# Cluster- und Cache-Betrieb

Diese Seite beschreibt, wie ein Webdesk mit mehreren Knoten (Cluster, "verteilter Betrieb") konfiguriert und betrieben wird und wie sich die Caches dabei verhalten. Sie richtet sich an Systemadministratoren und Consultants. Vorausgesetzt werden: alle Knoten greifen auf dieselbe Datenbank zu, ein erreichbarer Redis-Server und – bei Cache-Replikation über JMS – ein JMS-Broker. Die Schlüssel werden pro Knoten in der webdesk.properties gesetzt (siehe Seite "Konfigurationsdatei webdesk.properties und Umgebungsvariablen"). Ergänzend beschreiben Liste der Caches[1] den Zweck der Caches, Aktive Jobs[2] die Job-Überwachung inklusive Knotenstatus und Erweiterte Funktionen[3] die Wartungsaktionen.

## Überblick: Bausteine des Cluster-Betriebs

| Baustein | Aufgabe | Schalter |
|---|---|---|
| Cache-Replikation (EhCache) | Gleicht die EhCache-Caches zwischen den Knoten ab und transportiert Dienstaufrufe an die anderen Knoten (Speichern von Jobs und Job-Triggern, Übernahme von Systemparametern, Veröffentlichung des Clusterstatus). Transport wahlweise RMI (Multicast) oder JMS. | isDistributed, useJmsReplication |
| Redis | Gemeinsamer Speicher und Nachrichtenkanal (Pub/Sub) für die als "repliziert" definierten Anwendungs-Caches (Caffeine + Redis), für den Hibernate-Provider redis-caffeine und für die HTTP-Sessions (Spring Session). Die Redis-Anbindung wird aufgebaut, wenn isDistributed=true oder redis.useAsSessionStore=true gilt. | redis.* |
| Hibernate-Cache (Second-Level-Cache) | Cache für Datenbank-Entitäten. Anbieter global umschaltbar. | webdesk.hibernate.cache.provider |
| HTTP-Sessions | Sitzungen lokal im Speicher, in Redis (Spring Session) oder über den Tomcat-Session-Manager. | redis.useAsSessionStore, META-INF/context.xml |
| Jobs | Zuordnung der Job-Trigger zu einem Knoten. | PoOptions.clusterNodes, Trigger-Feld "Ausführen auf Clusternode" |

Bei isDistributed=false (Standard) ist jede Abstimmung zwischen Knoten abgeschaltet; die Caches arbeiten dann rein lokal. Ein gesetztes useJmsReplication wird ohne isDistributed=true ignoriert.

## Grundkonfiguration eines Knotens

| Eigenschaft | Erklärung | mögliche Wert(e) |
|---|---|---|
| isDistributed | Schaltet den verteilten Betrieb ein: Cache-Replikation, Redis-Anbindung für replizierte | true/false (Standard: false) |

| | | |
|---|---|---|
| | Caches, Cluster-Heartbeat und knotenspezifischer Workflow-Engine-Name. Muss auf allen Knoten gleich gesetzt sein. | |
| useJmsReplication | Bei true werden die EhCache-Caches über JMS repliziert, sonst über RMI mit Multicast-Erkennung. Nur wirksam bei isDistributed=true. Muss auf allen Knoten gleich gesetzt sein. | true/false (Standard: false, also RMI) |
| PoOptions.clusterNode | Name dieses Knotens; muss pro Knoten eindeutig sein. Ist kein Name gesetzt, zeigt der Clusterstatus LOCAL. Wird isDistributed=true per Umgebungsvariable oder JVM-Parameter gesetzt und weder dort noch in der webdesk.properties ein Knotenname angegeben, verwendet Webdesk beim Start die Umgebungsvariable HOSTNAME als Knotennamen. Ist isDistributed nur in der webdesk.properties gesetzt, findet diese Vorbelegung nicht statt. | Text, z.B. node1 |
| PoOptions.clusterNodes | Liste aller Knotennamen des Clusters. Wird in der Maske "Systemparameter" (Bean PoOptions, Eigenschaft clusterNodes) gepflegt und gilt für alle Knoten. Die Einträge müssen exakt den PoOptions.clusterNode-Namen der Knoten entsprechen: Der Clusterstatus erwartet von jedem Eintrag eine Statusmeldung, der Name des aktuellen Knotens wird immer zusätzlich berücksichtigt. Das Job-Feld "Ausführen auf Clusternode" bietet die Liste nur an, wenn sie mehr als einen Eintrag enthält. | Liste von Namen (Standard: ClusterNode1) |
| WfStartup.engineName | Name der Workflow-Engine des Knotens. Wird der Standardname bei isDistributed=true nicht geändert, bildet Webdesk den Namen aus Standardname und Knotenname (webdeskshark_<Knotenname>); im Normalfall muss der Schlüssel daher nicht gesetzt werden. | Text (Standard: webdeskshark) |

| | | |
|---|---|---|
| webdesk.cluster.heartbeatIntervalSeconds | Intervall in Sekunden, in dem ein Knoten seinen Clusterstatus und seinen Job-Überwachungsstand an die anderen Knoten veröffentlicht (auch sofort beim Start). Läuft nur bei isDistributed=true. 0 schaltet den Heartbeat ab, ein negativer Wert verwendet den Standard. | Zahl (Standard: 30) |
| webdesk.cluster.skipIdleJobMonitoringSnapshots | Bei true veröffentlicht ein Knoten ohne Jobs und mit gestopptem Scheduler im Heartbeat keinen Job-Überwachungsstand. | true/false (Standard: true) |

**Hinweis:** Die Schlüssel isDistributed, useJmsReplication, PoOptions.clusterNode, WfStartup.engineName, webdesk.cacheReplication.RMI.peerProvider.properties und webdesk.cacheReplication.JMS.peerProvider.properties können auch als JVM-Parameter oder Umgebungsvariable gesetzt werden. Der Name der Umgebungsvariable lautet WD_ plus Schlüssel in Großbuchstaben, Punkte werden zu Unterstrichen (z.B. WD_ISDISTRIBUTED, WD_POOPTIONS_CLUSTERNODE). Leere Umgebungsvariablen werden ignoriert.

## Cache-Replikation: RMI oder JMS

Der Transport gilt global für alle replizierten EhCache-Caches und wird über useJmsReplication gewählt. Zusätzlich nutzt Webdesk die Replikation, um Dienstaufrufe an die anderen Knoten zu senden. Das betrifft das Speichern von Jobs und Job-Triggern, die Übernahme von Systemparametern (Neu-Injektion der Bean-Eigenschaften) und die Veröffentlichung des Clusterstatus. Ist die Replikation gestört, erreichen diese Änderungen die anderen Knoten nicht.

| Eigenschaft | Erklärung | mögliche Wert(e) |
|---|---|---|
| webdesk.cacheReplication.RMI.peerProvider.properties | Provider-Properties bei RMI. Standard ist die automatische Erkennung über Multicast. timeToLive legt die Reichweite fest: 0 gleicher Host, 1 gleiches Subnetz, 32 gleicher Standort, 64 gleiche Region, 128 gleicher Kontinent, 255 unbeschränkt. Die Knoten müssen sich also in derselben Multicast-Gruppe befinden und Multicast muss im Netzwerk erlaubt sein. | Standard: peerDiscovery=automatic,multicastGroupAddres |
| webdesk.cacheReplication.peerListenerProperties | Einstellungen des RMI-Listeners (nur bei RMI). | Standard: socketTimeoutMillis=120000 |
| webdesk.cacheReplication.JMS.peerProvider.properties | Verbindungsparameter JMS-Broker (nur bei JMS): Adresse des Brokers sowie Benutzer und Passwort eines gültigen Broker-Zugangs. Webdesk startet keinen eigenen Broker. Die Namen von Topic und Queue sind fest vorgegeben und nicht änderbar. Der Beispielwert in der Vorlage enthält Standard- | Vorlage: providerURL=tcp://localhost:61616/,userName=<user>,password=< |

| | | |
|---|---|---|
| | Zugangsdaten, die durch die echten Zugangsdaten zu ersetzen sind. | |
| webdesk.cacheReplication.excludedCaches | Komma-getrennte Liste von Cache-Namen, die nicht repliziert werden sollen. Gilt für die in ehcache.xml-Dateien definierten Caches und, im Hibernate-Modus ehcache2, für Hibernate-Regionen. | Liste von Cache-Namen (Standard: leer) |
| webdesk.cacheReplication.listener.replicatePuts, .replicateUpdates, .replicateUpdatesOnly, .replicateRemovals, .replic | Wie auch bei replizierten Caches | true/false (Standard: true) |
| webdesk.cacheReplication.maxElementsInMemory, .timeToIdleSeconds, .timeToLiveSeconds, .overflowToDisk | Vorgabewerte für die Caches ohne eigene Definition. | Standard: 100 / 300 / 300 / false / true |

**Achtung:** webdesk.cacheReplication.listener.replicatePuts=false global zu setzen ist nicht zulässig. Der Versand von Dienstaufrufen zwischen den Knoten beruht darauf, dass das Ablegen eines Eintrags im internen Kommunikations-Cache repliziert wird; mit replicatePuts=false würden Job-Änderungen, Systemparameter und Clusterstatus nicht mehr bei den anderen Knoten ankommen.

**Achtung:** Der ältere Schlüssel webdesk.cacheReplication.peerProvider.properties (ohne RMI) wird nicht mehr ausgewertet. Mit ihm gesetzte Multicast-Adressen oder -Ports gelten nicht; es greift der Standard (230.0.0.1, Port 4446).

Die Vorgabewerte für Caches können zusätzlich durch eine Datei defaultCacheConfiguration.properties im Klassenpfad (WEB-INF/classes) übersteuert werden; sie hat Vorrang vor der webdesk.properties. Dass overflowToDisk standardmäßig aktiv ist, kann bei großen Caches zu höheren Antwortzeiten führen, weil Einträge dann von der Platte gelesen werden; Caches mit eigener Bean-Definition können dies mit der Bean-Eigenschaft overflowToDisk abschalten (Beispiel: PtmPermissionCacheBean.overflowToDisk=false in der webdesk.properties). Die Überschreibung per <Bean>.<Eigenschaft> wirkt nur auf Spring-Cache-Beans, nicht auf die Regionen der ehcache.xml-Dateien für den Hibernate-Cache.

## Redis

Redis wird für Sessions (Spring Session), für replizierte Anwendungs-Caches (Caffeine als lokale Ebene, Redis als gemeinsame Ebene, Ungültigkeitsmeldungen über den Pub/Sub-Kanal webdesk:cache-sync) und für den Hibernate-Provider redis-caffeine (Kanal webdesk:hibernate-cache-coherence) verwendet. Die Verbindung wird aufgebaut, sobald isDistributed=true oder redis.useAsSessionStore=true gilt. Ist weder redis.connection.url noch redis.clusterNodes gesetzt, verbindet sich Webdesk mit localhost Port 6379.

| Eigenschaft | Erklärung | mögliche Wert(e) |
|---|---|---|
| redis.useAsSessionStore | Bei true werden die HTTP-Sessions über Spring Session in Redis gehalten. | true/false (Standard: false) |
| redis.connection.url | Adresse eines einzelnen Redis-Servers. Übernommen werden Host, Port und Datenbank-Nummer; ein in der URL enthaltenes Passwort wird nur verwendet, wenn redis.connection.password leer ist. | z.B. redis://redis-host:6379/0 (Standard: leer, dann localhost:6379) |
| redis.clusterNodes | Komma-getrennte Liste der Knoten eines Redis-Clusters. Ist der Wert gesetzt, hat er Vorrang vor redis.connection.url; die | z.B. host1:6379,host2:6379 (Standard: leer) |

| | | |
|---|---|---|
| | Zeitüberschreitung für Redis-Befehle beträgt dann 2 Sekunden. | |
| redis.connection.password | Passwort für Redis. Nicht in Dokumentation, Tickets oder Logs weitergeben. | <your-redis-password> (Standard: leer) |
| redis.credentials.encrpted | Bei true ist redis.connection.password mit dem Webdesk-Verfahren verschlüsselt abgelegt. Der Schlüsselname ist im Produkt so geschrieben ("encrpted") und muss genau so verwendet werden. | true/false (Standard: false) |
| redis.session.cookie.jvmRoute | Route-Suffix des Session-Cookies JSESSIONID für Load Balancer, die über dieses Suffix routen (z.B. Apache mit mod_jk, Worker-Name wie worker2). Ist der Wert leer, liest Webdesk das Attribut jvmRoute des Engine-Elements aus der conf/server.xml des Tomcat. Der Wert ist nicht automatisch gleich PoOptions.clusterNode. Findet sich keine Route, wird im Log gewarnt und das Cookie ohne Suffix geschrieben. | Text (Standard: leer) |

Die redis.*-Schlüssel können auch als JVM-Parameter oder Umgebungsvariable gesetzt werden (z.B. WD_REDIS_USEASSESSIONSTORE, WD_REDIS_CONNECTION_URL); diese haben Vorrang vor der webdesk.properties.

## Hibernate-Cache (Second-Level-Cache)

Der Anbieter des Hibernate-Caches wird mit webdesk.hibernate.cache.provider in der webdesk.properties gewählt. Die Wahl gilt global und unabhängig von isDistributed. Die Variante redis-caffeine steht seit Webdesk 4.64 optional zur Verfügung.

| Wert | Wirkung |
|---|---|
| ehcache2 (Standard, auch bei leerem Wert) | EhCache-basierter Cache; bei isDistributed=true über RMI oder JMS repliziert. Die Regionen sind in den ehcache.xml-Dateien der Module definiert. |
| redis-caffeine | Redis als maßgeblicher gemeinsamer Speicher, ergänzt um einen lokalen Caffeine-Nahbereichs-Cache. Benötigt eine Redis-Anbindung, also isDistributed=true oder redis.useAsSessionStore=true; sonst scheitert der Start mit der Meldung, dass die Redis-Verbindung nicht verfügbar ist. |
| disabled | Schaltet Second-Level-Cache und Query-Cache ab. |

Ein anderer Wert verhindert den Start mit einer Fehlermeldung, die die erlaubten Werte nennt.

## Regeln für den Betrieb

- Alle Knoten einer Umgebung müssen denselben Wert verwenden.
- Eine Änderung des Werts erfordert einen Neustart der gesamten Umgebung; ein Mischbetrieb verschiedener Anbieter ist zu vermeiden.
- Der Redis-Server muss für redis-caffeine die Eviction-Richtlinie maxmemory-policy noeviction (oder ein gleichwertiges, dokumentiertes Verhalten bei verwaltetem Redis) haben. Webdesk prüft dies beim Start per CONFIG GET maxmemory-policy und bricht den Start bei anderen Richtlinien (z.B. allkeys-lru, volatile-lru) oder bei verweigertem CONFIG GET ab. Auch mit noeviction sollte der Speicherverbrauch von Redis überwacht werden, da ein voller Redis zum Ausfall führt.
- Lokale EhCache2-Cache-Dateien (Disk-Stand) dürfen beim Wechsel des Anbieters nicht weiterverwendet werden.

## Umstellung auf redis-caffeine

1. Wert für webdesk.hibernate.cache.provider für die gesamte Umgebung festlegen und ein Wartungsfenster planen, damit kein Mischbetrieb entsteht.
2. Prüfen, dass Redis mit maxmemory-policy noeviction arbeitet.
3. Die Redis-Schlüssel des Hibernate-Caches (Präfix wd:hibernate:l2:) vor dem Start löschen, da frühere Einträge nicht kompatibel sind. Beispiel für einen einzelnen Redis-Server (bei Redis Cluster pro Primärknoten ausführen):

```
redis-cli --scan --pattern 'wd:hibernate:l2:*' | xargs -r redis-cli del
```

4. Den Wert in der webdesk.properties aller Knoten setzen und die Knoten starten.
5. Im Log prüfen, dass keine dauerhafte Flut von Decodier-Warnungen zu Hibernate-Cache-Schlüsseln auftritt.

## Rückstellung oder Wechsel auf ehcache2 / disabled

1. Alle Knoten stoppen.
2. Bei bisherigem Modus redis-caffeine die Schlüssel wd:hibernate:l2:* löschen.
3. Den Wert in der webdesk.properties aller Knoten auf den Zielwert setzen.
4. Alle Knoten in einem Zug wieder starten.

## Feineinstellung des Nahbereichs-Caches (redis-caffeine)

Veränderbare Entitäten werden standardmäßig lokal zwischengespeichert; Redis bleibt maßgeblich. Wird ein Ereignis verpasst, kann ein Knoten einen Wert höchstens bis zum Ablauf von valueTtlMillis veraltet liefern. Die folgenden Schlüssel sind optional; die Standardwerte sind für den Normalbetrieb ausgelegt. Wo diese Schlüssel in einer Installation abgelegt werden, sollte mit dem Webdesk-Support abgestimmt werden.

| Eigenschaft | Erklärung | mögliche Wert(e) |
|---|---|---|
| wd.cache.hibernate.nearCache.valueTtlMillis | Gültigkeit eines lokal gelesenen Werts in Millisekunden. | Zahl (Standard: 3000) |
| wd.cache.hibernate.nearCache.coherenceStateTtlMillis | Dauer in Millisekunden, für die der lokal beobachtete Kohärenzzustand vorgehalten wird. | Zahl (Standard: 120000) |
| wd.cache.hibernate.nearCache.maxEntries | Maximale Anzahl lokaler Einträge. | Zahl (Standard: 20000) |
| wd.cache.hibernate.nearCache.reconnectStrategy | Verhalten nach einem Verbindungsabbruch zu Redis: | CLEAR (Standard) |

| | | |
|---|---|---|
| | lokale Werte werden verworfen. Die Werte LOAD und NONE werden abgelehnt. | |
| wd.cache.hibernate.nearCache.excludeMutableRegions | Komma-getrennte Liste von Regionen, die nie lokal gelesen werden dürfen (für besonders konsistenzkritische Regionen). | Liste (Standard: leer) |
| wd.cache.hibernate.nearCache.queryRegions | Komma-getrennte Liste von Query-Regionen, für die lokales Lesen erlaubt wird. | Liste (Standard: leer) |
| wd.cache.hibernate.region.entryTtlMillis | Lebensdauer eines Eintrags in Redis in Millisekunden. | Zahl (Standard: 3600000) |
| wd.cache.hibernate.readWrite.softLockTtlMillis | Lebensdauer der Soft-Locks in Redis in Millisekunden. | Zahl (Standard: 60000) |

Die Kennzahlen webdesk.hibernate.l2.redis.publish.failures, webdesk.hibernate.l2.redis.event.apply.failures und webdesk.hibernate.l2.redis.subscription.disconnects sollten im Normalbetrieb bei 0 bleiben, webdesk.hibernate.l2.redis.near.cache.disabled sollte 0 sein.

## HTTP-Sessions im Cluster

Welche Session-Art aktiv ist, zeigt die Maske "Clusterstatus" im Feld "Session-Mode" (siehe unten). Es gibt zwei voneinander unabhängige Einstellungen:

- **Spring Session mit Redis:** aktiv bei redis.useAsSessionStore=true (Standard: false). Das Session-Cookie bleibt JSESSIONID, optional mit Route-Suffix (redis.session.cookie.jvmRoute). Bei einem Knotenausfall findet der verbleibende Knoten die Session auch dann, wenn das Cookie noch das Suffix des ausgefallenen Knotens trägt. Bestimmte Session-Attribute werden bewusst nicht in Redis abgelegt, sondern bleiben auf dem Knoten, der sie angelegt hat (u.a. die Zustände von Cocoon-Formularen und Flowscripts, die Vaadin-Anwendung HR Expert, der angemeldete und der gewechselte Benutzer, der Skin). Der Load Balancer sollte Anfragen einer Session daher möglichst auf demselben Knoten halten.
- **Tomcat-Session-Manager (memcached-session-manager mit Redis):** wird ausschließlich über ein <Manager className="de.javakaffee.web.msm.MemcachedBackupSessionManager" .../>-Element in der META-INF/context.xml des Webdesk eingeschaltet. In der im WAR ausgelieferten context.xml ist dieses Element auskommentiert. Das Docker-Image des Webdesk bringt dagegen eine eigene context.xml mit, die in die Web-Anwendung kopiert wird und das Element aktiv enthält (mit memcachedNodes="redis://redis"); die Bibliotheken des Session-Managers sind im Image vorhanden. Die Maske "Clusterstatus" meldet "memcached-session-manager / Redis", sobald das Element in der context.xml gefunden wird, auch wenn zusätzlich redis.useAsSessionStore=true gesetzt ist.

**Achtung:** Vor der Inbetriebnahme muss pro Umgebung geklärt und im "Session-Mode" überprüft werden, welcher Mechanismus tatsächlich aktiv ist. Vor allem bei Docker-Betrieb ist die mitgelieferte context.xml zu beachten.

Für Formulare gibt es zusätzlich die Systemparameter enableFormStateClusterFailover (Standard: false) und useSessionToStoreFormState (Standard: true) der Bean PoOptions. Bei aktiviertem Failover wird der zuletzt gespeicherte Zustand bestimmter Formulare nach einem Knotenwechsel wiederhergestellt; mit useSessionToStoreFormState=true wird der Zustand in der Session (und damit in Redis) abgelegt, sonst in einem replizierten EhCache-Bereich.

## Jobs im Cluster

- Ob und mit wie vielen Threads auf einem Knoten Jobs ausgeführt werden, legen PoJobServiceTarget.active und noOfSchedulerThreads fest (siehe Systemparameter Portal & Organisation[4]).
- Arbeitet Webdesk als Cluster (isDistributed=true, mehr als ein Eintrag in PoOptions.clusterNodes und der eigene Knotenname ist enthalten), erscheint im Job-Editor bei den Triggern das Feld "Ausführen auf Clusternode". Ein Trigger wird nur auf dem Knoten eingeplant, dessen Name dort ausgewählt ist. Ein Trigger ohne Knotenzuordnung wird in dieser Konstellation auf keinem Knoten eingeplant (ausgenommen das einmalige Starten über "Job starten").
- Die Überwachung der Jobs aller Knoten beschreibt Aktive Jobs[5].

## Was nicht automatisch zwischen den Knoten abgeglichen wird

| Bereich | Verhalten und Maßnahme |
|---|---|
| MetaForm-Cache | Ist von der Replikation ausgenommen. Nach Änderungen an Formulardefinitionen, die nicht auf allen Knoten übernommen wurden, in der Maske "Erw. Funktionen" die Aktion "MetaForm-Cache auf allen Cluster-Knoten ungültig machen" ausführen. Sie leert den MetaForm-Cache auf dem aktuellen und auf den übrigen Knoten. |
| Weitere nicht replizierte Caches | Einige Caches sind bewusst lokal: unter anderem die Workflow-Prozessdefinitions-Caches (webdesk_processdefinitions, webdesk_processdefinitions2hide, webdesk_wfprocessdefinitiondao_methodcache), die CMIS-Sitzungen der Dokumentenverwaltung und der Cache at.workflow.webdesk.po.STATIC_NON_CLUSTERED. Änderungen daran wirken auf jedem Knoten nur lokal; siehe Erweiterte Funktionen[6] (Workflow-Cache aktualisieren) und Job refresh Cache[7]. |
| Lucene-Suchindex | Der Volltextindex liegt im lokalen Dateisystem jedes Knotens und wird beim Start des Knotens aktualisiert (Schlüssel webdesk.luceneIndexing, Standard: true). Die Aktionen "Lucene-Suchindex aktualisieren" und "Lucene-Suchindex bereinigen" in "Erw. Funktionen" wirken nur auf dem Knoten, auf dem sie ausgeführt werden. Der Unterordner des Index ist standardmäßig zufällig; mit dem JVM-Parameter webdesk.lucene.index.subdir kann er festgelegt werden. |
| Menüs, Berechtigungen und weitere Caches per Hand leeren | Die Aktionen zum Leeren in der Maske "Liste der Caches" und in "Erw. Funktionen" werden auf dem Knoten ausgeführt, auf dem sie aufgerufen werden. Eine gezielte Weitergabe an die anderen Knoten ist nur bei der MetaForm-Aktion vorgesehen; bei sonstigen Caches hängt die Wirkung auf andere Knoten davon ab, ob der Cache repliziert wird (Spalte "Distributed"). |

| | |
|---|---|
| Session-Attribute | Siehe Abschnitt "HTTP-Sessions im Cluster". |

## Maske "Liste der Caches" (po_showCaches)

Die Maske zeigt alle Caches des Knotens, auf dem die Anfrage landet. Mit dem Textfeld "Zeige nur Caches deren Namen folgenden Text beinhalten" kann die Liste gefiltert werden. Je Cache werden angezeigt: "Name", "Einträge", "Effizienz" (Trefferquote in Prozent), "Cache Typ", "Distributed" sowie die Konfiguration (maximale Einträge, Time-to-idle, Time-to-live, eternal). Mit der Option "Anzeige Speicherverbrauch" werden zusätzlich die Spalten "Speicher, in KB" und "Max. Speicher, in KB" sowie die Summen berechnet.

| Spalte "Distributed" | Bedeutung |
|---|---|
| Local | EhCache-Cache ohne Replikations-Listener: wird nicht zwischen Knoten abgeglichen. |
| Replicated | EhCache-Cache mit RMI-/JMS-Replikation oder Caffeine+Redis-Cache mit Abgleich über Redis. |
| Caffeine (Local) | Caffeine-Cache ohne Abgleich über Redis. |
| Terracotta-clustered / Unknown | EhCache-Cache mit Terracotta-Anbindung bzw. nicht erkannter Cache-Typ. |

Ist der verteilte Betrieb aktiv, zeigt die Maske zusätzlich den Text "Diese Instanz ist Teil eines Clusters und hat den Namen" mit dem Knotennamen und die Schaltfläche "Anzeige Replikationsmitglieder". Sie listet die der Replikation bekannten Partner (bei RMI URL und GUID, bei JMS den Namen); sind keine Partner bekannt, erscheint "No Caching Peers available". Mit "Leeren aller Caches" bzw. dem Symbol "Cache leeren" je Zeile werden Caches des aktuellen Knotens geleert (nach dem Leeren aller Caches ist der Workflow-Cache zu aktualisieren, siehe Liste der Caches[8]).

## Maske "Clusterstatus" (po_showClusterStatus)

Beim Öffnen veröffentlicht der Knoten seinen eigenen Status und wartet bis zu 2,5 Sekunden darauf, dass die Meldungen der konfigurierten Knoten eintreffen. Erscheinen danach konfigurierte Knoten weiterhin als fehlend, zeigt die Maske den Hinweis "Die Cluster-Replikation ist noch im Aufbau." mit den Namen der "Konfigurierte Knoten noch nicht sichtbar". Im Normalbetrieb senden alle Knoten ihren Status im Heartbeat (Standard: alle 30 Sekunden).

Die Kopfzeile zeigt "Clusterübersicht" (antwortende Knoten / konfigurierte Knoten), die Anzahl "Veraltet" und "Fehlt" sowie die Topologie:

- "Cluster-Modus": "Verteilt" oder "Einzelserver".
- "Cache-Replikations-Topologie": "RMI", "JMS" oder "Keine".
- "Session-Mode": "Spring Session / Redis", "memcached-session-manager / Redis", "Lokal im Speicher" oder "Unbekannt".

Die Tabelle listet je Knoten "Cluster-Knoten", "Status", "Job-Service" (gestartet oder gestoppt), "Threads", "Aktualisiert am", "Aktive Sessions", "Aktive Jobs", "Laufende Jobs", Speicher (frei, belegt, maximal) sowie "Version", "Revision" und "Branch". Abweichende Versionen zwischen den Knoten sind in den Spalten "Version", "Revision" und "Branch" sofort erkennbar. Die Werte in der Spalte "Status":

| Status | Bedeutung |
|---|---|
| CURRENT | Der Knoten, dessen Maske gerade angezeigt wird. |
| RESPONDING | Konfigurierter Knoten, dessen letzte Statusmeldung nicht älter als 180 Sekunden ist. |

| | |
|---|---|
| STALE | Die letzte Statusmeldung ist älter als 180 Sekunden (mögliche Ursachen: Knoten ausgefallen, Heartbeat abgeschaltet oder Replikation unterbrochen). |
| MISSING | Der Knoten steht in PoOptions.clusterNodes, es liegt aber keine Statusmeldung vor: Der Knoten hat noch nie gemeldet (nicht gestartet, anderer Name, Replikation funktioniert nicht) oder seine letzte Meldung ist verfallen (Statusmeldungen werden nach 300 Sekunden aus dem Cache entfernt, ein ausgefallener Knoten wechselt also von STALE nach MISSING). |
| OBSERVED_ONLY | Der Knoten sendet Status, steht aber nicht in PoOptions.clusterNodes (z.B. Schreibweise des Namens oder fehlender Eintrag). |

## Typische Symptome und Prüfungen

| Symptom | Prüfung |
|---|---|
| Der Eintrag "ClusterNode1" oder ein Knoten steht dauerhaft auf MISSING, obwohl alle Knoten laufen | PoOptions.clusterNodes enthält noch den Standardwert oder Namen, die von PoOptions.clusterNode abweichen. Namen auf allen Knoten und in der Liste abgleichen. |
| Andere Knoten sind nicht sichtbar (MISSING/ STALE), Topologie zeigt "Keine" | isDistributed=true auf allen Knoten und Neustart? Bei RMI: gleiche Multicast-Gruppe und Port, Multicast im Netz erlaubt, timeToLive passend. Bei JMS: Broker erreichbar, Zugangsdaten gültig, auf allen Knoten useJmsReplication=true. In der Maske "Liste der Caches" die Replikationsmitglieder prüfen. |
| Änderung an einem Job oder Systemparameter wirkt nur auf einem Knoten | Die Weitergabe läuft über die Cache-Replikation. Clusterstatus und Replikationsmitglieder prüfen; replicatePuts darf nicht auf false stehen. |
| Ein Job läuft auf keinem Knoten | Trigger ohne Eintrag im Feld "Ausführen auf Clusternode" werden im Cluster nicht eingeplant; Knotenstatus in Aktive Jobs[9] prüfen. |
| Formulare oder Masken zeigen auf einem Knoten alte Felder | MetaForm-Cache per Aktion auf allen Knoten ungültig machen. |
| Start bricht mit "Unsafe Redis maxmemory-policy ..." ab | Redis auf maxmemory-policy noeviction stellen (nur bei redis-caffeine). |
| Start bricht ab: "Unsupported value ... webdesk.hibernate.cache.provider" | Erlaubt sind ehcache2, redis-caffeine, disabled. |
| Nach dem Wechsel zu einem anderen Knoten erscheint die Anmeldeseite | "Session-Mode" im Clusterstatus prüfen. Bei Spring Session: gleiche Redis-Verbindung auf allen Knoten, redis.session.cookie.jvmRoute passend zur Route des Load Balancers (Warnung im Log, wenn keine Route gefunden wird). |

| | |
|---|---|
| Veraltete Daten nach Änderungen auf einem anderen Knoten (redis-caffeine) | Kennzahlen zu Publish-, Apply- und Verbindungsfehlern prüfen; gegebenenfalls Region in excludedMutableRegions aufnehmen. |

## Checkliste für die Inbetriebnahme

1. Pro Knoten einen eindeutigen PoOptions.clusterNode festlegen (WfStartup.engineName bleibt im Normalfall auf dem Standard), isDistributed=true und – falls gewünscht – useJmsReplication=true mit Broker-Zugang setzen.
2. Netzwerk prüfen: bei RMI Multicast (Standard 230.0.0.1:4446) zwischen den Knoten, bei JMS Zugriff auf den Broker, außerdem Zugriff aller Knoten auf Redis.
3. Redis-Verbindung (redis.connection.url oder redis.clusterNodes) auf allen Knoten setzen; Passwort nur aus einer geschützten Quelle.
4. Entscheiden, welche Session-Art verwendet wird (redis.useAsSessionStore oder Tomcat-Session-Manager) und Route des Load Balancers abstimmen (redis.session.cookie.jvmRoute).
5. Hibernate-Anbieter einheitlich festlegen (webdesk.hibernate.cache.provider); bei redis-caffeine Redis-Richtlinie noeviction und das Löschen der Schlüssel wd:hibernate:l2:* berücksichtigen.
6. Knoten starten; in der Maske "Systemparameter" die Liste PoOptions.clusterNodes mit allen Knotennamen füllen.
7. In der Maske "Clusterstatus" prüfen: alle Knoten RESPONDING bzw. CURRENT, Topologie wie geplant, einheitliche Versionen.
8. Jobs: für jeden Trigger den Knoten im Feld "Ausführen auf Clusternode" festlegen und in "Aktive Jobs" kontrollieren.

1. /daisy/webdesk-manual-en/admin/po-overview/3731-dsy/2662-dsy/2561-dsy.html
2. /daisy/webdesk-manual-en/admin/po-overview/3731-dsy/2662-dsy/1146-dsy.html
3. /daisy/webdesk-manual-en/admin/po-overview/3731-dsy/463-dsy.html
4. /daisy/webdesk-manual-en/admin/po-overview/3731-dsy/5122-dsy/5655-dsy.html
5. /daisy/webdesk-manual-en/admin/po-overview/3731-dsy/2662-dsy/1146-dsy.html
6. /daisy/webdesk-manual-en/admin/po-overview/3731-dsy/463-dsy.html
7. /daisy/webdesk-manual-en/admin/2674-dsy/3776-dsy/3779-dsy.html
8. /daisy/webdesk-manual-en/admin/po-overview/3731-dsy/2662-dsy/2561-dsy.html
9. /daisy/webdesk-manual-en/admin/po-overview/3731-dsy/2662-dsy/1146-dsy.html