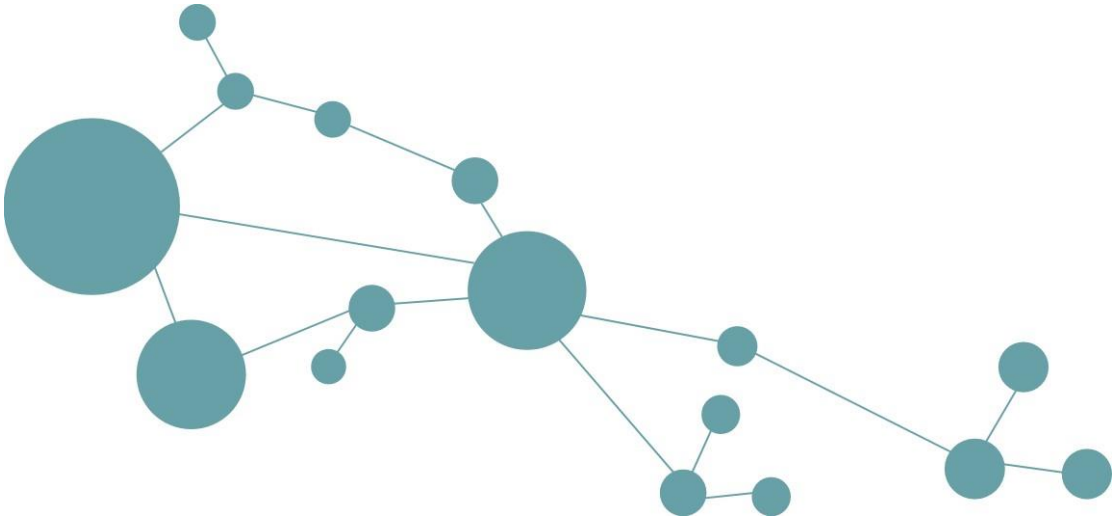


# REST-Bridge

i·views

# Inhaltsverzeichnis

# 1 Einführung

Die REST-Bridge Application ermöglicht den lesenden und schreibenden Zugriff auf K-Infinity über eine RESTful Services-Architektur. Die Schnittstelle steht als HTTP- oder HTTPS-Version zur Verfügung.

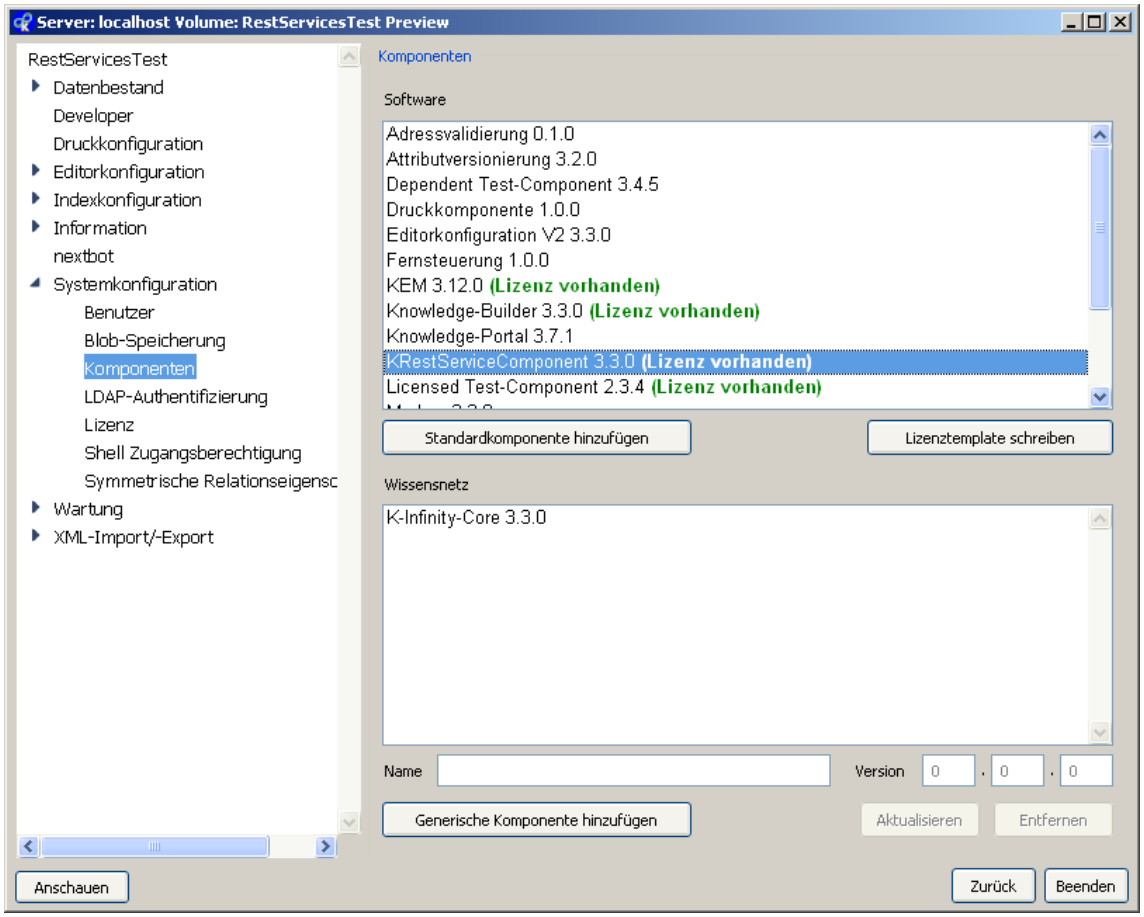
Die REST-Bridge läuft innerhalb der Standard-Bridge von K-Infinity (bridge.exe).

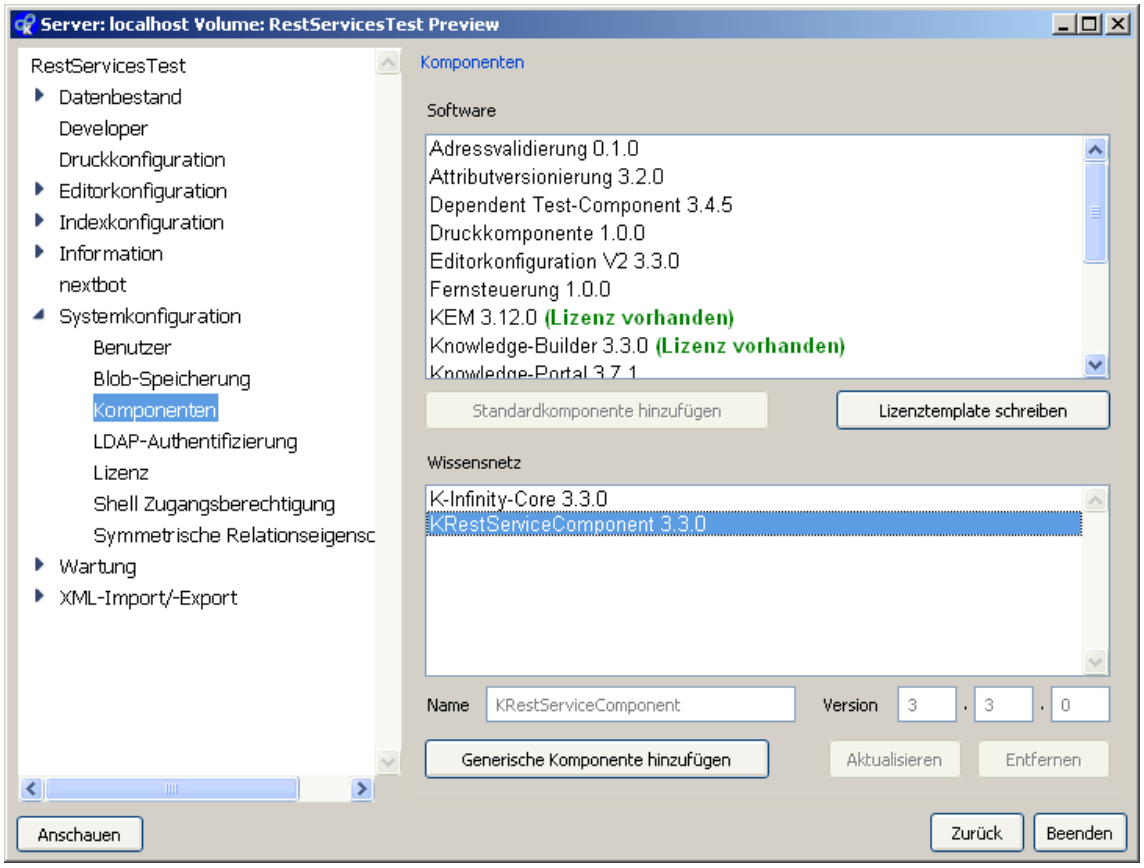
Die Schnittstelle wird vollständig durch Konfigurations-Individuen im Wissensnetz konfiguriert. Der Rückgabewert eines REST-Aufrufes ist eine beliebige Zeichenkette, in der Regel in einem Format, das der aufrufende Client gut weiterverarbeiten kann (z.B. XML oder JSON).

# 2 Installation

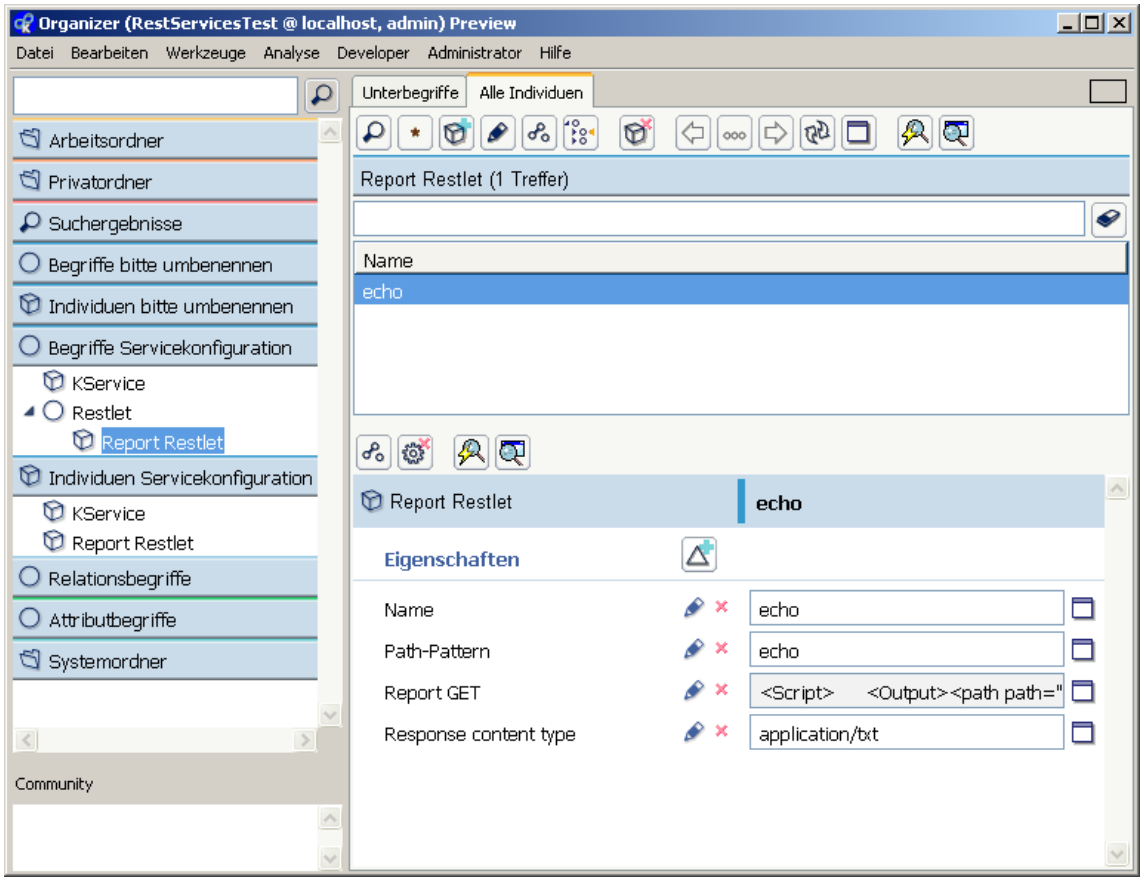
## 2.1 Volume vorbereiten

Durch das Hinzufügen der Softwarekomponente "KRestServiceComponent" im Admin-Tool wird im Wissensnetz das benötigte Schema angelegt.






Das Schema wird in einem Teilnetz des Wissensnetzes namens "Servicekonfiguration" angelegt:




## 2.2 Bridge konfigurieren

Die REST-Schnittstelle wird durch die Standard-Bridge-Komponente von K-Infinity bereitgestellt, sofern in der zugehörigen Konfigurationsdatei **bridge.ini** eine Kategorie **KHTTPRestBridge** bzw. **KHTTPSRestBridge** eingetragen ist:

```
[KHTTPRestBridge]
volume=name des Wissensnetzes
port=port, unter dem der Service erreichbar sein soll, default ist 8815
services=Liste von Service-IDs
```

Für die HTTPS-Version müssen in den Konfiguirationsdatei zusätzlich die Dateipfade für Zertifikat und Private Key angegeben werden:

```
[KHTTPSRestBridge]
volume=name des Wissensnetzes
port=port, unter dem der Service erreichbar sein soll, default ist 8815
services=Liste von Service-IDs
certificate=Name der .crt-Datei
privateKey=Name der .key-Datei
```

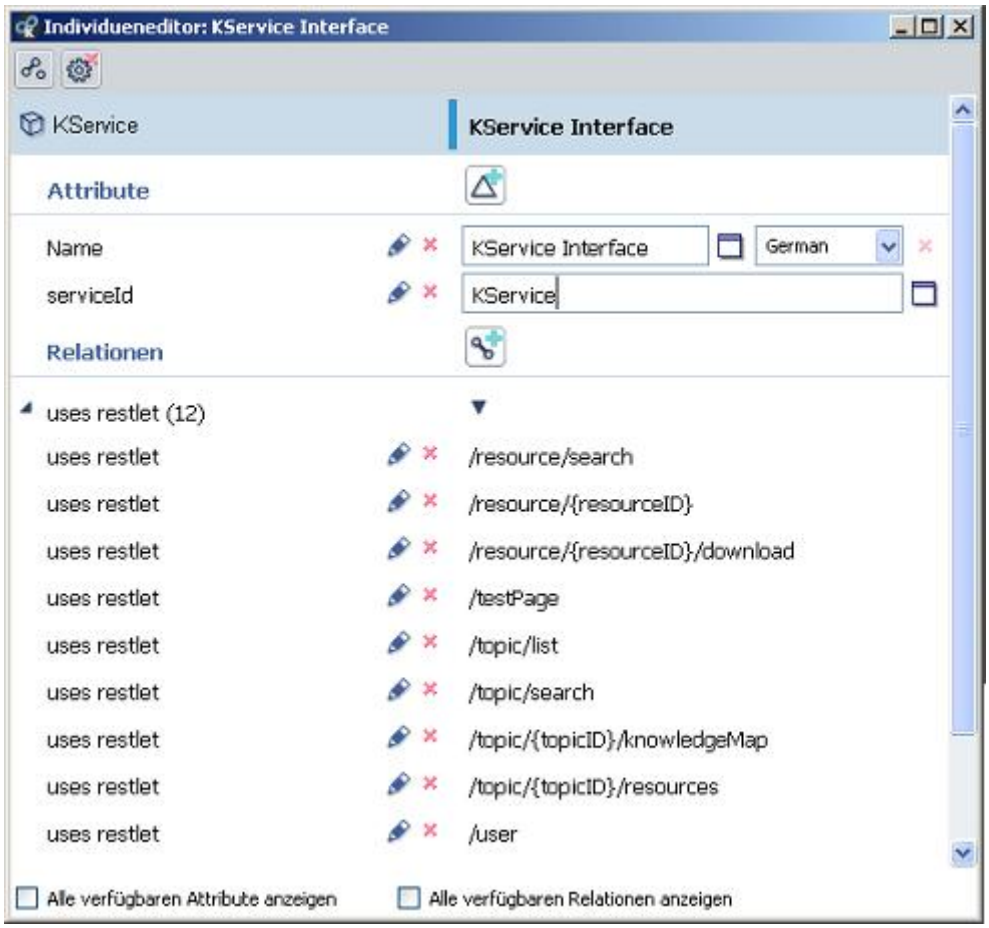
# 3 Konfiguration

## 3.1 Service Instanz definieren

Zunächst muss eine neue Service-Instanz angelegt werden. Sie haben die Möglichkeit mehrere Service-Instanzen zu konfigurieren, die von unterschiedlichen KService-Applikationen angesprochen werden. Erzeugen Sie dazu ein neues Individuum vom Begriff "KService".

An dieser Instanz muss ein frei wählbarer Name und die serviceId vergeben werden. Die serviceId muss dem serviceIdentifier in der kservice.properties entsprechen. Haben Sie diesen nicht geändert, ist der Defaultwert "KService".

Diese Instanz dient als Einstiegspunkt für die Webapplikation. Alle Restlets, die unter dieser Service-Instanz verfügbar sein sollen, müssen auch über die Relation "uses restlet" zugeordnet werden!




### 3.1.1 Reinitialisieren des Services

In diesem Abschnitt wird beschrieben, wie eine Reinitialisierung des Services durchgeführt werden kann, ohne die *Bridge* neu starten zu müssen.

Dieser Vorgang wird in zwei Schritte durchgeführt:

1. An dem *Rest-Service*, der reinitialisiert werden soll, muss das boolesche Attribut *Reinitialize service* mit gesetztem Häckchen angelegt werden.

2. Ein Request für ein bereits registriertes Restlet muss an die Bridge gesendet werden.

   -> Reinitialisieren des Services wird durchgeführt. Das Hächcken wird automatisch entfernt. Die erfolgreiche Durchfühung kann man im Bridge.log überprüfen.

Anmerkung: Sind mehrere Bridges für denselben Service konfiguriert, muss der obige Vorgang für jede Bridge einzeln wiederholt werden.

## 3.2 Restlets definieren

### 3.2.1 Restlets allgemein

Jede Methode, die Bestandteil der Service-API ist, wird durch ein eigenes Restlet-Individuum repräsentiert. Das Restlet wird durch einen frei definierbares Pfad-Muster adressiert. Hier sollten die allgemeinen Regeln für RESTful Services eingehalten werden. Im Pfad-Muster können in geschweiften Klammern variable Teile spezifiziert werden, die vom Restlet ausgewertet werden kann.

Beispiel:

```
echo/{message}
```

**Variablen**

Im Path-Pattern können Variablen definiert werden. Diese werden in geschweiften Klammern geschrieben. Diese Variablen sind required und somit fester Bestandteil des Path-Patterns. Optionale Variablen werden im QueryString übergeben.

*Beispiel*:

Pattern: http://servername:8080/api/customer/***{customerID}***/address
Aufruf: http://servername:8080/api/customer/***C4317***/address?***show=private***

***customerID*** ist eine required Variable. Für die Verarbeitung des Aufrufs, wird die Variable customerID mit dem Wert "*C4317*" befüllt. Die definierte Logik kann so den passenden Datensatz selektieren.

***show*** ist eine optionale Variable. Sie kann, muss beim Aufruf aber nicht angegeben werden. In diesem Beispiel würde das Restlet die Variable abfragen. Wenn sie gesetzt ist, wird nur die Private Adresse geliefert. Fehlt die Angabe, werden alle Adressen geliefert.

**Vordefinierte Variablen**

Zusätzlich zu den Variablen, die im Path-Pattern oder durch den Aufruf (z.B. im Body) angegeben werden, werden Variablen automatisch vom System gesetzt bzw. befüllt.

**requestBody**

Mit der Variablen *requestBody* kann auf den Inhalt des Bodys eines Requests zugegriffen werden.

Beispielaufruf zur Kontrollausgabe:

```
<Script>
    <If test="var(requestBody)/size() > 0">
        <do>
            <Output><path path="var(requestBody)"/></Output>
        </do>
        <else>
            <Output>Kein Body vorhanden</Output>
        </else>
    </If>
</Script>
```

**requestHeader**

In der Variablen *requestHeader* ist ein Dictionary enthalten, welches die Headereinträge des HTTP-Requestes enthält.

Mit Hilfe der Funktionen *groupKey* und *groupValue* kann auf die Inhalte zugegriffen werden.

Beispielaufruf zur Kontrollausgabe:

```
<XMLElement name="RequestHeader">
  <Path path="var(requestHeader)">
    <Each>
      <XMLElement name="HTTPHeader">
        <XMLAttribute name="name"><path path="groupKey()"/></XMLAttribute>
        <If test="groupValue()/size() = 1">
          <do>
            <path path="."/>
          </do>
          <else>
            <Path path="groupValue()">
              <Each>
                <XMLElement name="Value">
                  <path path="."/>
                </XMLElement><cr/>
              </Each>
            </Path>
          </else>
        </If>
      </XMLElement>
    </Each>
  </Path>
</XMLElement>
```

Ausschnitt des Results des Beispielaufrufes:

```
<RequestHeader>
    <HTTPHeader name="connection">keep-alive</HTTPHeader>
    <HTTPHeader name="accept-encoding">gzip,deflate</HTTPHeader>
    <HTTPHeader name="host">localhost:8818</HTTPHeader>
    <HTTPHeader name="accept-language">
        <Value>de-de</Value>
```

```
        <Value>de;q=0.8</Value>
        <Value>en-us;q=0.5</Value>
        <Value>en;q=0.3</Value>
    </HTTPHeader>
    <HTTPHeader name="accept">
        <Value>text/html</Value>
        <Value>application/xhtml+xml</Value>
        <Value>application/xml;q=0.9</Value>
        <Value>*/*;q=0.8</Value>
    </HTTPHeader>
</RequestHeader>
```

Tip: Für die Zerlegung eines GroupValues kann die Funktion *AnalyzeString* werwendet werden:

```
<AnalyzeString string="groupValue()" regex=",">
    <matchingSubstring/>
    <nonMatchingSubstring>
        <XMLElement name="value">
            <path path="regexNonMatch()"/>
        </XMLElement><cr/>
    </nonMatchingSubstring>
</AnalyzeString>
```

**responseHeader**

Die Variable *responseHeader* wird initial mit einem leeren Dictionary belegt. Somit ist diese Variable vorhanden und kann mit dem KPath-Ausdruck *atKeyPut()* befüllt werden.

```
<path path="var(responseHeader)/atKeyPut(einSchluessel, einWert)"/>
```

Wichtig: Der Schlüssel (*key*) muss eine Zeichenkette sein. Der Wert (*value*) darf eine Zeichenkette oder ein Ganzzahl sein, aber keine Menge.

**httpStatusCode**

Die Variable *httpStatusCode* kann auf eine beliebige Ganzzahl zwischen 100 und 599 gesetzt werden. Diese Variable wird dem HTTP-Response-Request als Statuscode gesetzt.

**Operation**

Zu den Regeln der RESTFul Services gehört die Unterscheidung der Operation durch die Art der Anfrage. Ein Http-GET-Request steht dabei für einen lesenden Zugriff. POST für eine modifikation, PUT für neu Erzeugen und DELETE für löschen.

Die Kservice-Applikation unterscheided zwischen diesen Request-Typen und erlaubt bei GET-Request keine Modifikation der Daten!  Wählen Sie deshalb bitte immer den passenden Request-Typ.

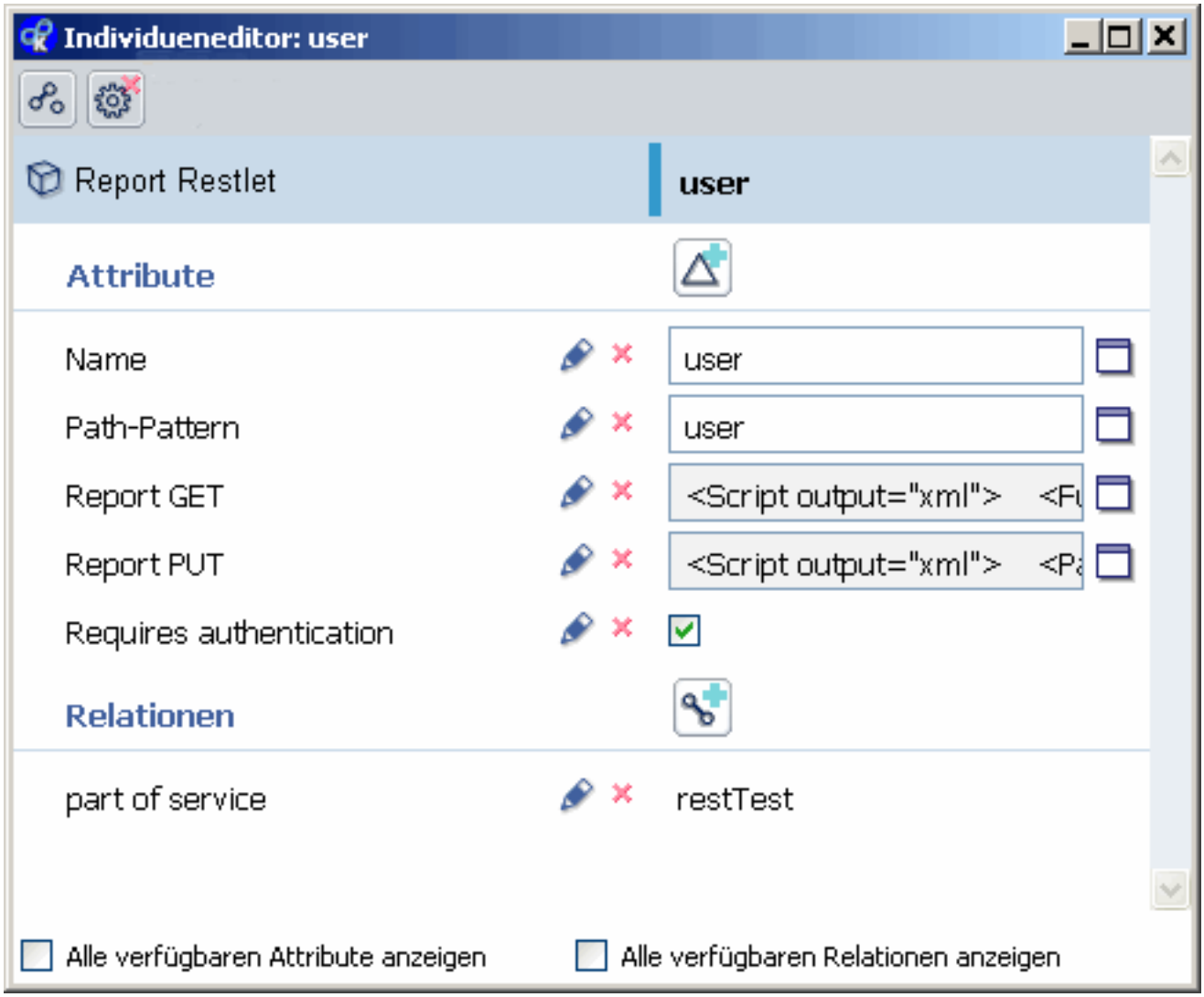
### 3.2.2 Report Restlet

Das Report-Restlet führt einen KScript-Report aus und schickt die generierten Ausgabe als Antwort zurück. Alle Variablen aus dem Pfad-Muster und die URL-Queryparameter stehen im KScript als Variablen zur Verfügung.

Beim ReportRestlet können folgende Eigenschaften konfiguriert werden:

| Eigenschaft | Erforderlich | Beschreibung |
|---|---|---|
| Path-Pattern | Ja | Pfad zum Restlet ab der Basis-URL. Variablen sind mit geschweiften Klammern zu versehen.<br>*Bsp: user/{userID}/details* |
| Report GET | Nein.<br>Standard: GET liefert http 404 | Report, der bei einem GET-Request ausgeführt werden soll. |
| Report POST | Nein.<br>Standard: POST liefert http 404 | Report der bei einem POST-Request ausgeführt werden soll. |
| Report PUT | Nein.<br>Standard: PUT liefert http 404 | Report der bei einem PUT-Request ausgeführt werden soll. |
| Report DELETE | Nein.<br>Standard: DELETE liefert http 404 | Report der bei einem DELETE-Request ausgeführt werden soll. |
| Response content type | Nein. Standard: text/plain | Content-Type des Response-Headers (z.B. text/html) |
| Requires authentica-ton | Nein. Standard: false | nein: Request ist offen zugänglich und läuft ohne Rechteprüfung<br><br>ja: User muss sich (per http Basic authorization) authentifizieren, der Request wird mit den Zugriffsrechten dieses Users ausgewertet. |
| Part of Service | Ja | Relation zur Service-Instanz, unter der das Restlet verfügbar sein soll |

*Konfigurationsbeispiel*




### 3.2.3 Blob Restlet

Das Blob-Restlet ermöglicht den Upload, Download und Löschen eines Blob-Attributes.

Beim Restlet lassen sich zwei Skript-Attribute anlegen, die folgende Objekte zurückliefern müssen.

| | Upload | | Download | Löschen |
|---|---|---|---|---|
| | neu | bestehend | | |
| Report to resolve attribute (rest.reportToResolveBlob) | Proto des neu anzulegenden Blob-Attributes | Existierendes Blob-Attribute, das verändert werden soll | Existierendes Blob-Attribute, das zurückgegeben werden soll | Existierendes Blob-Attribute, das gelöscht werden soll |

| | | | | |
|---|---|---|---|---|
| Report to resolve topic (rest.reportToResolveTopic) | Topic an dem das Blob-Attribut angelegt werden soll | - | - | - |

Beim Upload müssen im Body des HTTP-Requests folgende Paramter befüllt sein:

| | |
|---|---|
| file-name | Name bzw. Datei-Name des anzulegenden bzw. zu modifizierenden Blob-Attributes |
| con-tent | Binärdaten des Blobs |

Liste der Standardparameter:

| Variable | Required | Beschreibung |
|---|---|---|
| Path -Pattern | Ja | Pfad zum Restlet ab der Basis-URL. Variablen sind mit geschweiften Klammern zu versehen. *Bsp: download/{blobID}* |
| Requires autheticaton | Default : false | Gibt an ob sich der User authentifizieren muss |
| Execute as system user | Default: false | Ermöglicht den Download als Systemuser. |
| Part of Service | Ja | Relation zur Service-Instanz, unter der das Restlet verfügbar sein soll |

### 3.2.4 Static File Restlet

Mit Hilfe des Static File Restlet können statische Dateien über den REST-Server ausgeliefert werden. Dieser Restlet-Typ kann z.B. genutzt werden, um einfache Anwendungen vollständig durch den REST-Server und ohne Verwendung eines weiteren Webservers zu bedienen.

Zur Konfiguration wird lediglich das Attribut 'Path-Pattern' benötigt, in dem der Name eines Verzeichnisses im Arbeitsverzeichnis der REST-Bridge angegeben wird. Die Dateien in diesem und in untergeordneten Verzeichnissen können dann über die URL

```
http://{server:port}/{service name}/{path pattern}/{relative file path}
```

abgerufen werden.

## 3.3 Vordefinierte Services

### 3.3.1 Allgemeines

**Referenzen auf K-Infinity Objekte**

Wenn eine Referenz auf ein K-Infinity-Objekt in einer URL spezifiert wird, werden folgende Formate berücksichtigt.

1. Für Werte der Form **ID**<**Zahl**>_<**Zahl**> wird angenommen, dass es sich um eine Objekt-ID handelt über die das Objekt aufgelöst wird.
2. Ist mindestens ein **Tilde ("~")** im Wert enthalten, so wird der Teil vor dem ersten Tilde als interner Name eines Attributs interpretiert und der Teil hinter dem Tilde als Attributwert. Es wird ein eindeutiges Objekt gesucht, dass für dieses Attribut den vorgegebenen Wert hat.
3. Ansonsten wird der Wert als **interner Name** interpretiert. Es wird er Begriff mit dem entsprechenden internem Namen gesucht.

Beispiele:

| | |
|---|---|
| ID135_71384 | Referenziert das Objekt mit der Objekt-ID "ID135_71384" |
| Person | Referenziert den Begriff mit dem internen Namen "Person" |
| templateId~abc | Referenziert ein Objekt, dass ein Attribut mit dem internen Namen "templateId" und dem Attributwert "abc" hat. |

**Content-Disposition**

Über den Query-Parameter "**content-disposition**" eines Requests kann das Response-Header-Field "**Content-Disposition**" der zugehörigen "Response" gestetzt werden. Der Default-Wert ist ansonsten "inline".

### 3.3.2 Topic Icon

Über den folgenden Pfad kann man die Bilddatei zu einem gegebenen Topic laden. Wenn ein Individuum keine eigene Bilddatei hat, wird auf die Bilddatei des Begriffs zurückgegriffen, welche wiederum vererbar ist. Über den optionalen Parameter "size" kann man die Bilddatei mit der am ehesten passenden Größe selektieren, vorausgesetzt im Wissensnetz sind mehrere Bildgrößen hinterlegt.

**http://{server:port}/baseService/topicIcon/{topicID}?size=10**

### 3.3.3 Objektliste

Über den folgenden Pfad kann eine Objektliste im JSON-Format angefordert werden:

**http://{server:port}/baseService/{conceptLocator}/objectList**

Der Begriff der Objektliste wird über den Parameter "**conceptLocator**" referenziert, dem Format für Topic-Referenzen in der Rest-URL folgt. (siehe Verknüpfung)

Alternativ kann der "conceptLocator" auch den einen Prototyp (Individuum oder Begriff) des zu verwendenden Begriffs referenzieren.

Der optionale Parameter "**name**" bestimmt die Objektliste, die für die Ausgabe verwendet werden soll.

**Filter**

Über den optionalen und mehrwertigen Query-Parameter "**filter**" kann die Objektliste gefiltert werden. Ein Filter hat zwei mögliche Formen:

1. <Spalten-Name/Spalten-Nr.> ~ <Operator> ~ <Wert>
2. <Spalten-Name/Spalten-Nr.> ~ <Wert>

Mögliche Operatoren sind: equal, notEqual, greater, less, greaterOrEqual, lessOrEqual, equalCardinality, containsPhrase, covers, isCoveredBy, distance, fulltext, equalGeo, equalPresentTime, greaterOverlaps, greaterPresentTime, lessOverlaps, lessPresentTime, equalMaxCardinality, equalMinCardinality, overlaps, unmodifiedEqual.

**Sortierung**

Über den optionalen und mehrwertigen Query-Parameter "**sort**" kann die Objektliste sortiert werden. Die Reihenfolge der Sortierparameter bestimmt die Sortierpriorität. Die Angabe der Sortierung hat zwei mögliche Formen:

1. <Spalten-Name>
2. {-}<Spalten-Nr.>

Stellt man in Variante 2 ein Minus vor, wird absteigend sortiert, sonst aufsteigend.

**Startmenge der Liste setzen**

Über den optionalen und QueryParameter "**elements**"kann eine Komma-separierte Liste von Topic-Referenzen übergeben werden, die als Listenelemente verwendet werden sollen.

Da die Liste der Element ggf. sehr lang ist, kann der Request auch als POST geschickt und die Parameter als Form-Parameter übergeben werden.

**Startmenge der Liste über KPath setzen**

Über die optionalen Query-Parameter "**elementsPath**" und "**startTopic**" können die initialen Elemente der Liste berechnet werden. Sind diese Parameter nicht gegeben, besteht die Ausgangsmenge aus allen Individuen bzw. allen Unterbegriffen (im Falle einer Begriffs-Objektliste) des per "conceptLocator" festgelegten Begriffs.

Dabei ist "elementsPath" ein KPath-Ausdruck und "startTopic" eine Referenz auf das Topic, mit dem die Auswertung des KPath gestartet werden soll. Die Form des Parameters "startTopic" entspricht der des "conceptLocator".

**Vererbung**

Über den optionalen Query-Parameter "**disableInheritance**" kann die Vererbung unter-

drückt werden. Der Paramater macht nur Sinn, wenn kein "elementsPath" gesetzt ist.

**JSON-Ausgabeformat (Beispiel)**


```
{
 rows: [{
  topicID: "ID850_337000250",
  row: ["CS",
  "Schuckmann",
  "Christian",
  "240",
  "c.schuckmann@i-views.de",
  "10",
  "",
  "",
  "IT-Cluster P4, IT-Cluster P5"]
 },
 {
  topicID: "ID12068_152328826",
  row: ["",
  "Smith",
  "John",
  "",
  "",
  "",
  "",
  "",
  ""]
 },
 {
  topicID: "ID2438_366678497",
  row: ["SY",
  "Stoye",
  "Sabine",
  "421",
  "s.stoye@i-views.de",
  "",
  "",
  "",
  "Knowledge-Portal 3, Presales, SemGoRiCo, Support (HSNR)"]
 }],
 columnDescriptions: [{
  label: "Login",
  type: "string",
  columnId: "1"
 },
 {
  label: "Nachname",
  type: "string",
  columnId: "2"
 },
 {
  label: "Vorname",
  type: "string",
```

```
  columnId: "3"
 },
 {
  label: "Telefon",
  type: "string",
  columnId: "4"
 },
 {
  label: "Email",
  type: "string",
  columnId: "5"
 },
 {
  label: "Verfügbarkeit",
  type: "number",
  columnId: "6"
 },
 {
  label: "Aufwand",
  type: "string",
  columnId: "7"
 },
 {
  label: "erstellt am",
  type: "dateTime",
  columnId: "8"
 },
 {
  label: "Projekt",
  type: "string",
  columnId: "9"
 }]
}
```


### 3.3.4 Objeklistendruckvorlage

Über den folgenden Pfad kann eine Objektliste in eine 'Druckvorlage für Liste' (siehe Kapitel 'Schema der Druckvorlagen) gefüllt und das Resultat heruntergeladen werden:

**http://{server:port}/baseService/{conceptLocator}/objectList/printTemplate/{templateLocator}/{filenar**

Der Service funktioniert genau wie der Abruf einer Objektliste, trägt aber als zuätzlichen Parameter eine Referenz auf das Individuum des Begriffs 'Druckvorlage für Liste' im Wissensnetz.

"**templateLocator**" muss eines der unter "Allgemeines" beschriebenen Formate haben

Der optionale Pfad-Parameter "filename" wird nicht ausgewertet und dient dem besseren Browser-Handling.

Über das Header-Field "**Accept**" wird gesteuert, in welches Ausgabeformat konvertiert werden soll. Fehlt das Header-Field oder ist der Wert "***/***", findet keine Konvertierung statt. Mehrwertige Accept werden nicht unterstützt und resultieren in einer Fehlermeldung.

Über den optionalen Query-Parameter "**targetMimeType**" kann der Wert des "Accept" Header-Fields überschrieben werden. Dies ist notwendig, wenn man den Request aus einem Browser

aufrufen möchte und dort keinen Einfluss auf die Header-Fields hat.

### 3.3.5 Topic drucken

Über den folgenden Pfad kann ein Topic in ein Drucklistentemplate gefüllt und das Resultat heruntergeladen werden:

**http://{server:port}/baseService/{topicLocator}/printTemplate/{templateLocator}/{filename}**

"**templateLocator**" muss eines der unter "Allgemeines" beschriebenen Formate haben

Der optionale Pfad-Parameter "filename" wird nicht ausgewertet und dient dem besseren Browser-Handling.

Über das Header-Field "**Accept**" wird gesteuert, in welches Ausgabeformat konvertiert werden soll. Fehlt das Header-Field oder ist der Wert "***/***", findet keine Konvertierung statt. Mehrwertige Accept werden nicht unterstützt und resultieren in einer Fehlermeldung.

Über den optionalen Query-Parameter "**targetMimeType**" kann der Wert des "Accept" Header-Fields überschrieben werden. Dies ist notwendig, wenn man den Request aus einem Browser aufrufen möchte und dort keinen Einfluss auf die Header-Fields hat.

### 3.3.6 Dokumentformatkonvertierung

Über den folgenden Pfad kann ein Dokument in ein anderes Format umgewandelt werden (z.B. odt in pdf):

**http://{server:port}/baseService/jodconverter/service**

Der Service bildet den JOD-Konverter (siehe http://sourceforge.net/projects/jodconverter/) ab und dient der Abwärtskompatibilität für Installationen, die bisher mit dem JOD-Konverter betrieben wurden.

Damit der Service funktioniert muss LibreOffice (ab Version 4.0) installiert sein.

**Wichtig**: Apache OpenOffice ist nicht geeignet, da es kein Kommandozeilen-Interface zur Konvertierung bereitstellt

Die Konfigurationsdatei "bridge.ini" muss einen Eintrag enthalten, der auf die Datei "soffice" verweist:

```
[file-format-conversion]
sofficePath="C:\Program Files (x86)\LibreOffice 4.0\program\soffice.exe"
```