c7 – CLOUD SEVEN

Zentrale Parametrisierung

Mit Spaßfaktor

cook book

Dirk Goldbach
August 2013

Version 0.1

Inhaltsverzeichnis

Vorwort

Vor einigen Wochen stand wieder einmal das Thema Softwarekonfiguration auf meiner ToDo-Liste. Diesmal ging es um die seit langem überfällige Ersetzung einer dateiorientierten Implementierung im XML-Format durch eine Datenbanklösung.

Die Entscheidung dafür war bereits vor Jahren aus mehreren Gründen gefallen. Immer wieder standen die Techniker bei Softwareupdates vor dem Problem, dass sie einerseits die Konfigurationsdateien (wer gibt sich schon mit einer zufrieden?) des Kunden nicht austauschen können, ohne wichtige Einstellungen zu verlieren, andererseits aber hin und wieder neue Parameter dazu kommen. Blieb die aufwändige und fehleranfällige manuelle Prüfung mit Diff-Tools.

Der wesentliche Vorteil der zentralen Parametrisierung zeigt sich aber bei Ausfall eines Servers. Wie relaxed war kürzlich die Situation, als sich ein Kunde voller Panik meldete, weil eine Maschine mit zentraler Bedeutung über Nacht den Geist aufgegeben hatte aber sofort nach der Software-Installation auf einem anderen Rechner der Dienst wieder zur Verfügung stand, ohne dass jemand die Konfigurationsdateien aus dem Hut hätte zaubern müssen, denn die wären ja mit dem defekten Rechner gleich mit abgeraucht.

Angesichts der Vorteile, so fragte ich mich, ist es doch verwunderlich, dass es keine Standardlösung für die zentrale Parametrisierung verteilter Anwendungen gibt. Möglicherweise habe ich falsch recherchiert, denn fündig wurde ich nicht. Vielleicht liegt es an dem Umstand, dass die Installationen auf den verschiedenen Rechnern sich zwar die Konfiguration teilen können, der Komplexitätsgrad aber bei systembedingter Abweichung ansonster gleicher Parameter sprunghaft steigt. Letztendlich nicht unlösbar, aber offensichtlich hatte bislang keiner Lust auf eine generelle Lösung.

CLOUD SEVEN ist kein technischer Geniestreich. Es ist eine auf Basis realer Anforderungen konzipierte Lösung für den Entwickler- und Anwender-Hausgebrauch. Damit nicht nur Java-Programmierer in den Genuss von CLOUD SEVEN kommen, habe ich c7 (auch Software braucht einen Kosenamen) zusätzlich zur Java-API mit einer SOAP-Schnittstelle ausgestattet. Eine JMS-Schnittstelle ist geplant.

Und so ganz nebenbei ist mit c7 eine User Verwaltung mit frei vergebbaren Rechten entstanden.

Ein ganz wichtiger Aspekt bei der Erstellung von CLOUD SEVEN war für mich das einfache Handling und damit verbunden eine geringe Einarbeitungszeit. Ob mir das gelungen ist, können Sie nach der Lektüre dieses cook books – oder noch besser – nach Ausprobieren einer der Schnittstellen – sicher am besten beurteilen.

Einführung

Dieses cook book beschreibt die Möglichkeiten und die Anwendung von CLOUD SEVEN, die Inbetriebnahme der API, des SOAP-Servers und (zu gegebener Zeit) der JMS-Schnittstelle, es erläutert die 'intelligenten' Mechanismen für Instanzen und Mandanten und geht auf die Nutzung der API ein. Für SOAP und JMS existieren gesonderte Dokumente.

Ein eigenes Kapitel widmet sich der User Verwaltung, die einen wesentlich größeren Leistungsumfang angenommen hat, als ursprünglich geplant. Ebenso wird recht detailliert die Benutzung des integrierten Konfigurationseditors beschrieben.

Für exakte Informationen über die CLOUD SEVEN Java API existiert ein entsprechendes Javadoc: http://www.dirk-goldbach.de/c7/javadoc/index.html

Zur Vereinfachung (für den Autor) kann es vorkommen, dass im Text alternativ für CLOUD SEVEN die Kurzform c7 verwendet wird.

Konzept

Trotz des ganzen Drumherums bleibt das Thema ein simples: Parameter und deren Werte. CLOUD SEVEN dient der Verwaltung von Parametern. Der Unterschied zu herkömmlichen Konzepten, wie z.B. Key/Value Paaren ist, dass die Parameter in Form einer Struktur bzw. von Objekten repräsentiert werden.

Konfigurationsobjekte werden nicht in Dateien sondern in einer Datenbanktabelle abgelegt. Daraus lassen sich weitere Besonderheiten folgern:

- CLOUD SEVEN bedarf einer Datenbank
- die Konfiguration kann lokal oder zentral abgelegt werden
- mehrere völlig unterschiedliche Anwendungen können die selbe Konfigurationsquelle nutzen
- eine Anwendung kann parallel auf mehreren Rechnern ohne redundante Konfiguration installiert werden.

Für den sofortigen Einsatz oder auch für Testzwecke ist CLOUD SEVEN in der Lage, Derby embedded zu nutzen. Das bedeutet, dass sowohl das Framework als auch die Middleware ohne zusätzlichen Installations- oder Konfigurationsaufwand für die Datenbank lauffähig sind. Dieser Modus ist die Standardeinstellung von CLOUD SEVEN.

Die Konfigurationsobjekte werden *c7Property* genannt. Ein *c7*Property ist also ein Container für diverse Informationen, die Ergänzungen zum eigentlichen Konfigurationswert sind.

Im Sinne einer Allgemeingültigkeit und um einer möglichst breiten Anwendergruppe zur Verfügung zu stehen, bietet CLOUD SEVEN eine Java API, einen Webservice (SOAP) und später einen JMS Dienst (in Planung).

Wesentliche Aspekte des Konzepts sind der Instanz-Automatismus (Rechnerabhängigkeit) und die Möglichkeit der automatischen Unterscheidung von mandantenabhängigen Werten.

Hinzu kommt – ganz nebenbei – eine User Verwaltung mit Benutzern, Gruppen, Rechten und Aktionen, die den Anforderungen eigener Software angepasst werden können.

Zur Enttäuschung einer vermutlichen großen Anzahl potentieller Nutzer ist CLOUD SEVEN nicht als APP zu haben.

Das c7Property

Ein Parameter wird durch ein C7Property repräsentiert. C7Properties verfügen über diverse Attribute, die wichtigsten sind der Konfigurations-Key, der Konfigurations-Wert, bei verteilten Anwendungen der Instanz-Name und im Falle von abweichenden Werten in Abhängigkeit von Mandanten, der Mandanten-Name.

Keys

Die Identifizierung von Parametern erfolgt über Keys. Ein Key ist gewissermaßen der Name des Parameters. CLOUD SEVEN wertet die Keys case sensitiv aus, ebenso werden Sonderzeichen (wie z.B. Leerzeichen) als Bestandteil des Namens betrachtet. Wann immer auf einen Konfigurationswert zugegriffen werden muss, erfolgt dies über den Key.

CLOUD SEVEN erlaubt die mehrfache Vergabe identischer Keys. Wird über einen Key nach einem Wert gesucht, wird das erste passende Property zurück geliefert. Welches das ist, ist eher dem Zufall und dem der jeweils genutzten Datenbank beiwohnenden Hausgeist überlassen. Dieses Verhalten ist kein Bug sondern gewollt.

Auf die korrekte Schreibweise, auch Groß-Kleinschreibung, von Keys und Values ist zu achten.

Einfache Werte

Der Umgang mit einfachen Werten ist vergleichbar mit herkömmlichen Key/Value Konfigurationen. Ein Property könnte – ungeachtet weiterer, aber hier unbedeutender Informationen – diese Struktur haben:

```
confKey='user', confValue='admin'
```

So weit, so gut, nichts besonders Aufregendes.

Listen

Wie bereits erwähnt, können mehrere Properties den selben Key haben. Spätestens bei Verwendung der Listenfunktionalität wird der Grund dafür erkennbar.

Bsp.:

```
confKey='milestone', confValue='implementation'
```

```
confKey='milestone', confValue='test'
```

Die folgende Abfrage würde dementsprechend die verschiedenen Milestones auflisten. Zugegeben ein Beispiel, das nicht aus der Praxis stammt.

```
confKey='milestone'
```

Namespaces

Wie bereits erläutert, werden Parameter über deren Key identifiziert. Abhängig von der Anwendung kann es sinnvoll sein, mit 'Namespaces' zu arbeiten. Dazu wird der Key in mehrere Bestandteile gesplittet, die durch einen Punkt miteinander verbunden sind.

CLOUD SEVEN unterstützt den Umgang mit Namespaces durch Auflistung der Properties bei entsprechender Einschränkung des Schlüssels. Die Beispiele zeigen, wie dies genutzt werden könnte.

Bsp1.:

```
confKey='host.name.server1'
confKey='host.name.server2'
```

Mit einem solchen Konstrukt könnten alle Hostnamen durch folgende Abfrage ermittelt werden:

```
confKey='host.name'
```

Bsp2.:

```
confKey='server1.address'
confKey='server1.os'
```

Das folgende Konstrukt böte die Möglichkeit der Auflistung von Informationen über den server1.

```
confKey='server1'
```

Die Attribute des C7Property

Wie bereits erläutert, ist es zulässig, dass der selbe Konfigurations-Key mehreren Properties zugewiesen wird. Das ist insbesondere beim Handling von Namespaces von Interesse.

Attribut	Typ	Len[1]	Null[2]	Bedeutung

Id	Long		nein	Datenbank-ID. Wird von C7 automatisch vergeben. Darf nicht durch die Anwendung manipuliert werden. Kann mit Hilfe entspr. Methoden der Schnittstellen optional für den Zugriff auf ein Property genutzt werden.
ConfKey	String	120	nein	Der Konfigurations-Schlüssel. Sozusagen der Identifikator des Parameters. Muss nicht eindeutig sein.
ConfValue	String	200	nein	Der Wert des Parameters.
Instance	String	80	ja	Name des Rechners, auf dem die Software läuft.
ClientName	String		ja	Name eines Mandanten. Wenn dieser Wert nicht leer (null) ist, ist er Teil der Suchkriterien.
DependsTo	Long		ja	Referenz auf ein anderes 'übergeordnetes' C7Property. Damit können einfache Bäume aufgebaut werden.
Custom	Boolean		ja	Sagt aus, ob der Wert kundenspezifisch ist. Kann z.B. in Projekten genutzt werden, die vom Standard abweichen.
Description	String	200	ja	Kann eine Beschreibung der Bedeutung des Parameters sein.
Comment	String	200	ja	Kommentar zum Parameter oder zum Wert. Ganz nach Belieben.
Visible	Boolean		ja	Kann z.B. von einer GUI genutzt werden, um bestimmte Werte nicht darzustellen.
Crypted	Boolean		ja	Sagt aus, ob der Wert verschlüsselt ist. C7 legt die Werte verschlüsselt in der DB ab. Klartext wird per SOAP oder JMS nicht übermittelt. Wenn wahr, wird automatisch beim Schreiben ver- und beim Lesen entschlüsselt.
Persistent	Boolean		ja	Nicht persistente C7Properties werden nicht in der DB abgelegt und können nur zur Laufzeit genutzt werden.
Component	String	80	ja	Kann der Name einer Anwendung oder eines Teils einer Anwendung sein, zu der/dem das Property gehört.
InternalUse	Boolean		ja	Kann für interne Zwecke, z.B. die Ablauflogik einer Software, genutzt werden.
RuntimeType	String	20	ja	Hinweis auf die technische Bedeutung eines

				Properties. C7 nutzt die Typen 'INSTANCE' und 'CLASS'.
PropertyVersion	String	80	ja	Bei jeder Änderung eines C7Property wird dieser Wert inkrementiert. Nach erstmaligem Anlegen = 0
C7Version	String	80	ja	c7Version, unter der das Property erstmalig angelegt wurde. Wird von C7 ausgewertet.

[1] Longwerte sind hardware- und DB-abhängig. Boolean werden entweder als Boolean oder mittels mögl. kleiner Zahlenwerte in der DB repräsentiert.

[2] Zulässigkeit leerer Werte

Verteilte Anwendungen

Web-Anwendungen, die Forderung nach Lastverteilung und Verfügbarkeit oder auch Sicherheitskriterien führen vermehrt zur Entwicklung von Anwendungen, die gleichzeitig auf mehreren Systemen installiert und parallel betrieben werden.

Es liegt nahe, die Konfiguration solcher Programme nicht speziell für jeden Rechner zu erstellen und dort lokal abzulegen, sondern sie zentral vorzuhalten.

Was auf den ersten Blick recht simpel anmutet, offenbart bei genauerer Betrachtung ein kleines aber wichtiges Detail: nicht immer sind die Parameter für alle Systeme identisch. Typische Beispiele für abweichende Werte sind Verzeichnisse, Systembenutzer, IP-Adressen.

CLOUD SEVEN ermöglicht Konfigurationswerte mit gleichem Schlüssel aber abweichenden Werten in Abhängigkeit von dem Rechner, auf dem die Software läuft. Dazu nutzt CLOUD SEVEN ein spezielles Attribut der C7Properties.

Der Instanz-Automatismus

Geregelt wird die Abhängigkeit von Rechnern durch die sogenannte Instanz, die letztendlich dem Hostnamen entspricht. Zur Unterscheidung von instanzabhängigen und instanzunabhängigen Werten ist CLOUD SEVEN mit einem Automatismus ausgestattet, der beim Lesen der Konfigurationswerte erkennt, ob der Wert unabhängig von dem Rechner ist, auf dem die Software läuft, oder ob ein abweichender Wert zurückgeliefert werden muss.

Von außen betrachtet ist die Verwendung des Automatismus recht einfach. Das Handling beim Lesen von Properties ist in beiden Fällen das gleiche. Entscheidend für die effektive Nutzung ist, dass Properties mit rechnerabhängigen Werten mit dem entsprechenden Instanznamen geschrieben werden.

Zu beachten ist, dass die Art und Weise der Nutzung des Instanz- Automatismus abhängig von der verwendeten Schnittstelle (API, SOAP, JMS) geringfügig abweicht.

Mandantenabhängigkeit

Mandantentaugliche Software bringt besondere Herausforderungen mit sich. Bei gemeinsamer kostengünstiger Nutzung von Resourcen, wie z.B. Hardware, Datenbank oder Webservices durch voneinander unabhängige Anwendergruppen, müssen die (Un)Sichtbarkeit der Daten gesichert, Individuallösungen vertragsabhängig zur Verfügung gestellt und die Performance selbstverständlich hoch sein.

Dementsprechend können, ähnlich wie bei Nutzung unterschiedlicher Instanzen, Parameterwerte in Abhängigkeit von Mandanten abweichen. CLOUD SEVEN bietet Mechanismen, die die Verwaltung solcher Parameter mit ansonsten gleichem Schlüssel ermöglichen, sodass die Parametrisierung Ihrer Software weitestgehend unabhängig von Mandanten bleibt.

Voraussetzung für dieses Feature ist, dass Ihre Software die Mandanten durch einen eindeutigen Wert – in CLOUD SEVEN der sogenannte *clientName* – unterscheiden kann.

Inbetriebnahme

Voraussetzungen

Java: Java 7 (JDK für API, Runtime für SOAP und JMS)

Datenbank: Wahlweise Apache Derby, Oracle ab 10G, MySQL, PostgreSQL *

SOAP: CLOUD SEVEN bringt einen integrierten 'Standalone' Webservice mit.

JMS: Ist noch zu spezifizieren

Installation

- Laden Sie die Datei cloudseven_dist_xxx.jar herunter (xxx steht für die Versionsnummer).
- Legen Sie ein Installationsverzeichnis an.
- Kopieren Sie die Datei cloudseven.jar in dieses Verzeichnis.
- Entpacken Sie die o.g. Datei: jar -xvf cloudseven_dist_xxx.jar

Anschließend existieren in diesem Verzeichnis die Dateien CloudSeven.jar und c7.prop sowie ein Unterverzeichnis *lib* mit den notwendigen JDBC-Datenbanktreibern.

Für die Nutzung der API müssen die API selbst und die Treiber als Bibliothek in Ihre Anwendung eingebunden werden. Bei Bedarf können aktuelle Treiber per Download direkt vom Hersteller bezogen werden.

Derby kann ohne Installation als integraler Bestandteil von c7 'out of the box' genutzt werden. Das ist insbesondere dann sinnvoll, wenn die Software als JMS- oder SOAP-Service ihre Dienste tun soll.

Konfiguration

Habe ich es nicht gesagt? Keine Software ohne Konfiguration!

Auch C7 benötigt eine Basis-Konfiguration, die Informationen für das Laufzeitverhalten bereit hält. Dabei handelt es sich um eine typische Key/Value Datei.

Nachdem Sie ClOUD SEVEN installiert haben (s.o.) liegt dort eine Beispieldatei (c7.prop). Diese Datei dient als Vorlage und kann dementsprechend modifiziert werden.

Wichtig: Keys und Values sind – wie bei C7Properties übrigens auch – case sensitiv. Auf die korrekte Schreibweise ist also unbedingt zu achten.

CLOUD SEVEN benötigt lesenden Zugriff auf das Properties File. Wo es liegt, teilen Sie c7 bei Programmstart (SOAP, Jms) oder in der Initialisierungsphase (API) mit.

Kommentare in der Datei sollten mit einem vorangestellten hash # gekennzeichnet sein.

Keys und deren Bedeutung

Key	Bedeutung
connectionType	Typ der Anbindung von CLOUD SEVEN an eine Anwendung (also API, SOAP oder JMS).
databaseType	Das Datenbanksystem, in dem C7 Daten verwaltet. Aktuell werden Oracle, MySQL und PostgreSQL unterstützt.
databaseName	Name der Datenbank (identisch mit einem DB-Schema), in der die C7 Daten liegen.
databaseSchemaName	Name des Datenbankschemas, in dem die C7 Daten liegen.
oracleSid	Die Oracle SID identifiziert die installierte Datenbank. Wird ausschließlich für den Zugriff auf Oracle-Datenbanken benötigt.
databaseUser	Name des Datenbankbenutzers, der Zugriff auf die C7 Daten haben muss.
databasePassword	Passwort des Datenbankbenutzers.
databasePort	Port des DB Listeners.
databaseHost	Rechnername bzw. IP des DB-Servers.
soapPortUsage	Port über den die SOAP-Schnittstelle erreicht wird. Speziell für die Standard-Services.
soapPortAdmin	Port über den die SOAP-Schnittstelle erreicht wird. Speziell für die Administrations-Services.

Mögliche Werte für die Keys

connectionType

Value (default ist fett)	Bedeutung
CONNECTION_TYPE_API	Nutzung von CLOUD SEVEN als Java-Framework

	über die API
CONNECTION_TYPE_SOAP	Nutzung durch Zugriff über die SOAP-Schnittstelle
CONNECTION_TYPE_JMS	Nutzung durch Austausch von Nachrichten über einen Broker.

databaseType

Value (default ist fett)	Bedeutung
DATABASE_TYPE_MYSQL	Wird verwendet, falls MySQL die genutzte Datenbank ist.
DATABASE_TYPE_ORACLE	Wird verwendet, falls Oracledie genutzte Datenbank ist.
DATABASE_TYPE_POSTGRESQL	Wird verwendet, falls PostgreSQL die genutzte Datenbank ist.

databaseName

Value (default ist fett)	Bedeutung
CLOUD_SEVEN	String, vom DB-Administrator festgelegt.

databaseSchemaName

Value (default ist fett)	Bedeutung
CLOUD_SEVEN	String, vom DB-Administrator festgelegt.

oracleSid

Value (default ist fett)	Bedeutung
CLOUD_SEVEN	String, vom DB-Administrator festgelegt.

databaseUser

Value (default ist fett)	Bedeutung
C7	String, vom DB-Administrator festgelegt.

databasePassword

Value (default ist fett)	Bedeutung
+-cloudseven-+	String, vom DB-Administrator festgelegt.

databasePort

Value (default ist fett)	Bedeutung
CLOUD_SEVEN	Numerisch, vom DB-Administrator festgelegt.

databaseHost

Value (default ist fett)	Bedeutung
localhost	String, Rechnername, vom DB-Administrator festgelegt.

soapPortUsage

Value (default ist fett)	Bedeutung
4434	Numerisch, vom Administrator festgelegt. Kann mit soapPortAdmin identisch sein.

soapPortAdmin

Value (default ist fett)	Bedeutung
4444	Numerisch, vom Administrator festgelegt. Kann mit soapPortUsage identisch sein.

Setup

Nachdem Sie eine Datenbank mit Schema und User angelegt und das Properties File editiert haben, muss CLOUD SEVEN einmalig eingerichtet werden. Dieser Prozess umfasst das Anlegen von DB-Tabellen sowie das Eintragen von Standardwerten, wie z.B. der Rechte, Aktionen und des Users Admin.

Nach dem Setup steht Ihnen der Benutzer Admin mit allen notwendigen Rechten zur Verfügung: user = admin, password = +-easygoing+-

Das Setup ist ebenfalls eine Anwendung, die Sie entweder über das Properties File aktivieren oder – einfacher – deren Start sie direkt durch den entsprechenden Schalter auf der Konsole erzwingen.

Variante a): Aktivierung durch das Properties File

```
# Das SETUP ist eine Anwendung unter CLOUD SEVEN
application=setup
java -jar CloudSeven.jar --propertiesfile=c7.prop
```

Variante b): Überschreiben des Konfiugrationseintrags für die zu startende Anwendung durch Shellparameter

```
java -jar CloudSeven.jar --propertiesfile=c7.prop --application=setup
```

Nach erfolgreichem Setup kann dies nur wiederholt werden, wenn die DB-Tabellen zuvor gelöscht wurden. Ansonsten beschwert sich das Setup mit Fehlermeldungen darüber, dass die Tabellen bereits existieren.

Optionen

Das Verhalten von CLOUD SEVEN kann durch zusätzliche Optionen beeinflusst werden. Die Optionen können im Properties File gesetzt werden oder auch per API. Die Optionen werden ebenfalls als Key/Value-Paare gehandhabt.

Keys und deren Bedeutung

Key	Bedeutung
logLevel	Steuert die 'Sensibilität' des Loggings.
logDb	De-/Aktiviert das zusätzliche Logging in der Datenbank
logFile	Optionaler Name des LogFiles

Key	Bedeutung
loggingActive	De-/Aktiviert das Logging
synchronizerEntryMaxAge	Angabe, wie alt ein Synchronisationseintrag werden darf
synchronizerActive	De-/Aktiviert die Synchronisation mehrerer verteilter Caches
synchronizerClearTableInterval	Regelt den Zyklus, in dem die Synchronisationstabelle aktualisiert wird
synchronizerInterval	Zeitabstände, in der die Synchronisation im Hintergrund abläuft.
synchronizerClearCacheInterval	Abstände, in denen der lokale Cache auf veraltete Einträge überprüft wird.
cacheType	Typ des Cache, je nach Zuverlässigkeit der DB-Verbindung oder der Anwendung selbst
cacheActive	De-/Aktiviert das lokale Caching
garbageActive	De-/Aktiviert das Bereinigen des lokalen Caches
cacheGarbageInterval	Regelt die Zeitabstände der Cache-Bereinigung
cacheMaxGarbageTimeNoAccess	Gibt an, wie alt ein Eintrag im Cache ohne Zugriff werden darf
cacheMaxSize	Ist die max. Anzahl Einträge im Cache
soapSessionTimeOut	Dauer der Verfügbarkeit einer Session eines Users

Mögliche Werte für die Keys

logLevel

Value (default ist fett)	Bedeutung
DEBUG	Debug-Einträge sind für die detaillierte Fehlersuche gedacht. Die Anzahl der Log-Einträge kann sehr hoch sein.

	Dieses Level schließt die folgenden Level mit ein, d.h. INFO, WARNING und ERROR Meldungen werden ebenfalls geloggt.
INFO	Info-Einträge dokumentieren die Aktionen der Software. Das können z.B. die Schritte beim Laden der Software sein oder zeitgesteuerte Aktionen, die u.a. zum Cache-Garbage gehören. Dieses Level schließt die folgenden Level mit ein, d.h. WARNING und ERROR Meldungen werden ebenfalls geloggt.
WARNING	Warning-Einträge sind Hinweise auf nicht erwartete Zustände oder Probleme, die aber nicht zwangsläufig zu falschen Daten führen müssen. Dieses Level schließt Meldungen des Levels ERROR mit ein.
ERROR	Error-Einträge zeigen ernst zu nehmende Fehler. Solche Fehler führen zu Störungen der Software und können im schlechtesten Fall auch zu Datenfehlern führen.

logDb

Value (default ist fett)	Bedeutung
true	Aktiviert das zusätzliche Logging in die Datenbank. Dient der Information über das Laufzeitverhalten der Software. Detaillierte Fehlerinformationen, wie z.B. Call-Stacks, werden nicht geloggt.
false	Deaktiviert das DB-Logging. Unabhängig davon kann das Logging in die Datei fortgeführt werden.

logFile

Value (default ist fett)	Bedeutung
Dateiname	Vollständiger Dateiname incl. Pfad. Bei Windows-Systemen muss der Backslash (\) durch einen

	zweiten Backslash escaped werden (\\). Relative Pfadangaben sind möglich.

loggingActive

Value (default ist fett)	Bedeutung
true	Aktiviert das Logging. Dient der Information über das Laufzeitverhalten der Software.
false	Deaktiviert das Logging.

synchronizerEntryMaxAge

Value (default ist fett)	Bedeutung
129600000 (36 Stunden)	Einträge in der Synchronisationstabelle, die älter sind, als dieser Wert, werden gelöscht. Die Angabe erfolgt in Millisekunden.

synchronizerActive

Value (default ist fett)	Bedeutung
true	Aktiert die Synchronisation der Caches, wenn eine Anwendung verteilt auf verschiedenen Rechnern läuft. Durch die Synchronisation werden Änderungen von Properties (Neu, Ändern, Löschen) in den Caches aller Instanzen nachgezogen. Die Synchronisation erfolgt zeitverzögert (s. synchronizerInterval)
false	Deaktiviert die Synchronisation der Caches.

synchronizerClearTableInterval

Value (default ist fett)	Bedeutung
10800000 (3 Stunden)	Die lokale Synchronisationstabelle wird regelmäßig

	gelöscht. Der Parameter gibt das Intervall in Millisekunden an. Innerhalb dieser Zeitspanne sollten sich alle anderen Instanzen synchronisiert haben (s. synchronizerInterval).

synchronizerInterval

Value (default ist fett)	Bedeutung
300000 (5 Minuten)	Ist das Intervall der Synchronisation in Millisekunden. Diese Zeitspanne kann max. verstreichen, bis Properties, die auf einer anderen Instanz geändert wurden, synchronisiert sind.

synchronizerClearCacheInterval

Value (default ist fett)	Bedeutung
86400000 (24 Stunden)	Innerhalb dieses Intervalls wird der Synchronisations-Cache (nicht der Property-Cache) vollständig geleert. Das Bereinigen des Cache soll möglichen Datenfehlern in der Synchronisation vorbeugen. Die Angabe erfolgt in Millisekunden.

cacheType

Value (default ist fett)	Bedeutung
DYNAMIC	Der C7 Cache dient insbesondere der Performance. Dazu werden einzelne Konfigurationswerte und auch Property-Listen im Speicher vorgehalten. Ist der CacheType 'DYNAMIC', wird die Anwendung mit einem leeren Cache gestartet. Erst bei Zugriff auf Properties (Anlegen, Löschen, Lesen) erfolgt eine Ablage bzw. Aktualisierung im Cache.
STATIC	Setzt voraus, dass die Option cacheActive gesetzt ist (true). Wenn die Verfügbarkeit der DB während der

	Laufzeit nicht garantiert ist, oder wenn Properties zur Laufzeit nicht geändert werden, kann die gesamte Konfiguration zu Beginn in den Cache geladen werden. Das Schreiben von Properties erfordert den Zugriff auf die DB, auch bei statischem Caching. Wenn die DB zur Laufzeit nicht garantiert zur Verfügung steht, können evtl. nicht persistente Properties eine Lösung sein, da diese ja keinen Zugriff auf die DB erfordern. Bei Synchronisation wird der Cache gelöscht und vollständig neu geladen. Ist die DB nicht erreichbar, führt dies zwangsläufig zu Problemen. Daher sollte bei instabiler oder nicht garantierter DB-Verfügbarkeit die Synchronisation deaktiviert werden. Das regelmäßige Bereinigen des Cache wird im STATIC Modus nicht durchgeführt, auch wenn die Option garbageActive gesetzt wurde (true).

cacheActive

Value (default ist fett)	Bedeutung
true	Aktiviert das lokale Caching der Konfiguration.
false	Deaktiviert das lokale Caching der Konfiguration.

cacheGarbageActive

Value (default ist fett)	Bedeutung
true	Aktiviert das regelmäßige Aktualisieren des Cache. Dabei werden Einträge entfernt, auf die seit einer längeren Zeitstpanne nicht zugegriffen wurde (s. cacheMaxGarbageTimeNoAccess).
false	Deaktiviert das regelmäßige Aktualisieren des Cache.

cacheGarbageInterval

Value (default ist fett)	Bedeutung
60000 (1 Minute)	Im angegebenen Intervall wird der Cache auf veraltete Einträge überprüft.

cacheMaxGarbageTimeNoAccess

Value (default ist fett)	Bedeutung
3600000 (1 Stunde)	Ist die Zeitspanne, die ein Eintrag im Cache haben darf, ohne, dass auf ihn zugegriffen wurde. Ältere Einträge werden aus dem Cache entfernt. Die Angabe erfolgt in Millisekunden.

cacheMaxSize

Value (default ist fett)	Bedeutung
1000	Gibt die max. Anzahl Einträge im Cache an. Dabei wird zwischen Einzeleinträgen und Listeneinträgen unterschieden. Für beide Arten wird der genannte Cache gesondert zur Verfügung gestellt. D.h. es können n Einzeleinträge und n Listeneinträge existieren.

soapSessionTimeOut

Value (default ist fett)	Bedeutung
300000 (5 Minuten)	Gibt die Gültigkeit einer SOAP-Benutzersession in Millisekunden an. Greift ein Benutzer innerhalb dieser Zeitspanne nicht auf die SOAP-Schnittstelle zu, verfällt seine Sitzung und er muss sich erneut anmelden. Wert ist in Millisekunden angegeben.

CLOUD SEVEN Anwendungen

Integraler Bestandteil von CLOUD SEVEN sind lauffähige Anwendungen, die durch entsprechende Aufrufe in der Java Umgebung gestartet werden.

Starten einer CLOUD SEVEN Anwendung

Für die korrekte Ausführung einer CLOUD SEVEN Anwendung muss ein entsprechendes Properties File vorbereitet werden. Bei Start von CLOUD SEVEN können Sie u.a. den Namen des Properties File als Parameter übergeben. Wird keine Angabe über das Properties File beim Start gemacht, versucht CLOUD SEVEN auf die Datei c7.prop im Startverzeichnis zuzugreifen, also dort, von wo aus die Anwendung aufgerufen wird. Zusätzlich zum Properties File können Sie per Parameter den Typ der Anwendung steuern.

Achtung! Bei Windows-Systemen muss der Backslash in Pfadangaben entweder doppelt angegeben werden (\\ statt \) oder als einfacher Slash (/).

Sie können den konfigurierten Wert der zu startenden Anwendung (z.B. *editor* oder *server*) durch entspr. Shellparameter überschreiben.

Variante a): Aktivierung durch das Properties File

```
application=<Anwendung>
java -jar CloudSeven.jar [--propertiesfile=<Pfad>]
```

Beispiel

```
application=editor
java -jar CloudSeven.jar --propertiesfile=c7.prop
```

Variante b): Überschreiben des Konfiugrationseintrags für die zu startende Anwendung durch Shellparameter

```
java -jar CloudSeven.jar [--propertiesfile=<Pfad>] --application=<Anwendung>
```

Beispiel

```
java -jar CloudSeven.jar --propertiesfile=c7.prop --application=editor
oder
java -jar CloudSeven.jar --propertiesfile=c7.prop --application=server
```

Wird eine Server-Anwendung genutzt (also JMS, SOAP oder eine eigene), müssen weitere

Informationen im Properties File hinterlegt werden.

SOAP Server

Setzen Sie diesen Parameter

```
connectionType=CONNECTION_TYPE_SOAP
```

Alternativ zum Aufrufargument

```
application=server
```

JMS Dienst

Setzen Sie diesen Parameter

```
connectionType=CONNECTION_TYPE_JMS
```

Alternativ zum Aufrufargument

```
application=server
```

API Anwendung

Sie können Ihre eigene Anwendung als CLOUD SEVEN Applikation starten. Dafür müssen Sie im Properties File den vollständigen Namen der Klasse angeben, in der die main Methode liegt.

```
connectionType=CONNECTION_TYPE_API
userApplicationClass=de.myPackage.myClass
```

Zum Zeitpunkt des Aufrufs der main Methode Ihrer Anwendung ist die Initialisierung von CLOUD SEVEN bereits abgeschlossen. Die Parameter des Properties File stehen zur Verfügung, die Aufrufargumente werden vollständig durchgereicht.

Java API

Grundlegendes

CLOUD SEVEN wurde mit dem Ziel einer ausgereiften Verwaltung von Konfigurationswerten entwickelt. Für die SOAP- und JMS-Anwender entstand auf Basis dieser Lösung eine praktische und im Umfang für viele Anwender ausreichende User Verwaltung. Da eine Vorgabe für CLOUD SEVEN der – soweit mögliche – Verzicht auf fremde Frameworks war, wurde eine eigene Datenbank-Schnittstelle implementiert.

Konfigurationsverwaltung, User Verwaltung und die DB-Schnittstelle stehen als API zur Verfügung. Dabei ist zu berücksichtigen, dass die DB-Schnittstelle einen auf das Anlegen von Tabellen und das Erzeugen einfacher SQL-Statements eingeschränkten Umfang aufweist. Für Standardfälle lohnt sich die Betrachtung der letztgenannten Schnittstelle dennoch.

Mit Ausnahme von Enums und den Klassen für die Initialisierung (C7Option und C7configuration), ist der C7Manager die zentrale Klasse der meisten Aktionen.

Javadoc

Die ausführliche aktuelle Dokumentation können Sie von der Webseite downloaden:

http://c7.dirk-goldbach.de/download/

Nutzung der API

Wenn Ihre Anwendung nicht als CLOUD SEVEN Applikation laufen soll (s.o.), muss im Properties File lediglich der connectionType benannt werden. Die übrigen Parameter können – wenn Sie nicht in der Konfigurationsdatei stehen sollen – per Programmcode gesetzt werden. Ihr Programm muss lediglich die Initialisierung durchführen, bevor erstmalig auf C7Properties oder die User Verwaltung zugegriffen wird.

Initialisierung zum Programmstart

Soll die Anwendung als Teil von CLOUD SEVEN gestartet werden, wird dies mit dem connectionType im Properties File festgelegt. Außerdem muss die Klasse, in der die main Methode ist, konfiguriert werden.

```
connectionType=CONNECTION_TYPE_API
userApplicationClass=de.myPackage.myClass
```

Mit Ausnahme dieser beiden Einstellungen für CLOUD SEVEN Anwendungen, ist das Setzen der Parameter alternativ mittels Properties File oder anhand der Klasse C7Configuration

machbar. Außerdem bieten sich einige Enums zur Verwendung an. Die Methoden der Klasse C7Configuration sind statisch und gelten für die gesamte VM in der C7 läuft.

Codebeispiel Initialisierung

Im Beispiel werden die für den Programmstart notwendigen Parameter gesetzt.

```
C7Configuration.setDatabaseType(C7DatabaseType.DATABASE_TYPE_MYSQL);
C7Configuration.setDatabaseHost('myHostName');
C7Configuration.setDatabaseName('myDBName');
C7Configuration.setDatabaseUser('userName');
C7Configuration.setDatabasePassword('pwd');
C7Configuration.setDatabasePort(newLong(3306));
C7Configuration.setApplication('myApplication');

C7Manager.init();
```

Optionen

Mit Optionen kann das Verhalten der Software (von C7) manipuliert werden. Welche Optionen zur Verfügung stehen, können Sie dem Kapitel 'Inbetriebnahme' entnehmen.

Die Optionen müssen, wie auch die Initialisierungsparameter, ganz zu Beginn der Anwendung, also noch in der Initialisierungsphase und vor Aufruf der C7Manager.init() Methode, gesetzt werden.

Im Gegensatz zur Klasse für die Initialisierung, ist die Klasse C7Option statisch. Möglicherweise ist hier noch Potential für eine zukünftige Version.

Beachten Sie auch hier die Verwendung der Enums für das Loglevel und die Caching-Einstellungen.

Codebeispiel Optionen

Für den Anfang setzen wir im Beispiel den Pfad des Logfiles, das Loglevel, den Caching-Modus und vielleicht die max. Verweildauer ungenutzter Parameter im Cache. Das Beispiel setzt auf dem vorhergehenden Beispiel auf. Auf Windows-Systemen müssen die Backslashes von Pfadangaben mit einem weiteren Backslash escaped werden.

```
C7Configuration.setDatabaseType(C7DatabaseType.DATABASE_TYPE_MYSQL);
C7Configuration.setDatabaseHost('myHostName');
…
C7Option.setLogFile('C:/USERS/USER1/DOCUMENTS/c7Log.log');
C7Option.setLogLevel(C7LogLevel.INFO);
C7Option.setCacheType(C7CacheType.DYNAMIC);
C7Option.setCacheMaxTimeNoAccess(300000);
```

```
C7Manager.init('MyFirstApplication', configuration);
```

Handling von C7Properties

Der Zugriff auf C7Properties erfolgt mit Hilfe des C7Managers.

Nutzung des Instanz-Automatismus

Der Automatismus wird aktiviert durch

```
C7Manager.activateInstanceUsage(true);
```

Diese Einstellung gilt für die gesamte Laufzeit von CLOUD SEVEN, allerdings nur für die eine Instanz von CLOUD SEVEN. Wurde c7 mehrfach installiert, muss diese Einstellung – falls gewünscht – für jede Installation gesondert durchgeführt werden.

Instanz- bzw. hostabhängige Parameter müssen mit dem Hostnamen versehen werden. Um Abweichungen in der Schreibweise zu vermeiden empfiehlt es sich, den von C7 verwendeten Namen per Methode liefern zu lassen:

```
String instanceName = C7Manager.getInstanceName();
```

Ein neues C7Property würde dann beispielsweise angelegt mit

```
C7Property property = new C7Property('myKey', 'myValue', null, instanceName);
C7Manager.setProperty(property);
```

Existierende C7Properties können nachträglich mit dem Instanznamen versehen werden:

```
C7Property property = C7Manager.getProperty('myKey');
property.setInstance(C7Manager.getInstanceName);
C7Manager.setProperty(property);
```

Für den 'Umzug' einer Software auf eine andere Maschine gibt es die Möglichkeit den alten Instanznamen für alle Konfigurationswerte auf den neuen Instanznamen zu ändern. Nach Installation auf der neuen Maschine genügt der Aufruf:

```
C7Manager.movePropertyInstance('oldHostName', C7Manager.getInstanceName());
```

Das Leben kann so einfach sein.

Nutzung für verschiedene Mandanten

Um die Parametrisierung Ihrer Software mit CLOUD SEVEN mandantentauglich zu machen, müssen die API-Methoden von c7 mit den Signaturen verwendet werden, die die Angabe eines clientName erlauben bzw. es muss die Eigenschaft clientName am C7Property genutzt werden.

Versehen Sie beim Schreiben der Konfiguration die mandantenbezogenen Parameter mit dem Wert des clientName und die von Mandanten unabhängigen Parameter mit dem Wert null (default). Für den lesenden Zugriff verwenden Sie im Zweifelsfalle immer den clientName. CLOUD SEVEN prüft, ob es dem Schlüssel entspr. Einträge mit der passenden clientId gibt. Ist das nicht der Fall, wird nach allgemeingültigen, also mandantenunabhängigen Werten gesucht.

Auch wenn Ihre Software so implementiert ist, dass einige oder alle Funktionen innerhalb eines Mandantenkontexts ablaufen und die Aufrufe der Konfigurationsmethoden mit einer clientId erfolgen müssen, ist das kein Problem.

Codebeispiel für verschiedene Mandanten

Im Beispiel erhalten alle Mandanten die Priorität (wofür auch immer) mit dem Wert 1. Der besonders wichtige Mandant mit der clientName m-100 bekommt die Priorität 9.

```
C7Property priorityAlleMandanten = new C7Property();
priorityAlleMandanten.setConfKey('priority');
priorityAlleMandanten.setValue('1');
C7Manager.setProperty(priorityAlleMandanten);

C7Property priorityWichtigerMandant = new C7Property();
priorityWichtigerMandant.setConfKey('priority');
priorityWichtigerMandant.setValue('9');
priorityWichtigerMandant.setClientId('m-100);
C7Manager.setProperty(priorityWichtigerMandant);

// getPropertyInt(String confKey, String clientName, String instance, int default)
System.out.println(C7Manager.getPropertyInt('priority', null, null, 0));
→ 1
System.out.println(C7Manager.getPropertyInt('priority', 'm-100', null, 0));
→ 9
System.out.println(C7Manager.getPropertyInt('priority', 'm-50', null, 0));
→ 1
```

Caching

Das Caching dient dem beschleunigten Zugriff auf Parameter und ermöglicht eine bedingte Unabhängigkeit von der Datenbank zur Laufzeit. Das Caching kann zudem deaktiviert werden, beispielsweise, wenn die Anwendung über mehrere Instanzen verteilt läuft und die Konfiguration aller Instanzen stets aktuell sein muss (s. Synchronisation).

Für das Caching stehen zwei Betriebsarten zur Verfügung, das dynamische und das statische Caching.

Wird dynamisch gecached, startet die Anwendung zunächst ohne Zugriff auf anwendungsspezifische Konfigurationswerte. Erst bei Zugriff auf Parameter (Lesen, Schreiben, Löschen) werden die Parameter in den Cache geladen bzw. im Cache aktualisiert.

Damit der Speicher nicht unnötig belastet wird, kann die max. Anzahl vorgehaltener Einträge reglementiert werden. Einträge, auf die selten oder gar kein Zugriff mehr erfolgt, werden aus dem Cache entfernt.

Bei statischem Caching wird die Konfiguration während des Anwendungsstarts vollständig in den Cache geladen. Dieser Modus bietet sich an, wenn die Datenbankverbindung nicht konstant zur Verfügung steht. Bei entspr. Einstellungen und unter Nutzung von nicht persistenten Properties kann die Anwendung nach dem Programmstart ohne DB-Verbindung genutzt werden.

Damit bei verteilten Anwendungen die jeweils lokalen Caches nicht 'auseinander laufen', werden sie regelmäßig synchronisiert.

Cache Refreshing

Um Datenfehlern über lange Laufzeiten vorzubeugen, wird der Cache unabhängig von den Zugriffen auf die Parameter in vorgegebenen Intervallen geleert. Abhängig vom Caching-Modus werden die Parameter anschließend vollständig neu (STATIC) oder bei Bedarf bzw. Zugriff (DYNAMISCH) neu in den Cache geladen.

Cache Garbage

Selten genutzte Konfigurationswerte werden aus dem Cache entfernt. Die Bedeutung des Begriffs 'selten' kann anhand der Konfiguration bestimmt werden. Dies ist nicht nur sinnvoll, um unnützen Speicherverbrauch auszuschließen; die Limitierung der Anzahl von Parametern im Cache kann dazu führen, dass einige Werte nicht gecached werden können. Durch das Entfernen selten genutzter Parameter wird für andere Parameter Platz gemacht.

Synchronisation

Der Cache liegt im lokalen Arbeitsspeicher. Wenn Anwendungen verteilt sind und diese sich eine zentrale Konfiguration teilen, tritt bei Änderungen in der Datenbank durch die Software einer Instanz eine Diskrepanz zwischen den Instanzen auf. Das liegt daran, dass ja bei Zugriff auf einen Konfigurationswert dieser zunächst aus den lokalen Caches gelesen wird.

Die Synchronisation prüft veränderte Werte in der Datenbank und aktualisiert die lokalen Caches.

Abhängig von der Anwendung bzw. dem Anspruch auf Aktualität der Konfigurationswerte muss entschieden werden, ob das Caching genutzt wird und wenn ja, wie viel Zeit bis zur Synchronisation verstreichen darf, die verschiedenen Instanzen also abweichende Werte haben dürfen.

Die Synchronisation erfolgt in der Regel automatisiert, kann zusätzlich mittels spezieller Methoden bzw. Requests der API, SOAP oder JMS beeinflusst werden.

Logging

Zur Überwachung und zur Fehlerbehebung der Anwendung werden Informationen während der Laufzeit in eine Logdatei geschrieben. Um welche Informationen es sich dabei handelt kann sehr unterschiedlich sein, von großem Interesse ist selbstverständlich ein unerwartetes bzw. unerwünschtes Verhalten der Software sowie Fehler, die im Zusammenhang mit Daten entstehen.

Ob das Logging aktiv ist, wie und wohin gelogged wird, bestimmen Sie über diverse Optionen, die am Properties File oder am Objekt C7Option gesetzt werden.

Damit das Logging die Anwendung nicht bremst (z.B. bei wartenden Schreibprozessen in die Datenbank), werden die Meldungen der Software in einem nebenläufigen Prozess abgearbeitet. Dennoch entspricht die Reihenfolge der Logeinträge dem Auslösen des Loggings in der Software.

Loglevel

Typischerweise wird die Granularität der Informationen anhand sogenannter Loglevel gesteuert. Abhängig davon, wie detailliert der Anwender über Zustände der Software in Kenntnis gesetzt werden will, kann er das über die Level steuern. Gleichzeitig stellen die Loglevel so etwas wie eine Gewichtung dar, die Auskunft über die Bedeutung des Log-Eintrags gibt.

Value (default ist fett)	Bedeutung
DEBUG	Debug-Einträge sind für die detaillierte Fehlersuche gedacht. Die Anzahl der Log-Einträge kann sehr hoch sein. Dieses Level schließt die folgenden Level mit ein, d.h. INFO, WARNING und ERROR Meldungen werden ebenfalls geloggt.
INFO	Info-Einträge dokumentieren die Aktionen der Software. Das können z.B. die Schritte beim Laden der Software sein oder zeitgesteuerte Aktionen, die u.a. zum Cache-Garbage gehören. Dieses Level schließt die folgenden Level mit ein, d.h. WARNING und ERROR Meldungen werden ebenfalls geloggt.
WARNING	Warning-Einträge sind Hinweise auf nicht erwartete Zustände oder Probleme, die aber nicht

	zwangsläufig zu falschen Daten führen müssen. Dieses Level schließt Meldungen des Levels ERROR mit ein.
ERROR	Error-Einträge zeigen ernst zu nehmende Fehler. Solche Fehler führen zu Störungen der Software und können im schlechtesten Fall auch zu Datenfehlern führen.

Das Loglevel wird im Properties File oder am C7Option-Objekt festgelegt.

Logdatei

Die Ausgaben erfolgen in der Datei c7Log.log, die im Startverzeichnis der Anwendung angelegt wird.

Der Name sowie das Ablageverzeichnis der Datei können über die Optionen gesteuert werden. Die Optionen wiederum können im Properties File oder bei Verwendung der Java API alternativ am Objekt C7Options gesetzt werden.

Optionales Logging in der Datenbank

Zur Überwachung der Anwendung können die Log-Ausgaben zusätzlich in die Datenbank geschrieben werden. Die Tabelle hat den Namen C7LOG und liegt im selben Schema, wie die C7Properties-Tabelle. Um das Logging in der DB zu aktivieren muss die Option logDb=true sein (per Properties File oder API C7Option).

Da die Loglevel in einer eigene Spalte stehen (LOGLEVEL), kann z.B. diese Spalte auf ERROR oder WARNING durch entspr. Monitoring-Tools getriggert werden.

Die wichtigsten Attribute der Tabelle C7LOG:

Feldname	Bedeutung
LOGLEVEL	Die Gewichtung bzw. das Level für den der Eintrag erfolgt.
CODE	Ein numerischer Fehlercode aus C7. Dient automatisierter Auswertung. Ist die numerische Repräsentation von LOGINFO.
LOGINFO	Eine allgemeine Information über die zum Zeitpunkt des Logeintrags von C7 ausgeführte Aktion. Ist die textuelle Repräsentation von CODE.
EXCEPTION	Ein entstandener Fehler oder eine Warnung. Wenn der Eintrag keinen Fehler repräsentiert, ist das Feld leer (null).
MESSAGE	Eine weiterführende Information zu LOGINFO. Kann z.B. die

	Ursache für den Logeintrag oder mehr Details zu einem Zustand der Software liefern.
LASTUPDATE	Gibt Auskunft darüber, wann der Eintrag erfolgte.

Editor

Der CLOUD SEVEN Editor ist ein einfaches Werkzeug, das Administratoren die Bearbeitung der Konfiguration auf der Konsole erlaubt. Die wichtigsten Eigenschaften der C7Properties werden mit dem Editor angezeigt und geändert.

Vor Benutzung des Editors muss das Setup von Cloud Seven abgeschlossen sein (s.a. Kapitel **Installation / Inbetriebnahme / Setup**).

Starten des Editors

Alternativ zum Aufruf per Parameter kann der Editor als Anwendung im Properties File hinterlegt werden:

```
application=editor
```

Die Anwendung unterscheidet zwischen Anzeigemodi (?, LS) und Bearbeitungsmodi (EDIT, DELETE, MOVE).

Dialogbetrieb

Anzeigemodus ?

Im Anzeigemodus, der durch ein ? vor dem Prompt kenntlich gemacht wird, werden Konfigurationspfade oder Konfigurationswerte angezeigt. Der Anwender hat folgende Möglichkeiten:

Eingabe eines Konfigurations-Key (confKey). Groß-/Kleinschreibung beachten! Entspricht der Key einem Namespace, wird eine Liste der Keys geliefert.

```
? >> server
    server.linux
    server.windows
? >> server.linux
    server.linux =>
       Value    : DMZ
```

Switchen zwischen einfachem und komplexem Modus.

simple: Suche über Key; Mandant und Instanz mit != null werden ignoriert

complex: Einschränkung auf Mandant und Instanz

```
? >> simple
Modus simple
? >> os
    os =>
        Value   : linux

? >> complex
Modus complex
? >> os
Client >> 100
Instance (Host) >> 192.168.130.1
    os =>
        Value   : linux
        Client  : m-100
        Instance: 192.168.130.1
        Comment : Beispielkommentar
        Desc.   : null
        Custom  : false
        Internal: false
        Visible : false
? >>
```

Beenden der Anwendung mit exit oder quit.

Bearbeitung (EDIT/DELETE/MOVE)

Bearbeitung durch die Modi EDIT(edit) bzw. DELETE(delete) oder MOVE (move)

EDIT

Modus zum Editieren oder Neuanlage von Konfigurationseinträgen

```
? >> edit
 EDIT ConfKey >> os
 ConfValue >> linux
 EDIT ok? (y/yes) >> yes
 // nur bei yes/y wird gespeichert.
```

simple: nur Key und Value werden editiert; Daten mit clientId/instance != null bleiben

unberührt

complex: alle relevanten Daten werden editiert

```
EDIT ConfKey >> complex
Modus complex
EDIT ConfKey >> os
ConfValue >> linux
Client >> 100
Instance (Host) >> 192.168.130.1
Comment >> Beispielkommentar
Visible? (true) >> true
Internal? (false) >>                  // <- Für Defaultwerte genügt auch Return
Custom? (false) >> true
Crypted? (false) >>                   // <- Für Defaultwerte genügt auch Return
EDIT ok? (y/yes) >> yes
```

quit: Verlassen des Editier-Modus

```
EDIT confKey >> quit
? >>
```

Abbruch des Bearbeitungsmodus mit exit oder quit.

DELETE

Modus zum Löschen

```
DELETE ConfKey >> os
    os: linux
ok? (yes/no) >>
```

simple: Suche über Key; Daten mit clientId/instance != null bleiben unberührt

complex: Mandant und Instanz müssen passen

```
DELETE ConfKey >> complex
Modus complex
DELETE ConfKey >> os
Client >> 100
Instance (Host) >> 192.168.130.1
    os: linux
DELETE ok? (yes/no) >>
```

Abbruch des Bearbeitungsmodus mit exit oder quit.

MOVE

Verschieben der Properties von einer Instanz in eine andere. Die Angabe leerer Werte für alte und neue Instanz ist zulässig.

Achtung! Mit diesem Kommando kann unbeabsichtigt die von Instanzen unabhängige Konfiguration auf eine bestimmte Instanz umgehängt werden.

```
? >> mv
MOVE old Instance >> 192.168.130.1
new Instance >> SERVER-EMEA
MOVE ok? (yes/no) >> yes
MOVE old Instance >>
```

LS

Liste der Properties

•simple: für alle Mandanten

```
? >> simple
Modus simple
? >> ls
    os
     Value     linux
     Client    100
     Instance  192.168.130.1
```

complex: auf Mandant eingeschränkt, instance wird ignoriert

```
? >> complex
Modus complex
? >> ls
ClientId >> 100
  os
   Value     windows
   Client    20
   Instance  192.168.130.2

  os
   Value     linux
   ClientId  null
   Instance  null

  os
```

```
    Value     sun-solaris
    Client    100
    Instance  192.168.130.1

   os
    Value     linux
    Client    500
    Instance  null
```

Quit / Exit

Mit quit oder exit kann in den meisten Situationen einer der Bearbeitungsmodi verlassen werden (Ausnahme ist z.B., wenn die Eingabe einer Instanz erwartet wird).

Im Anzeigemodus wird mit quit oder exit die Anwendung beendet.

Shortcuts

Kommandos können abgekürzt werden:

y=yes, q=quit, x=exit, del=delete, mv=move

-h=help, -s=simple, -c=complex

```
 ? >> complex
 Modus complex
 ? >> -s
 Modus simple
 ? >> del
 DELETE ConfKey >>
 ? >> -h
   Einfaches Werkzeug zur Administration von Konfigurationswerten (SmsProperty)

   Die Anwendung kennt einen Anzeigemodus (?) und zwei Bearbeitungsmodi
(EDIT/DELETE).
   ...
   ...

 // u.s.w.
```

Parameteraufruf von der Shell

Die Anwendung kann mit Parametern direkt von der Konsole aufgerufen werden. Aktuell werden nur die Hilfe und die Anzeige von Konfigurationswerten unterstützt.

Syntax:

```
 $ cloudSeven.sh --help
 $ cloudSeven.sh --key=<Konfigurations Key> [--c | --complex [--client=<Client
Name>] [--instance=<Instanz/Server>]]
```

Beispiele:

```
// Anzeige der Hilfe
$ cloudSeven.sh --help
// Simple Modus (client und instance sind null)
$ cloudSeven.sh --key=os
    os =>
        Value   : linux
// Complex Modus (client und instance sind null)
$ cloudSeven.sh --key=os
    os =>
        Value   : linux
        Client  : null
        Instance: null
        Comment : Ein Kommentar
        Desc.   : Standard-Betriebssystem für alle Mandanten und Instanzen
        Custom  : false
        Internal: false
        Visible : false


// Complex Modus (Einschränkung auf clientId=100 UND instance=null)
$ cloudSeven.sh --key=os --c --clientId=100
    os =>
        Value   : Windows
        Client  : 100
        Instance: null
        Comment : Windows nur für diesen Mandanten auf allen Instanzen
        Desc.   : null
        Custom  : false
        Internal: false
        Visible : false
```

```
// Complex Modus (Einschränkung auf instance=192.168.130.1 UND clientId=null)

$ cloudSeven.sh --key=os --c --instance=192.168.130.1
    os =>
        Value   : sun solaris
        Client  : null
        Instance: 192.168.130.1
        Comment : Sun läuft auf dieser Kiste, unabhängig vom Mandanten
        Desc.   : null
        Custom  : false
        Internal: false
        Visible : false
```

User Verwaltung

Unabhängig von den übrigen Funktionalitäten kann die User Verwaltung für Entwickler oder Administratoren nützlich sein, die nach einem Benutzer/Rechte/Aktionen-Konzept suchen, denn genau auf diesen Komponenten basiert die CLOUD SEVEN User Verwaltung.

Die User Verwaltung steht ebenfalls für die API, SOAP und JMS zur Verfügung und stellt die Mandantenunterstützung bereit.

Um die User Verwaltung zu nutzen, müssen Sie die für die Inbetriebnahme von CLOUD SEVEN notwendigen Schritte durchführen.

Konzept

Die User Verwaltung besteht aus den vier Elementen *Benutzer*, *Rechte-Klassen*, *Rechte* und *Aktionen*.

Benutzer setzen sich zusammen aus einem Usernamen und einem Passwort.

Rechte-Klassen sind vergebbare Rechte. Sie können in einer Baumstruktur voneinander abhängig sein.

Rechte sind Benutzern zugewiesene Rechte-Klassen.

Aktionen sind mit Rechte-Klassen verbundene Tätigkeiten.

Vor dem Aufruf einer Funktion oder Methode wird zunächst geprüft, ob der Benutzer bekannt ist und ob er grundsätzlich zum Aufruf von Methoden zugelassen ist (Login-Recht). Anschließend wird die für die Funktion oder Methode stellvertretende Aktion gesucht. Anhand der Aktion wird die notwendige Rechte-Klasse festgelegt und die Rechte des Benutzers abgeglichen.

Rechte und Aktionen

Rechte-Klassen sind die Menge aller an Benutzer vergebbaren Rechte. Rechte-Klassen können in Form einer Baumstruktur voneinander abhängig sein. Eine Rechte-Klasse deckt alle ihr untergeordneten Rechte-Klassen ab.

Beispiel:

ADMIN → ADMIN_PING

→ ADMIN_SHUTDOWN

Wird einem Benutzer das Recht ADMIN erteilt, verfügt er implizit solange über die Rechte ADMIN_PING und ADMIN_SHUTDOWN, bis ihm eines der beiden Rechte oder beide wieder entzogen werden.

Rechte-Klassen werden durch ihre Namen repräsentiert, die – so meine Empfehlung – aus

Großbuchstaben bestehen sollten.

Aktionen werden den Rechte-Klassen im Verhältnis n:1 zugeordnet. Verschiedene Aktionen verweisen also möglicherweise auf die selbe Rechte-Klasse. So weiß CLOUD SEVEN, über welches Recht ein Benutzer verfügen muss, um eine Aktion auszuführen.

Hypothetisches Beispiel (jede Standard-Aktion in CLOUD SEVEN wird durch ein Standard-Recht abgedeckt):

Rechte-Klasse KUNDEN_VERWALTEN → Aktion KUNDE_ANLEGEN
→ Aktion KUNDE_LESEN
→ Aktion KUNDE_LÖSCHEN

Bekommt ein Benutzer das Recht KUNDEN_VERWALTEN, darf er alle Aktionen an Kundendaten ausführen.

CLOUD SEVEN Standard Rechte und Aktionen

Es ist möglich Rechte-Klassen und Aktionen passend zu Ihren Anforderungen zu definieren. Wie das geht, wird in den entsprechenden Kapiteln zur API, SOAP und JMS erklärt.

CLOUD SEVEN bringt bereits einige Rechte-Klassen und Aktionen mit, die insbesondere für administrative Tätigkeiten und den Umgang mit C7Properties interessant sind.

Standard-Rechte-Klassen in CLOUD SEVEN

Rechte-Klasse	Übergeordnete Rechte-Klasse
LOGIN	-
ADMIN	-
ADMIN_SHUTDOWN	ADMIN
ADMIN_PING	ADMIN
ADMIN_CACHE_CLEAR	ADMIN
ADMIN_CACHE_SYNC	ADMIN
ADMIN_CACHE_GARBAGE	ADMIN
ADMIN_MOVE_INSTANCE	ADMIN
USER	-

Rechte-Klasse	Übergeordnete Rechte-Klasse
USER_WRITE	USER
USER_READ	USER
USER_DELETE	USER
PROPERTY	-
PROPERTY_WRITE	PROPERTY
PROPERTY_READ	PROPERTY
PROPERTY_DELETE	PROPERTY
RIGHT	-
RIGHT_READ	RIGHT
RIGHT_WRITE	RIGHT
RIGHT_DELETE	RIGHT

Standard-Aktionen in CLOUD SEVEN

Aktion	Notwendige Rechte-Klasse
LOGIN	LOGIN
ADMIN_SHUTDOWN	ADMIN_SHUTDOWN
ADMIN_PING	ADMIN_PING
ADMIN_MOVE_INSTANCE	ADMIN_MOVE_INSTANCE
ADMIN_CACHE_CLEAR	ADMIN_CACHE_CLEAR
ADMIN_CACHE_SYNC	ADMIN_CACHE_SYNC
ADMIN_CACHE_GARBAGE	ADMIN_CACHE_GARBAGE
USER_WRITE	USER_WRITE
USER_READ	USER_READ

Aktion	Notwendige Rechte-Klasse
USER_DELETE	USER_DELETE
PROPERTY_WRITE	PROPERTY_WRITE
PROPERTY_READ	PROPERTY_READ
PROPERTY_DELETE	PROPERTY_DELETE
RIGHT_READ	RIGHT_READ
RIGHT_WRITE	RIGHT_WRITE
RIGHT_DELETE	RIGHT_DELETE

Mandantenabhängige User Verwaltung

Als Erweiterung von CLOUD SEVEN bietet die User Verwaltung die Möglichkeit, Benutzer, Rechte-Klassen und Rechte mandantenbezogen zu organisieren. So sind Anforderungen realisierbar, wie z.B. die, dass ein Benutzer zwar die Kunden von Mandant A sehen oder bearbeiten kann, nicht aber die von Mandant B.

Um die Elemente der User Verwaltung mandantenspezifisch zu organisieren, muss Ihre Software – wie sonst auch unter CLOUD SEVEN – die Mandantenkennung (clientName) nutzen bzw. auswerten. Auf die Vorgehensweisen wird ebenfalls in den Kapiteln zu den Schnittstellen eingegangen.

Nutzung der User Verwaltung

Um die User Verwaltung in Ihre Software einzubinden müssen einige Vorbereitungen getroffen werden:

- Bei Bedarf Anlegen eigener Rechte-Klassen
- Bei Bedarf Anlegen eigener Aktionen
- Abhängig von der verwendeten Schnittstelle muss vor Aufruf 'sensibler' Funktionen bzw. Verwendung 'sensibler' Daten eine entsprechende CLOUD SEVEN Funktionalität verwendet werden.

Wird ein neuer Benutzer angelegt, wird ihm implizit das Login-Recht zugewiesen. Alle

anderen Aktionen und Rechte müssen explizit den Benutzern zur Verfügung gestellt werden. Wurde ein User einem bestimmten Mandanten zugewiesen, kann er nur Aktionen ausführen, die entweder mandantenunabhängig sind oder für diesen Mandanten gelten.

Zur 'Deaktivierung' eines Benutzers ohne Löschung aller ihm zugewiesenen Rechte oder des Benutzers selbst, genügt es, ihm das Recht des Logins zu entziehen.

Zusätzlich zu den Standard-Aktionen und Standard-Rechten besteht die Möglichkeit, weitere Aktionen und Rechte zu generieren.

API

Der Zugriff auf die UserVerwaltung erfolgt mit Hilfe des C7SecurityManager

SOAP

Für SOAP steht ein Service mit entsprechenden Aktionen zur Verfügung

Rechtliche Hinweise

Dieses Dokument, das Konzept, die Realisierung/Implementierung, die (entgeltliche oder unentgeltliche) Verbreitung von CLOUD SEVEN sowie alle weiterführenden Rechte im Zusammenhang mit CLOUD SEVEN sind allein dem Urheber Dirk Goldbach vorbehalten.