

Wissensnetzwerke im Grid

# Ressourcen-Föderator

Arbeitspaket 2.1

Januar 2011

Denis Hünich



---

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>2</b>
<b>2</b>	<b>Architektur</b>	<b>3</b>
<b>3</b>	<b>Installationsanleitung</b>	<b>4</b>
<b>4</b>	<b>Konfiguration</b>	<b>5</b>
4.1	Ressourcen-Föderator-Einstellungen - adapter-config.xml . . . . .	5
4.2	Server-Einstellungen - server.config . . . . .	8
4.3	Logging-Einstellungen - log4j.properties . . . . .	8
<b>5</b>	<b>Verfügbare Connector</b>	<b>9</b>
5.1	UNICORE . . . . .	9
5.2	iRODS . . . . .	10
5.3	MySQL . . . . .	11
5.4	PostgreSQL . . . . .	12
5.5	Stellaris . . . . .	13
<b>6</b>	<b>Betrieb</b>	<b>14</b>

## 1 Einleitung

Dieses Dokument beschreibt die Komponente Ressourcen-Föderator der WisNetGrid-Infrastruktur. Mit dieser Komponente ist es möglich Datenquellen wie Datenbanken (z.B. MySQL) oder Datenmanagementsysteme(z.B. iRODS) anzubinden und in einem einheitlichen Namensraum über WebDAV<sup>1</sup> anzuzeigen. Dieses Dokument beschreibt kurz die Architektur und gibt danach eine Installationsanweisung. Weiterhin wird die Konfiguration des Ressourcen-Föderators und die verfügbaren Connector (siehe Abschnitt 5) beschrieben. Zum Schluß wird die Inbetriebnahme des Ressourcen-Föderators erklärt.

---

<sup>1</sup>Spezifikation zu finden unter: <http://www.webdav.org/specs/rfc4918.html>

---

## 2 Architektur

Der Ressource-Föderator bindet Ressourcen wie z.B. Grid-Datenmanagement- oder Datenbanksysteme an und stellt sie als einheitlichen Namensraum dar. Wie im Bild 1 ersichtlich ist der Ressourcen-Föderator in die folgenden drei Komponenten eingeteilt:

- Jetty-Webserver (Jetty)<sup>2</sup>
- Apache Camel Routing Engine (Camel)<sup>3</sup>
- Milton WebDAV-Server (Milton)<sup>4</sup>

*Jetty* ist ein in Java geschriebener Webserver und nimmt die Anfragen des Nutzers entgegen und gibt sie über ein Servlet an *Camel* weiter. *Camel* ist eine regelbasierte Routing- und Konvertierungsengine und ermöglicht die Definition einer Route bestehend aus Endpunkten und sogenannten *Processor*. Durch die Route wird ein Container (Exchange) geschickt der Eigenschaften und Header gekapselt in Ein- und Ausgabenachrichten (Messages) verwalten kann. *Manager* (siehe 1) erzeugt solch ein *Exchange* Objekt und über die folgenden *Processor* zum Endpunkt, welcher *Milton* startet, geschickt:

- Preparation
- Location
- Credentials
- Milton-ResourceFactory

Die einzelnen *Processor* bearbeiten die Anfrage und bereiten die nötigen Eingabedaten für Milton auf. *Preparation* liest die angefragte URI aus und löst unter anderem den Namen für die angefragte Datenquelle, sowie den relativen Pfad innerhalb der Datenquelle heraus. Zudem werden Informationen über den anfragenden Nutzer (z.B. Username, Session-Id) bereitgestellt. Die gesammelten werden in *Exchange* gespeichert. In *Location* werden mithilfe des gefundenen Datenquellennamens Informationen zur angeforderten Datenquelle (spezifiziert in einer Konfigurationsdatei, siehe Abschnitt ) geladen und an *Exchange* weitergegeben. *Credentials* lädt die nötigen Credentials um den Nutzer bei der angefragten Datenquelle zu autorisieren (authentifizieren) und legt sie ebenfalls in *Exchange* ab. Der letzte Processor *Milton-ResourceFactory* fordert den nötigen Connector, um mit der Datenquelle interagieren zu können, an. Dieser liegt in *Milton* verständlicher Form vor und wird ebenfalls in *Exchange* gespeichert. Am Endpunkt der Route werden die in *Exchange* gesammelten Informationen an *Milton* übergeben und *Milton* gestartet. *Milton* überprüft die Art der Anfrage (z.B. Lesen oder Schreiben) und gibt sie zusammen mit dem relativen Pfad der Datenquelle an den Connector weiter. Dieser führt die

---

<sup>2</sup><http://jetty.codehaus.org/jetty/>

<sup>3</sup><http://camel.apache.org/>

<sup>4</sup><http://milton.ettrema.com/>

Anfrage aus und *Milton* gibt die Antwort WebDAV konform über *Camel* und *Jetty* an den Nutzer zurück. Tritt ein Fehler innerhalb der Route auf wird ein *Processor* gestartet welcher den Fehler ausliest und einen HTTP-Status-Code samt Fehlerbeschreibung generiert. Dieser wird dann an den Nutzer (über den *Jetty*) übergeben und die Route abgebrochen.

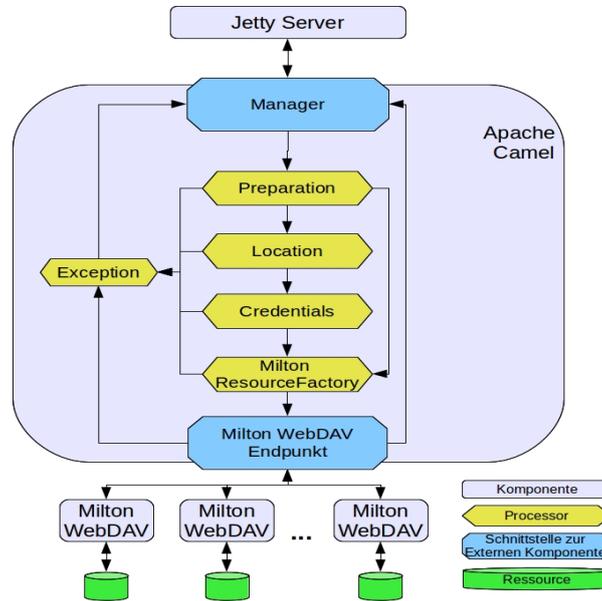


Abbildung 1: Struktur der Ressourcen-Föderators

### 3 Installationsanleitung

Die Datei `resource-federator-x.x.x.zip` in den gewünschten Ordner entpacken. Nach dem entpacken befinden sich folgende Ordner im Verzeichnis:

**bin** Start- und Stop-Skripte

**conf** Konfigurationsdateien

**doc** Installationsanweisung und andere Dokumentationen

**lib** Java Bibliotheksdateien (jar files)

**logs** Log-Dateien

Für den Ressourcen-Föderator muß Java JDK von Oracle oder OpenJDK in der Version 6 oder später vorliegen.

---

## 4 Konfiguration

Im Verzeichnis *\$Installationsverzeichnis/conf* befinden sich die Dateien:

**adapter-config.xml** Enthält die Konfiguration für die einzelnen Connector und der Sicherheitsinfrastruktur

**server.config** Konfiguration für den Webserver

**logging.properties** Konfiguration für das Logging des Ressourcen-Föderators

**server-keystore.jks** Keystore mit den privaten Schlüsseln und vertrauenswürdigen Zertifikaten von den Servern (z.B. UNICORE)

**server.pem** Öffentlicher Schlüssel des Servers (passend zum gespeicherten privaten Schlüssel im Keystore)

### 4.1 Ressourcen-Föderator-Einstellungen - adapter-config.xml

Die Datei *adapter-config.xml* konfiguriert im oberen Teil die Anbindung an die Sicherheitsinfrastruktur[1] und anschließend die Connector. Im folgenden werden die Konfigurationsmöglichkeiten Stückweise an Beispielen erklärt.

#### 4.1.1 Login-URL für den SSO-Server

Um den WisNetGrid-Nutzer autorisieren und seine gespeicherten Credentials abrufen zu können ist es notwendig einen SSO-Server anzugeben. Die URL für den SSO-Server ist beim zuständigen Administrator zu erfragen oder bei einem eigenen SSO-Server dessen URL zu nutzen.

```
<s:bean id="wisnetgrid.integrator.server.sso.loginurl" class="java.lang.String">
  <s:constructor-arg value="https://127.0.0.1:9999/sso-server/login"/>
</s:bean>;
```

**Aufistung 1:** SSO-Server-Konfiguration

#### 4.1.2 REST-Service-URL

Der SSO-Server schaltet zusätzlich eine URL für REST-Services frei. Diese URL ist ebenfalls beim zuständigen Administrator zu erfragen oder bei einem eigenen die festgelegte URL zu nutzen.

```
<s:bean id="wisnetgrid.integrator.server.sso.serviceurl" class="java.net.URL">
  <s:constructor-arg value="https://127.0.0.1:8183/sso-service"/>
</s:bean>
```

**Aufistung 2:** REST-Konfiguration

### 4.1.3 Agent für den Zugriff auf die SSO-Komponenten

Konfiguriert einen Agenten der den Zugriff auf die SSO-Komponenten vornimmt.

Argumente	Beschreibung
1	Die zuvor definierte REST-Service-URL
2	Pfad zum Keystore (bei Nutzung von SSL nötig)
3	Passwort für den Keystore (bei Nutzung von SSL nötig)
4	Pfad zum Truststore (bei Nutzung von SSL nötig)
5	Passwort für den Truststore (bei Nutzung von SSL nötig)
6	Zeitintervall zum Säubern des Session-Caches
7	Zeitintervall zum Auffrischen des Session-Caches

```
<s:bean id="wisnetgrid.integrator.server.sso.agent" class="wisnetgrid.
  security.sso.client.advanced.AuthenticatingSSOAgentImpl" depends-on="
  wisnetgrid.integrator.server.sso.serviceurl">
  <s:constructor-arg ref="wisnetgrid.integrator.server.sso.serviceurl"/>
  <s:constructor-arg value="conf/server-keystore.jks"/> <!-- SSL -->
  <s:constructor-arg value="the!agent"/> <!-- SSL -->
  <s:constructor-arg value="conf/server-keystore.jks"/> <!-- SSL -->
  <s:constructor-arg value="the!agent"/> <!-- SSL -->
  <s:constructor-arg value="60"/> <!-- cached sessions clean-up rate in
    seconds -->
  <s:constructor-arg value="180"/> <!-- cached sessions refresh rate in
    seconds -->
</s:bean>
```

#### Aufistung 3: SSO-Agent-Konfiguration

### 4.1.4 SSO-User-Realm

Konfiguriert den SSO-User-Realm.

Argumente	Beschreibung
1	Name des SSO-User-Realms
2	Übergabe des zuvor definierten SSO-Agents

```
<s:bean id="wisnetgrid.integrator.server.userRealm" class="wisnetgrid.
  adapter.server.security.SSOUserRealm" depends-on="wisnetgrid.integrator
  .server.sso.agent">
  <s:constructor-arg value="WisNetGrid_Data_Integrator"/>
  <s:constructor-arg ref="wisnetgrid.integrator.server.sso.agent"/>
</s:bean>
```

#### Aufistung 4: SSO-User-Realm-Konfiguration

### 4.1.5 Verschlüsselung von Credentials

Diese Konfiguration ist nötig um Credentials welche mit dem öffentlichen Schlüssel des Ressourcen-Föderators verschlüsselt wurden wieder zu entschlüsseln.

Argumente	Beschreibung
1	Pfad zum Keystore mit dem privaten Schlüssel
2	Passwort für den im Keystore gespeicherten privaten Schlüssel
3	Alias für den im Keystore gespeicherten privaten Schlüssel

```
<s:bean id="wisnetgrid.integrator.server.credential.keystore" class="
  wisnetgrid.adapter.core.security.CredentialDataKeystore">
  <s:constructor-arg value="conf/server-keystore.jks" />
  <s:constructor-arg value="the!agent" />
  <s:constructor-arg value="demo_agent" />
</s:bean>
```

#### Aufistung 5: Verschlüsselung von Credentials

### 4.1.6 Angabe der Connector

Die zu nutzenden Connector (siehe Abschnitt 5) können hier definiert werden und werden beim Start des Ressourcen-Föderators geladen. Beispielhaft ist ein Connector mit aufgeführt.

```
<s:bean id="loc1" class="wisnetgrid.adapter.connectors.sql.mysql.
  MYSQLLocation">
  <s:constructor-arg value="mysql-test" />
  <s:constructor-arg value="localhost" />
  <s:constructor-arg value="3306" />
  <s:constructor-arg value="wisnetgrid" />
</s:bean>

<s:bean id="wisnetgrid-locator" class="wisnetgrid.adapter.locator.Locator"
  scope="singleton">
  <s:property name="initialLocations">
    <s:list>
      <s:ref bean="loc1" />
      <!-- ... -->
    </s:list>
  </s:property>
</s:bean>
```

#### Aufistung 6: Angabe der genutzten Connector

Der Rest der Konfigurationsdatei sollte nur von Entwicklern oder erfahrenen Nutzern geändert werden.

## 4.2 Server-Einstellungen - server.config

Die Einstellungen für den Webserver werden in der Datei server.config vorgenommen. Der Nutzer hat dafür die folgenden Einstellungsmöglichkeiten:

Name	Standardwert	Beschreibung
wisnetgrid.adapter.routes.config	conf/adapter-config.xml	Pfad zur Konfigurationsdatei des Ressourcen-Föderators
wisnetgrid.integrator.server.port	8080	Port über den der Webserver erreichbar sein soll
wisnetgrid.integrator.server.host	127.0.0.1	Host auf dem der Webserver zu erreichen sein soll
wisnetgrid.integrator.server.ssl	false	Kommunikation über SSL (true/false)
wisnetgrid.integrator.server.ssl_port	8443	Port für die SSL-Kommunikation
wisnetgrid.integrator.server.ssl.keystore	-	Pfad zum Keystore mit dem Schlüssel für SSL
wisnetgrid.integrator.server.ssl.keystorePassword	-	Passwort für den Keystore

## 4.3 Logging-Einstellungen - log4j.properties

Der Ressourcen-Föderator verwendet das Logging-Framework Apache Log4J, an dessen Dokumentation <sup>5</sup> für tiefgreifende Einstellungsoptionen verwiesen wird. Diese Datei erfordert nach der Installation (siehe Abschnitt 3) keinerlei Änderungen. Dennoch zeigt die nachfolgende Tabelle etwaige Anpassungsmöglichkeiten auf.

<sup>5</sup>Log4J-Dokumentation - <http://www.jajakarta.org/log4j/jakarta-log4j-1.1.3/docs/documentation.html>

---

Name	Standardwert	Beschreibung
log4j.rootLogger	INFO, Appender	Anstelle <i>INFO</i> sind noch folgende Log-Level möglich: <ul style="list-style-type: none"> <li>• OFF</li> <li>• WARN</li> <li>• TRACE</li> <li>• FATAL</li> <li>• ERROR</li> <li>• DEBUG</li> <li>• ALL</li> </ul>
log4j.appender.Appender.File	logs/resc-federator.log	Pfad zur zu füllenden Log-Datei

## 5 Verfügbare Connector

Diese Abschnitt beschreibt die bestehenden Connector und die Einstellungsmöglichkeiten für den Administrator des Ressourcen-Föderators. Die Einstellung werden in der Konfigurationsdatei (Abschnitt 4.1) des Adapters definiert. Im folgenden werden die Connector und ihre Konfigurationsmöglichkeiten erklärt und an ein Code-Beispiel gegeben.

### 5.1 UNICORE

Mit diesem Connector kann die Datenverwaltung der UNICORE-Middleware angesprochen werden.

Operation	Daten	Metadaten
Lesen	JA	NEIN
Schreiben	JA	NEIN

Parameter	Beschreibung	Konstruktor-argument
name	Frei wählbarer Name für die angebundene Datenquelle. Dieser Name wird als Teil der URI benutzt und muß einzigartig für den jeweiligen Ressourcen-Föderator sein.	1
endpointURL	URL für die Datenquelle	nein
endpointRefType	Reference-Endpoint-Type	nein
keystore	Pfad zum Keystore	nein
keystoreAlias	Alias vom Key/Zertifikat	nein
keystorePassword	Passwort für den Key/Zertifikat	nein
keystoreType	Typ des Keystores	nein
truststore	Pfad zum Truststores	nein
truststorePassword	Passwort für den Truststore	nein
needDelegation	Delegation (true/false)	nein

```

<s:bean id="loc1" class="wisnetgrid.adapter.connectors.unicore.
  UNICORELocation">
  <s:constructor-arg value="unicore-test" />
  <s:property name="endpointURL" value="https://localhost:8080/WISNETGRID
    /services/StorageManagement?res=default_storage" />
  <s:property name="keystore" value="conf/server-keystore.jks" />
  <s:property name="keystorePassword" value="the!agent" />
  <s:property name="keystoreAlias" value="agent" />
  <s:property name="needDelegation" value="true" />
</s:bean>

```

### Auflistung 7: UNICORE Connector

## 5.2 iRODS

Dieser Connector bindet das Grid-Datenmanagementsystem iRODS an. Es werden die Dateien und Ordner des gewählten Home-Verzeichnisses von iRODS wiedergegeben und über *PROPFIND/PROPPATCH* Operationen auf die Metadaten ausgeführt.

Operation	Daten	Metadaten
Lesen	JA	JA
Schreiben	JA	JA

Parameter	Beschreibung	Konstruktor-argument
name	Frei wählbarer Name für die angebundene Datenquelle. Dieser Name wird als Teil der URI benutzt und muß einzigartig für den jeweiligen Ressourcen-Föderator sein.	1
endpointURL	URL für die Datenquelle	nein
host	Name/IP vom Host	2
port	Port-Nummer	3
zone	Name der iRODS Zone	4
resource	Name der iRODS Ressource	5
homeDir	Pfad des iRODS Home-Verzeichnisses	6

```

<s:bean id="irods" class="wisnetgrid.adapter.connectors.irods.IRODSLocation">
  <s:constructor-arg value="irods-test" />
  <!-- Host -->
  <s:constructor-arg value="localhost" />
  <!-- Port -->
  <s:constructor-arg value="1247" />
  <!-- iRODS Zone -->
  <s:constructor-arg value="tempZone" />
  <!-- iRODS Resource -->
  <s:constructor-arg value="tempResc" />
  <!-- iRODS Home-Verzeichnis -->
  <s:constructor-arg value="/tempZone/home/wisnetgrid" />
</s:bean>

```

### Aufistung 8: iRODS Connector

## 5.3 MySQL

Connector für MySQL-Datenbanken. Tabellen werden als Verzeichnis dargestellt und Einträge werden in der eingestellten Ansicht dargestellt.

Operation	Daten	Metadaten
Lesen	JA	NEIN
Schreiben	NEIN	NEIN

Parameter	Beschreibung	Konstruktor-argument
name	Frei wählbarer Name für die angebundene Datenquelle. Dieser Name wird als Teil der URI benutzt und muß einzigartig für den jeweiligen Ressourcen-Föderator sein.	1
endpointURL	URL für die Datenquelle	nein
host	Name/IP vom Host	2
port	Port-Nummer	3
database	Name der gewünschten Datenbank	4
view	Auswahl der Ansicht für die Tabelleneinträge. Zur Zeit sind die folgenden zwei Ansichten auswählbar: <ul style="list-style-type: none"> <li>• Aufteilung der Einträge in mehreren Dateien mit je 1000 Einträgen (z.B. 1-1000.xml) (Standard wenn keine Ansicht definiert wurde)</li> <li>• Alle Einträge in eine Datei (entries.xml)</li> </ul>	nein

```

<s:bean id="sql" class="wisnetgrid.adapter.connectors.sql.mysql.
  MySQLLocation">
  <!-- Name der Datenquelle -->
  <s:constructor-arg value="mysql-test" />
  <!-- Host -->
  <s:constructor-arg value="localhost" />
  <!-- Port -->
  <s:constructor-arg value="3306" />
  <!-- Name der Datenbank -->
  <s:constructor-arg value="test" />
  <!-- Auswahl der Ansicht -->
  <s:property name="view" value="wisnetgrid.adapter.connectors.sql.
    SQLResourceViewAll" />
</s:bean>

```

**Auflistung 9:** MySQL Connector

## 5.4 PostgreSQL

Die Einstellungsmöglichkeiten für den Connector zu einer PostgreSQL-Datenbank sind bis auf die Definition des Connectors identisch zu den MySQL-Connector-Einstellungen (siehe 5.3).

Operation	Daten	Metadaten
Lesen	JA	NEIN
Schreiben	NEIN	NEIN

```
<s:bean id="sql" class="wisnetgrid.adapter.connectors.sql.mysql.
  PostgreSQLLocation">
  <!-- Name der Datenquelle -->
  <s:constructor-arg value="postgresql-test" />
  <!-- Host -->
  <s:constructor-arg value="localhost" />
  <!-- Port -->
  <s:constructor-arg value="3306" />
  <!-- Name der Datenbank -->
  <s:constructor-arg value="test" />
  <!-- Auswahl der Ansicht -->
  <s:property name="view" value="wisnetgrid.adapter.connectors.sql.
    SQLResourceViewAll" />
</s:bean>
```

#### Auflistung 10: PostgreSQL Connector

## 5.5 Stellaris

Connector für das Metadatenmanagementsystem Stellaris.

Operation	Daten	Metadaten
Lesen	JA	NEIN
Schreiben	JA	NEIN

Parameter	Beschreibung	Konstruktor-argument
name	Frei wählbarer Name für die angebundene Datenquelle. Dieser Name wird als Teil der URI benutzt und muß einzigartig für den jeweiligen Ressourcen-Föderator sein.	1
endpointURL	URL für die Datenquelle	nein
host	Name/IP vom Host	2

```
<s:bean id="sql" class="wisnetgrid.adapter.connectors.stellaris.
  STELLARISLocation">
  <!-- Name der Datenquelle -->
  <s:constructor-arg value="stellaris-test" />
  <!-- Host -->
  <s:constructor-arg value="localhost" />
```

#### Auflistung 11: Stellaris Connector

---

## 6 Betrieb

Nach Installation und korrekter Konfiguration lässt sich der Ressourcen-Föderator mittels der Skripte unter *\$Installationsverzeichnis/bin* in Betrieb nehmen.

Betriebssystem	Start	Neustart	Stop
Unix/Linux	resc-federator.sh start	resc-federator.sh restart	resc-federator.sh stop
Windows	start.bat	-	CTRL + C bzw. entsprechenden Java- Prozess über den Taskmanager stoppen

Etwaige Probleme beim Start sind standardmäßig unter *\$Installationsverzeichnis/logs/adapter.log* und *\$Installationsverzeichnis/logs/startup.log* einsehbar (siehe Abschnitt 4.3).

## Literatur

[1] Jason Milad Daivandy. *Single Sign-On Server*, 2012. Dokumentation.