Intex PureSpa in IP-Symcon steuern

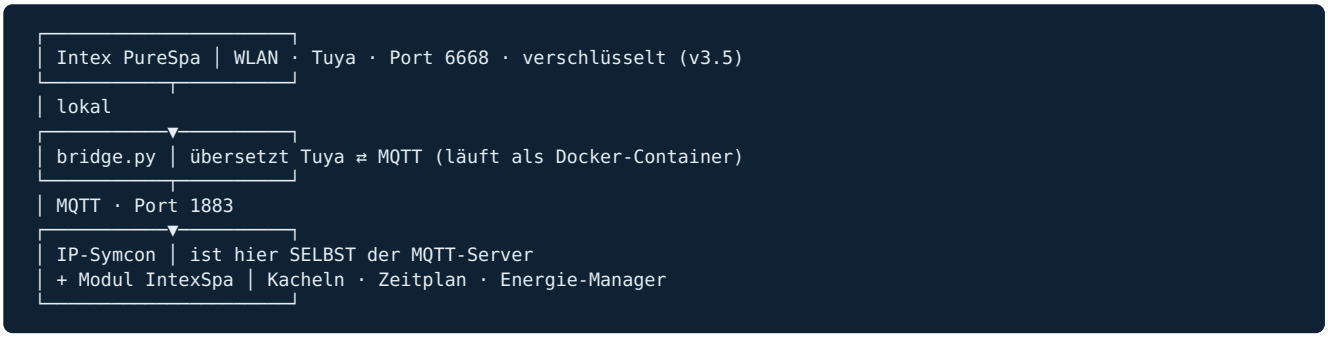
Die komplette Schritt-für-Schritt-Anleitung

Diese Anleitung bringt deinen **Intex PureSpa** (ab Baujahr 2024, Typcode endet auf **„TY"**) vollständig in **IP-Symcon 9.0** – **lokal**, ohne Cloud-Zwang.

> **Hinweis:** Diese Anleitungen wurden **mit Hilfe von KI (Claude)** erstellt und sorgfältig geprüft. Arbeite sicherheitsrelevante Schritte (Strom, WLAN, Geräte) trotzdem mit eigenem Verstand ab.
>
> **Ganz wichtig vorab:** Die 2024er-Modelle sind **Tuya**-Geräte. Du koppelst den Spa deshalb mit der **„Smart Life"-App** (von Tuya), **nicht** mit der „INTEX Link"-App. Über INTEX Link kommst du **nicht** an den Schlüssel, den wir für die lokale Steuerung brauchen.

Wie das Ganze funktioniert (in einfachen Worten)

Dein Spa „spricht" das **Tuya-Protokoll** – lokal, aber **verschlüsselt**. IP-Symcon versteht das nicht direkt. Deshalb gibt es eine kleine **Brücke** (`bridge.py`), die zwischen Spa und IP-Symcon übersetzt.



> **Merksatz:** Spa ⇄ Brücke ⇄ MQTT ⇄ IP-Symcon.

Was du brauchst (Checkliste)

	Voraussetzung
[ ]	Intex PureSpa **ab 2024** (Typcode endet auf **TY**), im WLAN
[ ]	Smartphone mit der App **Smart Life** (von Tuya)
[ ]	Kostenloses **Tuya-Entwicklerkonto** (nur zum einmaligen Auslesen)
[ ]	Gerät mit **Python** (PC/Mac/Raspberry Pi) für das Auslese-Tool
[ ]	**NAS/Server mit Docker** (z. B. Asustor/Synology + Portainer)
[ ]	**IP-Symcon 9.0**
[ ]	**GitHub-Konto** (zum Einbinden des Moduls)

Die 12 Schritte im Überblick

Schritt	Thema
1	Spa mit der **Smart Life**-App koppeln
2	Tuya-Entwicklerkonto & Projekt anlegen
3	**Smart-Life-App per QR-Scan mit dem Projekt verknüpfen**
4	Local Key & Device-ID mit tinytuya auslesen

5	Brücke `bridge.py` konfigurieren
6	Brücke als Docker-Container starten
7	IP-Symcon als MQTT-Server einrichten
8	Modul auf GitHub hochladen
9	Modul installieren & Instanz anlegen
10	Heizen bei PV-Überschuss (Energie-Manager)
11	Push-Nachricht & manueller Vorrang
12	Fehlerbehebung & Wartung

> Alle benötigten Dateien liegen im Ordner `dateien/`. Arbeite die Schritte **der Reihe nach** ab – jeder baut auf dem vorherigen auf.

Kleines Glossar

Begriff	Bedeutung
Tuya	Technik-Plattform hinter vielen Smart-Home-Geräten (auch Intex 2024+)
Smart Life	die Tuya-App, mit der du den Spa koppelst
Local Key	geheimer Schlüssel deines Geräts für die **lokale** Entschlüsselung
Device-ID	eindeutige Kennung deines Spa bei Tuya
MQTT	einfache „Nachrichten-Post" zwischen Programmen
Brücke	das Programm `bridge.py`, übersetzt Tuya ⇄ MQTT
DP / Datenpunkt	einzelne Funktion des Geräts (z. B. DP 108 = Heizung)
Instanz	ein konkretes Gerät/Objekt in IP-Symcon

Schnellreferenz – deine Eckdaten

Was	Wert
Spa-Modell	Intex PureSpa 28457–28464 „SALT & JET", Bj. 2024
Spa-IP	`192.168.X.X`
Device-ID	`DEINE_DEVICE_ID`
Local Key	`DEIN_LOCAL_KEY`
Tuya-Protokoll	`3.5`
Tuya-Port (lokal)	`6668`
MQTT-Broker	IP-Symcon (Modul „MQTT Server"), Port `1883`
Docker-Netz	`intexspa_default`
Brücken-Ordner (NAS)	`/volume1/Docker/intexspa/bridge`
GitHub-Repo	`https://github.com/DEIN_GITHUB_NAME/IPSymcon-IntexSpaMQTT`

Weiter mit Schritt 1 (`01_Spa_mit_SmartLife_koppeln.md`)

1 · Spa mit der Smart-Life-App koppeln

> **Ziel:** Dein Spa ist in der **Smart-Life-App** eingebunden und hat eine **feste IP-Adresse**. Nur über Smart Life (Tuya) kommst du später an den Local Key.
>
> **Du brauchst:** Smartphone · 2,4-GHz-WLAN + Passwort · Zugang zum Router
>
> **Wichtig:** Nimm **Smart Life** (oder „Tuya Smart"), **nicht** „INTEX Link". Ein Tuya-Gerät ist immer nur mit **einem** Konto aktiv gekoppelt. War dein Spa schon in der INTEX-Link-App, **entferne ihn dort zuerst** (das ist umkehrbar).

1.1 · Smart-Life-App installieren & Konto anlegen

1. Lade **„Smart Life – Smart Living"** (Anbieter: Tuya / Volcano Technology) aus dem App-Store.
2. Registriere ein Konto mit **E-Mail + Passwort**. - **Kein Gastkonto** verwenden – das wird sonst später kommentarlos gelöscht. - Diese Zugangsdaten brauchst du in **Schritt 3** wieder.

1.2 · Handy ins richtige WLAN

- Verbinde dein Handy mit dem **2,4-GHz-WLAN**, in das auch der Spa soll.
- Tuya koppelt nur über **2,4 GHz**, nicht über 5 GHz.

1.3 · Spa in den Kopplungsmodus bringen

- Schalte den Spa ein.
- Halte am Bedienteil die **WLAN-/Timer-Taste** gedrückt, bis das **WLAN-Symbol schnell blinkt**. Schnelles Blinken = bereit zum Koppeln. *(Je nach Modell heißt die Taste etwas anders – such das WLAN-Symbol.)*

1.4 · Gerät in Smart Life hinzufügen

1. Öffne **Smart Life** → oben rechts auf **„+" → Gerät hinzufügen**.
2. Die App sucht automatisch nach Geräten in der Nähe → dein Spa sollte erscheinen → **Hinzufügen** antippen. - Erscheint nichts: **„Manuell hinzufügen"** → Kategorie **„Kleingerät"** bzw. **„Sonstiges"** wählen.
3. **WLAN-Namen und -Passwort** eingeben (2,4 GHz).
4. Warte, bis die App **„Erfolgreich hinzugefügt"** meldet.

> Klappt die Kopplung nicht: Spa-Bedienteil vom Strom trennen, 10 s warten, neu starten, Kopplungsmodus erneut auslösen (schnelles Blinken) und 1.4 wiederholen.

1.5 · Feste IP-Adresse vergeben

1. Öffne deinen Router (Fritzbox: **Heimnetz → Netzwerk**) und suche den Spa.
2. Notiere die **IP-Adresse** → deine: `192.168.X.X`
3. Aktiviere **„immer diese IPv4-Adresse zuweisen"** (DHCP-Reservierung), damit sie sich nie ändert.

> **Hat's geklappt?** - Der Spa lässt sich in **Smart Life** schalten. - Du kennst seine **feste IP**.
>
> **Wenn's klemmt** - **Kein Gerät gefunden:** Handy wirklich im **2,4-GHz**-Netz? Spa im schnellen Blinkmodus? - **„Gerät bereits gebunden":** Spa noch in INTEX Link (oder anderem Konto) → dort entfernen, dann erneut koppeln.

2 · Tuya-Entwicklerkonto & Projekt anlegen

> **Ziel:** Ein **kostenloses** Tuya-Entwicklerkonto mit einem „Cloud-Projekt" anlegen. Darüber kommst du später (Schritt 3 + 4) an den Local Key. **Nur einmal nötig.**
>
> **Du brauchst:** PC mit Browser

2.1 · Konto anlegen

1. Gehe auf **https://iot.tuya.com**.
2. Oben rechts **„Sign Up"** → mit E-Mail registrieren (kann ein anderes Konto sein als das Smart-Life-Konto). Bestätige die E-Mail und melde dich an.

2.2 · Cloud-Projekt erstellen

1. Im Menü links: **Cloud → Development**.
2. Button **„Create Cloud Project"**.
3. Ausfüllen: - **Project Name:** frei wählbar, z. B. „IntexSpa" - **Industry / Development Method:** Standard lassen - **Data Center: „Central Europe Data Center"** wählen *(unbedingt das europäische – sonst erscheinen deine Geräte später nicht)*
4. **Create** → es öffnet sich ein Fenster **„Authorize API Services"** → einfach mit **OK / Authorize** bestätigen.

2.3 · IoT Core & Authorization sicherstellen

1. Projekt öffnen → oben Reiter **„Service API"**.
2. Es müssen **„IoT Core"** und **„Authorization"** gelistet sein. - Fehlt etwas: **„Go to Authorize"** / **„Add Authorization"** und beide hinzufügen.

2.4 · Zugangsdaten notieren

Projekt → Reiter **„Overview"**. Dort findest du zwei Werte:

Feld	wofür
Access ID / Client ID	brauchst du in **Schritt 4** (tinytuya)
Access Secret / Client Secret	brauchst du in **Schritt 4** (tinytuya)

Beide notieren (Secret über das Auge-Symbol einblenden).

> **Hat's geklappt?** Dein Cloud-Projekt existiert, Data Center = **Central Europe**, **Access ID** und **Access Secret** sind notiert.
>
> **Wenn's klemmt** – Später kein Auslesen möglich? Das „IoT Core"-Trial ist abgelaufen → **Cloud → Cloud Services → IoT Core** kostenlos verlängern.

3 · Smart-Life-App per QR-Scan verknüpfen

> **Ziel:** Du verbindest dein **Smart-Life-Konto** (mit dem gekoppelten Spa) mit deinem **Tuya-Projekt**. Erst dadurch „sieht" das Projekt deinen Spa – und nur dann liefert tinytuya in Schritt 4 den Local Key.
>
> **Du brauchst:** PC mit dem geöffneten Tuya-Projekt (Schritt 2) · dein Handy mit der **Smart-Life-App** (Schritt 1)
>
> So funktioniert es: Im Tuya-Projekt erzeugst du einen **QR-Code**. Den scannst du **in der Smart-Life-App**. Dadurch werden alle Geräte deines Smart-Life-Kontos mit dem Projekt verknüpft.

3.1 · Im Tuya-Portal die Verknüpfungs-Seite öffnen

Genau hier findest du alles auf **https://iot.tuya.com**:

1. Menü links: **Cloud → Development**.
2. In der Projektliste auf **deinen Projektnamen** klicken (aus Schritt 2).
3. Oben im Projekt auf den Reiter **„Devices"** (Geräte) klicken.
4. Direkt darunter erscheinen mehrere **Unter-Reiter**. Wähle **„Link Tuya App Account"** (auf Deutsch teils „Tuya-App-Konto verknüpfen").
5. Klicke auf den Button **„Add App Account"**. → Es öffnet sich ein Fenster mit einem **QR-Code**.

> **Merke den Pfad:** *Cloud → Development → [dein Projekt] → Devices → Link Tuya App Account → Add App Account → QR-Code.*

3.2 · QR-Code mit der Smart-Life-App scannen

Wichtig: **nicht** mit der normalen Handy-Kamera scannen, sondern **in der Smart-Life-App** – nur so wird die Verknüpfung hergestellt.

1. Öffne am Handy die **Smart-Life-App**.
2. Tippe unten rechts auf **„Profil"** (bzw. „Ich" / „Me").
3. Tippe oben rechts auf das **Scan-Symbol** (kleines Quadrat/Strichcode-Symbol neben dem Zahnrad).
4. **Scanne den QR-Code** vom PC-Bildschirm ab.
5. Bestätige in der App mit **„Bestätigen" / „Confirm"**.

3.3 · Verknüpfung im Portal prüfen

1. Zurück im Tuya-Portal erscheint dein Konto nun unter **„Link Tuya App Account"** in der Liste.
2. Wechsle auf den Unter-Reiter **„All Devices"** (alle Geräte). - Oben ggf. das richtige **Data Center** wählen (**Central Europe**), sonst bleibt die Liste leer.
3. Dein **Spa** muss jetzt in der Geräteliste auftauchen (mit Name, Online-Status und einer **Device ID**).

> Tipp: Klick auf den Spa → dort siehst du u. a. die **Device ID**. Die brauchst du gleich in Schritt 4. Den **Local Key** zeigt das Portal hier zwar nicht offen an – den holt tinytuya automatisch.

> **Hat's geklappt?** Dein Spa erscheint im Tuya-Projekt unter **Devices → All Devices**.
>
> **Wenn's klemmt** - **Liste leer / Spa fehlt:** richtiges **Data Center** (Central Europe) gewählt? Gleiches Smart-Life-Konto gescannt, in dem der Spa gekoppelt ist (Schritt 1)? - **QR lässt sich nicht scannen:** wirklich über **Profil → Scan-Symbol** in der Smart-Life-App, nicht mit der Kamera-App.

4 · Local Key & Device-ID auslesen (tinytuya)

> **Ziel:** Mit dem Tool **tinytuya** den geheimen **Local Key**, die **Device-ID** und die **Protokoll-Version** holen. Diese Werte sind das Herzstück der lokalen Steuerung.
>
> **Du brauchst:** Computer mit **Python 3** (am besten im selben WLAN) · **Access ID** + **Access Secret** aus Schritt 2 · verknüpftes Konto aus Schritt 3

4.1 · tinytuya installieren

Terminal / Eingabeaufforderung öffnen und eingeben:

```
pip install tinytuya
```

4.2 · Den Assistenten starten

```
python -m tinytuya wizard
```

Der Assistent fragt der Reihe nach:

Frage	Deine Eingabe
API Key	deine **Access ID** (Schritt 2.4)
API Secret	dein **Access Secret**
Device ID	eine Geräte-ID aus dem Projekt (Schritt 3.3: Spa anklicken → Device ID)
Region	**eu** (Europa)

Weitere Fragen mit **Y** (ja) bestätigen.

4.3 · Ergebnis abholen

Der Assistent legt die Datei `devices.json` an. Dort deinen Spa heraussuchen und notieren:

Feld in `devices.json`	Bedeutung	Dein Wert
`id`	Device-ID	`DEINE_DEVICE_ID`
`key`	**Local Key**	`DEIN_LOCAL_KEY`
`version`	Tuya-Protokoll	`3.5`
`product_id`	Produkt-ID	`DEINE_PRODUCT_ID`

Die lokale **IP** kennst du aus Schritt 1 (`192.168.X.X`). Optional bestätigen:

```
python -m tinytuya scan
```

> (Dafür müssen im Netzwerk **UDP 6666/6667/7000** und **TCP 6668** frei sein.)

4.4 · Datenpunkte (DPs) deines Spa

Bereits ermittelt und in Brücke/Modul hinterlegt – nur zur Info:

DP	Funktion		DP	Funktion
104	Power (Ein/Aus)		**103**	Hygienisierung
108	Heizung		**109**	Soll-Temperatur (°F)
106	Filter (manuelle Taste)		**110**	Ist-Temperatur (°F)

107	Luftsprudler		**117**	Heizstatus (lesen)
105	Jets			

Hat's geklappt? Du hast **Device-ID**, **Local Key** und **Version** notiert.

Wenn's klemmt - **„No devices found":** Region/Data Center falsch (Schritt 2.2) oder App nicht verknüpft (Schritt 3). - **Key geht später nicht:** Nach **Neukopplung/Reset** ändert sich der Key → Schritt 4 wiederholen.

Behandle den Local Key wie ein Passwort. In der Community-Version dieser Anleitung ist er bewusst **nicht** enthalten.

5 · Die Brücke `bridge.py` konfigurieren

> **Ziel:** Deine Werte (IP, Device-ID, Local Key) in die Brücke eintragen. Die Brücke verbindet den Spa mit IP-Symcon.
>
> **Du brauchst:** Datei `dateien/bridge.py` · Werte aus Schritt 4 · den **Container-Namen deines IP-Symcon** (für MQTT_HOST – siehe Schritt 7)

5.1 · Datei öffnen

Öffne `dateien/bridge.py` in einem Text-Editor.

5.2 · Werte eintragen

Ganz oben im Abschnitt **„Konfiguration"**:

```
SPA_IP = "192.168.X.X"
SPA_ID = "DEINE_DEVICE_ID"
SPA_KEY = "DEIN_LOCAL_KEY"
SPA_VERSION = 3.5

MQTT_HOST = "<Name-deines-IP-Symcon-Containers>" # Name deines IP-Symcon-Containers (gleiches Docker-Netz)
MQTT_PORT = 1883
BASE = "intexspa" # MQTT-Basis-Thema – muss mit dem Modul übereinstimmen
```

Danach **speichern**.

5.3 · Was bedeuten die Felder?

Feld	Bedeutung
`SPA_IP` / `SPA_ID` / `SPA_KEY` / `SPA_VERSION`	deine Werte aus Schritt 4
`MQTT_HOST`	Adresse des MQTT-Servers. Da **IP-Symcon selbst** der Server ist und im selben Docker-Netz läuft → hier den **Container-Namen von IP-Symcon** eintragen (nicht „mosquitto"). Alternativ die IP des Symcon-Hosts.
`BASE`	Grund-Thema aller Nachrichten. Auf `intexspa` lassen – das Modul erwartet genau das.

5.4 · Gut zu wissen (macht die Brücke automatisch)

- Liest den Spa regelmäßig und meldet den Status an IP-Symcon.
- Befehle laufen als `intexspa/set/<funktion>/<wert>` – der **Wert steht im Thema**, nicht im Nachrichtentext.
- Federt eine Tuya-Eigenheit ab: Direkt nach dem Schalten hinkt die volle Abfrage kurz nach – die Brücke hält den geschalteten Wert, damit die Kachel nicht zurückspringt.

> **Hat's geklappt?** `bridge.py` enthält deine vier Spa-Werte und einen sinnvollen `MQTT_HOST`. *(Starten kommt im nächsten Schritt.)*
>
> **Wenn's klemmt** – Unsicher beim `MQTT_HOST`? Trag ihn erst in **Schritt 7** final ein (dort legst du den MQTT-Server an und kennst den Container-Namen).

6 · Die Brücke als Docker-Container starten

> **Ziel:** Die Brücke läuft dauerhaft und startet nach einem NAS-Neustart von selbst – **ohne** bei jedem Start etwas neu zu installieren.
>
> **Du brauchst:** NAS/Server mit **Docker + Portainer** · Dateien `bridge.py` und `intexspa-stack.yml` · einmalig **Konsole/SSH** (6.2)

6.1 · Dateien ablegen

1. Ordner `/volume1/Docker/intexspa/bridge/` anlegen und `bridge.py` hineinkopieren.
2. Leeren Ordner `/volume1/Docker/intexspa/deps/` anlegen (dort liegen später die Pakete dauerhaft).

6.2 · Pakete EINMALIG installieren

Damit der Container später **nichts** installiert, die Pakete einmal in den deps-Ordner holen (NAS-Konsole / SSH):

```
docker run --rm -e PYTHONUSERBASE=/deps \
 -v /volume1/Docker/intexspa/deps:/deps \
 python:3.12-slim pip install --user tinytuya paho-mqtt
```

> ⏱Dauert kurz und ist **nur einmal** nötig.

6.3 · Stack starten

1. **Portainer → Stacks → Add stack**.
2. Name z. B. `intexspa`.
3. **Kompletten Inhalt** von `intexspa-stack.yml` in das Editorfeld kopieren.
4. **Deploy the stack** klicken.

Der Container … - … läuft mit `restart: unless-stopped` (kommt nach NAS-Neustart von allein), - … führt nur `python -u /app/bridge.py` aus (installiert nichts mehr), - … hängt im Netz `intexspa_default` (dasselbe Netz wie IP-Symcon).

6.4 · Log prüfen

Öffne in Portainer das **Log** des Containers `intexspa-bridge`. Erwartet:

```
RAW DPS: { ... }
Status veroeffentlicht: { ... }
```

Sobald du später in IP-Symcon schaltest, erscheint zusätzlich:

```
Befehl empfangen: intexspa/set/...
```

> **Hat's geklappt?** Im Log erscheinen Statuszeilen ohne Fehler, der Container bleibt **„running"**.
>
> **Wenn's klemmt** - `ModuleNotFoundError: tinytuya` → Schritt 6.2 (im richtigen deps-Ordner) wiederholen. - **Container findet IP-Symcon nicht** → Netzname muss exakt `intexspa_default` sein; `MQTT_HOST` = IP-Symcon-Container-Name. - **Anderer Docker-Ordner?** → nur die beiden `volumes`-Pfade im Stack anpassen.

7 · IP-Symcon als MQTT-Server einrichten

> **Ziel:** IP-Symcon wird selbst zum **MQTT-Server (Broker)**. Darüber redet die Brücke mit IP-Symcon.
> **Du brauchst:** Zugriff auf die **IP-Symcon-Verwaltungskonsole**

7.1 · MQTT-Server-Instanz anlegen

1. **Objektbaum** → Rechtsklick → **Objekt hinzufügen → Instanz**.
2. Nach **„MQTT Server"** suchen und auswählen.
3. Beim Anlegen nach dem **Gateway (Server Socket)** gefragt: - neuen Server-Socket anlegen, **Port 1883**, **aktiv**.
4. **Übernehmen**.
5. Instanz anklicken und die **Objekt-ID notieren** (Reserve für Schritt 9).

7.2 · Erreichbarkeit sicherstellen

- Die Brücke (Docker) muss **Port 1883** erreichen.
- Brücke und IP-Symcon laufen im selben Docker-Netz `intexspa_default` → die Brücke nutzt den **Container-Namen von IP-Symcon** als `MQTT_HOST` (Schritt 5).
- Diesen Namen jetzt – falls noch offen – in `bridge.py` eintragen und den Brücken-Container **neu starten**.
- Hast du im MQTT-Server **Benutzer/Passwort** gesetzt, müssen diese auch in die Brücke. *(Standard hier: ohne.)*

> **Gut zu wissen:** Der MQTT-Server liefert empfangene Nachrichtentexte **HEX-codiert** an Module (Thema bleibt lesbar). Das Spa-Modul rechnet das selbst zurück – du musst nichts tun.

> **Hat's geklappt?** MQTT-Server-Instanz **aktiv**, Server-Socket lauscht auf **1883**.
> **Wenn's klemmt** – Brücke verbindet nicht? `MQTT_HOST` = exakter Container-Name von IP-Symcon? Beide im Netz `intexspa_default` ?

8 · Modul auf GitHub hochladen

> **Ziel:** Das IP-Symcon-Modul liegt in einem **öffentlichen GitHub-Repo**, damit IP-Symcon es laden kann.
> **Du brauchst:** GitHub-Konto · Datei `dateien/IPSymcon-IntexSpaMQTT.zip`

8.1 · Repo anlegen

1. Auf **github.com** einloggen (`DEIN_GITHUB_NAME`).
2. Oben rechts **„+" → New repository**.
3. **Repository name:** `IPSymcon-IntexSpaMQTT`
4. Sichtbarkeit **Public** → **Create repository**.

8.2 · Inhalt richtig hochladen

> **Wichtigster Punkt:** Die Datei `library.json` **muss ganz oben** im Repo liegen (direkt auf der Startseite sichtbar). In einem Unterordner findet IP-Symcon das Modul **nicht**.

1. Entpacke `IPSymcon-IntexSpaMQTT.zip` auf dem PC. Du siehst:

`library.json ← muss in die Wurzel README.md LICENSE IntexSpaMQTT/ (Ordner: module.php, module.json, form.json, tile.html)`

2. Im neuen Repo auf **„uploading an existing file"** klicken.
3. **Den Inhalt** (lose Dateien **und** den Ordner `IntexSpaMQTT`) hineinziehen – **nicht** den umschließenden Ordner.
4. Unten **„Commit changes"**.

8.3 · Kontrolle

Auf der Repo-Startseite muss `library.json` **direkt sichtbar** sein:

```
library.json richtig
IntexSpaMQTT
README.md
```

> Siehst du nur **einen** Unterordner, ist die Struktur falsch → Dateien löschen und 8.2 mit dem *Inhalt* wiederholen.

> **Hat's geklappt?** `library.json` ist auf der Repo-Startseite direkt sichtbar.
> **Updates später:** ZIP-Inhalt erneut hochladen (Dateien ersetzen). Die Versionsnummer in `library.json` ist schon erhöht → löst in IP-Symcon das Update aus.

9 · Modul installieren & Instanz anlegen

> **Ziel:** Das Modul ist in IP-Symcon geladen, eine Spa-Instanz ist angelegt, und du kannst den Spa steuern.
> **Du brauchst:** das GitHub-Repo (Schritt 8) · laufende Brücke (Schritt 6) · MQTT-Server (Schritt 7)

9.1 · Modul laden

1. Verwaltungskonsole → **Kern Instanzen → Modules** (Doppelklick).
2. Auf **„+"** klicken und die Repo-URL eintragen:

```
https://github.com/DEIN_GITHUB_NAME/IPSymcon-IntexSpaMQTT
```

3. **OK**. Das Modul erscheint und muss **grün** (fehlerfrei) sein. - **Rot?** → Schritt 8 prüfen (liegt `library.json` oben?). Danach Eintrag anklicken → **Aktualisieren**, Konsole neu verbinden.

9.2 · Instanz anlegen

1. **Objektbaum** → Rechtsklick auf die gewünschte Kategorie → **Objekt hinzufügen → Instanz**.
2. Nach `IntexSpaMQTT` suchen → auswählen.
3. Wird nach einem **Gateway** gefragt → den **MQTT-Server** aus Schritt 7 wählen.

9.3 · Konfigurieren

1. **MQTT-Basis-Thema:** `intexspa` (muss zur Brücke passen).
2. Falls kein Gateway verbunden: Button **„Mit MQTT Server verbinden"**.
3. **Übernehmen**.

9.4 · Funktion testen

- Im Objektbaum erscheinen die Schalter (Power, Heizung, Filter, Sprudler, Jets, Hygienisierung), Ist-/Soll-Temperatur und der Zeitplan.
- Schalte testweise **Power** → der Spa muss reagieren, der Status zieht nach.
- Die **Kachel** zeigt Steuerung und Zeitplan.

9.5 · Update einer bestehenden Instanz (für später)

1. Neues ZIP ins Repo.
2. **Modules → Aktualisieren** → Konsole neu verbinden.
3. Instanz **Übernehmen** (**nicht löschen!**). Variablen/Verknüpfungen bleiben erhalten.

> **Hat's geklappt?** Du kannst den Spa aus IP-Symcon schalten, der Status ist live, die Kachel zeigt alles.
> **Wenn's klemmt** - **Schalter ohne Wirkung / Status springt zurück** → Brücken-Log prüfen (kommt `Befehl empfangen ...` ?), `MQTT_HOST` + Netz prüfen, aktuelles Modul nutzen. - **„Datenfluss inkompatibel"** → aktuelles Modul verwenden (GUIDs passen dort).

10 · Heizen bei PV-Überschuss

> **Ziel:** Der Spa heizt **automatisch**, wenn deine Photovoltaik genug Überschuss liefert – und der **Akku hat Vorrang**. Die Energie-Logik macht der offizielle **„Energie Manager"** von IP-Symcon; das Spa-Modul liefert nur einen Schalter und erledigt die Reihenfolge (erst Strom, dann Heizung).
>
> **Du brauchst:** eine Variable mit deinem **Überschuss in Watt** (siehe 10.5) · das installierte Spa-Modul (Schritt 9)

10.1 · Modul vorbereiten

1. Spa-Instanz öffnen → **„PV-Heizung-Schalter bereitstellen"** aktiv lassen → **Übernehmen**.
2. Dadurch existiert die Schaltvariable **„PV-Heizung (Energie Manager)"** (in der Visualisierung ausgeblendet, im Energie Manager aber wählbar).

10.2 · Energie Manager anlegen

1. **Objektbaum → Instanz hinzufügen → „Energie Manager"** *(fehlt er? → Module Store → „Energie Manager" installieren).*
2. Konfiguration öffnen: - **Modus:** `Absolut` - **Verfügbare Leistung (W):** deine **Überschuss-Variable in Watt** - **Invertieren:** so setzen, dass ein Überschuss **positiv** angezeigt wird
3. **Übernehmen**.

10.3 · Spa als Verbraucher eintragen

Im Energie Manager **„Verbraucher → Hinzufügen"**:

Feld	Wert
Verbraucher	Variable **„PV-Heizung (Energie Manager)"**
Maximaler Verbrauch	**2200 W** *(einziger Ort für die Watt-Angabe)*
Mindestlaufzeit	z. B. **600 s** *(verhindert Takten)*
Nachlaufzeit	z. B. **240 s** *(überbrückt Wolken – darf kurz aus dem Akku)*
Bedingung	**leer lassen!**

> **Keine Bedingung eintragen.** Der manuelle Vorrang wird komplett im Spa-Modul geregelt (Schritt 11). Eine Bedingung hier würde das stören.

10.4 · „Akku zuerst" – ohne ihn als Gerät einzutragen

Wird dein Speicher vom **Wechselrichter selbst** geregelt, trägst du ihn **nicht** als steuerbaren Speicher ein. „Akku zuerst" entsteht über die Überschuss-Zahl:

```
Verfügbare Leistung = Einspeisung + Akku-Ladeleistung (positiv = Überschuss)
```

Wirkung: Der Spa heizt erst, wenn über die Akkuladung hinaus genug übrig ist. Bei Wolken stützt der Akku kurz (Nachlaufzeit), danach geht der Spa aus und der Akku lädt weiter.

10.5 · kW → W umrechnen (falls deine Quelle in Kilowatt liefert)

Der Energie Manager rechnet in **Watt**. Liefert deine Variable **kW**:

1. **Neue Variable** anlegen: Typ **Integer**, Name z. B. „PV Überschuss W".
2. **Skript** anlegen:

```php „Bei Aktualisierung" lässt sich in manchen Ablaufplan-Versionen nicht stabil > speichern – **„Bei Änderung"** reicht hier völlig. 4. Im Energie Manager als **„Verfügbare Leistung (W)"** die **W-Variable** wählen. --- > **Hat's geklappt?** Bei genügend Überschuss schaltet der Energie Manager > „PV-Heizung" ein, der Spa geht an. Bei Mangel schaltet er nach der Nachlaufzeit ab. > **Wenn's klemmt** > - **Schaltet nie ein** → Überschuss positiv? (Invertieren).
```

Richtige W-Variable? > „Maximaler Verbrauch" = 2200? **Keine** Bedingung? > - **Wert viel zu klein (z. B. 3 statt 3000)** → kW nicht ×1000 umgerechnet (10.5).

?>

11 · Push-Nachricht & manueller Vorrang

> **Ziel:** - **Manuelles Heizen über die Kacheln hat Vorrang** – der Energie Manager darf eine von Hand eingeschaltete Heizung **nicht** ausschalten. - **Genau eine** Push-Nachricht, wenn du manuell heizt und der Energie Manager abschalten *würde* (kein Überschuss) – als Erinnerung, selbst auszuschalten. - **Sonst keine** Push-Meldungen.

11.1 · Wie das Verhalten funktioniert

Schaltest du die Heizung über die Kachel ein, merkt sich das Modul den manuellen Vorrang. Will der Energie Manager dann abschalten, **ignoriert** das Modul das Abschalten (deine Schaltung gewinnt) und sendet **einmalig** die Erinnerung.

> Damit das greift: im Energie Manager beim Verbraucher **keine Bedingung** setzen (Schritt 10.3).

11.2 · Der richtige Push-Weg in Symcon 9.0

Mit der **Kachel-Visualisierung** funktioniert die alte Funktion `WFC_PushNotification` **nicht** (nur fürs alte WebFront). Richtig ist `VISU_PostNotification` – genau das nutzt das beiliegende Skript.

11.3 · Handy als Push-Empfänger registrieren

1. Öffne am Handy die **IP-Symcon-App** und logge dich einmal in deine **Kachel-Visualisierung** ein.
2. Konsole → Instanz der **Kachel-Visualisierung** öffnen → Reiter **„Benachrichtigungen"**. Dein Handy muss dort gelistet sein.
3. Sende von dort eine **Testnachricht**. Kommt sie an, ist der Push-Weg in Ordnung. *(Voraussetzung: gültige App-Subskription.)*

11.4 · Push-Skript anlegen

1. **Skript** anlegen, Inhalt aus `dateien/Spa_Push.php` einfügen, speichern. Es findet deine Kachel-Visualisierung automatisch und sendet per `VISU_PostNotification`.
2. Test: Rechtsklick → **Ausführen**. Erwartet: „Push an 1 Kachel-Visualisierung(en) gesendet". Notfalls oben im Skript `$visuID` fest auf die ID setzen.

11.5 · Skript ins Modul eintragen

Spa-Instanz öffnen → Feld **„Push-Skript für Erinnerung"** → dein Skript auswählen → **Übernehmen**.

> **Hat's geklappt?** Heizung über die Kachel einschalten, wenn gerade **kein** Überschuss da ist: Die Heizung muss **anbleiben**, und es kommt **eine** Push-Nachricht.
>
> **Wenn's klemmt** - **Kein Push:** Kachel-Visualisierung (nicht WebFront)? Handy im Reiter „Benachrichtigungen" gelistet? Subskription gültig? Testnachricht (11.3) ok? - **Heizung geht trotzdem aus:** Im Energie Manager ist beim Verbraucher doch eine **Bedingung** gesetzt → entfernen (Schritt 10.3).

12 · Fehlerbehebung & Wartung

> **Ziel:** Probleme schnell lösen und das System sauber aktuell halten.

12.1 · Schnelle Hilfe bei typischen Problemen

Symptom	Ursache & Lösung
Spa lässt sich nicht in Smart Life koppeln	2,4-GHz-WLAN? Schneller Blinkmodus? Evtl. zuerst aus „INTEX Link" entfernen (Schritt 1).
Spa fehlt im Tuya-Portal	Data Center = **Central Europe**? App per QR mit gleichem Konto verknüpft (Schritt 3)?
Kachel springt nach dem Schalten zurück	Im aktuellen Modul gelöst. Aktuelles Modul verwenden.
Spa reagiert nicht	Brücken-Log: kommt `Befehl empfangen …` ? `MQTT_HOST` = IP-Symcon-Container-Name? Beide im Netz `intexspa_default` ?
Status veraltet / hinkt nach	Normal direkt nach lokalem Schalten (Tuya). Echte Geräte-Pushes ziehen sofort nach.
„Datenfluss inkompatibel" am Gateway	Aktuelles Modul nutzen (GUIDs passen zum MQTT-Server).
Automatik schaltet aus, Heizung bleibt an	Im aktuellen Modul gelöst (Heizung AUS → kurz warten → Strom AUS). Aktuelles Modul verwenden.
Neue Modul-Variablen fehlen nach Update	Instanz **Übernehmen**; ggf. Modules → Aktualisieren + Konsole neu verbinden.
Modul in „Modules" ist rot	`library.json` muss im Repo ganz oben liegen (Schritt 8).
Push kommt nicht	Kachel-Visualisierung → `VISU_PostNotification` (Schritt 11); Handy registriert? Subskription?
Local Key passt nicht mehr	Spa wurde neu gekoppelt → Schritt 4 wiederholen.
Energie Manager schaltet nie	Überschuss positiv (Invertieren)? Max 2200? **Keine** Bedingung? Richtige W-Variable?

12.2 · Wartung & Updates

Brücke ändern 1. `bridge.py` in `/volume1/Docker/intexspa/bridge/` ersetzen. 2. Container `intexspa-bridge` **neu starten**.

Modul ändern 1. Neues ZIP ins GitHub-Repo (Inhalt ersetzen). 2. **Modules → Aktualisieren** → Konsole neu verbinden. 3. Instanz **Übernehmen** *(nicht löschen)*.

12.3 · Wichtige Fakten (zum Nachschlagen)

- Kopplung läuft über **Smart Life** (Tuya), **nicht** INTEX Link.
- IP-Symcon-MQTT-Server liefert Nachrichtentexte **HEX-codiert** an Module (Thema lesbar).
- Befehle: Wert steht im **Thema** (`intexspa/set/<name>/<wert>`), nicht im Text.
- Tuya: Befehle nicht zu schnell hintereinander (Ein/Aus jeweils mit kurzer Pause).
- Temperaturen kommen in **Fahrenheit** vom Gerät – Brücke/Modul rechnen in °C um.
- DP-Zuordnung siehe **Schritt 4.4**.

12.4 · Umzug auf eine Symbox (später)

`MQTT_HOST` in der Brücke auf die **Symbox-IP** setzen, Brücken-Container im passenden Netz starten – der Rest bleibt gleich.

12.5 · Wenn du nicht weiterkommst

Halte bereit: das **Brücken-Log**, die **Modul-Version** (aus `library.json`) und eine kurze Beschreibung „erwartet vs. passiert".

Geschafft! Dein Intex PureSpa läuft jetzt lokal in IP-Symcon.