Testbericht

External Task Worker Stabilität bei Verbindungsverlust

Paket:	@5minds/processcube_app_sdk v8.2.1
Datum:	13.02.2026
Testumgebung:	Docker-Compose (Engine, Authority, PostgreSQL) + lokale Next.js-Testanwendung
Komponenten:	ExternalTaskWorkerProcess, ExternalTaskAdapter

---

1. Zusammenfassung

In Version 8.2.0 des ProcessCube App SDK wurden mehrere kritische Probleme im External Task Worker Mechanismus identifiziert, die dazu führten, dass Worker bei Verbindungsverlust zur Engine oder Authority nicht zuverlässig wiederhergestellt wurden. Alle identifizierten Probleme wurden in Version 8.2.1 behoben und erfolgreich getestet.

> **Ergebnis: Alle Tests bestanden.** Die Worker erholen sich jetzt zuverlässig von Verbindungsverlusten zur Engine und Authority.

---

2. Identifizierte Probleme

2.1 Exponentielles Backoff funktionierte nicht (kritisch)

Problem: Der Retry-Zähler (`connectionRetryCount`) wurde bei jedem Verbindungsversuch sofort auf 0 zurückgesetzt. Da die Zurücksetzung synchron nach dem `start()`-Aufruf erfolgte — aber bevor der asynchrone Error-Callback ausgelöst wurde — startete jeder Retry-Zyklus wieder bei Versuch 1/6. Das exponentielle Backoff (1s → 2s → 4s → 8s → 16s → 30s) griff nie; stattdessen wurde immer mit konstantem 1-Sekunden-Delay retried.

Auswirkung: Bei einem Engine-Ausfall hämmerten die Worker die Engine permanent mit 1-Sekunden-Intervall an, anstatt die Last durch exponentielles Backoff zu reduzieren.

Behebung: Ein `isReconnecting`-Flag verhindert jetzt das Zurücksetzen des Zählers während eines Retry-Zyklus. Der Zähler wird erst nach einer erfolgreichen Verbindung zurückgesetzt.

2.2 Token-Refresh-Zyklus gab nach 10 Sekunden auf (kritisch)

Problem: Bei Fehlern im Token-Refresh (z.B. Authority nicht erreichbar) gab es nur 5 Retry-Versuche mit festem 2-Sekunden-Delay. Nach insgesamt 10 Sekunden wurden alle Worker beendet und der Token-Refresh-Zyklus wurde permanent gestoppt. Eine Wiederherstellung war ohne App-Neustart nicht möglich.

Auswirkung: Ein kurzer Ausfall der Authority (>10 Sekunden) führte zum vollständigen und dauerhaften Verlust aller External Task Worker.

Behebung: Der Token-Refresh verwendet jetzt exponentielles Backoff (1s → 2s → 4s → ... → 60s max) und gibt nie auf. Worker werden bei Token-Refresh-Fehlern nicht mehr beendet. Der Zyklus retried unbegrenzt, bis die Authority wieder erreichbar ist.

2.3 IPC-Fehler konnten den Token-Refresh-Zyklus crashen (kritisch)

Problem: Wenn ein Worker-Prozess während des Token-Refresh beendet oder disconnected wurde, warf der `send()`-Aufruf für die Identity-Aktualisierung eine Exception. Diese wurde fälschlicherweise als Token-Refresh-Fehler gezählt und beschleunigte das Erreichen des Retry-Limits.

Auswirkung: Das Beenden einzelner Worker konnte den gesamten Token-Refresh-Zyklus zum Absturz bringen.

Behebung: Jeder `send()`-Aufruf ist nun einzeln in `try/catch` umgeben. Fehler beim Senden werden als Warning geloggt, beeinflussen aber nicht den Token-Refresh-Zyklus.

2.4 App startete nicht bei verzögerter Authority (mittel)

Problem: Der initiale Token-Fetch beim App-Start hatte keinen Retry-Mechanismus. Wenn die Authority beim Start noch nicht bereit war, schlug der External Task Adapter sofort und endgültig fehl.

Auswirkung: In Container-Umgebungen, in denen die Authority einige Sekunden länger zum Starten braucht als die App, konnten External Tasks nie gestartet werden.

Behebung: Der initiale Token-Fetch hat nun 10 Retry-Versuche mit exponentiellem Backoff (bis max. 30s pro Versuch).

2.5 Worker-Start-Exception nicht als Verbindungsfehler behandelt (mittel)

Problem: Wenn `externalTaskWorker.start()` synchron eine Exception warf, wurde kein Reconnect ausgelöst und der Worker-Prozess beendete sich ohne Wiederherstellungsmöglichkeit.

Behebung: Der `start()`-Aufruf ist jetzt in `try/catch` umgeben. Bei Fehler wird die gleiche Reconnect-Logik wie bei asynchronen Verbindungsfehlern ausgelöst.

2.6 Token-Refresh-Zyklus bei Worker-Restart nicht wiederhergestellt (mittel)

Problem: Wenn der Token-Refresh-Zyklus einmal gestoppt hatte und der Adapter einen Worker-Prozess neu startete, wurde der Zyklus nicht wieder aufgenommen. Die Worker liefen dann mit veralteten Tokens.

Behebung: Beim Adapter-Restart wird nun geprüft, ob der Token-Refresh-Zyklus aktiv ist. Falls nicht, wird ein neuer Token geholt und der Zyklus neu gestartet.

---

3. Testumgebung

3.1 Infrastruktur

Komponente	Version / Image	Port
ProcessCube Engine	`5minds/processcube_engine:16.2.0-hotfix-develop`	8000
Authority	`5minds/processcube_authority:7.1.1`	11560
PostgreSQL	`postgres:15.2`	5432
Next.js Test-App	Lokale Testanwendung mit npm-linked SDK v8.2.1	—

3.2 Konfiguration

```
PROCESSCUBE_ENGINE_URL=http://localhost:8000
PROCESSCUBE_AUTHORITY_URL=http://authority:11560
PROCESSCUBE_EXTERNAL_TASK_WORKER_CLIENT_ID=external_task_worker
PROCESSCUBE_EXTERNAL_TASK_WORKER_CLIENT_SECRET=external_task_worker_secret
```

3.3 Test-Worker

Ein External Task Worker (`test-task/external_task.ts`) mit Topic `test-task`, der eine einfache Aufgabe verarbeitet und ein Ergebnis zurückgibt.

---

4. Testszenarien und Ergebnisse

4.1 Engine-Ausfall ohne Last

Szenario: Engine wird bei laufenden Workern gestoppt und nach ca. 60 Sekunden wieder gestartet.

Durchführung:

1. Testanwendung mit External Task Worker starten

2. Worker registriert sich erfolgreich bei der Engine
3. Engine-Container stoppen (docker stop)
4. Worker-Verhalten im Log beobachten
5. Engine-Container starten (docker start)
6. Recovery-Verhalten im Log beobachten

Erwartetes Verhalten:

- Worker erkennt Verbindungsfehler
- Exponentielles Backoff: 1s → 2s → 4s → 8s → 16s → 30s
- Nach Engine-Neustart: Automatische Wiederverbindung

Ergebnis: BESTANDEN

Beobachtetes Verhalten:

```
Connection error for topic test-task, retrying in 1000ms (attempt 1/6)
Connection error for topic test-task, retrying in 2000ms (attempt 2/6)
Connection error for topic test-task, retrying in 4000ms (attempt 3/6)
Connection error for topic test-task, retrying in 8000ms (attempt 4/6)
Connection error for topic test-task, retrying in 16000ms (attempt 5/6)
```

Nach dem Neustart der Engine:

```
Started external task worker [...] for topic test-task
```

Der Worker erholte sich vollständig und nahm das Polling wieder auf.

4.2 Authority beim App-Start nicht verfügbar

Szenario: Testanwendung wird gestartet, während die Authority noch nicht erreichbar ist.

Durchführung:

1. Authority-Container stoppen
2. Testanwendung starten
3. Verhalten des initialen Token-Fetch im Log beobachten
4. Authority-Container starten
5. Recovery-Verhalten im Log beobachten

Erwartetes Verhalten:

- Initialer Token-Fetch schlägt fehl
- Exponentielles Backoff: 1s → 2s → 4s → 8s → ... (bis max. 30s, 10 Versuche)
- Nach Authority-Start: Token-Fetch erfolgreich, Worker starten normal

Ergebnis: BESTANDEN

Beobachtetes Verhalten:

```
Failed to fetch initial token set (attempt 1/10), retrying in 1000ms
Failed to fetch initial token set (attempt 2/10), retrying in 2000ms
Failed to fetch initial token set (attempt 3/10), retrying in 4000ms
```

Nach dem Start der Authority:

```
Started external task worker [...] for topic test-task
```

Die Worker starteten erfolgreich, sobald die Authority erreichbar war.

4.3 Engine-Ausfall unter Last (zyklische Prozessinstanzen)

Szenario: Ein Prozessmodell wird so konfiguriert, dass es den External Task zyklisch aufruft. Während aktiver Verarbeitung wird die Engine gestoppt und wieder gestartet.

Durchführung:

1. Testanwendung mit External Task Worker starten
2. Prozessinstanz starten, die den External Task zyklisch aufruft
3. Verarbeitung im Log bestätigen
4. Engine-Container stoppen während aktiver Verarbeitung
5. Worker-Verhalten im Log beobachten
6. Engine-Container starten
7. Recovery-Verhalten im Log beobachten

Erwartetes Verhalten:

- Worker erkennt Verbindungsfehler trotz laufender Verarbeitung
- Exponentielles Backoff greift korrekt
- Nach Engine-Neustart: Worker verbindet sich und nimmt Verarbeitung wieder auf

Ergebnis: BESTANDEN

Beobachtetes Verhalten:

Der Worker erkannte den Verbindungsverlust, durchlief das exponentielle Backoff und stellte nach dem Engine-Neustart die Verbindung wieder her. Die zyklische Verarbeitung wurde ohne manuellen Eingriff fortgesetzt.

---

5. Konfigurierbare Parameter

Parameter	Umgebungsvariable	Standard	Beschreibung
Worker-Reconnect-Versuche	`PROCESSCUBE_APP_SDK_ETW_RETRY`	6	Max. Verbindungsversuche pro Worker bei Engine-Ausfall

Worker-Backoff (max)	—	30s	Max. Wartezeit zwischen Worker-Reconnect-Versuchen
Adapter-Restarts	—	6 in 5 Min	Max. Worker-Neustarts durch den Adapter pro Zeitfenster
Token-Refresh-Retries	—	unbegrenzt	Token-Refresh gibt nie auf, retried mit Backoff
Token-Refresh-Backoff (max)	—	60s	Max. Wartezeit zwischen Token-Refresh-Versuchen
Initialer Token-Fetch	—	10 Versuche	Versuche beim App-Start, Token von Authority zu holen
Initialer Token-Backoff (max)	—	30s	Max. Wartezeit zwischen initialen Token-Fetch-Versuchen

6. Geänderte Dateien

Datei	Änderungen
`ExternalTaskWorkerProcess.ts`	Backoff-Counter-Bug behoben, start() abgesichert, Reconnect-Logik in wiederverwendbare Funktion extrahiert
`ExternalTaskAdapter.ts`	Token-Refresh robust gemacht (gibt nie auf), IPC-Kommunikation abgesichert, initialen Token-Fetch mit Retry versehen, Refresh-Zyklus-Recovery bei Worker-Restart

7. Fazit

Alle sechs identifizierten Schwachstellen im External Task Worker Mechanismus wurden behoben. Die Tests zeigen, dass die Worker sich jetzt zuverlässig von Verbindungsverlusten erholen — sowohl bei Engine-Ausfällen (mit und ohne Last) als auch bei Authority-Ausfällen. Das exponentielle Backoff funktioniert korrekt und reduziert die Last auf die Infrastruktur während eines Ausfalls.

Die Änderungen sind rückwärtskompatibel. Bestehende Konfigurationen funktionieren ohne Anpassung. Optional kann über die Umgebungsvariable `PROCESSCUBE_APP_SDK_ETW_RETRY` die Anzahl der Worker-Reconnect-Versuche angepasst werden.