cnf`
macOS	/Applications/openitcockpit-agent/config.cnf /Applications/openitcockpit-agent/`customchecks.cnf`

Die Konfiguration des Agenten kann an den folgenden zwei Stellen, als Startparameter oder Konfigurationsoption, vorgenommen werden.

1. Als Parameter zum Programmstart

Die in der weiter unten stehenden Liste als Startparameter definierten Optionen können der auszuführenden Startdatei des Agenten als Parameter übergeben werden.

(Von dieser Verwendung im produktiven Betrieb wird abgeraten!)

Beispielkommando

```
/usr/bin/openitcockpit-agent-python3.linux.bin --config /etc/openitcockpit-agent/config.cnf --verbose -s --port
80 --config-update-mode
```

2. In den Konfigurationsdateien

config.cnf

```
[default]

# Determines in seconds how often the agent will schedule all internal checks
interval = 30

# Port of the Agents buil-in web server
port = 3333

# Bind address of the build-in web server
address = 0.0.0.0

# If a certificate file is given, the agent will switch to https only
# Example: /etc/ssl/certs/ssl-cert-snakeoil.pem
certfile =

# Private key file of the given TLS certificate
# Example: /etc/ssl/private/ssl-cert-snakeoil.key
keyfile =

# Try to enable auto ssl mode for webserver
try-autossl = true

# File paths used for autossl (default: /etc/openitcockpit-agent/... or C:\Program Files\openitcockpit-
agent\...):
# Example: /etc/openitcockpit/agent.csr
autossl-csr-file =
# Example: /etc/openitcockpit/agent.crt
autossl-crt-file =
# Example: /etc/openitcockpit/agent.key
autossl-key-file =
# Example: /etc/openitcockpit/server_ca.crt
autossl-ca-file =

# Print most messages
verbose = false

# Print all messages with stacktrace
# For developers
stacktrace = false

# Enable remote read and write of THIS config file and custom checks definition
# Examples:
#   Read config: curl http://0.0.0.0:3333/config
#   Write config: curl -X POST -d '{"config": {"interval": "60", "port": "3333", "address": "0.0.0.0",
"certfile": "/etc/ssl/certs/ssl-cert-snakeoil.pem", "keyfile": "/etc/ssl/private/ssl-cert-snakeoil.key",
"verbose": "true", "stacktrace": "false", "config-update-mode": "true", "auth": "", "customchecks": "",
"temperature-fahrenheit": "false", "oitc-host": "", "oitc-url": "", "oitc-apikey": "", "oitc-interval": "60",
"oitc-enabled": "false"}, "customchecks": {}}' http://0.0.0.0:3333/config
config-update-mode = false

# Enable Basic Authentication
# Disabled if blank
# Example: auth = user:password
auth =

# Remote Plugin Execution
# Path to config will where custom checks can be defined
customchecks = /etc/openitcockpit-agent/customchecks.cnf
```

```
# Return temperature values as fahrenheit
temperature-fahrenheit = false

# Try to check docker containers and return stats in default output
dockerstats = false

# Try to check qemu virtual machines and return stats in default output
qemustats = false

# Enable default cpu status check
cpustats = true

# Enable default sensor status check
sensorstats = true

# Enable default process status check
processstats = true

# Add process child ids to the default process status check (computationally intensive)
processstats-including-child-ids = false

# Enable default network status check
netstats = true

# Enable default disk status check
diskstats = true

# Enable default network I/O calculation
netio = true

# Enable default disk I/O calculation
diskio = true

# Enable default windows services status check
winservices = true

# Enable default systemd services status check
systemdservices = true

# If you have an Alfresco enterprise instance, JMX is configured and java installed, you can enable
alfrescostats
alfrescostats = false

# Set your Alfresco JMX username
alfresco-jmxuser = monitorRole

# Set your Alfresco JMX password
alfresco-jmxpassword = change_asap

# Set your Alfresco host address
alfresco-jmxaddress = 0.0.0.0

# Set your Alfresco JMX port
alfresco-jmxport = 50500

# Set your Alfresco JMX path (path behind the JMX address "service:jmx:rmi:///jndi/rmi://0.0.0.0:50500")
alfresco-jmxpath = /alfresco/jmxrmi

# Set you custom Alfresco JMX query. Leave empty to use the default.
alfresco-jmxquery =

# Path to the java binary (java need to be installed on agent host system in case you want to use alfrescostats)
```

```
alfresco-javapath = /usr/bin/java


# By default openITCOCKPIT will pull check results from the openITCOCKPIT Agent.
# In a Cloud environments or behind a NAT network it could become handy
# if the openITCOCKPIT Agent will push the results to your openITCOCKPIT Server
[oitc]

# Enable Push Mode
enabled = false

# The UUID of the Host.
# You can find this information in the openITCOCKPIT interface
# Example: 402357e4-dc34-4f5b-a86d-e59cfbb3ffe7
hostuuid =

# Address of your openITCOCKPIT Server
# Example: https://openitcockpit.io/receiver
url =

# API-Key of your openITCOCKPIT Server
apikey =

# Determines in seconds how often the agent will push
# check results to your openITCOCKPIT Server
interval = 60
```

customchecks.cnf

```
[default]
# max_worker_threads should be increased with increasing number of custom checks
# but consider: each thread needs (a bit) memory
max_worker_threads = 8

#[check_users]
# command = /usr/lib/nagios/plugins/check_users -w 5 -c 10
# interval = 30
# timeout = 5
# enabled = true

#[check_load]
# command = /usr/lib/nagios/plugins/check_load -r -w .15,.10,.05 -c .30,.25,.20
# interval = 60
# timeout = 5
# enabled = true
```

Zur Verfügung stehen folgende Konfigurationsoptionen in der Sektion **[default]** der Datei **config.cnf**:

Fett markierte Werte entsprechen jeweils der Standard Einstellung.

Parameter	Startparameter	Beispielwert	Beschreibung
interval	-i --interval	**30**	Legt in Sekunden fest, wie oft der Agent die integrierten Standard Checks ausführt
port	-p --port	**3333**	Port des in den Agenten eingebauten Web Servers
address	-a –address	**127.0.0.1**	IP Adresse, auf welcher der eingebaute Web Server laufen soll

auth	--auth	user:password	Definiert und aktiviert HTTP Basic Auth
verbose	-v --verbose	true / **false** *Diese Angabe wird nur in der INI-Datei Konfiguration benötigt*	Hinzufügen des Startparameters aktiviert die Ausgabe von Informationen und Fehlermeldungen
stacktrace	-s --stacktrace	true / **false** *Diese Angabe wird nur in der INI-Datei Konfiguration benötigt*	Hinzufügen des Startparameters aktiviert die Ausgabe von möglichen Stacktraces
config-update-mode	--config-update-mode	true / **false** *Diese Angabe wird nur in der INI-Datei Konfiguration benötigt*	Aktiviert den Fernzugriff zum Lesen und Schreiben neuer Konfigurationen. Konfigurationsaktualisierung über POST Anfragen. /config um die aktuelle Konfiguration über den Webserver des Agenten abzurufen.
temperature-fahrenheit	--temperature-fahrenheit	true / **false** *Diese Angabe wird nur in der INI-Datei Konfiguration benötigt*	Ändert Temperaturangaben, sofern aktiviert, auf die Einheit Fahrenheit. Standard ist Celsius
	-h --help		Gibt eine Hilfemeldung aus und beendet das Programm
	-c --config	[..]/config.cnf	Pfad zur Konfigurationsdatei des Agenten
customchecks	--customchecks	[..]/customchecks.cnf	Pfad zur Konfigurationsdatei für benutzerdefinierte Checks
dockerstats	--dockerstats	true / **false** *Diese Angabe wird nur in der INI-Datei Konfiguration benötigt*	Versucht Docker Container zu überprüfen und fügt deren Status der Standardausgabe hinzu
qemustats	--qemustats	true / **false** *Diese Angabe wird nur in der INI-Datei Konfiguration benötigt*	Versucht virtuelle QEMU Maschinen zu überprüfen und fügt deren Status der Standardausgabe hinzu (nur für Linux, Beta)
cpustats	--no-cpustats	**true** / false *Diese Angabe wird nur in der INI-Datei Konfiguration benötigt*	Deaktiviert den CPU-Status Standardcheck
sensorstats	--no-sensorstats	**true** / false *Diese Angabe wird nur in der INI-Datei Konfiguration benötigt*	Deaktiviert den Sensor-Status Standardcheck
processstats	--no-processstats	**true** / false *Diese Angabe wird nur in der INI-Datei Konfiguration benötigt*	Deaktiviert den Prozess-Status Standardcheck
processstats-including-child-ids	--processstats-including-child-ids	true / **false** *Diese Angabe wird nur in der INI-Datei Konfiguration benötigt*	Hinzufügen von untergeordneten Prozess-IDs zur Standardprüfung des Prozessstatus (rechenintensiv)
netstats	--no-netstats	**true** / false *Diese Angabe wird nur in der INI-Datei Konfiguration benötigt*	Deaktiviert den Netzwerk-Status Standardcheck
diskstats	--no-diskstats	**true** / false *Diese Angabe wird nur in der INI-Datei Konfiguration benötigt*	Deaktiviert den Festplatten-Status Standardcheck
netio	--no-netio	**true** / false *Diese Angabe wird nur in der INI-Datei Konfiguration benötigt*	Deaktiviert die Netzwerk I/O Berechnung
diskio	--no-diskio	**true** / false *Diese Angabe wird nur in der INI-Datei Konfiguration benötigt*	Deaktiviert die Festplatten I/O Berechnung

winservices	--no-winservices	**true** / false *Diese Angabe wird nur in der INI-Datei Konfiguration benötigt*	Deaktiviert den Windows Services Status Standardcheck (nur für Windows)
systemdservices		**true** / false	Deaktiviert den systemd Services Status Standardcheck

Das Hinzufügen folgender Parameter kann den SSL verschlüsselten Webserver (https) aktivieren:

Parameter	**Startparameter**	**Beispielwert**	**Beschreibung**
certfile	--certfile	/path/to/cert.pem	Wenn eine Zertifikatsdatei angegeben ist, verwendet der Webserver ausschließlich das https Protokoll
keyfile	--keyfile	/path/to/key.pem	Pfad zum privaten Schlüssel der Zertifikatsdatei (wird für die Angabe der „certfile“ Option benötigt)
try-autossl	--try-autossl	**true** / false *Diese Angabe wird nur in der INI-Datei Konfiguration benötigt*	Versucht den automatischen SSL Modus für den Webserver zu aktivieren. Benötigt die Konfiguration eines openITCOCKPIT Servers

Mit folgenden Parametern können die Dateipfade für die „autossl“ Option zum automatischen Erzeugen von SSL Zertifikaten angepasst werden:

Parameter	**Startparameter**	**Beispielwert**	**Beschreibung**
autossl-csr-file	--autossl-csr-file	*/etc/openitcockpit/agent.csr*	Pfad zur csr Datei (Zertifikatanfrage)
autossl-crt-file	--autossl-crt-file	*/etc/openitcockpit/agent.crt*	Pfad zur crt Datei (Zertifikat)
autossl-key-file	--autossl-key-file	*/etc/openitcockpit/agent.key*	Pfad zur key Datei (Zertifikatsschlüssel)
autossl-ca-file	--autossl-ca-file	*/etc/openitcockpit/server_ca.crt*	Pfad zur ca Datei (Zertifikat der Zertifizierungsstelle)

Mit folgenden Parametern kann eine Alfresco Enterprise Instanz überwacht werden:

Parameter	**Startparameter**	**Beispielwert**	**Beschreibung**
alfrescostats		true **/ false**	Wenn Sie eine Alfresco Enterprise Instanz haben, JMX konfiguriert und Java auf dem Agent Hostsystem installiert ist, können Sie Alfrescostats aktivieren
alfresco-jmxuser		monitorRole	Definiert deinen Alfresco JMX Benutzernamen
alfresco-jmxpassword		change_asap	Definiert dein Alfresco JMX Passwort
alfresco-jmxaddress		0.0.0.0	Definiert deine Alfresco Host Adresse
alfresco-jmxport		50500	Definiert deinen Alfresco JMX Port
alfresco-jmxpath		/alfresco/jmxrmi	Definiert deinen Alfresco JMX Pfad (Pfad hinter der JMX Adresse "service:jmx:rmi:///jndi/rmi://0.0.0.0:50500")
alfresco-jmxquery			Definiert deine eigene Alfresco JMX Abfrage. Leer lassen, um den Standard zu nutzen.
alfresco-javapath		/usr/bin/java	Pfad zur ausführbaren Java Datei deines Systems (Java muss auf dem Agent-Host-System installiert sein, wenn alfrescostats verwenden werden soll)

Zur Verfügung stehen folgende Konfigurationsoptionen in der Sektion **[oitc]** der Datei **config.cnf**:

Die Verwendung dieser Optionen, ausgenommen „enabled“ und „interval“, wird für die Funktionalität der „autossl“ Option benötigt.

Parameter	Startparameter	Beispielwert	Beschreibung
enabled		true / **false** *Diese Angabe wird nur in der INI-Datei Konfiguration benötigt*	Aktiviert den Push Modus Benötigt die Konfiguration eines openITCOCKPIT Servers
hostuuid	--oitc-hostuuid	*402357e4-d……*	UUID des openITCOCKPIT Hosts
url	--oitc-url	*https://openitcockpit.io/receiver*	Adresse des openITCOCKPIT Servers
apikey	--oitc-apikey	*1XC8nZ2On……*	API-Key des openITCOCKPIT Servers
interval	--oitc-interval	*30*	Legt in Sekunden fest, wie oft der Agent die Prüfungs-ergebnisse an den Server sendet.

Sofern eine Konfigurationsdatei für eigene Checks angegeben ist, können folgende Parameter zur Konfiguration dieser verwendet werden.

Sektion **[default]** der Datei **customchecks.cnf**:

Parameter	Beispielwert	Beschreibung
max_worker_threads	**8**	Maximale Anzahl an Threads, die zur Bearbeitung benutzerdefinierter Checks verwendet werden dürfen

Soll ein eigener Check hinzugefügt werden, so muss diesem ein eindeutiger Name zugewiesen werden. Dieser wird als Sektion in eckige Klammern geschrieben.

Sektion **[username]** der Datei **customchecks.cnf**:

Parameter	Beispielwert	Beschreibung
command	whoami	Kommando des auszuführenden Checks oder Pfad zur auszuführenden Datei
interval	120	Legt in Sekunden fest, wie oft der Agent den definierten Check ausführt
timeout	5	Legt in Sekunden fest, nach welcher Zeit der Check spätestens abgebrochen werden soll
enabled	true / false	Aktiviert oder deaktiviert den Check

Führe ein PowerShell Script als customcheck aus

Stellen Sie sicher, dass der Pfad zur Datei customcheck.cnf in Ihrer config.cnf unter C:\Program Files\it-novum\openitcockpit-agent\ konfiguriert ist.

Passe das Timeout für den customcheck bei Bedarf an. Versuche es mit einem Timeout von 10 Sekunden. Wenn immer noch ein Timeout auftritt, empfehlen wir, den Timeout-Wert schrittweise zu erhöhen.

Ihr Befehl sollte wie dieser Beispielbefehl aussehen.

Beispiel Powershell Skript

```
Write-Host "Hallo customcheck"
exit 2
```

In C:\Program Files\it-novum\openitcockpit-agent\customchecks.cnf

```
[check_xyz]
command = powershell.exe -nologo -noprofile -File "C:\example.ps1"
interval = 60
timeout = 10
enabled = true
```

Öffne eine CMD als Administrator.

Die Ausführung von Powershell-Skriptdateien muss aktiviert werden.

Den Agent neu starten, um die angepasste Konfiguration anzuwenden.

CMD als Administrator

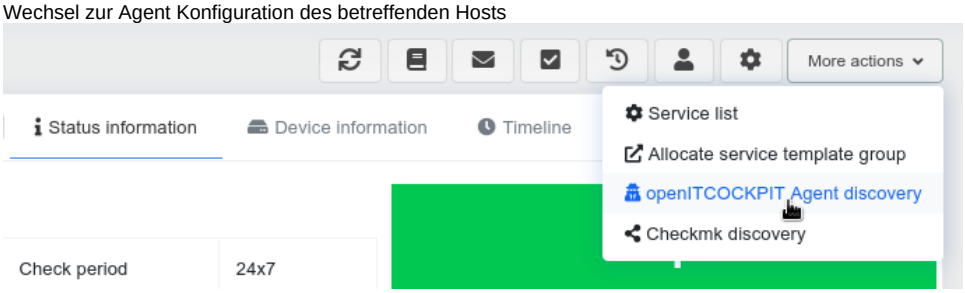
```
powershell Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope LocalMachine
sc.exe stop oitcAgentSvc
sc.exe start oitcAgentSvc
```

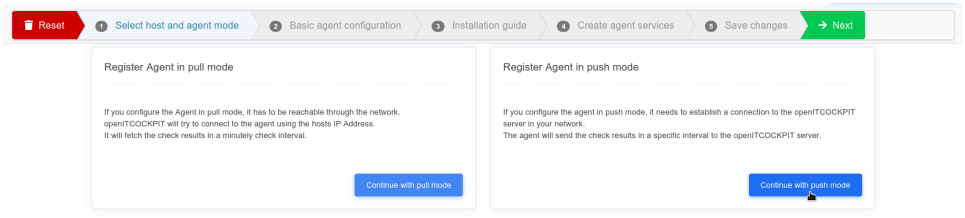
Danach kann der neue Check als Service mithilfe der openITCOCKPIT Agent Configuration einem Host hinzugefügt werden.

(Folge den Anweisungen in Benutzung)

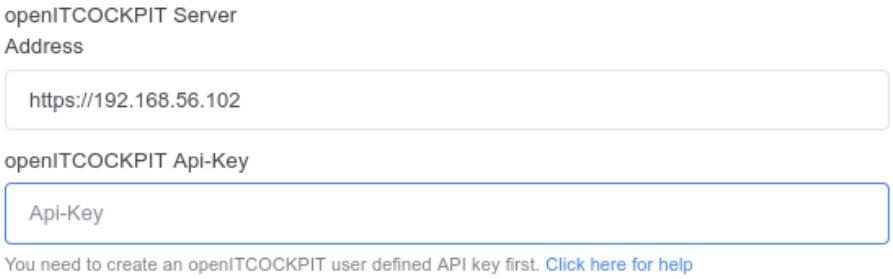
Nachträglicher Wechsel zwischen Pull- und Push-Mode

Der Pull-Mode wurde konfiguriert, es sind bereits Services angelegt und im Monitoring

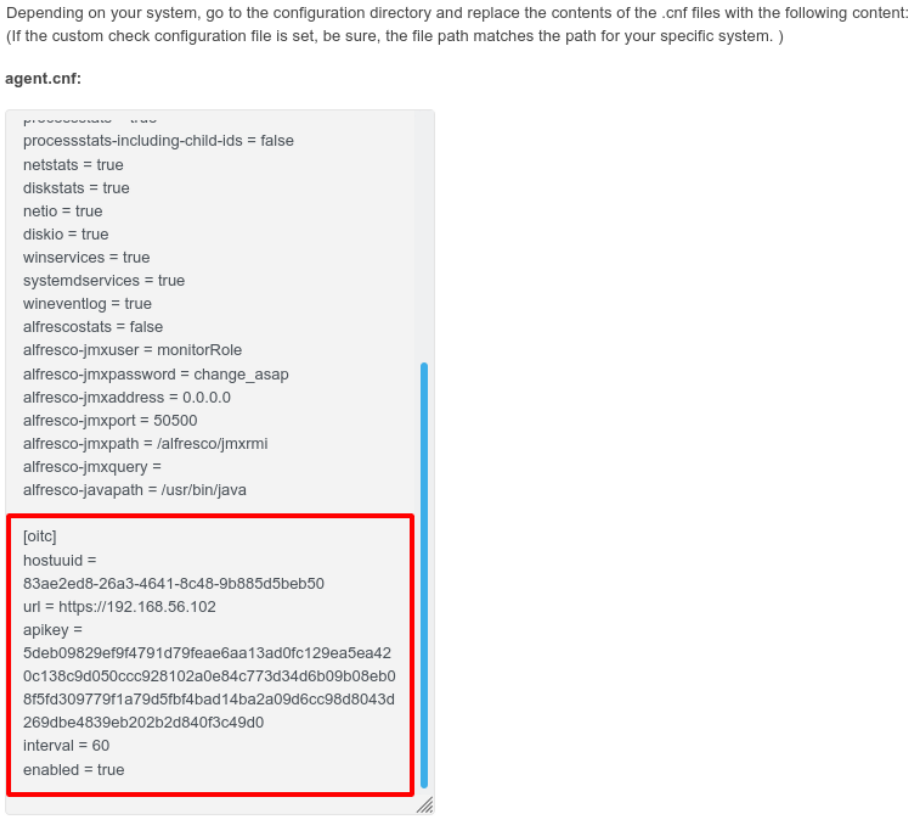
1. Wechsel zur Agent Konfiguration des betreffenden Hosts
2. 

3. Wähle den Push-Mode
4. 

5. Generiere einen Api-Key und füge ihn ins Feld "openITCOCKPIT Api-Key" ein.



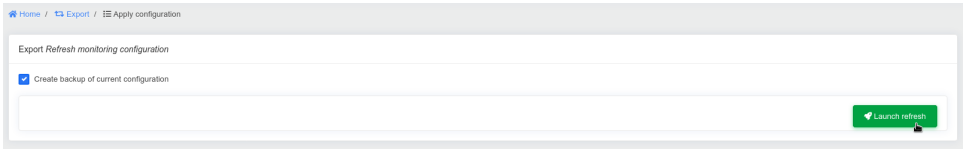
6.
7. Kontrolliere, ob die angezeigte openITCOCKPIT Server Adresse vom Host des Agents erreichbar ist. Korrigiere diese wenn nötig. Das Protokoll (https) muss immer mit angegeben werden.
8. Drücke auf **Next** und kopiere den unteren Teil der generierten Agent-Konfiguration (ab [oitc]). Dieser Teil muss in die Konfigurationsdatei des Agent auf dem zu überwachenden System eingefügt werden. Sollte bereits ein [oitc] Bereich in der Konfiguration bestehen, muss dieser überschrieben werden!

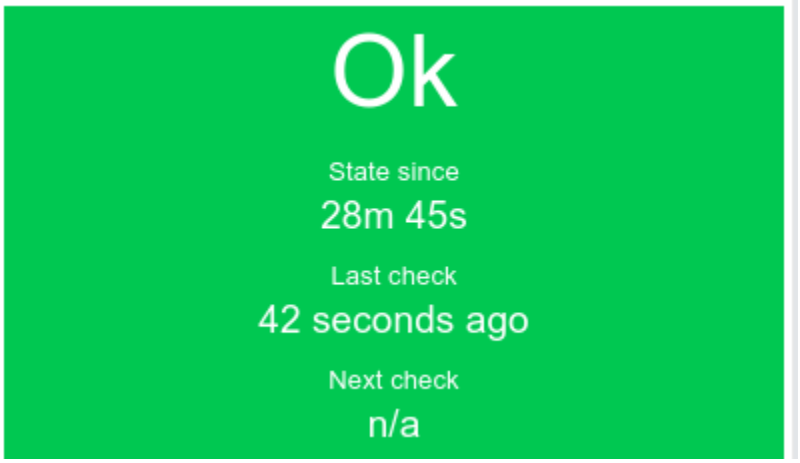


9.
10. Danach muss der Agent neu gestartet werden, um die Konfiguration zu laden. Eine kurze Anleitung dazu findet sich im openITCOCKPIT unter der generierten Konfiguration. Es können folgende Befehle verwendet werden:
11.

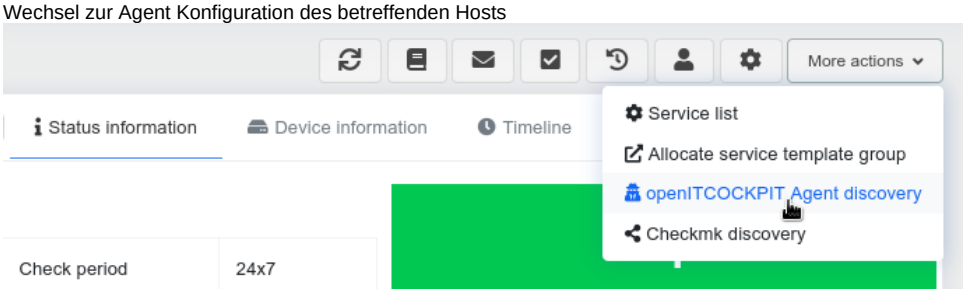
Windows CMD	`sc stop oitcAgentSvc && sc start oitcAgentSvc`
Linux	`systemctl restart openitcockpit-agent`
macOS	`sudo launchctl stop com.it-novum.openitcockpit.agent` `sudo launchctl start com.it-novum.openitcockpit.agent`

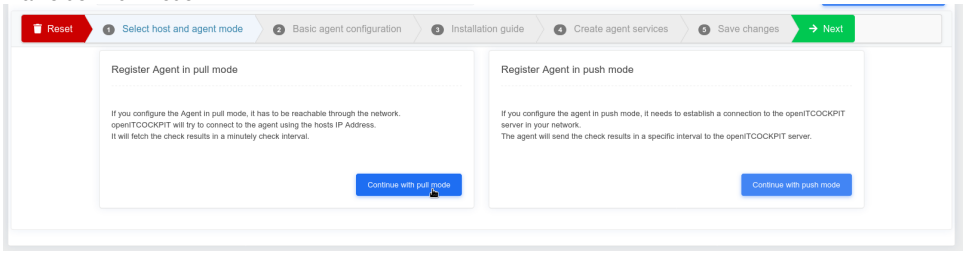
12. Nach dem Neustart des Agent sollte etwa 5-10 Sekunden gewartet werden, damit der Agent die ersten Check Ergebnisse ans openITCOCKPIT senden kann. Es wird mit einem Klick auf **Next** fortgefahren.
13. Auf Wunsch können weitere Services zum Monitoring hinzugefügt werden. In jedem Fall (auch ohne eine Auswahl) muss noch ein mal auf **Next** geklickt werden, um die Konfiguration zu speichern.
14. Zuletzt muss ein Export durchgeführt werden, um die Änderungen an das Monitoring System zu übergeben.

15. 

16. Die durch den Agent überwachten Services des Hosts sollten sich nun jeden Minute automatisch aktualisieren.
17. 

Der Push-Mode wurde konfiguriert, es sind bereits Services angelegt und im Monitoring

1. Wechsel zur Agent Konfiguration des betreffenden Hosts
2. 

3. Wähle den Pull-Mode
4. 

5. Sofern der Webserver Port des Agenten zuvor geändert wurde, muss diese Änderung im Agent-Konfigurationsinterface des openITCOCKPIT angepasst werden.
6. Bevor fortgefahren werden kann, muss der Agent auf dem zu überwachenden Host gestoppt werden. Dazu können folgende Befehle für das jeweilige System verwendet werden:
7. | | |
|---|---|
| Windows CMD | `sc stop oitcAgentSvc` |
| Linux | `systemctl stop openitcockpit-agent` |
| macOS | `sudo launchctl stop com.it-novum.openitcockpit.agent` |
8. Drücke auf **Next** um die generierte Konfiguration angezeigt zu bekommen. Öffne die Konfigurationsdatei des Agent auf dem zu überwachenden System und stelle sicher, dass im [oitc] Bereich die Option "enabled = false" gesetzt ist!

Depending on your system, go to the configuration directory and replace the contents of the .cnf files with the following content:
(If the custom check configuration file is set, be sure, the file path matches the path for your specific system.)

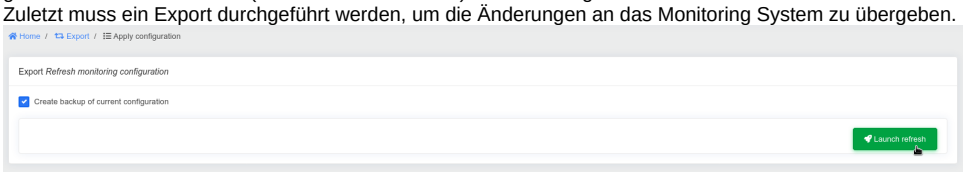
agent.cnf:

```
dockerstats = false
qemustats = false
cpustats = true
sensorstats = true
processstats = true
processstats-including-child-ids = false
netstats = true
diskstats = true
netio = true
diskio = true
winservices = true
systemdservices = true
wineventlog = true
alfrescostats = false
alfresco-jmxuser = monitorRole
alfresco-jmxpassword = change_asap
alfresco-jmxaddress = 0.0.0.0
alfresco-jmxport = 50500
alfresco-jmxpath = /alfresco/jmxrmi
alfresco-jmxquery =
alfresco-javapath = /usr/bin/java

[oitc]
hostuuid =
url =
apikey =
interval = 60
enabled = false
```

9.
10. Danach muss der Agent gestartet werden, um die Konfiguration zu laden. Dies kann wie folgt getan werden:
11.

Windows CMD	`sc start oitcAgentSvc`
Linux	`systemctl start openitcockpit-agent`
macOS	`sudo launchctl start com.it-novum.openitcockpit.agent`

12. Es wird mit einem Klick auf **Next** fortgefahren.
13. Die angezeigten Optionen stammen aus veralteten, zwischengespeicherten Daten, die vom Agent zuvor im Push-Mode ans openITCOCKPIT gesendet wurden. Es muss (mit oder ohne Auswahl) mit **Next** fortgefahren werden.
14. Zuletzt muss ein Export durchgeführt werden, um die Änderungen an das Monitoring System zu übergeben.
15. 

Zugriff auf den Webserver des Agent

Der Agent stellt die Ergebnisse der Checks über den integrierten Webserver zur Verfügung. Die Ausgabe erfolgt im JSON Format.

Der Webserver kann, sofern nicht anders konfiguriert, über die Angabe einer URL aus der IP Adresse des ausführenden Hosts, in einem Web-Browser aufgerufen werden.

z.B. http://0.0.0.0:3333/

Die komplette Struktur der Ausgabe ist der echten Ausgabe des laufenden Agenten zu entnehmen.

Ausgegeben werden z.T. folgende übergeordnete Objekte im JSON Format, welche jeweils weitere Objekte mit den Check Ergebnissen enthalten.

- disks (Speichergeräte mit Mountpoint, Dateisystem und Speicherplatzangaben)
- disk_io (Lese- und Schreibstatistiken der Speichergeräte)
- net_io (Eingabe- und Ausgabestatistiken der Netzwerkgeräte)
- net_stats (Netzwerkgeräte mit zur Verfügung stehender Geschwindigkeit, ...)
- sensors (Angeschlossene Sensoren, z.B. Temperatur der CPU, Akkustatus)
- cpu_total_percentage (Verwendete CPU Rechenzeit in Prozent)
- cpu_percentage (Verwendete CPU Rechenzeit pro Kern in Prozent)
- cpu_total_percentage_detailed (CPU Rechenzeit (in %) je Systemressource)
- cpu_percentage_detailed (CPU Rechenzeit (in %) je Systemressource pro Kern)
- system_load (System Load 1, 5, 15 als Array)
- users (Am System angemeldete Benutzer, deren Terminal (PID), Loginzeitpunkt)
- memory (Informationen zum Arbeitsspeicher, verwendet, aktiv, gepuffert, ...)
- swap (Informationen zum Auslagerungsspeicher, insgesamt, verwendet, ...)
- processes (Informationen zu laufenden Prozessen, CPU, Arbeitsspeicher, PID, ...)
- agent (Version des Agenten, Zeitpunkt des letzten Checks, Systemversion, ...)
- dockerstats (Aktive Docker Container, ID, CPU, Arbeitsspeicher, Block IO, PID)
- qemustats (Informationen zu aktiven QEMU Maschinen (auf einem Proxmox))

.....

Konfiguration per API abrufen / aktualisieren

Die Konfiguration per API sollte aus Sicherheitsgründen erst nach einer erfolgreichen SSL-Konfiguration aktiviert werden.

Warnung: Die Remotecodeausführung ist möglich, wenn das Zertifikat gestohlen wurde oder kein SSL konfiguriert wurde.

Sie können die Hauptkonfiguration, sowie die Konfiguration für benutzerdefinierte Checks im laufenden Betrieb aktualisieren, indem Sie eine Post-Anfrage mit json-formatierten Daten senden.

Benötigte Anpassung der Standardkonfiguration:

config.cnf

```
[oitc]

# Enable remote read and write of THIS config file and custom checks definition
config-update-mode = true
```

Beispiel: Konfiguration lesen

```
curl http://0.0.0.0:3333/config
```

Beispiel: Konfiguration schreiben

```
curl -X POST -d '{"config": {"interval": "60", "port": "3333", "address": "0.0.0.0", "verbose": "true",
"stacktrace": "false", "config-update-mode": "true", "temperature-fahrenheit": "false", "oitc-host": "", "oitc-
url": "", "oitc-apikey": "", "oitc-interval": "60", "oitc-enabled": "false"}, "customchecks": {}}' http://0.
0.0.0:3333/config
```

Beispiel: Konfiguration schreiben (aus Datei)

```
curl -X POST -d @new_config.json http://0.0.0.0:3333/config -u user:pass
```

Beispieldatei: new_config.json

```
{
   "config":{
      "interval":"30",
      "port":"3333",
      "address":"0.0.0.0",
      "certfile":"",
      "keyfile":"",
      "try-autossl":"true",
      "autossl-folder":"/etc/openitcockpit-agent",
      "verbose":"false",
      "stacktrace":"false",
      "config-update-mode":"true",
      "auth":"",
      "customchecks":"/etc/openitcockpit-agent/oitc_customchecks.conf",
      "temperature-fahrenheit":"false",
      "dockerstats":"true",
      "qemustats":"true",
      "cpustats":"true",
      "sensorstats":"true",
      "processstats":"true",
      "processstats-including-child-ids":"false",
      "netstats":"true",
      "diskstats":"true",
      "netio":"true",
      "diskio":"true",
      "winservices":"true",
      "systemdservices":"true",
      "alfrescostats":"false",
      "alfresco-jmxuser":"monitorRole",
      "alfresco-jmxpassword":"test123",
      "alfresco-jmxaddress":"10.10.10.1",
      "alfresco-jmxport":"50500",
      "alfresco-jmxpath":"/alfresco/jmxrmi",
      "alfresco-jmxquery":"",
      "alfresco-javapath":"/usr/bin/java",
      "oitc-hostuuid":"",
      "oitc-url":"",
      "oitc-apikey":"",
      "oitc-interval":"60",
      "oitc-enabled":"false"
   },
   "customchecks":{
      "default": {
         "max_worker_threads": 8
      },
      "username": {
         "command": "whoami",
         "interval": 30,
         "timeout": 5,
         "enabled": "1"
      },
      "uname": {
         "command": "uname -a",
         "interval": 15,
         "timeout": 5,
         "enabled": "0"
      }
   }
}
```

Konfiguration lesen und speichern mit HTTPS und autossl.

Folgendes Kommando könnte auf einem openITCOCKPIT Server ausgeführt werden.

Beispiel: Konfiguration lesen und speichern (HTTPS/autossl)

```
curl https://192.168.122.1:3333/config --cert /opt/openitc/agent/server_ca.pem --key /opt/openitc/agent
/server_ca.key -k -o agentconfig.json
```

Beispiel: Konfiguration schreiben (HTTPS/autossl)

```
curl -X POST -d @agentconfig.json https://192.168.122.1:3333/config --cert /opt/openitc/agent/server_ca.pem --
key /opt/openitc/agent/server_ca.key -k
```

Konfiguration per API über einen anderen Host

Die Konfiguration per API sollte im Standard nur über die Agent Konfiguration im openITCOCKPIT vorgenommen werden.

Manuelles Ändern der Konfiguration kann dazu führen, dass openITCOCKPIT den Agent nicht mehr erreichen kann.

Ändern von z.B. dem Webserver Port oder die De-/Aktivierung der HTTPS Verschlüsselung sollte nur über openITCOCKPIT erfolgen!

Soll die Konfiguration über die Agent API prudiktiv geschehen, empfehlen wir für diesen Zweck ein eigenes Zertifikat zu erstellen.

Dieses kann dann auch einfach auf einen anderen Host kopiert werden, von welchen die Konfiguration aktualisiert wird.

Erstellung eines eigenen Zertifikats durch die openITCOCKPIT Agent CA

```
openssl genrsa -out customcert.oitc.key 4096

openssl req -new -sha512 -key customcert.oitc.key -subj "/C=US/ST=CA/O=MyOrg, Inc./CN=customcert.oitc" -out
customcert.oitc.csr -config <(cat /etc/ssl/openssl.cnf | sed "s/RANDFILE\s*=\s*\$ENV::HOME\/\.rnd/#/")

openssl x509 -req -in customcert.oitc.csr -CA /opt/openitc/agent/server_ca.pem -CAkey /opt/openitc/agent
/server_ca.key -CAcreateserial -out customcert.oitc.crt -days 365 -sha512
```

Beispiel: Nutzung des eigenen Zertifikats zum Lesen der Agent Konfiguration

```
curl https://192.168.122.1:3333/config --cert customcert.oitc.crt --key customcert.oitc.key -k
```

Zugriff auf den verschlüsselten Agent Webserver

Erstelle eine .p12-Datei, die als Zertifikat in einen Webbrowser (wie Firefox) importiert werden kann, um auf den verschlüsselten Webserver des Agenten zugreifen zu können.
Generiere benutzerdefinierte Client-Zertifikate wie im vorhergehenden Abschnitt beschrieben (Erstellung eines eigenen Zertifikats durch die openITCOCKPIT Agent CA).

Erstelle das Browserzertifikat aus dem generierten Clientzertifikat

```
cat customcert.oitc.crt customcert.oitc.key > both_customcert.oitc.pem
openssl pkcs12 -export -in both_customcert.oitc.pem -out both_customcert.oitc.p12
```

openITCOCKPIT Integration - Nice to know :)

- Host anlegen (Button: Create host and and setup Agent)

- Agent auf dem Host einrichten (zum Anlegen der Standardcecks)

- Für Push Mode mit automatischer Zertifikatgenerierung muss dem Agent vertraut werden. Diese Berechtigung kann jederzeit wieder entzogen werden.

- Agent Konfiguration:

  - Basic Mode (Pull from Agent HTTP Webserver)
  - Advanced Security Mode (Pull from Agent HTTPS Webserver or Push) mit automatischer Zertifikatgenerierung
  - Agent Ausgabe wird ausgewertet und die zu überwachenden Daten sollten ausgewählt werden
  - Konfiguration speichern

- openITCOCKPIT erstellt die Services auf Basis der checkspezifischen Agent-Servicetemplates (dort können Standardwerte angepasst werden)

  - Services werden passiv konfiguriert und bekommen ein "dummy Kommando", sowie den service_type = 16 in der Datenbank.

- Exportieren

  - Der Host bekommt einen aktiven, automatisch generierten Service, welcher jede Minute eine Verbindung zum Agent herstellt, um die aktuellen Daten abzufragen.

- Check Daten, die vom Agent im Push-Modus ans openITCOCKPIT gesendet werden, werden direkt ausgewertet. Die letzte Version wird immer in der Datenbank gespeichert, um eine spätere flüssige Agent Konfiguration zu ermöglichen.

Zertifikatauthentifizierung

openITCOCKPIT ist im Standard die CA (Zertifizierungsstelle) für die Generierung der benötigten Zertifikate des Agents für eine HTTPS Verbindung.

Push Modus:

Der Agent erstellt eine Zertifikatanfrage, die an den openITCOCKPIT Server gesendet wird.

Im openITCOCKPIT muss in der **Agent Overview** im Bereich **Untrusted Agents** dem Agent mit entsprechendem Host und IP manuell vertraut werden.

- wenn dem Agent noch nicht vertraut wird, bekommt dieser eine entsprechende Fehlermeldung und versucht es nach 10 Minuten erneut.
- wurde dem Agent vertraut, wird die nächste Anfrage mit dem Zertifikat beantwortet.

Pull Modus:

Wurde in der openITCOCKPIT Konfigurationsoberfläche für den Agent die AutoSSL Option aktiviert, wird nach dem Erstellen der Services eine Verbindung zum Agent Webserver aufgebaut, um sich eine neue Zertifikatanfrage zu holen.

Da in der Konfigurationsdatei des Agents die Option "try-autossl" im Standard aktiviert ist, wird eine Zertifikatanfrage generiert und zurückgegeben. Ist diese Option deaktiviert, passiert nichts weiter.

Wurde eine gültige Zertifikatanfrage ans openITCOCKPIT zurückgegeben, wird diese signiert und das daraus resultierende Zertifikat (sowie das aktuelle CA Zertifikat) an den Agent gesendet.

Für den Zugriff auf den Agent-Webserver wird dann ein Client-Zertifikat einer konfigurierten CA für die Anfrage benötigt.

Anfragen an den Agent-Webserver, ohne ein von der selben CA ausgestelltes Zertifikat, werden abgewiesen.

Bekannte Probleme

C - Kompatibilität bei Linux Systemen

"glibc" bzw. "libc6" ist eine zentrale C-Bibliothek auf Linux Systemen. Diese wird mindestens in Version 2.17 benötigt, um den auf Python basierten Agent (ohne Binary) ausführen zu können.

Möchten Sie den Agent auf einem System mit einer "glibc" bzw. "libc6" Version < 2.17 verwenden, nehmen Sie bitte Kontakt mit unserem Support auf.

Fehler bei den Checks "Network device IO", "Processes" oder "Windows services"

Auf Grund eines Fehlers in der Version openITCOCKPIT 4 Beta müssen auf einigen Systemen folgende Servicetemplates aktualisiert werden.

OITC_AGENT_NET_IO

OITC_AGENT_PROCESSES

OITC_AGENT_WINDOWS_SERVICES

Zum Beheben des Fehlers können folgende SQL Statements ausgeführt werden.

```
use openitcockpit;
update servicetemplatecommandargumentvalues set value='' where value=' ' and servicetemplate_id in (select id
from servicetemplates where template_name in
('OITC_AGENT_NET_IO','OITC_AGENT_PROCESSES','OITC_AGENT_WINDOWS_SERVICES'));
update servicecommandargumentvalues set value='' where value=' ' and service_id in (select id from services
where service_type=16 and servicetemplate_id in (select id from servicetemplates where template_name in
('OITC_AGENT_NET_IO','OITC_AGENT_PROCESSES','OITC_AGENT_WINDOWS_SERVICES')));
```

Disk Usage checks auf Windows sind nicht verfügbar

Bis Version 1.0.3 des Agents sind Disk Usage checks auf Windows system nicht verfügbar, solange ein Optisches Laufwerk ohne Datenträger/image vorhanden ist. Beim Scan des Agents werden daraufhin keinerlei Disk Usage Checks angegeben.

Das Problem wird durch ein Fehler in der psutil Bibliothek, welche vom Agent genutzt wird, verursacht.

Mit Version 1.0.4 des Agents ist das problem behoben.