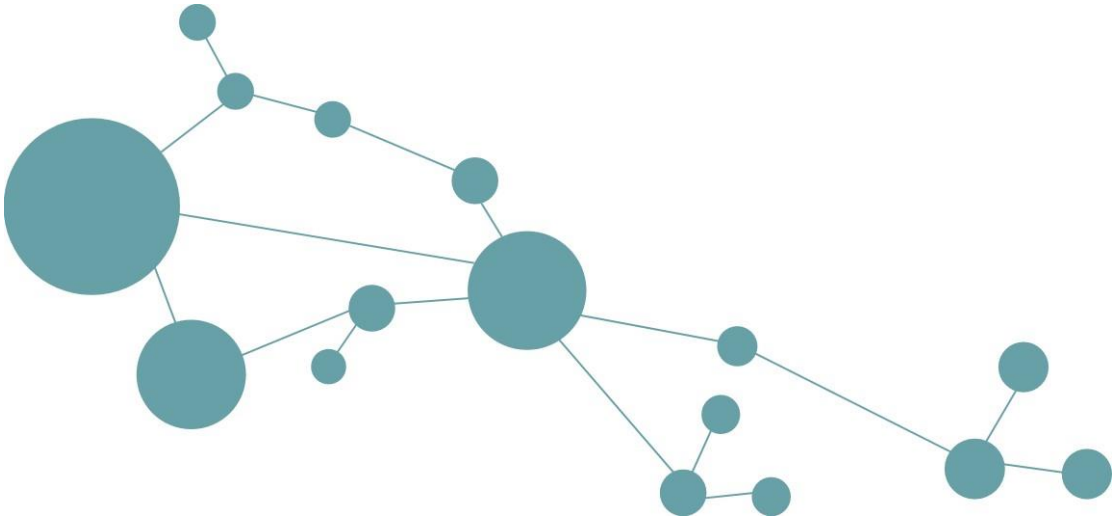

K-Infinity Server 5.4

i·views

Inhaltsverzeichnis

1 Überblick

Der K-Infinity Server sorgt für konsistente und persistente Datenhaltung und für die Aktualität der Daten auf den angeschlossenen K-Infinity Clients.

Die Datenhaltung erfolgt in einer objektorientierten Datenbank, die durch ein optimistisches Transaktionssystem kooperatives Arbeiten auf dem Wissensnetz ermöglicht.

In seiner Funktion als Kommunikationszentrale sorgt der K-Infinity Server für die Synchronisation von Clients und Services. Als Basismechanismus stellt er hierfür einen geteilten Objektraum und aktive Updates zur Verfügung.

Technische Daten:

- Multi-Platform Executable auf Basis der VisualWorks Smalltalk Virtual Machine (mediator.exe bzw. mediator.im).
- Konfigurierbarer TCP/IP Server-Port für die Kommunikation mit den Clients, Standard bei K-Infinity 3.2 ist es 30053.

2 Systemvoraussetzungen

Der K-Infinity Server ist Plattform-unabhängig und läuft auf allen gängigen Betriebssystemen, z. B. Windows, Solaris und Linux (ab Kernel 2.2).

OS	Version	Prozessor	Unterstützt	64 Bit VM
Windows	7,Vista, 2000, XP, 2003	x86	ja	nein
	2003	IA64	x86 Emulation	nein
Linux	RedHat EL 4, SLES9+, u.a.	x86	ja	ja
	RedHat EL 4, SLES9+, u.a.	IA64	x86 Emulation	nein
	RedHat EL 4, SLES9+, u.a.	PPC	ja	nein
Mac	OSX 10.4+	x86	ja	nein
	OSX 10.4+	PPC	ja	nein

3 Installation

Der K-Infinity Server benötigt prinzipiell keine spezielle Installation, d.h. er ist ad hoc aus einem beliebigen Verzeichnis startbar.

Es ist dabei darauf zu achten, dass die notwendigen Zugriffsrechte (lesen/schreiben/erzeugen) für das Arbeitsverzeichnis des Servers und alle Unterverzeichnisse gesetzt sind.

3.1 Startparameter

Dem Mediator Prozess können beim Start diverse Parameter mit übergeben werden. Die meisten Parameter können aber auch in der mediator.ini angegeben werden, sodass man den Mediator mit einer einfachen Kommandozeile starten kann. Dabei gilt, dass auf der Kommandozeile angegebene Parameter Präzedenz vor evtl. doppelt in der .ini-Datei angegebenen Parametern haben.

Die komplette Liste der möglichen Startparameter gibt der Mediator beim Aufruf mit dem Parameter "-?" aus.

`-port <num>`

Startet den Server mit der Portnummer <num>. Ohne diese Angabe wird Port 30009 verwendet.

`-password <passwd>`

Setzt das Passwort zum Stoppen des Servers auf <passwd>

`-clientTimeout <sec>`

Setzt die Zeit, innerhalb der sich ein Client automatisch melden muss, auf <sec> Sekunden. Der Wert sollte mindestens auf 600 gesetzt werden (was auch der Standardwert ist).

`-baseDirectory <directory>`

Setzt das Verzeichnis, in dem sich das Verzeichnis "volumes" befindet. Neben dem Unterverzeichnis "volumes" werden hier auch standardmäßig die Verzeichnisse für Backups und Downloads angelegt. Dieser Parameter hieß früher "-volumes".

Die folgenden Parameter stellen Kommandos an das Mediator Executable, um bestimmte Aufgaben auszuführen, ohne danach als Server für Wissensnetze zu fungieren.

`-quickRecover <volume>`
`-recover <volume>`

Falls der Mediator unordnungsgemäß beendet wird (z.B. Absturz des Rechners), bleiben in Volumes, die in Benutzung waren, Lock-Dateien stehen. Das Volume kann dann nicht mehr betreten werden. Um das Lock aufzuheben, kann man mit dem Aufruf von -quickRecover <volume> das Lock entfernen. Der Aufruf schlägt fehl, wenn (mögliche) Inkonsistenzen gefunden wurden. In diesem Fall muss der Startparameter -recover verwendet werden.

`-stop -password <passwd>`

Stoppt den Server auf 'localhost' mit den gegebenen Parametern (port und password)

`-stop -host <hostname> -password <passwd>`

Stoppt den Server auf hostname mit den gegebenen Parametern (port und password)

Achtung:

Das Arbeitsverzeichnis beim Aufruf muss hier das Verzeichnis sein, welches das „volumes"-

Verzeichnis enthält. Der „-volumes“-Parameter wirkt hier also nicht.

`-backup <volume> [-host hostname] [-port portnumber] [-password <passwd>]`

`-startBackup <volume> [-host hostname] [-port portnumber] [-password <passwd>]`

`-waitForBackup <volume> [-host hostname] [-port portnumber] [-password <passwd>]`

Da das Backup des BlockfileSystems asynchron erfolgt, wird mit -startBackup ein Backup gestartet, ohne auf dessen Beendigung gewartet werden muss. Mit -waitForBackup kann man nun im Falle eins laufendes Backups den Aufrufer blockieren, bis das Backup fertig ist. Um ein Backup zu Starten und die Beendigung abzuwarten, kann man über den Parameter -backup ein Backup starten.

`-bfscommand <volume> <command>`

Führt Kommandos aus, die vom BlockFileSystem erkannt werden.

Kommandozeilen-Parameter für das Logging:

`-nolog`

Schaltet Logging ab

`-loglevel <Integer>`

Konfiguriert, welche Meldungen im Log erscheinen sollen:

- 0: Alle Meldungen inklusive Debug-Ausgaben
- 10 (Standardwert): Alle Meldungen außer Debug-Ausgaben
- 20: Nur Warnungen und Fehlermeldungen
- 30: Nur Fehlermeldungen

`-logfile <Dateiname>, -log <Dateiname>`

Name der Log-Datei, die statt der Standard-Log-Datei verwendet wird. Dieser Parameter muss auf jeden Fall verändert werden, wenn mehrere Clients im selben Arbeitsverzeichnis gestartet werden sind.

`-debug`

Schaltet das Logging auf debug-mode

`-log <logname>`

Setzt die Logdatei auf <logname>.

3.2 Konfigurationsdatei "mediator.ini"

Einige Mediator-Einstellungen können auch in der Konfigurationsdatei mediator.ini festgelegt werden. Der Aufbau der Datei sieht folgendermaßen aus:

```
[Default]
parameterName1=parameterWert1
parameterName2=parameterWert2
...
```

Folgende Parameter sind an dieser Stelle einsetzbar:

`baseDirectory=<Verzeichnis>`

Setzt das Verzeichnis, in dem sich das Verzeichnis volumes befindet. Sollte dieser Wert auf volumes enden, so wird dieses Verzeichnis direkt verwendet, ohne noch zusätzlich darunter ein Verzeichnis volumes anzulegen.

`volumesDirectory=<Verzeichnis>`

In diesem Verzeichnis liegen die Wissensnetze. Als Standardwert ist an dieser Stelle 'volumes' eingetragen.

`backupDirectory=<Verzeichnis>`

Gibt das Verzeichnis an, in welches Wissensnetz-Backups geschrieben und zum Wiederherstellen auch gelesen werden. Nur vollständige Verzeichnisnamen erlaubt, keine Relativ-Angabe.

`networkBufferSize=<Größe in Bytes>`

Gibt die Größe des Puffers an, der für das Senden/Empfangen von Daten verwendet wird. Der Standardwert ist 20480. In manchen Infrastrukturen kann man durch Angabe von

networkBufferSize=4096

einen höheren Durchsatz erreichen.

`autoSaveMaxModifiedClusters=<Anzahl der Cluster>`

gibt an, ab welcher Anzahl gelandener Cluster automatisch gespeichert wird, da erst dann ein Auslagern möglich ist.

`autoSaveTimeInterval=<Warteintervall in Sekunden>`

gibt an in Sekunden, wie lang nach dem letzten Cluster-Speichern gewartet wird, bis wieder automatisch gespeichert wird. Dient der Einstellung des Speicherverhaltens, durch Angabe eines langen Intervalls kann dem Mediator zu häufiges Speichern ausgeredet werden.

`clientTimeout=<Timeout in Sekunden>`

gibt die Zeit in Sekunden an, die ein verbundener Client maximal keine Alive-Nachricht geschickt haben darf, bevor der Mediator ihn als inaktiv erachtet und ihn ausschliesst.

`skipVolumesCheck=<true|false>`

gibt an, ob die normalerweise nach dem Start des Mediators durchgeführte Überprüfung der vorhandenen Volumes ausgelassen wird

Logging-Einstellungen:

`loglevel = <Integer>`

Konfiguriert, welche Meldungen im Log erscheinen sollen:

- 0: Alle Meldungen inclusive Debug-Ausgaben
- 10 (Standardwert): Alle Meldungen ausser Debug-Ausgaben
- 20: Nur Warnungen und Fehlermeldungen
- 30: Nur Fehlermeldungen

`debug = true/false`

Setzt den Log-Level bei true auf 0, bei false auf 10

`channels = <Channel1> [,<Channel2>,...]`

Namen von Channelfiltern. Mit Hilfe der Channelfilter werden nur Log-Meldungen ausgegeben, die zu den angegebenen Channelfiltern gehören. Der Name eines Channelfilters deutet darauf hin, zu welchem Themengebiet die Log-Ausgaben gehören. Welche Channelfilter möglich sind, erfährt man in der Kommandozeile mit Hilfe des Parameters -availableChannels.

`logfile = <Dateiname>`
`log = <Dateiname>`

Name der Log-Datei, die statt der Standard-Log-Datei verwendet wird.

`maxLogSize = <size>`

die maximale Grösse des Logfiles, ab der die alte Logdatei archiviert wird und eine neue angebrochen wird. Bei Werten kleiner als 1024 wird die Angabe als in MB verstanden

`notifiers = <Name1> [,<Name2>,...]`

Namen von Notifiern. Ein Notifier wird bei Log-Ausgaben ab einem bestimmten Level aktiviert (typischerweise bei Fehlermeldungen). Siehe Abschnitt „Notifier“.

`logprefix = <Prefix1> [, <Prefix2>,... ]`

Zusätzliche Daten, die bei jeder Log-Ausgabe hinzugefügt werden

- $timestamp$: Zeitstempel der Log-Ausgabe
- pid : Prozess-ID der Anwendung
- $proc$: ID des aktuellen Smalltalk-Threads
- $alloc$: belegter Speicher der VM (in Megabyte)
- $free$: Freier Speicher der VM (in Megabyte)
- $incGC$: Status inkrementelle GCs
- os : Information über OS
- cmd : Kommandozeile
- $build$: Build-Version
- $coast$: COAST-Version

Bei einem Prefix, der nicht in dieser Liste enthalten ist, wird der Prefix unverändert ausgegeben

Speicher-Einstellungen:

Die folgenden drei Parameter dienen zur Konfiguration der Speicherzuteilung und -nutzung. Erlaubt ist die Angabe von Werten entweder in Megabyte oder in tatsächlichen Byte, wobei die Annahme gilt, dass sich Werte kleiner als 1048576 auf Megabyte-Angaben beziehen.

`maxMemory=<Integer, in MB>`

Maximal erlaubte Hauptspeicherbelegung. Mininmal 50 MB, standardmäßig gesamter physikalisch vorhandener Hauptspeicher (unter Windows) bzw. 512 MB.

`growthRegimeUpperBound=<Integer, in MB>`

Hauptspeicherbelegung ab der verstärkt versucht wird, Speicher freizugeben. Standardmäßig 0.6 * maxMemory.

`freeMemoryBound=<Integer, in MB> [10]`

Falls belegter, aber nicht mehr benötigter Speicher diese Grenze überschreitet, wird er wieder freigegeben.

Mail-Notifier Einstellungen:

Log-Ausgaben können über Mail-Notifier zusätzlich zum Eintrag in die Log-Datei per Mail versendet werden. Sinnvollerweise wählt man einen nicht zu geringen Schwellwert (Log-Level = 30), ab dem eine Mail versendet wird.

Für jeden angegebenen Namen aus der Notifier-Liste muss eine Konfiguration angegeben werden, die im Abschnitt [Name] angegeben wird.

Folgende Angaben sind für alle Notifier gleich:

- class = <Klassenname> : Art des Notifiers (z. Zt. nur COAST.CoastMailNotifier)
- level = <Integer> : Log-Level, ab dem der Notifier aktiv wird

Spezifische Angaben für COAST.CoastMailNotifier;

- attachLog = <Boolean> : Wenn true wird ein Error-Log als Attachment an die Mail angehängt
- sender = <Mailadresse> : Absender der Mail
- recipient = <Mailadresse> : Empfänger der Mail
- smtpHost = <Hostname> : Mailserver
- smtpPort = <Integer> : Port des Mailservers. Standardwert = 25 / 465 (TLS/SSL)
- retries = <Integer> : Anzahl der Versuche, die Mail bei einem Fehlschlag nochmals zu versenden
- retryDelay = <Integer> : Wartezeit zwischen zwei fehlgeschlagenen Versuchen, eine Mail zu versenden
- username = <Nutzername> : optionaler Anmeldename, benötigt für Authentifizierung
- password = <Password> : optionales Passwort, benötigt für Authentifizierung
- tls = true/false : gesicherte Verbindung (TLS/SSL) verwenden. Standardwert = false. Falls true müssen username und password gesetzt sein.

TLS/SSL ist für Authentifizierung nicht zwingend erforderlich, aber dringend zu empfehlen. Die Kompatibilitätserweiterung STARTTLS wird nicht unterstützt.

Beispiel:

```
notifiers = errorMailNotifier

[errorMailNotifier]
class = COAST.CoastMailNotifier
```

```
level = 30
;Spezifische Angaben für Mail-Versand
attachLog = true
sender = sender@hostname.de
recipient = receiver@hostname.de
smtpHost = mailserver.hostname.de
retries = 3
retryDelay = 30
```

Die angegebene Konfiguration definiert einen Mail-Notifier, die alle Fehlermeldungen per Mail verschickt.

3.3 Installation als Windows-Dienst

Zur Installation als Windows-Dienst werden Administrator-Rechte benötigt.

`-installAsService <name>`

Installiert den Server als Dienst unter dem angegebenen Namen. Alle Startparameter außer *stop* werden dabei übernommen. Ein evtl. gleichnamiger Dienst wird zuvor deinstalliert. Das aktuelle Verzeichnis, aus dem die Installation als Dienst gestartet wird, wird als Arbeitsverzeichnis verwendet. Auf einem System können mehrere Server installiert werden, wenn Arbeitsverzeichnisse, Ports und Servicenamen sich nicht überschneiden. Nach der Installation empfiehlt sich eine Anpassung der Startart des Dienstes auf „automatisch“ und ein automatischer Neustart des Dienstes im Fehlerfall.

`-deinstallService <name>`

Enfernt den angegebenen Dienst. Die Dateien des Dienstes bleiben erhalten. Vorsicht: Es wird nicht überprüft ab der angegebene Dienst tatsächlich ein K-Infinity Server oder ein anderer Dienst ist.

Alternativ kann der Dienst auch mit dem Service Control-Tool von Microsoft deinstalliert werden:

`sc delete <name>`

Parameter, die nicht mehr zur Verfügung stehen:

`-installService <service name> <service runtime path>`

Bitte stattdessen den Parameter *-installAsService* verwenden.

3.4 Sicherheitskonzept des Mediators

Der K-Infinity Server ist eine generische Komponente, die nicht nur für K-Infinity verwendet werden kann. Daher kennt der Server keinerlei benutzerspezifische Berechtigungen. Dennoch ist es möglich, den Server so einzurichten, dass sich nur authentifizierte Anwender anmelden können und diese dann nur die Operationen ausführen dürfen, zu denen sie autorisiert sind.

Jede Anwendung (Client und Server) enthält ein RSA-Schlüsselpaar, das je ausgelieferter Anwendung eindeutig ist. Den öffentlichen Schlüssel kann man über die Information erhalten (KB: Menü „Werkzeuge“, „Info“, dann die Schaltfläche „RSA-Key kopieren“) bzw. für Konsolen-Anwendungen per Aufruf mit dem Parameter -showBuildID. Die hierdurch exportierte Build-Information enthält den öffentlichen RSA-Exponenten (rsa.e_1) und RSA-Modul (aufgeteilt auf mehrere rsa.n_x) sowie eine MD5 Prüfsumme dieser Informationen (buildID).

Beispiel einer Build-Information:

```
[buildID.90A1203EFB957A58C2268AD8FE3CC5A3]
rsa.n_1=93D516DF61395258AA21A91B33E8EE67
rsa.n_2=B07C6FC5023DBB18F2201CF723C8F5DD
rsa.n_3=78941FB7C10D20988FEDFC6BD02CF3B7
rsa.n_4=E4567751843C38F055ED791AA7505278
rsa.n_5=23D94BB9EAB2E23F21DBEAA3DD2D2776
rsa.n_6=CE8B81564645DA85C85E9A78BB6E6B41
rsa.n_7=28A646D4868C38E00AE4810601B1EE9F
rsa.n_8=4FF5C35F873E6ED4F65F0FE8B4B45307
rsa.e_1=010001
```

Möchte man nun, dass sich nur eine bestimmte Menge Client-Anwendungen mit dem Server verbinden kann, so muss man im Server die jeweiligen Abschnitte in die mediator.ini übertragen. Beim Verbindungsaufbau überträgt der Client seine buildID. Wenn der Mediator einen passenden Eintrag enthält, so wird er die Client-Authentizität prüfen. Andernfalls wird er eine Verbindung nur aufbauen, wenn es gar keine Einträge zu Build-Informationen in seiner Ini-Datei gibt. Somit kann beispielsweise verhindert werden, dass sich veraltete Client-Anwendungen oder modifizierte Client-Anwendungen mit dem Mediator verbinden.

Umgekehrt können auch in der Client-Anwendung entsprechende buildIDs für die Mediatoren in die jeweilige ini-Datei eingetragen werden, um eine Verbindung zu einem kompromittierten oder veralteten Server zu verhindern.

So kann man zum Beispiel eine Umgebung einrichten, in der nur mit der aktuellsten Software auf die Produktivdaten zugegriffen werden kann, aber auf die Server mit den Testdaten auch von einer Entwicklungsumgebung aus. Die Anwendersoftware wiederum kann nur auf den Produktivserver oder auf den Testserver zugreifen.

Konfiguriert man weder Server noch Client, so verhält sich die Installation wie in den Vorgängerversionen: Jede Anwendung kann sich mit jedem Server verbinden (sofern die Protokollversion übereinstimmt).

Seit der Version 5.4 des Servers benötigt man zum Durchführen administrativer Befehle das Server-Passwort als Parameter (als Kommando-Zeilen-Parameter oder über die Verwaltung per Administrationswerkzeug). Damit nicht alle Administratoren Kenntnis dieses Passwortes haben müssen, kann einzelnen Benutzern mit Adminrecht für einzelne Operationen das Passwort zugewiesen werden. Die jeweilige Operation kann der Anwender dann nach erfolgreicher Anmeldung im Admintool ohne Eingabe des Server-Passwortes ausführen.

4 Betrieb

4.1 Herunterfahren des Servers

Der K-Infinity Server lässt sich manuell durch Aufruf mit dem Parameter -stop beenden. Bei der Installation als Windows-Dienst muss der Server mit der Diensteverwaltung gestoppt

werden.

Unter UNIX sowie beim Betrieb als Windows-Dienst, wird der Server beim Herunterfahren des Betriebssystems ordnungsgemäß beendet.

Zum Ausführen dieses Befehls muss mittels Parameter -password das richtige Serverpasswort übermittelt werden.

4.2 Speicherung und Backup von Wissensnetzen

4.2.1 Verzeichnisstruktur

Das Basisverzeichnis des K-Infinity-Servers weist folgende Struktur auf:

```
volumes/
   wissensnetzName/
     wissensnetzName.cbf
     wissensnetzName.cdr
     wissensnetzName.cfl
     wissensnetzName.lock (wenn das Wissensnetz geöffnen ist)

backup/
  wissensnetzName/
    <zehnstellige Nummer>/
      wissensnetzName.cbf
      wissensnetzName.cdr
      wissensnetzName.cfl
```

4.2.2 Speicherung von Wissensnetzen

Wissensnetze werden im Dateisystem im Unterverzeichnis "volumes" des Basisverzeichnisses des K-Infinity-Servers abgelegt. In diesem Verzeichnis wird für jedes Wissensnetz ein Unterverzeichnis mit entsprechendem Namen angelegt. Eine Datei mit der Dateiendung '.lock' zeigt an, dass ein Wissensnetz gerade in Verwendung ist.

4.2.3 Backup von Wissensnetzen

Die Wissensnetzverzeichnisse dürfen auf keinen Fall direkt kopiert werden, so lange der K-Infinity-Server läuft. Zu diesem Zweck besitzt der Server einen Backup-Service, der einen konsistenten Stand des Wissensnetzes in einen Backup-Bereich kopiert. Dieser Backup-Bereich muss in regelmäßigen Abständen gesichert werden (z.B. im Rahmen einer allgemeinen Backup-Strategie).

Der Ort, an dem die Backups angelegt werden kann über den Eintrag

```
backupDirectory=<Verzeichnis>
```

in der Datei "mediator.ini" festgelegt werden. Ohne diese Angabe wird das Unterverzeichnis "backup" des Basisverzeichnisses verwendet.

Der Backup-Service des K-Infinity-Servers kann auf zwei Arten angestoßen werden:

1. Durch einen direkten Request an den Serverprozess (z.B. vom K-Infinity-Administrationstool

aus)

2. Durch Einträge in der Datei 'jobs.ini' im Arbeitsverzeichnis des K-Infinity-Servers. Diese Datei kann pro Wissensnetz eine Rubrik mit folgenden Einträgen enthalten:

```
;Uhrzeit, zu dem das Backup gestartet wird
backupTime=00:45

;Turnus in Tagen - hier täglich
backupInterval=1

;Wird dieser Parameter angegeben, so werden nur die letzten 5 Backups dieses Wissensnetzes im Arch
backupsToKeep=5
```

Der Eintrag 'backupInterval' ist optional.

4.3 Garbage Collection

Ohne Garbage Collection wächst das Wissensnetz kontinuierlich mit der der Verwendung. Folglich ist es sinnvoll, von Zeit zu Zeit eine Bereinigung (Garbage Collection) durchzuführen. Wie die Datensicherung kann die Garbage Collection jederzeit manuell (z.B. mit einem speziellen Administrationswerkzeug) oder automatisch gestartet werden.

Die Garbage Collection kann - je nach Netzgröße - viel Zeit und Arbeitsspeicher in Anspruch nehmen. Bei der Durchführung auf großen Netzen empfiehlt es die Garbage Collection ohne verbundene Klients (z.B. KBuilder und JobClients) und ohne weitere aktive Prozesse (z.B. Backup) zu starten.

4.3.1 Automatische Garbage Collection: Aufbau der Datei jobs.ini

Die automatische Garbage Collection wird durch einen Eintrag in der Datei 'jobs.ini' konfiguriert, z.B.

```
[volume1]
garbageCollectTime=00:55
garbageCollectInterval=7
```

Dieser Eintrag in der jobs.ini sorgt dafür, dass das Netz mit Namen "volume1" im Abstand von "7" Tagen jeweils um "00:55" Uhr garbage-collected wird. Für das Interval ist der Standardwert "1" (also täglich), der Zeitpunkt muss angegeben werden.

Bei der Angabe der Netznamen in eckigen Klammern ist die Verwendung der Platzhalter "*" und "?" erlaubt, Groß- und Kleinschrift wird ignoriert.

4.3.2 Manueller Start der Garbage Collection

Alternativ kann die Garbage Collection auch durch spezielle Aufrufparameter des K-Infinity-Servers gesteuert werden:

-startGC <volume> -host <hostname>	Startet die Garbage Collection auf dem Netz mit Namen <volume> auf einen ggfs. entfernten Mediator auf Rechner <hostname> (optional inkl. Portangabe).
-stopGC <volume> -host <hostname>	beendet eine ggfs. auf dem Mediator <hostname> laufende Garbage-Collection des Netzes mit dem Namen <volume>.
-infoGC <volume> -host <hostname>	Informiert über den aktuellen Stand der Garbage Collection.

Diese Kommandos werden mit Hilfe eines Mediator-Executables an einen anderen bereits laufenden Mediator übermittelt.

Als weitere Möglichkeit bietet sich das Starten der Garbage Collection über das Admin-Tool an.

Zum Ausführen dieser Befehle muss mittels Parameter -password das richtige Serverpasswort übermittelt werden.

4.4 Betrieb unter Unix

Unter UNIX reagiert der Server auf folgende Signale:

`SIGTERM/SIGHUP`

Beendet den Server

`SIGUSR2`

Der Server startet einen sofortigen Backup aller Wissensnetze, die in der jobs.ini-Datei für Backup spezifiziert sind (siehe auch Abschnitt über Backup).

4.5 Betrieb im Cluster

Der Mediator kann in einem Cluster betrieben werden. In einer Cluster-Umgebung wird i.d.R. eine laufende Spiegelung der Verzeichnisse und damit des Wissensnetzes vorgenommen. Fällt der Teil des Clusters, auf dem der Mediator läuft aus, wird automatisch ein neuer Mediator gestartet, der dann den Zugriff auf das Wissensnetz verwaltet.

Bei Ausfall des ersten Mediators kann es passieren, dass der Mediator keine Zeit mehr hat, das Wissensnetz in einen konsistenten Zustand zu bringen, das Netz damit eine Inkonsistenz aufweist und die "lock"-Datei des alten Mediators noch im entsprechenden Verzeichnis bestehen bleibt. Damit der neue Mediator in der Lage ist, die "lock"-Datei zu löschen, muss folgender Parameter in die mediator.ini hinzugefügt werden.

```
host=NameDesClusters
```

In diesem Fall können alle Mediatoren mit diesem ini-Eintrag auch gesperrte volumes anderer Mediatoren, die beim Start den selben Wert in der mediator.ini ausgelesen hatten, entsperren. "NameDesClusters" ist frei wählbar, muss aber den Regeln entsprechen, die für Hostnamen gelten (keine Leerzeichen, Doppelpunkte, o.ä.)

Eine Konsistenzprüfung des volumes läuft beim Starten des Mediators automatisch ab. Soweit möglich, wird das Wissensnetz in einen konsistenten Zustand versetzt und der Betrieb läuft normal weiter

4.6 Problembehebung

Falls der K-Infinity-Server während des Betriebs nicht ordnungsgemäß heruntergefahren wurde (z.B. Absturz des Rechners), bleiben bei geöffneten Wissensnetzen die Sperren bestehen. Beim Öffnen eines gesperrten Wissensnetzes wird diese Sperre erkannt und - falls möglich - entfernt.

Falls der Mediator eine Inkonsistenz erkennt, kann in der Kommandozeile durch den Aufruf des Mediators mit den Parametern -quickRecover / -recover das Wissensnetz geprüft und Inkonsistenzen soweit möglich repariert werden.

Sollte eine Auflösung der Inkonsistenzen wider Erwarten nicht möglich sein, muss auf eine Sicherungskopie zurückgegriffen werden.

4.7 Kommandos des BlockFileSystems

Mögliche Kommandos für das BlockFileSystem sind:

```
copyto {target volume}
```

Ein Volume wird umkopiert. Das neue Volume darf noch nicht existieren. Bei diesem Vorgang werden die Blöcke umstrukturiert, wodurch der Vorgang länger als eine reine Dateikopiervorgang dauert.

Zum Ausführen dieses Befehls muss mittels Parameter -password das richtige Serverpasswort übermittelt werden.