SOC-Connector Installations- und Bedienungsanleitung

DFN-CERT

DFN CERT®

Inhaltsverzeichnis

Ziel und Zweck dieses Dokuments

Dieses Dokument beschreibt die Schritte, um den SOC-Connector lokal in Betrieb zu nehmen und Logdaten zur Analyse in die DFN.Security Infrastruktur einleiten zu können.

1 Inbetriebnahme des SOC-Connectors

1.1 Voraussetzungen

1.1.1 Hardware

Die benötigte Hardware hängt stark von der Menge der transferierten Log-Meldungen ab, da u.a. auch eine erste Normalisierung der Daten vom SOC-Connector vorgenommen wird.

Für die Teilnahme am DFN-Security Basisdienst ist üblicherweise eine einfache Ausstattung ausreichend:

- CPU: 2 Kerne
- Hauptspeicher: 4 GB
- SSD: 20 GB

1.1.2 Betriebssystem

Die Installation findet auf einem Linux-System statt. Aktuell sind die Installation und der Betrieb auf den folgenden Betriebssystemen getestet:

Linux-Distribution	Version
Debian	11 (Bullseye)

Andere Linux-Distributionen werden mit groSSer Sicherheit auch funktionieren, sind jedoch (noch) nicht getestet.

Grundsätzlich kann die Installation auf einer VM erfolgen.

1.1.3 Folgende Software muss auf dem Linux-System vorhanden sein

- Übliche Linux-Kommandozeilen-Tools (z.B. `sed`)
- Eine aktuelle Bash-Shell
- Ein OCI-kompatibler Container-Manager, z.B. Podman oder Docker. Die Empfehlung ist Podman im "rootless"-Betrieb zu verwenden:
  - Installationsanleitung für Podman
    https://podman.io/getting-started/installation#installing-on-linux

– Installationsanleitung für Docker
https://docs.docker.com/engine/install/

Container-Manager	Getestete Versionen
Podman	3.4.2, 4.3.1
Docker	20.10.5

Andere Versionen (ab Podman 3.0 bzw. ab Docker 20.10) funktionieren wahrscheinlich auch, sind aber aktuell noch nicht explizit getestet.

1.1.4 Netzwerkverbindungen nach auSSen

Der SOC-Connector-Host **muss** Port 443 der folgenden Server ansprechen können:

Zielserver	Hostname	Zweck

Concentrator	`concentrator-ber-1.soc.dfn.de`
	`concentrator-ber-2.soc.dfn.de`
	`concentrator-ber-3.soc.dfn.de`
	`concentrator-erl-1.soc.dfn.de`
	`concentrator-erl-2.soc.dfn.de`
	`concentrator-erl-3.soc.dfn.de`
		Einlieferung von Logdaten für Analyse (Produktion)

Incubator	`incubator-stream.soc.dfn.de`	Funktionstest/Einlieferung (Stage)
Incubator	`incubator.soc.dfn.de`	Funktionstest/Kontrolle (Stage)

Dabei darf die Kommunikation nicht über einen HTTPS-Proxy laufen!

Der SOC-Connector-Host **sollte** Port 443 der folgenden Server ansprechen können:

Zielserver	Hostname	Zweck/Alternativen
Container-Image-Repository	`repo.dfn-cert.de`	Als Workaround kann das Image als Datei über einen anderen Host heruntergeladen und manuell lokal installiert werden

Der SOC-Connector-Host **sollte** selbst **nicht** von auSSen erreichbar sein.

1.1.5 Netzwerkverbindungen nach innen

Der SOC-Connector dient zur Weiterleitung von Log- bzw. Ereignisdaten aus dem internen Netz in die DFN.Security Infrastruktur. Zu diesem Zweck muss er selbst von allen Systemen erreichbar sein, die direkt Logdaten einleiten sollen.

Derzeit erwartet der SOC-Connector Daten im Syslog-Format und per Syslog-Protokoll. Der Standard-Port für Syslog-Kommunikation ist der privilegierte Port `514`. Der SOC-Connector verwendet per Default jedoch den unprivilegierten Port `1514`, damit es keine Kollision mit einem eventuell schon vorhandenen Syslog-Prozess gibt und damit der SOC-Connector auch ohne `root` bzw. `sudo`-Rechte betrieben werden kann. Der Syslog-Port kann während des Setups bei Bedarf geändert werden.

In jedem Fall **muss** der gewählte Syslog-Port intern von allen Hosts aus erreichbar sein, die direkt Logdaten einliefern sollen.

Falls der privilegierte Port `514` verwendet wird, **muss** sichergestellt sein, dass auf dem SOC-Connector-Host keine andere Syslog-Instanz läuft und diesen Port bereits belegt!

Der SOC-Connector muss von sich aus keine anderen internen Systeme erreichen können.

1.2 Installationsvorbereitung

Die Installation und Einrichtung des SOC-Connectors kann grundsätzlich von einem beliebigen User durchgeführt werden. Dieser User wird im Folgenden `$SOC_USER` genannt. Es sind dabei allerdings die folgenden Punkte zu beachten:

- Wenn **Docker** (und nicht **Podman**) als Container-Manager verwendet werden soll (s.o.), muss die Einrichtung und der Betrieb entweder von `root` durchgeführt werden, oder `$SOC_USER` muss `sudo`-Rechte besitzen, um **Docker** aufrufen zu können.
- Wenn der privilegierte Port `514` (Default: `1514`) als Eingangsport für Syslog-Meldungen verwendet werden soll (s.o.), muss die Einrichtung und der Betrieb entweder von `root` durchgeführt werden, oder `$SOC_USER` muss `sudo`-Rechte besitzen, um diesen Port öffnen zu können.

Für die weiteren Installationsschritte muss ein geeignetes, leeres Verzeichnis angelegt werden, welches `$SOC_USER` gehören sollte und in welchem `$SOC_USER` Dateien anlegen, verändern und ausführen darf. Dieses Installationsverzeichnis wird im Folgenden `$SOC_DIR` genannt.

Empfehlung:

- Podman als Container-Manager verwenden.
- Unprivilegierten Port `1514` als Syslog-Port verwenden.
- Benutzer `soc` für Installation und Betrieb des SOC-Connectors anlegen.
- Als Installationsverzeichnis `/home/soc/soc-connector/` verwenden.

1.3 Beschaffung der notwendigen Artefakte

1.3.1 Installationsskript

Eventuell haben Sie das Skript `soc-connector.sh` im Rahmen des Onboarding-Prozesses bereits per E-Mail oder auf anderem Wege erhalten. Falls nicht, laden Sie das Skript `soc-connector.sh` aus dem SOC-Connector Download-Bereich herunter und kopieren Sie es in das Installationsverzeichnis (`$SOC_DIR`). Stellen Sie sicher, dass das Skript ausführbar ist:

```
cd $SOC_DIR
cp ..../soc-connector.sh .
chmod 755 soc-connector.sh
```

1.3.2 Client-Zertifikat für die Verbindung in die DFN.Security Infrastruktur

Für die sichere Einlieferung von Logdaten in die DFN.Security Infrastruktur sind zwingend ein entsprechendes TLS-Client-Zertifikat und der dazugehörige private Schlüssel erforderlich. Das Zertifikat muss von der **DFN.Security SOC Access CA** ausgestellt sein.

Ein entsprechendes Zertifikat und den Schlüssel haben Sie im Rahmen des Onboarding-Prozesses erhalten.

Die beiden Dateien (`\*-cert.pem` und `\*-key.pem`) müssen ebenfalls in das Installationsverzeichnis (`$SOC_DIR`) kopiert werden. Sie müssen dort für `$SOC_USER` lesbar sein.

1.4 Durchführung der Installation

1.4.1 Empfohlene Variante mit Podman und Default-Port

Wechseln Sie in das Installationsverzeichnis (`$SOC_DIR`) und rufen Sie als User `$SOC_USER` das SOC-Connector-Skript mit dem Befehl `setup` auf:

```
./soc-connector.sh setup
```

Das Skript prüft einige Installations- und Betriebsvoraussetzungen und loggt die Ergebnisse auf der Konsole. Beim ersten Aufruf wird u.a. das benötigte Container-Image aus dem zentralen Repository heruntergeladen und installiert. Dieser Vorgang kann u.U. einige Minuten dauern.

Wenn das `setup`-Kommando ohne Fehler durchläuft und mit der folgenden Meldung beendet wird, ist der SOC-Connector vollständig eingerichtet und kann getestet werden:

```
OK   - SOC-Con\-nec\-tor setup completed!
```

Falls das `setup`-Kommando mit einer Fehlermeldung abbricht, so kann der ausgegebene Fehler-Code `Ennn` verwendet werden, um mehr zu erfahren über die Ursache und zu möglichen Schritten zur Behebung des Problems. Suchen Sie dazu auf dieser Seite nach dem Fehler-Code.

1.4.2 Alternative: Docker als Container-Manager, Ausführung als `root`

Um Docker (und nicht Podman) als Container-Manager zu verwenden, kann beim Setup die Option `--docker` angegeben werden:

```
./soc-connector.sh setup --docker
```

In diesem Fall muss die Installation und auch der Betrieb des SOC-Connectors grundsätzlich als `root` durchgeführt werden!

1.4.3 Alternative: Docker als Container-Manager, Ausführung als normaler User

Um Docker (und nicht Podman) als Container-Manager zu verwenden, aber ohne den Benutzer `root` zu verwenden, kann beim Setup zusätzlich die Option `--sudo` angegeben werden:

```
./soc-connector.sh setup --docker --sudo
```

Voraussetzung hierbei ist natürlich, dass der verwendete User (`$SOC_USER`) die entsprechenden `sudo`-Rechte besitzt. Das Skript wird dann ggf. nach dem Passwort des Benutzers fragen, um die `sudo`-Funktion zu aktivieren.

1.4.4 Alternative: Docker als Container-Manager, `514` als Syslog-Port, Ausführung als normaler User

Um zusätzlich auch den privilegierten Port `514` als Syslog-Port zu verwenden, kann die Option `--syslog-port 514` hinzugefügt werden:

```
./soc-connector.sh setup --docker --sudo --syslog-port 514
```

1.4.5 Alternative: Podman als Container-Manager, `514` als Syslog-Port, Ausführung als normaler User

Das ganze geht natürlich auch mit Podman. Die Option `--sudo` ist wegen des privilegierten Ports hierfür dann auch erforderlich:

```
./soc-connector.sh setup --sudo --syslog-port 514
```

1.5 Status des SOC-Connectors

Mit Hilfe des `status`-Kommandos kann jederzeit geprüft werden, in welchem Zustand sich der SOC-Connector befindet:

```
./soc-connector.sh status
```

Nach erfolgreichem Setup sollte dieser Status ausgegeben werden:

```
INFO  - Setup: complete  Mode: stage  Connector: stopped
```

Dabei sind die folgenden Statuswerte und -kombinationen möglich:

Setup	Mode	Connector	Beschreibung	Typische Aktionen
`incomplete`	-	-	Setup wurde noch nicht (erfolgreich) durchgeführt.	`setup`
`complete`	`stage`	`stopped`	Setup wurde erfolgreich durchgeführt, SOC-Connector im Stage-Modus, SOC-Connector läuft nicht	`start`, `deploy`
`complete`	`stage`	`running`	Setup wurde erfolgreich durchgeführt, SOC-Connector im Stage-Modus, SOC-Connector läuft	`test`, `stop`
`complete`	`production`	`stopped`	Setup wurde erfolgreich durchgeführt, SOC-Connector im Produktionsmodus, SOC-Connector läuft nicht	`start`, `reset`
`complete`	`production`	`running`	Setup wurde erfolgreich durchgeführt, SOC-Connector im Produktionsmodus, SOC-Connector läuft	`stop`

Die Tabelle zeigt auch einige der wichtigsten Kommandos, die im jeweiligen Status möglich sind. Grundsätzlich sind - je nach Status - immer nur bestimmte Aktionen möglich.

1.6 Einfacher Test des SOC-Connectors im Stage-Modus

Hinweis: Direkt nach erfolgreichen Setup befindet sich der SOC-Connector zunächst grundsätzlich im **Stage**-Modus. D.h. eventuell eingehende Log-Meldungen werden **nicht** an das produktive DFN.Security SOC-System, sondern an den sogenannten **SOC-Incubator** geschickt! Der SOC-Incubator nimmt keinerlei Analyse der eingehenden Daten vor und die Daten werden auch nur für eine sehr kurze Zeit zwischengespeichert und dann automatisch gelöscht.

Der einzige Zweck des SOC-Incubators ist es, eine zum produktiven SOC-System kompatible Schnittstelle für Tests zur Verfügung zu stellen und die eingehenden Daten an den Aufrufer (und nur an den!) zurückzuliefern, damit geprüft werden kann, ob die Daten vollständig ankommen und korrekt verarbeitet werden können.

Ein einfacher Testlauf des SOC-Connectors in Stage-Modus besteht aus diesen Kommandos, die anschlieSSend im Detail erläutert werden:

- `setup` - Der erfolgreiche Setup ist Grundvoraussetzung (s.o.).
- `status` - Prüfen, dass sich der SOC-Connector im Stage-Modus befindet (s.o.).
- `start` - Den SOC-Connector starten.

- `test` - Einige automatische Test-Meldungen an den SOC-Incubator-Server schicken und auf Bestätigung warten.
- `stop` - Den SOC-Connector wieder stoppen.

1.6.1 Start des SOC-Connectors

Bevor der Testlauf gestartet werden kann, muss der SOC-Connector gestartet werden:

```
./soc-connector.sh start
```

Wie oben ausgeführt, werden im Stage-Modus keine Daten an das produktive System übertragen und es werden insbesondere keine Daten ausgewertet oder dauerhaft gespeichert.

1.6.2 Senden einiger automatischer Test-Meldungen und Warten auf die Empfangsbestätigung des Incubators:

Dann kann der Test-Lauf gestartet werden:

```
./soc-connector.sh test
```

Die Ausgabe des SOC-Connectors sollte etwa so aussehen:

```
INFO  - submitting log message #1/3      : SOC-Con\-nec\-tor/Incubator test
    message 1674486316
INFO  - SOC System confirmed log message : SOC-Con\-nec\-tor/Incubator test
    message 1674486316
INFO  - submitting log message #2/3      : SOC-Con\-nec\-tor/Incubator test
    message 1674486328
INFO  - SOC System confirmed log message : SOC-Con\-nec\-tor/Incubator test
    message 1674486328
INFO  - submitting log message #3/3      : SOC-Con\-nec\-tor/Incubator test
    message 1674486339
INFO  - SOC System confirmed log message : SOC-Con\-nec\-tor/Incubator test
    message 1674486339
```

Diese Zeilen zeigen, dass drei Test-Meldungen übertragen wurden und korrekt angekommen sind. Der Empfang wurde vom SOC-Incubator explizit bestätigt.

Hinweis: Insbesondere beim ersten Aufruf kann der Test-Vorgang ein paar Minuten dauern oder sogar mal in ein Timeout laufen. In diesem Fall bitte das `test`-Kommando erneut ausführen.

Wenn der Testlauf wiederholt fehlschlägt, obwohl der Setup erfolgreich war, bitte an das SOC-Team bei DFN-CERT wenden.

1.6.3 Stop des SOC-Connectors

Nach dem Test sollte der SOC-Connector wieder gestoppt werden:

```
./soc-connector.sh stop
```

1.7 Umfassender Funktionstest des SOC-Connectors

Wenn der einfache Funktionstest erfolgreich war, sollte ein umfassender Text mit echten Log-Meldungen durchgeführt werden.

Auch der umfassende Test wird im Stage-Modus durchgeführt. D.h. es werden keine Daten an das produktive System geschickt. Die übertragenen Daten werden nicht analysiert und nach kurzer Zeit automatisch gelöscht.

Ein umfassender Testlauf des SOC-Connectors in Stage-Modus besteht aus diesen Kommandos, die anschlieSSend im Detail erläutert werden:

- `setup` - Der erfolgreiche Setup ist Grundvoraussetzung (s.o.).
- `status` - Prüfen, dass sich der SOC-Connector im Stage-Modus befindet (s.o.).
- `start` - Den SOC-Connector starten, um eingehende Meldungen an den SOC-Incubator zu schicken.
- `sample` - Der SOC-Connector fordert vom SOC-Incubator eine Bestätigung der eingehenden Daten an.
- `stop` - Den SOC-Connector wieder stoppen.

1.7.1 Start des SOC-Connectors

Bevor der Testlauf gestartet werden kann, muss der SOC-Connector gestartet werden:

```
./soc-connector.sh start
```

Wie oben ausgeführt, werden im Stage-Modus keine Daten an das produktive System übertragen und es werden insbesondere keine Daten analysiert oder dauerhaft gespeichert.

1.7.2 Aktivierung der Feedback-Funktion

Um zu sehen, ob der SOC-Connector erfolgreich Daten an den SOC-Incubator senden kann, und ob die Daten dort im erwarteten Format ankommen, wird das Kommando `sample` gestartet. Der Testlauf kann jederzeit mit `Ctrl-C` gestoppt werden:

```
./soc-connector.sh sample
```

Auf der Konsole erscheint diese Ausgabe, wobei etwa alle 10 Sekunden ein weiterer Punkt hinzukommt:

```
        waiting for confirmation ...
```

Aus einem anderen Fenster (sogar von einem anderen Host!) können nun gezielt Log-Meldungen an den SOC-Connector geschickt werden:

```
logger --port <PORT> --server <IP/HOSTNAME> --tcp This is my very important log
    message!
```

- Das Feld `<PORT>` muss dem beim Setup gewählten SYSLOG_PORT entsprechen (Default: 1514)

- Das Feld `<IP/HOSTNAME>` muss der IP-Adresse bzw. dem Namen des SOC-Connector-Hosts entsprechen. Ggf. muss hier `localhost` eingesetzt werden!
- Als Log-Meldung am Ende der Zeile kann ein beliebiger Text eingegeben werden.

Hinweis: Es ist unbedingt ratsam, nicht nur ein paar "manuelle" Log-Meldungen mit dem `logger`-Befehl zu erzeugen, sondern tatsächliche Log-Meldungen aus den zu überwachenden System zu verwenden. Idealerweise nicht zu viele - ein paar Beispiele reichen in der Regel aus, um die Übertragung und v.a. das Format zu prüfen.

Die vom SOC-Incubator empfangenen Meldungen sollte nach wenigen Sekunden in dem Terminalfenster zu sehen sein, in dem `./soc-connector.sh sample` gestartet worden ist. Die Meldung erscheint dabei im JSON-Format. Die einzelnen Felder der Syslog-Meldung sollten korrekt erkannt und besetzt sein:

```
INFO  - SOC System confirmed log message : {
  "appInst": "myapp",
  "appName": "myapp",
  "body": "This is my very important log message!",
  "event_raw": "<13>1 2023-01-23T16:34:27.097246+01:00 myhostname myapp - - [
     timeQuality tzKnown=\"1\" isSynced=\"0\"] This is my very important log
     message!",
  "facility": "1",
  "hostName": "myhostname",
  "priority": "13",
  "procId": "-",
  "severity": "5",
  "structuredData": "[timeQuality tzKnown=\"1\" isSynced=\"0\"]",
  "syslogTag": "myapp",
  "timestamp": "2023-01-23T16:34:27.097246+01:00"
}
```

Hinweis: Besonders bei den folgenden Feldern sollte auf den korrekten Inhalt geachtet werden:

- `event_raw`
  - Das ist die Meldung, so wie sie beim SOC-Incubator angekommen ist. Alle anderen Felder sind hieraus extrahiert worden.
  - Idealerweise wird die Meldung gemäßRFC 5424 übertragen: Dann ist gewährleistet, dass alle Felder korrekt erkannt werden.
- `timestamp`
  - Stimmt die Uhrzeit in Kombination mit der Zeitzone? Dieser Wert muss unbedingt korrekt sein, da ansonsten die Auswertung der Log-Meldungen eventuell nicht optimal funktioniert!
  - Ggf. prüfen, welcher Zeitstempel in `event_raw` übergeben wurde.
  - Für Zeitstempel ohne Zeitzone wird i.d.R. als Zeitzone UTC angenommen.
  - Falls der Zeitstempel gar nicht erkannt werden kann, wird die aktuelle Zeit verwendet.
  - Im Format RFC 5424 können Zeitstempel in High-Precision-Format übergeben werden. Das ist zwar keine zwingende Voraussetzung für die Analyse der Daten, kann aber sehr hilfreich sein, um Vorfälle im Detail zu rekonstruieren.

- `body`
  - Hier sollte der "Freitext"-Teil der Meldung stehen.
  - Falls die strukturierten Felder am Anfang der Zeile nicht korrekt erkannt werden konnten, können hier Fragmente dieser Felder übrig bleiben. Die korrekte und vollständige Erkennung des Meldungstexts ist essenziell für die weitere Verarbeitung der Daten!
- `appInst`, `hostName`, `procId`, `facility`, `severity`
  - Hier sollte der Inhalt der entsprechenden Syslog-Felder erkannt worden sein (sofern vorhanden).

1.7.3 Stop des SOC-Connectors

Nach dem Test sollte der SOC-Connector wieder gestoppt werden:

```
./soc-connector.sh stop
```

2 Betrieb des SOC-Connectors

Wenn die Testläufe erfolgreich waren und die Log-Meldungen im gewünschten Format ankommen und korrekt ausgewertet werden können, kann der Produktiv-Modus aktiviert werden.

2.1 Wechsel in den Produktiv-Modus

Die Aktivierung des SOC-Connectors im Produktiv-Modus besteht aus diesen Kommandos, die anschlieSSend im Detail erläutert werden. Ein erfolgreicher Setup sowie Tests (s.o.) sind natürlich Voraussetzung!

- `status` - Prüfen, dass sich der SOC-Connector im Stage-Modus befindet (s.o.).
- `deploy` - Den SOC-Connector umschalten in den Produktiv-Modus.
- `status` - Prüfen, dass sich der SOC-Connector nun im Produktiv-Modus befindet.
- `start` - Den SOC-Connector starten, um eingehende Meldungen an den SOC-Concentrator zu schicken.

2.1.1 Umschalten vom Stage-Modus (SOC-Incubator) zum Produktiv-Modus (SOC-Concentrator):

Nach erfolgreich durchgeführten Tests kann der SOC-Connector für den produktiven Betrieb aktiviert werden:

```
./soc-connector.sh deploy
```

Die Status-Ausgaben (Kommando `status`) sollte jetzt so aussehen:

```
INFO  - Setup: complete  Mode: production  Connector: stopped
```

Hinweis: Das `setup`-Kommando kann jederzeit erneut ausgeführt werden, z.B. um einzelne Konfigurationseinstellungen zu ändern. Danach befindet sich der SOC-Connector aber **grundsätzlich** im Stage-Modus! D.h. das `deploy`-Kommando muss dann - nach den erforderlichen Tests - ebenfalls erneut ausgeführt werden!

2.1.2 Start des SOC-Connectors für den Produktionsbetrieb

Der Start des SOC-Connectors erfolgt denn mit dem schon bekannten `start`-Kommando:

```
./soc-connector.sh start
```

Im Produktionsbetrieb werden die eingehenden Logs an die zentrale DFN.Security SOC-Infrastruktur geschickt und dort ausgewertet. Die Verarbeitung und Speicherung erfolgt gemäSS Datenschutz-Bestimmungen und vertraglichen Vereinbarungen.

3 Wartung des SOC-Connectors

3.1 Update des SOC-Connectors

Falls neue Versionen des SOC-Connector-Skripts oder -Images zur Verfügung stehen, werden die Teilnehmer in geeigneter Weise informiert.

Eine neue Version des `soc-connector.sh`-Skripts muss in das Verzeichnis `$SOC_DIR` kopiert werden und ersetzt dann die aktuelle Version. Die Auslieferungsnotiz enthält zusätzliche Informationen, falls weitere Schritte zur Inbetriebnahme erforderlich sind.

Eine neue Version des Container-Images kann mit dem `update`-Kommando heruntergeladen und aktiviert werden:

```
./soc-connector.sh update
```

Falls ein neues Image geladen wurde, muss nach dem Update das `setup`-Kommando ausgeführt werden. AnschlieSSend sollten die Tests (s.o.) durchgeführt werden. Zum Schluss kann dann wieder in den Produktiv-Modus gewechselt werden.

3.2 Zurücksetzen der Konfiguration

Um mit der Konfiguration “bei null” zu beginnen, kann der Reset-Befehl verwendet werden:

```
./soc-connector.sh reset
```

AnschlieSSend können optional die Unterverzeichnisse `setup`, `conf` und `logs` gelöscht werden. Sie werden beim erneuten Ausführen des `setup`-Kommandos automatisch angelegt und bestückt.

3.3 Löschen des Containers

Normalerweise sollte der Container bei jedem Stop automatisch gelöscht und beim Start wieder aus dem Image neu erstellt werden. In seltenen Fällen klappt das Löschen nicht. Dann kann der Container manuell gelöscht werden:

```
./soc-connector.sh delete
```

3.4 Entfernen des SOC-Connectors

Um den SOC-Connector vollständig zu entfernen, können diese Schritte durchgeführt werden:

```
cd $SOC_DIR

# Stoppen (falls gestartet)
./soc-connector stop

# Container löschen (falls nicht automatisch gelöscht)
./soc-connector delete

# Container-Image löschen
./soc-connector purge

# Unterverzeichnisse löschen
rm -rf setup conf logs
```

Hinweis: Das Skript `soc-connector.sh` sowie das SOC-Access-Zertifikat und der zugehörige Schlüssel werden durch diese Befehle nicht gelöscht. Sie können natürlich auch entfernt werden, wenn sie nicht mehr benötigt werden.