

Synixx — Handbuch für den Betrieb

Dieses Handbuch richtet sich an die Person, die Synixx auf einem Server installiert, aktuell hält und sichert — etwa die IT einer Hochschule oder eines Unternehmens. Für die Einrichtung von Kursen und Zugangscode siehe das *Handbuch für Administration*. Die konkrete Installation auf dem SmartBPM-NAS ist in `docs/hosting/README.md` beschrieben und dient hier als Beispiel.

1. Was Synixx ist

Synixx ist **eine Node.js-Anwendung** (Express + Socket.io) mit mehreren Oberflächen. Welche davon ein Server anbietet, bestimmt die **Edition**:

EDITION (<code>SYNIXX_EDITION</code>)	ENTHÄLT	TYPISCHE ADRESSE
<code>bpmn</code>	Synixx BPMN: Einzelplatz-Modellierung mit Simulation, ohne KI	<code>/bpmn</code>
<code>bpmn-team-praxis</code>	Synixx BPMN Team (Unternehmen, Schulung): gemeinsames Modellieren in Räumen, mit KI	<code>/bpmn-team</code>
<code>bpmn-team-campus</code>	Synixx BPMN Team, Campus-Ausgabe (Hochschulen): wie oben, dazu der Forschungsbereich — ab Werk ausgeschaltet	<code>/bpmn-team</code> , <code>/forschung</code> , <code>/bewertung</code>
<code>quiz-basis</code>	Synixx-Live: Quiz für Vorlesungen	<code>/</code> (Spieler), <code>/admin.html</code>
<code>quiz-plus</code>	Quiz mit Planning Poker und Live-Demos	wie oben
<code>full</code>	Alles (Standard), einschließlich Forschungsbereich	alle

Auf einer Instanz mit Synixx BPMN Team (ohne Quiz) leitet `/bpmn` auf `/bpmn-team` weiter — jede Adresse zeigt genau ein Produkt. Die früheren Namen `bpmn-studio` , `bpmn-studio-praxis` und `bpmn-studio-campus` gelten weiter. Die früheren Adressen `/bpmn-live` und `/bpmn-studio` leiten dauerhaft auf `/bpmn` bzw. `/bpmn-team` weiter. Einen unbekannten Namen meldet das Log als Warnung; der Server läuft dann als `full` . Seiten anderer Editionen liefert der Server mit **404** aus. Für getrennte Zielgruppen (öffentlich / Partner / intern) betreiben Sie **getrennte Instanzen** mit je eigenem Datenbaum — eine Instanz mit Edition `full` für alles würde Bibliothek, Zugangscode und Räume zwischen allen Gruppen teilen.

Für die Weitergabe eines Teilprodukts als Quelltext erzeugt `node scripts/mk-edition.mjs` `<edition>` ein Paket ohne die nicht benötigten Teile. Im Praxis-Paket ist der Forschungscode damit physisch nicht enthalten.

Lizenz: Zusatzfunktionen freischalten

Die Edition legt fest, welche Teile eine Installation **überhaupt** enthält; sie kann nur einschränken. Die Zusatzfunktionen **Team** (Räume, gemeinsames Bearbeiten, Fernbedienung), **KI**, **Forschung**, **Externe Dienste** (echte Aufrufe in der Simulation) und **Agenten-Rückrufe** (Aufträge an KI-Agenten, die sich später über einen offenen Eingang melden; setzt Externe Dienste voraus) laufen zusätzlich nur mit einer gültigen, von Synixx signierten Lizenzdatei. Es gilt immer die Schnittmenge aus Edition und Lizenz.

LIZENZ	ERGEBNIS
keine, ungültig oder verändert	Grundausstattung
gültig	Grundausstattung + lizenzierte Zusatzfunktionen (soweit in der Edition enthalten)
abgelaufen (nach dem Tag „gültig bis“)	Grundausstattung, ohne Neustart
aufgerufen unter einer nicht lizenzierten Adresse	unter dieser Adresse Grundausstattung, ohne Neustart

KI ist auch ohne Team lizenzierbar; Forschung und Externe Dienste setzen Team voraus.

Ablage: `data/synixx-lizenz.json` im Datenordner (oder ein eigener Pfad über `SYNIXX_LICENSE_FILE`). Ein Admin kann die Datei auch über **☰ → Verwaltung → Lizenz...** hochladen.
Freischalten: Datei ablegen und den Server neu starten (`docker compose restart`) — erst beim Start werden die Bausteine der Zusatzfunktionen geladen. Ablauf und Adressbindung prüft der Server dagegen bei jeder Anfrage. Der Start meldet im Log, welche Lizenz gilt.

Für Partner, die später erweitern wollen, empfiehlt sich die Edition `bpmn-team-praxis` : Team und KI sind im Paket enthalten, aber ohne passende Lizenz gesperrt — eine neue Lizenzdatei genügt, eine neue Auslieferung ist nicht nötig. Wer dauerhaft nur die Grundausstattung bekommt, erhält die Edition `bpmn` , in der Team und KI physisch fehlen.

Grenze: Die Unterschrift schützt gegen das Umkonfigurieren, nicht gegen das Umprogrammieren des ausgelieferten Quelltexts. Was ein Partner nie bekommen soll, gehört deshalb nicht in sein Paket.

2. Voraussetzungen

- **Docker** mit Compose (empfohlen) oder **Node.js ≥ 22.13**.
- Etwa **1 GB RAM** je Instanz; Speicher für die Daten (Modelle, Versionen, Medien) — für Kurse reichen wenige GB.

- Ein **Reverse-Proxy mit HTTPS** (z. B. Nginx Proxy Manager, Caddy, Traefik). Synixx nutzt WebSockets; der Proxy muss sie durchreichen.
- Optional für KI: API-Schlüssel eines Anbieters (Anthropic, Google, OpenAI) oder ein eigener Modellserver (Ollama bzw. OpenAI-kompatibles Gateway).

3. Installation mit Docker

1. Quelltext auf den Server bringen (Git-Checkout oder Editions-Paket).
2. Datenordner anlegen, zum Beispiel `/srv/synixx/daten/data`.
3. Eine `.env` neben der `docker-compose.yaml` anlegen (Abschnitt 4).
4. Bauen und starten:

```
docker compose build
docker compose up -d
```

5. Im Reverse-Proxy die Adresse auf den Container-Port weiterleiten (intern Port 3000) und ein Zertifikat einrichten.
6. Prüfen: `https://<adresse>/api/version` liefert Version und Baudatum.

Die Bundles (BPMN-, DMN-Editor usw.) baut das Image beim `build` selbst.

Datenbaum

Der Container erwartet seine Daten unter `/app/data`:

<code>data/</code>	
<code> bpmn/bibliothek/</code>	Modellbibliothek (geschützt), darin <code>.access.json</code> und <code>.connections.json</code> (Verbindungen externer Dienste, Rechte 600)
<code> bpmn/raeume/</code>	Räume und Gruppenordner (geschützt)
<code> bpmn/forschung/</code>	Forschungsbereich: Aufgaben, Studien mit Ergebnissen und Bewertungen, Forschungsräume (nur Campus/Voll)
<code> quiz/</code>	Quiz-Dateien und Medien (nur Quiz-Editionen)
<code> ai-knowledge/</code>	Wissensdokumente für die KI (optional)
<code> .app-config.json</code>	Einstellungen des Quiz-Admins

Die Ordner entstehen beim ersten Start von selbst. Bibliothek, Räume und Forschungsbereich sind über das Web **nicht** direkt abrufbar (403), nur über die angemeldeten Schnittstellen.

4. Konfiguration (`.env`)

Die wichtigsten Einträge:

VARIABLE	BEDEUTUNG
SYNIXX_EDITION	Edition, siehe Abschnitt 1
SYNIXX_LICENSE_FILE	Pfad der Lizenzdatei (Vorgabe <code>data/synixx-lizenz.json</code>), siehe Abschnitt 1
ADMIN_PASSWORD	Admin-Passwort dieser Instanz. Lang, eigenes je Instanz. Keine <code>\$</code> , <code>#</code> , Anführungszeichen oder Leerzeichen — Compose deutet <code>\$</code> als Variable.
CORS_ORIGIN	Die öffentliche Adresse, z. B. <code>https://bpmn-team.example.org</code>
LOG_LEVEL	<code>info</code> (Standard), <code>warn</code> , <code>debug</code>
BPMN_MODELS_DIR , TEAM_ROOMS_DIR	Nur setzen, wenn Bibliothek oder Räume woanders liegen sollen
GUEST_FILE_TTL_MS	Wie lange Dateien im Gast-Raum bleiben (Standard 14 Tage)
ACCESS_TZ	Zeitzone, in der „Gültig ab/bis“ der Zugangscodes gilt (Standard <code>Europe/Berlin</code>). Unabhängig von der Zeitzone des Containers, der meist auf UTC läuft.
TEAM_CHECKPOINT_MS	Abstand der automatischen Zwischenstände (Standard 15 Minuten, 0 = aus)
SERVICES_FILE	Ort des Verbindungs-Verzeichnisses der externen Dienste (Vorgabe <code>bpmn/bibliothek/.connections.json</code>). Es enthält Schlüssel im Klartext — in die Sicherung, aber nicht weitergeben.
AGENT_CALLBACK_BASE	Öffentliche Adresse, unter der Agenten ihre Rückrufe schicken, z. B. <code>https://bpmn-team.example.org</code> . Leer = aus der Verbindung des Browsers abgeleitet (über <code>X-Forwarded-Proto</code> / <code>-Host</code> des Proxys).
RESEARCH_ENABLED	Nur Campus/Voll: <code>true</code> schaltet den Forschungsbereich beim allerersten Start ein. Danach gilt der Schalter auf <code>/forschung</code> (gespeichert in <code>bpmn/forschung/.research.json</code>).
RESEARCH_DIR	Nur setzen, wenn der Forschungsbereich woanders liegen soll

KI (nur Editionen `bpmn-team-praxis`, `bpmn-team-campus` und `full`):

VARIABLE	BEDEUTUNG
AI_ENABLED	true schaltet den KI-Assistenten ein
AI_PROVIDER	Voreingestellter Anbieter: claude , gemini , openai , qwen , ollama
AI_CLAUDE_API_KEY , AI_GEMINI_API_KEY , AI_OPENAI_API_KEY	Schlüssel der Cloud-Anbieter
AI_CLAUDE_MODEL , AI_GEMINI_MODEL	Standardmodell; leer = Voreinstellung im Code
AI_QWEN_URL , AI_QWEN_API_KEY	Eigener OpenAI-kompatibler Modellserver
AI_OLLAMA_EMBED_URL	Ollama für die Wissenssuche (Modell bge-m3)
AI_WHISPER_URL	Spracherkennung (faster-whisper)
AI_TTS_URL	Sprachausgabe (OpenAI-kompatibles /v1/audio/speech)

Die Schlüssel bleiben auf dem Server; der Browser sieht sie nie. Ohne Schlüssel bzw. URL ist der jeweilige Anbieter im KI-Panel einfach nicht verfügbar.

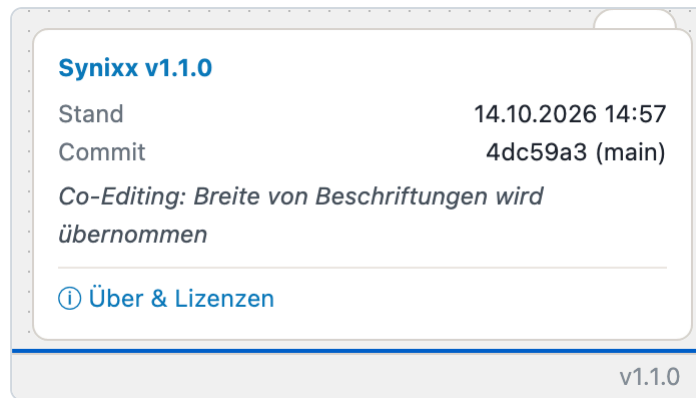
Wichtig: Umgebungsvariablen werden beim **Erstellen** des Containers übernommen. Nach einer Änderung der `.env` reicht `docker restart` nicht — nötig ist `docker compose up -d` (erstellt den Container neu).

5. Aktualisieren

```
# neuen Quelltext holen (git pull oder Dateien kopieren), dann:
docker compose build      # ohne Dienstnamen: baut ALLE Instanzen
docker compose up -d
```

Betreiben Sie mehrere Instanzen aus einer Compose-Datei, hat jede ein eigenes Image. `docker compose build <einer>` aktualisiert nur diese eine — die anderen laufen unbemerkt mit dem alten Stand weiter.

Kontrolle nach dem Update: Unten rechts in jeder Oberfläche steht die Versionsnummer. Ein Klick zeigt Stand und Baudatum.



Die Versionsanzeige. Angemeldete Administratoren sehen zusätzlich den Commit und eine Warnung, falls Server und Dateien nicht zusammenpassen.

Von der Kommandozeile:

```
curl -s https://<adresse>/api/version
```

Das `builtAt` aller Instanzen muss nach einem Update übereinstimmen.

GEÄNDERT	NÖTIG
<code>.env</code>	<code>docker compose up -d</code>
Quelltext	<code>docker compose build</code> + <code>up -d</code>
<code>docker-compose.yaml</code>	<code>docker compose up -d</code>
Daten im Datenordner	nichts

Forschungsstudien ohne offenen Browser (nur Campus)

Studien laufen entweder im Browser einer forschenden Person oder unbeaufsichtigt auf einem Server, etwa einem eigenen KI-Rechner. Dafür gibt es einen **Ausführer** (`scripts/research-agent.mjs`): Er meldet sich mit einem Code mit Forschungs-Freigabe an, holt beauftragte Studien ab und arbeitet sie in einem unsichtbaren Browser ab. Er braucht Node.js ≥ 22 , das Paket `playwright` mit Chromium und Netzzugang zur Instanz.

VARIABLE	BEDEUTUNG
<code>SYNIXX_URL</code>	Adresse der Instanz
<code>SYNIXX_RESEARCH_CODE</code>	Zugangscode mit „Forschungsbereich nutzen“
<code>RESEARCH_AGENT_NAME</code>	Name des Ausführers, z. B. <code>nas</code> oder <code>pgx</code> ; auf <code>/forschung</code> als Ziel wählbar

Die Sprachmodelle ruft weiterhin der Synixx-Server auf; die KI-Schlüssel bleiben dort. Einrichtung mit Docker-Beispiel: `docs/forschung/ausfuehrer.md`.

6. Sichern und Wiederherstellen

Zu sichern ist **nur der Datenordner** (`/app/data` im Container). Er enthält:

- die Modellbibliothek und die ZugangsCodes (`.access.json`),
- die Lizenzdatei (`synixx-lizenz.json`),
- alle Räume mit Gruppenordnern, Versionen (`.versions/` , gzip-komprimiert), dem Live-Stand gemeinsamer Bearbeitung (`.state.ydoc`) und der Raumliste (`.rooms.json`),
- Diskussionen und Prozess-Dossiers (`_process.meta.json` je Raum),
- den Forschungsbereich (`bpmn/forschung/`): Aufgabenkorporus, Studien mit allen Läufen, Kennzahlen und Bewertungen. Codes von Bewertenden stehen dort nur als Prüfsumme, nie im Klartext.

Alle Dateien werden atomar geschrieben, eine Sicherung im laufenden Betrieb ist unkritisch. Für einen konsistenten Gesamtstand vorher `docker compose stop` .

Wiederherstellen: Container stoppen, Datenordner zurückspielen, starten. Beim Start meldet das Log Räume ohne Ordner oder Ordner ohne Raum, löscht aber nichts.

Was der Server selbst aufräumt

WAS	WANN
Dateien im Gast-Raum	nach 14 Tagen
Räume eines abgelaufenen oder gelöschten Codes	48 Stunden nach Ablauf, nur wenn leer
Selbst gegründete Räume	nach 7 Tagen ohne Aktivität, nur wenn leer
Forschungsräume und Studien	nie automatisch
Alte Versionen	gestaffelt ausgedünnt; höchstens 100 je Datei
KI-Mitschnitte (falls eingeschaltet)	nach 30 Tagen

7. Sicherheit

- **Ein Passwort je Instanz.** Das Admin-Passwort öffnet alle Daten dieser Instanz.
- **Datenbäume trennen.** Eine öffentlich erreichbare Instanz bekommt einen eigenen Datenordner — niemals den einer internen Instanz.
- **KI-Schlüssel nur dort, wo KI gebraucht wird.** Die Edition `bpmn` hat keine KI; Schlüssel gehören dort nicht hin.
- **ZugangsCodes sind keine Passwörter.** Sie öffnen Bibliotheksordner und Räume, sind aber kurz und werden weitergegeben. Vertrauliches gehört in Ordner, die nur ein bestimmter Code sieht, und Codes bekommen ein Ablaufdatum.

- **Forschungsbereich:** ab Werk ausgeschaltet. Dann sind seine Räume und Daten auch für Administratoren gesperrt; erreichbar bleibt ihnen nur der Schalter auf `/forschung`. Codes mit der Rolle *Bewertung* sehen nur ihre Bewertungsseite, keine Studien oder Ergebnisse.
- **Externe Dienste:** Der Server ruft nur Verbindungen auf, die ein Admin freigegeben hat, nur deren Pfade und Methoden. Private, Loopback- und Link-Local-Adressen sind gesperrt — geprüft nach der Namensauflösung —, außer die Verbindung erlaubt das interne Netz ausdrücklich. Weiterleitungen werden nicht verfolgt, Antworten auf Größe und Zeit begrenzt. Personenbezogene Daten, die ein Modell an einen Dienst schickt, verlassen das Haus; das bleibt Sache des Betriebs.
- **Rückruf-Eingang der Agenten** (`POST /api/agent-rueckruf/<Marke>` , nur mit dem Paket Agenten-Rückrufe): offen erreichbar, damit Agenten außerhalb antworten können. Jede Marke ist zufällig, gilt für einen Auftrag und verfällt (Gültigkeit je Verbindung); mit Rückruf-Schlüssel muss der Agent den Rumpf signieren (HMAC-SHA256, sonst 401). Der Rumpf ist auf 64 KB begrenzt (`AGENT_CALLBACK_MAX_BYTES`), je Absender gilt eine Ratenbegrenzung (`AGENT_CALLBACK_RATE_MAX` je Minute). Der Proxy muss den Pfad durchreichen; sonst ist nichts einzurichten.
- **Externe Dienste in Camunda 8:** Exportierte Modelle rufen die Dienste über den REST-Connector von Camunda auf; Schlüssel gehören dort in die Secrets des Connector-Laufzeitdienstes (Namen nennt der Hinweis beim Deployen). Ab Camunda 8.9 liest dieser Dienst nur Umgebungsvariablen mit dem Präfix `SECRET_ : {{secrets.X}}` steht dort als `SECRET_X` . Die Variablen wirken erst, wenn der Container neu erstellt wird (`docker compose up -d --force-recreate connectors`). Aufträge an Agenten rufen in Camunda den **Webhook-Eingang der Konnektorlaufzeit** zurück (`/inbound/<id>` , Standard-Port 8086) — er muss für den Agenten erreichbar sein, und der Rückruf-Schlüssel gehört als Secret dorthin.
- Anmeldeversuche und Socket-Ereignisse sind pro Verbindung begrenzt (Rate-Limit). Überschreitungen erscheinen im Log.

8. Rechtliches

- **Lizenzen:** `https://<adresse>/lizenzen` listet alle verwendeten Open-Source-Bestandteile mit Lizenztext. Erreichbar über die Versionsanzeige („Über & Lizenzen“).
 - Das Wasserzeichen „Powered by bpmn.io“ auf der Zeichenfläche ist Teil der bpmn.io-Lizenz und darf nicht entfernt oder verdeckt werden.
 - **Impressum und Datenschutzerklärung** stellt der Betreiber für seinen öffentlichen Betrieb selbst bereit. Bei eingeschalteter KI werden Modellinhalte an den gewählten Anbieter übertragen; das gehört in die Datenschutzerklärung. Das gilt ebenso für Forschungsstudien: Aufgabentexte und Bilder gehen an die gewählten Anbieter. Mit einem eigenen Modellserver bleiben die Daten im Haus.
-

9. Fehlersuche

BEOBAHTUNG	URSACHE UND ABHILFE
Neue Funktion fehlt nach dem Update	Image nicht neu gebaut, oder nur eine von mehreren Instanzen. <code>build</code> ohne Dienstnamen, dann <code>up -d</code> . Versionsanzeige prüfen.
<code>.env</code> -Änderung wirkt nicht	Container nicht neu erstellt: <code>docker compose up -d</code> .
Verbindung bricht ständig ab	Reverse-Proxy reicht keine WebSockets durch (<code>Upgrade</code> / <code>Connection</code> -Header).
KI-Punkt im Panel rot	Schlüssel fehlt oder ist ungültig; der Tooltip nennt den Grund. Ein aufgebrauchtes Kontingent zeigt sich erst beim ersten Chat.
Gemeinsame Bearbeitung zeigt veraltete Stände	Browser-Cache: Hard-Reload. Der Server liefert Skripte mit <code>no-cache</code> ; hängt ein vorgeschalteter Cache dazwischen, diesen für <code>/js/</code> und <code>/css/</code> abschalten.
Log-Ausgabe	<code>docker compose logs -f <dienst></code> ; mehr Details mit <code>LOG_LEVEL=debug</code> .