GEFÖRDERT VOM
Bundesministerium
für Bildung
und Forschung

WissGrid

Dokumentation

Arbeitspaket 3 - Langzeitarchivierung von Forschungsdaten

Integration der Systeme Fedora Commons und iRODS[1]

Autoren	Arbeitspaket 3 - Langzeitarchivierung von Forschungsdaten
Editoren	Panzer, Samadi
Datum	13.04.2012
Dokument Status	Endfassung
Dokument Version	1.0

[1]Diese Arbeit wurde vom WissGrid Projekt erstellt. Dieses Projekt wird gefördert vom Bundesministerium für Bildung und Forschung (BMBF).

Inhaltsverzeichnis

1 Einleitung

Das Arbeitspaket 3 (AP3) im WissGrid-Projekt[2] beschäftigt sich u.a. mit Fragen der Langzeitarchivierung (LZA) in der Bandbreite von für sich stehenden, einfachen Datenarchiven, bis zur Integration von (virtuellen) Forschungsumgebungen mit Repositorien. Im Rahmen des AP3 wird dazu ein Software-Stack zur langfristigen und vertrauensvollen Verwaltung[3] großer Datenmengen entwickelt, welcher von Fach-Communities, in D-Grid installiert und an die eigenen Bedürfnisse anpasst werden kann.

Ein Kernproblem bei der Entwicklung einer Langzeitarchivierungsarchitektur ist, dass die Anwendungsbereiche zwar den Bedarf an einem Repository zur langfristigen Verwaltung und Speicherung ihrer Daten teilen, sich aber in ihren Nutzungsszenarien und Integrationsmöglichkeiten eines Repositories sehr unterscheiden. Aufgrund der Verschiedenheit der Fach-Communities und ihrer Anforderungen, wird eine Differenzierung in verschiedene Anwendungsprofile vorgenommen, die unterschiedliche Integrationsvarianten behandeln, womit die wesentlichen Nutzungsszenarien abdecken werden. Der entwickelte Software-Stack unterstützt die nachfolgend aufgeführten Anwendungsprofile und die damit verbundenen Integrationsmöglichkeiten, gemäß den Ausführungen in der Spezifikation einer generischen Langzeitarchitektur[4] und der Grid-Repository Spezifikation[5]:

- "Repositorien als Archiv-Backend für das Grid" (sog. Profil A)
  Kernanforderung des Repositories ist hier die Unterstützung von Grid-Workflows (Grid-Jobs). Daten-Objekte werden vornehmlich im Grid erzeugt und bearbeitet, weshalb die Repository-Anbindung an das Grid durch entsprechende Protokolle notwendig ist.
- "Daten-Grid als Storage für Repositorien" (sog. Profil B)
  Hier ist die Anbindung des Repositories an Nutzerumgebungen die Kernanforderung. Daten-Objekte werden in (interaktiven, kollaborativen) Nutzerumgebungen erzeugt und bearbeitet. Die Unterstützung üblicher Schnittstellen- bzw. Kommunikationsmechanismen, zur Interaktion zwischen Nutzerumgebung und Repository, ist bei diesem Profil besonders wichtig.

In den nun folgenden Unterkapiteln wird vertiefend auf die Eigenschaften und Anwendungsszenarien der beiden Profile eingegangen.

1.1 Repositorien als Archiv-Backend für das Grid

Diese Integrationsvariante wird für Anwendungsfälle eingesetzt, bei denen in einem Repository gespeicherte Daten-Objekte (Dateien) direkt, möglichst einfach und effizient in

[2] http://www.wissgrid.de/

[3] Der Begriff Verwaltung umfasst dabei auch Aufgaben, die über CRUD-Operationen (create, read, update und delete) hinaus gehen, z.B. das Anlegen und Pflegen von Metadaten, Beziehungen oder Indizes, die Realisierung von Bit Preservation Aspekten, sowie das Bereitstellen entsprechender Schnittstellen.

[4] Generischen Langzeitarchivierung für D-Grid. WissGrid, 2010

[5] WissGrid-Spezifikation: Grid-Repository. WissGrid, 2010

Grid-Jobs bzw. Grid-Workflows eingebunden werden können sollen. Wesentlich hierfür ist die Referenzierbarkeit von Objekten mit Hilfe geeigneter persitenter Identifikatoren (PIDs) und die Unterstützung, im Grid anwendbarer Kommunikationsmechanismen, d.h. die Unterstützung von Grid-Protokollen für den Import und Export (z.B. GridFTP, iRODS iCommands). Die Grid-Repository Spezifikation bezeichnet dieses Anwendungsszenario mit Profil A.

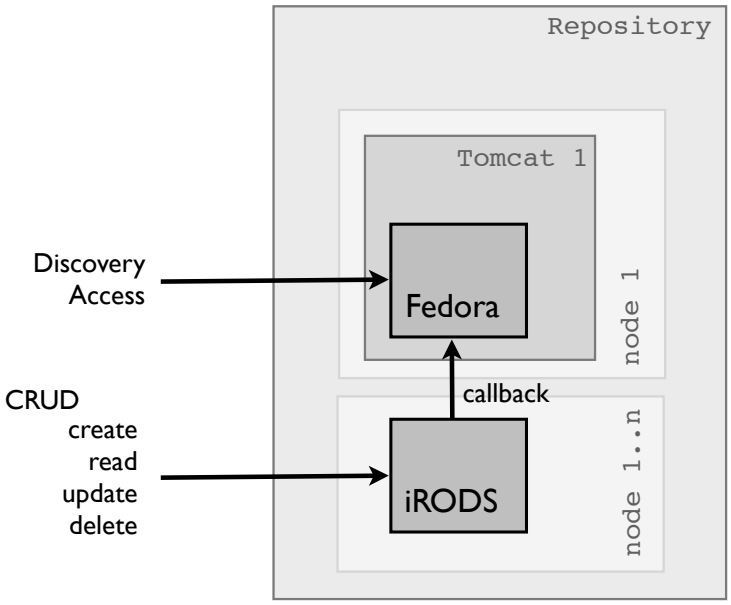


Abbildung 1: Repositorien als Archiv-Backend für das Grid

Community-Anforderungen: Die *Erzeugung und Verarbeitung großer Datenmengen im Grid*, deren ständige Verfügbarkeit, sowie ihre vertrauenswürdige Aufbewahrung über lange Zeiträume.

Anwendungsszenario: Grid-Jobs beziehen Daten direkt aus dem Repository und schreiben ihre Ergebnisse auch dorthin zurück. Ein explizites "staging" der Daten vor der Verarbeitung und übergabe an den Dienst entfällt, da sie über die PID referenziert oder über ein iRODS iCommand bezogen und in die Job-Beschreibung (-Ausführung) integriert werden können.

Technische Umsetzung: In der Grid-Repository Spezifikation wird iRODS[6] als System zur technischen Umsetzung festgelegt. iRODS ist ein regel-basiertes Data-Hosting- und Storage-System. Wegen seines eingeschränkten Schnittstellen-Angebots wird es jedoch oft sinnvoll sein, dem iRODS-Server einen Fedora[7]-Server voranzustellen, um von dessen Funktionalitäten zu profitieren. Fedora ist ein Daten- und Metadatenmanagementsystem, das Funktionalitäten zum Verwaltung von Daten-Objekten, Metadaten und Beziehungen bietet; es ermöglicht die Strukturierung von Objekten und deren Klassifizierung; es erlaubt die Erzeugung virtueller Repräsentationen zur Laufzeit (d.h. abgeleiteter Objekte); und bietet eine breitere Schnittstelle als iRODS. In dem kooperativen Sys-

[6] Integrated Rule-Oriented Data System
[7] Flexible Extensible Data Object Architecture

tem, bestehend aus iRODS und Fedora, dient Fedora quasi als Metadatenkatalog und Exportschnittstelle für iRODS und unterstützt mit seinen Funktionen die Aufgabenbereiche Content Preservation und Data Curation. iRODS behandelt Aspekte der Bit Preservation, wie z.B. die Durchführung von Integritätschecks

1.2 Daten-Grid als Storage für Repositorien

Hierbei handelt es sich um eine Integrationsvariante, bei der die Integration des Repositories mit anderen Anwendungen, wie z.B. Forschungsumgebungen, im Vordergrund steht. Die Grid-Repository Spezifikation sprich dabei vom Profil B, welches von den Anwendungsfällen geleitet ist, die Grid bzw. D-Grid Storage Resourcen zum Abspeichern und verlässlichen Aufbewahren von Objekten nutzen wollen. Im Kontext dieser Integrationsvariante werden Objekte insbesondere in interaktiven Umgebungen erzeugt und kollaborativ bearbeitet. Dazu ist es auf Repository-Seite erforderlich, Schnittstellen anzubieten, die auf offenen und standardisierten Technologien beruhen (auf Basis von REST, SOAP oder JMS), um eine leichte und komfortable Anbindung von existierenden Nutzerumgebungen an das Repository zu unterstützen. Die Interoperabilität mit anderen Anwendungen steht somit im Vordergrund.

Community-Anforderungen: Auch hier ist die Verwaltung großer Datenmengen eine wesentliche Anforderung, wobei insbesondere (viele) kleine Objekte und zugehörige bzw. abgeleitete Erschließungsmaterialien (z.B. XML-Daten oder Annotationen zu Bildern) zu verwalten sind. Darüber hinaus ist es aus Nutzersicht erforderlich, eigene Objekt-Modelle (in Fedora-Terminologie: Content-Modelle) definieren, Objekten Metadaten zuweisen, sowie Beziehungen zwischen Objekten festlegen zu können.

Anwendungsszenario: Es werden Daten in einer (interaktiven) Nutzerumgebung (kollaborativ) erzeugt oder bearbeitet.

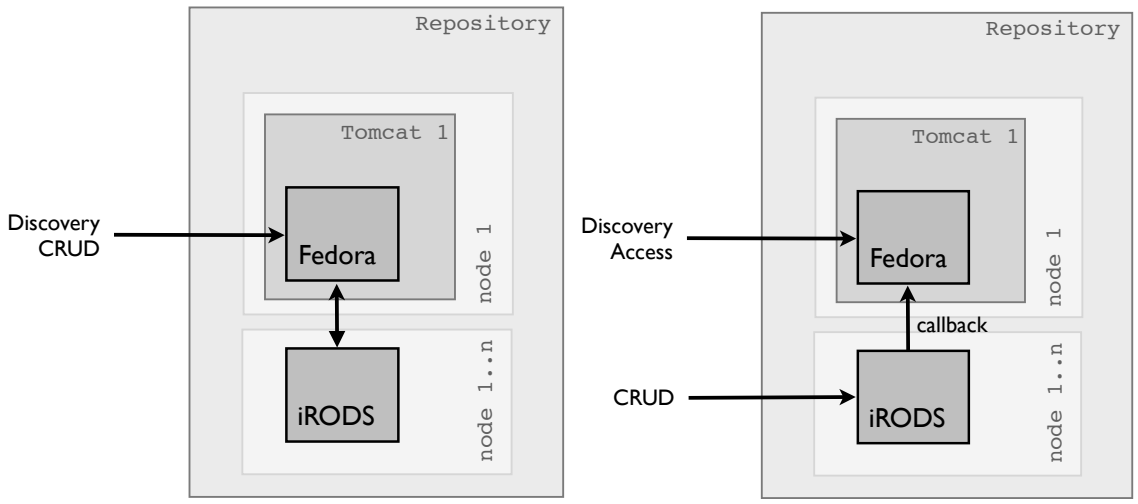
Technische Umsetzung: In der Grid-Repository Spezifikation wurde die Kombination von iRODS und Fedora zur technischen Umsetzung bereits vorgegeben.

Bei dieser Integrationsvariante lassen sich die beiden, in Abbildung 2 dargestellten Varianten unterscheiden, wobei die Unterschiede in der Zugriffsart und der Sichtbarkeit des iRODS-Servers liegen:

a) Fedora mit *transparentem* iRODS als Backend, d.h. der Zugriff erfolgt ausschließlich über Fedora (Web-Schnittstelle). Der Einsatz von iRODS als Speicherungs-Lösung bleibt für Nutzer verborgen.

b) Fedora mit *sichtbarem* iRODS als Backend. Hier ist eine direkte Interaktion mit dem iRODS-System möglich, d.h. der Zugriff kann auch direkt über das iRODS-System (per iCommands[8] oder Jargon-API[9]) erfolgen. Dies kann wünschenswert sein, um z.B. ganze Verzeichnisse mit einem Aufruf zu übertragen (Massen-Ingest[10]) oder die von iRODS angebotene Optimierung (Parallelisierung) der Übertragung auszunutzen.

[8] Kommandozeilen Tools, zur Interaktion mit dem iRODS-System
[9] Java API zur Interaktion mit dem iRODS-System
[10] Als "Ingest" wird im Archivierungs-Kontext das Einbringen von Daten in das Repository verstanden.



a) Zugriff auf Managed Content b) Zugriff auf Referenced Content

Abbildung 2: Daten-Grid als Storage für Repositorien

Im folgenden wird diese Unterscheidung jedoch vernachlässigt, da die Unterschiede nicht in der technischen Umsetzung, sondern lediglich in der Nutzung des Systems liegen und die Konfiguration jeweils dieselbe ist.

1.3 Überblick der System-Komponenten und Integrationsaspekte

Die System-Konfiguration besteht im wesentlichen aus: Fedora, iRODS, dem `wissgrid-service`-Modul, verschiedenen Fedora Plug-ins und auf Fedora aufsetzenden Diensten. Um die Nutzeranforderungen erfüllen zu können, wird Fedora und iRODS in den vorgestellten Integrationsvarianten kombiniert eingesetzt, um in dem entstehenden kooperativen System die jeweiligen Stärken der Teilsysteme (vgl. Tabelle 4 auf Seite 14) auszunutzen. Die beiden Systeme Fedora und iRODS sind jeweils als eigenständige, autonome Systeme entwickelt, deren eigentliches Anwendungsszenario vorsieht, dass Anwendungen direkt auf ihnen aufsetzen. Durch die Integration der beiden Systeme für das Repository in AP3 ist es erforderlich, dass sie sich austauschen, d.h. den aktuellen Datenbestand ständig synchronisieren. Die Mechanismen, welche die Synchronisation zwischen Fedora und iRODS sicherstellen, werden in AP 3 als Callback[11] und Forward[12] bezeichnet (siehe Kasten auf Seite 11).

Für die Integration von iRODS und Fedora wurden mehrere Komponenten entwickelt, sowie Anpassungen an den Systemen iRODS und Fedora vorgenommen. Die wesentlichen Entwicklungen im Rahmen von AP3 sind auf Fedora-Seite das iRODS-Akubra Plug-In, mit dem es Fedora möglich ist, iRODS als Speicherungsressource zu verwenden, und auf iRODS-Seite die Erweiterung und Anpassung von Regeln, sowie die Entwicklung von Microservices, um Ereignisse (z.B. ingest oder delete) in iRODS für die Systemsyn-

[11] Zieht relevante Ereignisse im iRODS-System in Fedora nach, um die Integrität zu gewährleisten.
[12] Wie Callback, jedoch von Fedora ausgehend.

chronisation abfangen zu können. Zudem wurde mit der Web-Komponente `wissgrid-service` ein SOAP-basierter Service entwickelt, der eine weitergehende Verarbeitung, wie bspw. die Indexierung, Versionierung oder das Setzen von Metadaten, vornimmt.

Das iRODS-Akubra Modul ermöglicht es der Fedora Storage Ebene, Managed Content in iRODS zu speichern. In Vorgriff auf die Ausführungen in den folgenden Kapiteln, sei bereits hier darauf hingewiesen, dass es sich bei *Managed Content* um Daten-Objekte handelt, die sich unter der Kontrolle von Fedora befinden, d.h. physisch an das Fedora-System übergeben und von diesem gespeichert wurden. Im Gegensatz dazu steht *Referenced Content* nicht unter der Kontrolle von Fedora, er wird ausserhalb von Fedora gespeichert und in Fedora nur über eine URL referenziert (zu Content-Typen bzw. Kontollgruppen siehe 2.1 auf Seite 10)).

1.4 Aufbau des Dokuments

Nachdem nun ein grober Überblick auf Anwendungsbereiche und Komponenten gegeben wurde, werden in Kapitel 2 die eingesetzten Systeme und System-Komponenten, ihre Aufgaben und das Zusammenspiel detaillierter betrachtet. Zudem werden spezielle Objekt-Typen, die zur Abbildung von iRODS-Verzeichnissen und -Dateien auf Fedora-Objekte erforderlich sind, eingeführt. Im 3. Kapitel wird auf die Systemkonfiguration, das Zusammenspiel der System-Komponenten und mögliche Integrationsvarianten eingegangen. Fragen der technischen Realisierung werden im 4. Kapitel betrachtet und im 5. Kapitel folgt ein Überblick auf die Installation und Benutzung, wobei eine detaillierte Installationsbeschreibung in der im Anhang befindlichen `README.pdf` zu finden ist. Das 6. Kapitel zeigt auf, was in der aktuellen Version implementiert ist, gibt Hinweise zur möglichen Verbesserung des Systems und hebt die Veränderungen zur Vorversion (bzgl. der Software) hervor und in Kapitel 7 finden Entwickler Einstiegspunkte, um das System zu verstehen oder zu ändern.

Hinweis:

Bei den in diesem Dokument beschriebenen Ergebnissen handelt es sich um eine prototypische Entwicklung, deren Eignung für den Produktivbetrieb nicht gewährleistet werden kann.

2 Systemkomponenten und -Elemente

In diesem Kapitel werden die einzelnen Systemkomponenten und ihre Eigenschaften detaillierter vorgestellt, und Objekt-Typen eingeführt, welche zum Abbilden neuer iRODS Daten-Objekte auf Fedora Objekte oder Datastreams erforderlich sind.

2.1 Fedora

Die Fedora Commons Repository Software ist ein System zum Daten- und Metadatenmanagement. Dazu definiert Fedora ein Objekt, das Fedora Digital Object (FDO), welches als Container für verschiedene Informationen dient und wie in der Einleitung bereits erwähnt, Daten-Objekte, Metadaten und Beziehungen[13] abbildet (siehe Abbildung 3). Durch die Möglichkeit FDOs in Beziehung zu setzen, lassen sich Objekte strukturieren. Über Content Models kann diese Strukturierung abstrakt spezifiziert werden, und auch die Zuordnung von FDO-spezifischem Verhalten ist über das Model möglich (siehe Seite 12).

Das FDO ist die zentrale Komponente im Fedora-System. Es handelt sich dabei um ein zusammengesetztes Objekt, das über eine fedora-weit (bei entsprechendem PID-System auch absolut) eindeutige PID identifiziert wird, das system-spezifische, aber auch objektbeschreibende Metadaten speichert und letztendlich die eigentlichen Daten-Objekte (Datastreams in Fedora Terminologie) direkt als Inline-XML enthält oder per fedora-interner Id oder externer Referenz adressiert.

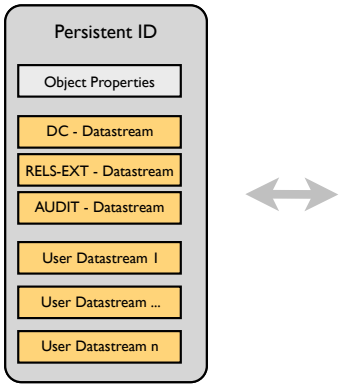
Das FDO ist in dem Sinne objekt-orientiert, dass es Daten-Objekte, d.h. Dateien oder per HTTP referenzierbare Ressourcen verwaltet und in einen größeren Kontext stellt, indem es diese zu Objekten (FDOs) zusammenfasst. Die FDOs können wiederum durch Setzen von Beziehungen in einen noch weiteren Kontext gestellt werden, womit beliebige, benutzerdefinierte Objekt-Strukturen abgebildet werden können. Damit unterscheidet sich der Objektbegriff in Fedora deutlich von dem im iRODS-Kontext, wo Daten-Objekte[14], d.h. Dateien, adressiert werden. Ein FDO ist eine dem übergeordnete Einheit und seine Beschreibung erfolgt über eine XML-Struktur, dem sog. FOXML[15] durch ein XML-Schema (foxml1-1.xsd) beschrieben).

Die zu jedem FDO gehörenden Object Properties beschreiben administrative Metadaten, welche zur Verwaltung der FDOs dienen, z.B. OwnerId, State, CreatedDate, LastModifiedDate. Alle anderen, in Abbildung 3 aufgezeigten Elemente sind Datastreams. Prinzipiell kann ein FDO beliebig viele Datastreams referenzieren, wobei eine zu große Anzahl für ein schlechtes Objektdesign spräche. Dies sollte durch Strukturierung der Objekte und Setzen von Beziehungen verhindert werden (Content-Modelling).

Die konkrete Strukturierung eines FDO ist stark abhängig vom Anwendungsbereich. So könnte es z.B

- im einfachsten Fall aus einem einzelnen Datastream (Daten-Objekt) bestehen, oder

[13] FDOs untereinander, Daten-Objekten untereinander und auch FDOs mit Daten-Objekte
[14] Das Gegenstück zum iRODS Daten-Objekt ist auf fedora-seite der Datastream.
[15] Fedora Object XML




```
<foxml:digitalObject PID="uniqueID">
  <foxml:objectProperties>
    <foxml:property/>
    <foxml:property/>
     ...
  </foxml:objectProperties>
  <foxml:datastream ID="DS1"
       STATE="A"
         CONTROL_GROUP="R"
           VERSIONABLE="true">
       <foxml:datastreamVersion ID="DS1.0"/>
         </foxml:contentLocation TYPE="URL"
              REF="http://www.xyz.de/a.pdf"/>
       </foxml:datastreamVersion>
          ...
  </foxml:datastream>
     ...
</foxml:digitalObject>
```

Abbildung 3: Zusammenhang von FDO und FOXML

- alle Digitalisate[16] zu einem Buch enthalten und zu jedem Digitalisat eine OCR-Datei[17] und ggf. eine TEI-Datei mit weitergehenden Informationen enthalten, oder
- ein FDO könnte ein Forschungsprojekt abbilden und zu jeder Versuchsreihe oder jedem Teilprojekt würde ein eigenes FDO erstellt (ggf. wieder in weitere FDOs aufgeteilt), welches dessen Ergebnisse enthält.

Letztendlich ist die Strukturierung vom Anwendungsbereich abhängig und muss von den Communities vorgenommen werden; dies gehört zum Aufgabenbereich der Data Curation.

Abhängig von Art und Ort der Speicherung der Daten-Objekte unterscheidet Fedora die folgenden vier Typen von Datastreams, Kontrollgruppen in Fedora-Terminologie:

Internal XML Content - (X): Bei diesem Typ werden die Daten (Daten-Objekte) direkt im FOXML, als Inline-XML abgelegt.

Managed Content - (M): Bei Managed-Content ist Fedora nicht nur für die Verwaltung zuständig, sondern führt auch die Speicherung der Daten-Objekte über seine Low-level-Storage-Ebene durch, d.h. Fedora bekommt das Daten-Objekt auch physisch, als Datei übergeben und hat die volle Kontrolle über das Daten-Objekt.

Externally Referenced Content - (E): Im Gegensatz zu Managed-Content wird hier nur eine objekt-referenzierende URL auf das extern gespeicherte Daten-Objekt abgelegt. Die physische Speicherung erfolgt ausserhalb und somit nicht unter Kontrolle von Fedora. Beim Bezug des Daten-Objektes, wird dieses durch Fedora "gestreamed", was insbesondere dann sinnvoll ist, wenn der Benutzer keine Zugriffsbe-

[16] Gescannte Bilder
[17] Das Ergebnis einer automatischen Texterkennung innerhalb von Bildern

rechtigung auf den Speicherort hat, oder um den konkreten Speicherungsort zu verbergen.

Redirect Referenced Content - (R): Auch hier wird, wie bei (E), nur eine objekt-referenzierende URL abgelegt. Allerdings erfolgt hier beim Zugriff kein Streaming durch Fedora, stattdessen wird ein URL-Redirect vorgenommen. Das ist insbesondere dann interessant, wenn ein Streaming- bzw. Media-Server zum Einsatz kommt.

Neben beliebig vielen benutzerdefinierten Datastreams kann ein FDO über die folgenden vier system-spezifischen Datastreams verfügen, welche administrative, wie auch beschreibende Metadaten pflegen.

DC (Dublin Core): Jedes FDO enthält standardmäßig diesen Datastream, ggf. wird er automatisch erzeugt. Über diesen Datastream können beschreibende Metadaten zum FDO gesetzt werden, welche bei der Basissuche von Fedora berücksichtigt werden.

RELS-EXT (Relationships-External): Objekt-zu-Objekt-Beziehungen werden in diesem Datastream als Inline-XML abgebildet. Der Datastream enthält dafür ein RDF-Element, welches alle entsprechenden Beziehungen dieses Objektes enthält.

RELS-INT (Relationships-Internal): Hier werden Beziehungen von Datastreams zu Objekten oder anderen Datastreams abgebildet. Auch dieser Datastream gehört zur Kontrollgruppe Internal XML Content, d.h. Inline-XML .

Audit: Auch der Audit-Datastream wird als Inline-XML spezifiziert, er speichert alle objektverändernden Ereignisse und bildet darüber eine Objekthistorie ab.

Benutzer interagieren in dem kooperativen Repository entweder über iRODS oder über Fedora mit dem System. Diese Interaktion kann abwechselnd über iRODS oder Fedora erfolgen, eine einzelne Aktion, wie bspw. ein Ingest erfolgt jedoch nur über eine Seite.

Callback: Erfolgt die Interaktion über iRODS, d.h. finden Aktivitäten (ingest, update, delete) über iRODS statt, so muss die Aktivität auch auf Fedora stattfinden, um die Integrität des Systems zu gewährleisten. Dies wird über einen sog. Callback realisiert, der die Rückmeldung an das Datenverwaltungssystem Fedora vornimmt und das Gesamtsystem damit synchron hält.

Forward: Ereignisse im Fedora-System werden per Forward an iRODS weitergeleitet, um die Synchronisation von Fedora ausgehend zu realisieren.

Über FDOs und Datastreams aggregiert Fedora somit Nutzdaten und diese Aggregation hat insbesondere Konsequenzen beim Löschen von FDOs. Datastreams der Kontrollgruppen Internal XML Content und Managed Content werden mit dem FDO gelöscht,

weil sie Teil des FDO sind oder ausschließlich über das FDO adressiert werden können. Das ist bei Referenced Datastreams anders, da diese lediglich referenziert werden. Es ist Fedora nicht automatisch möglich festzustellen, dass das referenzierte Daten-Objekt gelöscht oder geändert wurde. In der Gegenrichtung hat das Löschen eines Datastreams über Fedora keinen Einfluss auf das referenzierte Daten-Objekt, es wird lediglich aus dem Fedora-System entfernt und bleibt an seinem Ablageort erreichbar. Die Ursache dafür ist, dass sich das externe Daten-Objekt nicht unter Kontrolle von Fedora oder iRODS befindet.

Eine Ausnahme bildet Referenced Content (beider Gruppen), der in dem kooperativen Repository gespeichert und per Callback in Fedora registriert wurde. Hier erfolgt das Löschen daher von iRODS ausgehend über den Callback, d.h. das Löschen wird über den Callback im Fedora-System nachgezogen; von Fedora ausgehend wird das Löschen eines FDOs oder eines Datastreams per Forward im iRODS-System nachgeholt.

FDOs können mit anderen FDOs und auch externen Ressourcen in Beziehung stehen. Mit dem o.g. Content-Model bietet Fedora einen generischen Ansatz, FDOs intern (Art und Anzahl zugehöriger Datastreams) und extern (Umfang in Beziehung stehender Objekte) zu strukturieren. Dazu können FDOs mit einem oder mehreren Content-Modellen in Beziehung stehen, worüber die Struktur eines FDO festgelegt werden kann. Zugleich wird darüber eine Klassifizierung der FDOs ermöglicht, und das oben erwähnte Objekt-Design eines FDO kann über ein oder mehrere Content-Modelle festgelegt werden. Zudem kann ein Content-Model Verhalten definieren, über welches mit diesem Content-Model assoziierte FDOs verfügen. Dieses model-spezifische Verhalten wird durch in Beziehung setzen mit Service-Definitions (Schnittstellen-Spezifikationen) und Service-Deployments (Implementierungen) festgelegt. Damit können auf dem Daten-Objekt die festgelegten Services (Operationen) aufgerufen und virtuellen Objekte bzw. virtuellen Repräsentationen zur Laufzeit erzeugt werden. Konkret handelt es sich bei den virtuellen Objekten um Transformationen der original Daten-Objekte, die über die definierten Services erzeugt werden. Als Transformationen wären z.B. denkbar: Formattransformation von einem Bild-Typ in einen anderen; Transformation einer XML-Datei über ein XSL-Stylesheet (d.h. über XSLT und XSL-FO); Berechnungen auf Basis von Daten-Objekten oder Anfragen wie “liefere die n-te Seite eines PDF-Dokuments“. Durch die Möglichkeit virtuelle Objekte zu liefern, ist Fedora für den Aufgabenbereich der Content Preservation gut vorbereitet. Die Aufgabe der Modellierung ist Bestandteil der Data Curation und muss von der Community durchgeführt werden, da diese Modellierung von den konkreten Daten-Objekten und Kollektionen, aber auch Community-Anforderungen abhängt.

Zusammenfassung: Die Stärken des Fedora Systems sind seine Formatneutralität, die Möglichkeit, Objekte in Beziehung zu setzen (per RDF) oder mit Metadaten zu belegen, sowie eine weitergehende Strukturierung zu realisieren. Fedora ermöglicht eine Versionierung von Daten-Objekten, die Erzeugung virtueller Repräsentationen, den Export von Objekten z.B. im METS-Format, sowie die einfache Anbindung externer Systeme über die REST-, SOAP- oder Messaging-Schnittstelle.

2.2 iRODS

iRODS ist ein Speicherungsinfrastruktur- oder Data-Hosting-System, das eine verteilte, virtuelle Speicherung ermöglicht, wobei man über iRODS auf logischen Objekten und Strukturen arbeitet. Objekte und Strukturen im Kontext von iRODS sind Dateien und Verzeichnisse (bzw. Daten-Objekte und Kollektionen in iRODS-Terminologie). iRODS kennt keinen speziellen Objektbegriff wie Fedora. Es besteht lediglich die Möglichkeit, Dateien auf Dateisystemebene über Verzeichnisse zu gruppieren und darüber zu strukturieren. Das Wissen über die verwalteten iRODS-Objekte ist in der iRODS-Komponente iCAT (Katalog), einer relationalen Datenbank, abgelegt. Dabei bilden mehrere iRODS-Server eine Zone, welche sich einen iCAT teilen bzw. gemeinsam nutzen. Zur Speicherung und Replikation können und sollten verschiedene Speicherungs-Ressourcen eingebunden werden, was auch mit Blick auf Lastverteilung und Verfügbarkeit sinnvoll ist.

Das System kann über Regeln (text-basiert) und Microservices (C-Routinen) angepasst oder erweitert werden, über Regeln auch zur Laufzeit. Über Regeln kann auf Systemereignisse reagiert werden, es ist aber auch möglich, die Regeln in bestimmten zeitlichen Intervallen, explizit zu aktivieren. Als beispielhafte Reaktionen auf Erreignisse seinen die Replikation auf verschiedene Speicher-Geräte (-Ressourcen), die Berechnung von Checksummen, deren regelmäßige Überprüfung (Integritätstests) und ggf. der Austausch einer defekten Datei durch eine korrekte Kopie genannt. Realisiert wird dies über die iRODS Rule-Engine, welche das System-Verhalten über Regeln und Microservices steuert. Ein Abbildung von Abläufen (Workflows) ist durch entsprechende Regel- und Microservice-Kombinationen möglich.

Durch Verbinden von iRODS-Elementen, d.h. Daten-Objekten (Dateien), Kollektionen (Verzeichnissen), Ressourcen (Speicherressource) und Ressource-Gruppen, sowie iRODS-Benutzern, mit AVUs[18], ist es möglich, diese Elemente mit Metadaten zu belegen, Metadaten abzufragen oder zu ändern. Das direkte Setzen von Beziehungen zwischen iRODS-Elementen ist nicht möglich. Einem iRODS-Element können jedoch beliebig viele AVUs hinzugefügt werden und Beziehungen ließen sich z.B. darüber abbilden, dass ein Daten-Objekt, die PID (oder den Objekt-Pfad) des *Eltern-Daten-Objekts* als AVU speichert. Die Abbildung von Beziehungen über RDF und die Verwaltung der Beziehungen in einem Triple-Store[19], mit der Möglichkeit der Suche darauf, ist in Fedora vergleichsweise komfortabel gelöst.

Prinzipiell kann iRODS durch rule-basierte Formattransformationen, z.B. bei Ingest, Access oder auf Abruf, Unterstützung für Aufgaben der Content Preservation bieten. Wenn die Transformation immer durchgeführt werden soll, dann ließe sich dies noch relativ einfach lösen. Wenn die Transformation jedoch in Abhängigkeit bestimmter Bedingungen erfolgen soll, dann wird die Entwicklung der Rules schnell komplex und auch hier erscheint die Lösung in Fedora, mit den virtuellen Repräsentationen, komfortabler.

Zusammenfassung: Die Besonderheiten bzw. Stärken des iRODS-Systems sind seine Ereignisbezogenheit, Anpassbarkeit über Regeln und Microservices, die Möglichkeit Bit Preservation Aspekte (Replikation, Integritätstests, Austausch defekter Kopien) über Ru-

[18] AVU: Attribute Value Unit, d.h. Key-Value-Paare mit optionaler Einheit.

[19] Ein Triple-Store ist eine spezialisierte Datenbank, die auf die Speicherung von Triples optimiert ist. Im RDF werden diese Triple jeweils durch Subjekt-Prädikat-Objekt-Gruppen gebildet.

les und Microservices zu realisieren, sowie die Möglichkeit der Abbildung von Abläufen über Regeln.

	Schwächen	Stärken
iRODS	• eingeschränkte, proprietäre Schnittstelle • Umgang mit Metadaten unkomfortabel • keine Beziehungen • kein Objektbegriff • Unterstützung für Content Preservation schwierig (Transformationen per Rule/ Microservice) • Schwache Unterstützung für Data Curation (über AVUs)	• Rule-Engine • Unterstützung für Bit Preservation durch Regeln und Microservices • Parallelisierung • Unterstützung für Bit Preservation
Fedora	• keine direkte Unterstützung für Bit Preservation, d.h. keine automatische Replikation o. Integritätskontrolle • sequentieller Ingest • keine Unterstützung für Bit Preservation	• offene und standardisierte Schnittstellen • Metadatenverwaltung • Beziehungen per RDF abgebildet, Speicherung in Triple-Store • Objektbegriff • Strukturierungsmöglichkeiten über Content-Modelling • Gute Unterstützung für Content Preservation und Data Curation
kooperatives Repository	• Benutzer- und Rechteverwaltung • Synchronisation des Datenbestandes	• offene und standardisierte Schnittstellen • Metadatenverwaltuung • Beziehungen per RDF abgebildet, Speicherung in Triple-Store • Objektbegriff • Unterstützung für Bit Preservation durch Regeln und Microservices • Parallelisierung

Abbildung 4: Gegenüberstellung: iRODS u. Fedora

2.3 `wissgridservice`

Der `wissgridservice` ist ein eigenständiger, SOAP-basierten Web-Service, der den Callback (siehe Abbildung 5), die Versionierung von Daten-Objekten und deren Indexierung über Solr realisiert. iRODS-Events werden dazu an den Service weitergereicht, welcher die Verarbeitung übernimmt und zudem `AVUs` (z.B. `PID` des FDOs oder die `DsId`[20] des Daten-Objektes) auf den zugehörigen iRODS-Elementen setzt, um darüber die Abbildung (Mapping) zwischen Fedora und iRODS-Objekten zu realisieren.

Das `wissgridservice`-Modul vermittelt zwischen Fedora und iRODS, mit dem Ziel, deren Kopplung gering zu halten. Über diese Service-Komponente wird somit ein Teil der Ereignisbehandlung außerhalb des iRODS-Servers vorgenommen, um das System ohne großen Aufwand mit einer anderen Storage-Infrastruktur (z.B. dCache als Alternative zu

[20] Jeder Datastream wird über eine, im Kontext des FDO eindeutige, Datastream-Id identifiziert.

iRODS) nutzen zu können. In diesem Fall muss *nur* der Callback-Mechanismus, welcher in der `wissgridservice`-Komponente gekapselt ist, angepasst werden.

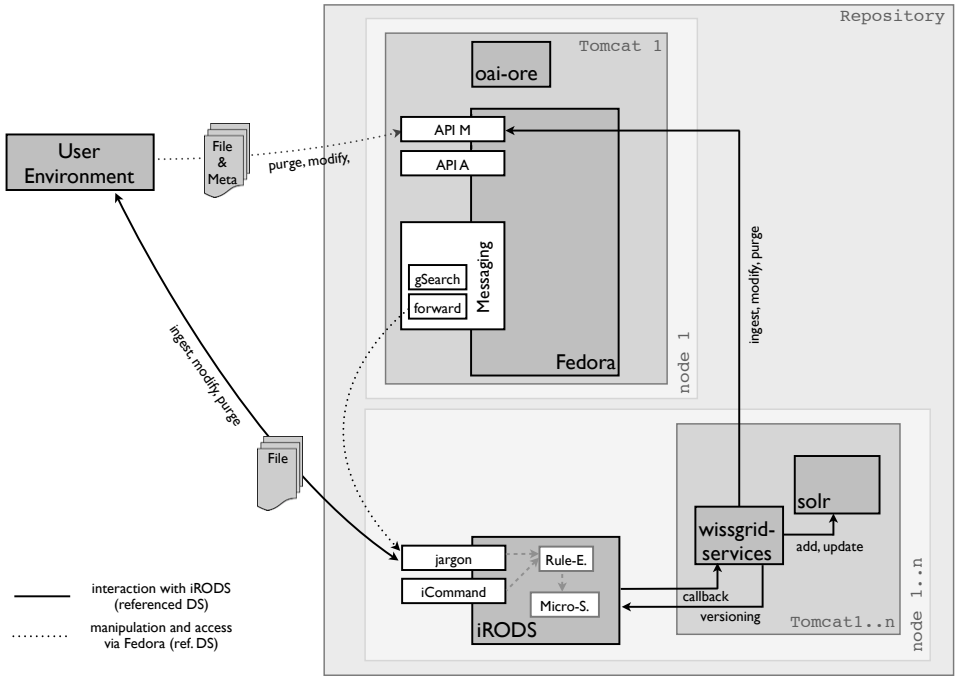


Abbildung 5: Callback und Forward

2.4 Davis WebDAV Schnittstelle

Daten-Objekte vom Typ Referenced Content werden in Fedora mit einer URL-Referenz auf diese Objekte verwaltet. Da iRODS-Ressourcen über das Schema (`irods://...`) adressiert werden, Fedora aber nur File- und HTTP-basierte Referenzen unterstützt, bedarf es eines Mechanismus, um diese Diskrepanz zu überbrücken. Das Davis WebDAV Modul realisiert für das iRODS-System eine HTTP-Schnittstelle und ermöglicht darüber den Zugriff auf Daten-Objekte per HTTP. Die in iRODS gespeicherten und in Fedora referenzierten Daten-Objekte bekommen als Datastream Location eine HTTP-basierte URL eingetragen. Die WebDAV-Komponente läuft als eigenständiger Service und kann unabhängig vom iRODS-Server installiert werden, d.h. sie muss nicht beim iRODS-Server lokalisiert sein.

2.5 SRW/SRU, OAI-ORE und OAI-PMH und Resource Index

Das Schnittstellen-Angebot des Fedora-Server kann durch Plug-Ins oder externe Dienste erweitern werden. Im WissGrid-Kontext sind insbesondere die OAI- sowie ORE-Provider, GSearch und Pazpar2, der Resource Index und der Resource Index Search Service von Interesse.

- GSearch[21] ist eine externe Komponente, die über das Fedora Messaging System eingebunden wird und mit welcher das Indexieren von Fedora-Objekten und Text-basierten Inhalten von Datastreams realisiert wird. Fedora selbst bietet nur eine Basissuche, wobei dort das Suchen auf der Grundlage des Dublin Core Datastreams durchführt wird. Mit GSearch ist auch eine Volltext-Suche auf den Inhalten aller Daten-Objekte möglich. Zudem stellt GSearch die Schnittstelle für Suche-Anfragen bereit und bindet Search Engines wie Lucene, Solr oder Zebra ein.

- Pazpar2 wird zur verteilten Suche in verschiedenen Solr-Indizes eingesetzt. Es wird über das *mod_proxy*-Modul des Apache HTTP Servers eingebunden. Die zu durchsuchenden Indizes werden sowohl auf Fedora-Seite (über GSearch/Solr), als auch auf iRODS-Seite (über Solr) erstellt.

- Der Basic Fedora OAI-PMH Provider Service bietet eine Implementierung für das OAI-PMH[22]. Über die PMH-Schnittstelle von Fedora wird ein standardisierter Zugang zu den im System verwalteten Metadaten gestattet.

- Der Fedora ORE Provider Service bietet eine Implementierung für das OAI-ORE[23] Protokoll und wird als eigenständige Dienst (Webanwendung) betrieben.

- Resource Index und Resource Index Search Service: Der Resource Index ist ein Fedora Server Plug-In zum Indexieren von RDF-Beziehungen. Die Abfrage des Resource Index läuft über den eigenständigen Dienst Resource Index Search Service (RISearch), der als Webanwendung implementiert ist und welcher derzeit iTQL und SPO als Anfragesprache unterstützt (SPARQL wird noch nicht unterstützt).

2.5.1 Apache HTTP Server (`mod_Proxy`)

Der für Pazpar2 eingesetzte Apache HTTP Server, wird auch dazu eingesetzt, Fedora vor Nutzern verborgen zu halten, was durch Anwendung von URL-Mappings und URL-Rewriting über das Proxy-Modul (`mod_proxy`) des Apache Servers erreicht wird. Durch das Proxy-Modul kann man eine einheitliche URL-Formatierung (Adressierung) realisieren, d.h. system-spezifische URLs, wie `http://<host>:<port>/fedora`, lassen sich maskieren bzw. mappen. So wäre es denkbar, den Fedora-Server über `http://<host>/repository` zu adressieren und den Request per `mod_proxy` , lokal umzulenken oder auf einen entfernten Server zu leiten. Das dahinter liegende System und insbesondere der tatsächliche `Host` und `Port` bleiben für Benutzer transparent.

2.6 Objekt-Typen und Objekt-Mapping

Wie sich aus den vorangegangenen Ausführungen ableiten läßt, kann eine Interaktion, je nach Profil bzw. Integrationsvariante, sowohl über iRODS, als auch über Fedora erfolgen. Beim Zugriff über iRODS stellt sich die Frage, *was* auf ein FDO bzw. Datastream

[21] Fedora Generic Search Service
[22] Open Archives Initiative - Protocol for Metadata Harvesting
[23] Open Archives Initiative - Object Reuse and Exchange

(fedora-seitig) abgebildet werden soll, und *wie* diese Abbildung erfolgen kann? Dazu wurden drei Objekt-Typen definiert und über das `wissgridservice`-Modul implementiert, womit die Abbildung von iRODS Verzeichnissen oder iRODS Daten-Objekten nach Fedora realisiert wird:

Single-Object: Jedes Daten-Objekt wird auf ein FDO abgebildet, das über genau einen benutzer-definierten Datastream (das Daten-Objekt) verfügt. iRODS Verzeichnisse haben hier kein Fedora Gegenstück.

Recursive-Object: Jedes iRODS Verzeichnis wird auf ein FDO abgebildet. Dateien innerhalb des Verzeichnisses werden auf Datastreams dieses FDOs abgebildet. Enthaltene Verzeichnisse bilden eigenständige, d.h. neue FDOs, welche mit dem übergeordneten, durch eine RDF-Beziehung (`fedora:isMemberOfCollection`) verbunden sind. Dies setzt sich rekursiv fort.

Multi-Object: Der Umfang eines FDO variiert - dieser kann aus einer Gruppe von n Daten-Objekten und einer Beschreibungsdatei (METS, BAG-IT oder OAI-ORE) bestehen, wobei nur der Ingest der Beschreibungsdatei zu einem Callback nach Fedora führt und damit zur Erzeugung eines FDOs. Alle anderen iRODS Daten-Objekte werden ausschließlich im Rahmen der Verarbeitung der Beschreibungsdatei behandelt. Aus der Beschreibungsdatei wird die Lokalität der zugehörigen Daten-Objekte ermittelt und damit werden die Daten-Objekte dem neuen FDO als Datastreams hinzugefügt. Wichtig ist, dass alle zugehörigen Daten-Objekte, beim Verarbeiten der Beschreibungsdatei, in iRODS vorliegen müssen. Die Auflösung, ob es sich um eine Beschreibungsdatei handelt, erfolgt über eine Namenskonvention.

Die Konfiguration des Mappings erfolgt über eine Property-Datei (siehe 5.5 auf Seite 30), in welcher eine feste Zuordnung zwischen Verzeichniszweig und Objekt-Typ festgelegt wird, d.h. alle Verzeichnisse bzw. Daten-Objekte unterhalb eines bestimmten Verzeichnisses sollen vom festgelegten Objekt-Typ sein und entsprechend behandelt werden.

Beim Ingest über Fedora (Managed Content, vgl. Seite 10) erfolgt die Zuordnung von Datastreams zu FDOs implizit, da der entsprechenden API-Methode `addDatastream` die `PID` des zugehörigen FDOs übergeben werden muss. Die Speicherung erfolgt dabei durch die Fedora Storage Ebene und der Benutzer hat nur sehr begrenzten Einfluss auf die Ablage (Daten-Verzeichnis ist konfigurierbar), und überhaupt keinen Einfluss auf die Strukturierung der Daten. Es handelt sich bei diesen Datastreams um von Fedora kontrollierten Content.

3 Design, Konfiguration und Interaktion

In diesem Kapitel wird auf die Systemkonfiguration, das Zusammenspiel der Systemkomponenten und mögliche Integrationsvarianten eingegangen

Die Interaktion erfolgt beim Fedora-System über dessen REST- oder SOAP-Schnittstellen (API-A für Access, API-M für Manipulation), beim iRODS-System, über dessen Kommandozeilen Schnittstelle (iCommands) oder über die java-basierte Jargon-Core API. Das Davis WebDAV Modul realisiert eine Webschnittstelle, über welche per Browser, WebDAV-Clients oder anderen HTTP-basierten Mechanismen auf iRODS Daten-Objekte zugegriffen werden kann. Fedora nutzt zur Kommunikation mit dem iRODS-System die Jargon-Core API. Der Callback, d.h. die Rückmeldung von Ereignissen auf dem iRODS-System an das Fedora-System, erfolgt über einen SOAP-basierten Webservice, d.h. das `wissgridservice`-Module.

Im folgenden wird zunächst auf die Integrationsvariante *Daten-Grid als Storage für Repositorien* (Profil B) eingegangen, weil das Repository hier von beiden Seiten (iRODS und Fedora) genutzt wird und damit alle denkbaren Interaktionsmöglichkeiten aufgezeigt werden. Damit verkürzen sich die Ausführungen zur Integrationsvariante *Repositorien als Archiv-Backend für das Grid* (Profil A) wesentlich.

3.1 Integrationsvariante: Daten-Grid als Storage für Repositorien

Zu Profil B wurden die beiden Varianten: *Fedora mit transparentem iRODS als Backend* und *Fedora mit sichtbarem (direkt nutzbarem) iRODS als Backend* unterschieden (vgl. Abb. 2 auf Seite 7), wobei die Unterschiede im wesentlichen in der Nutzung des Systems und dem Bewußtsein der Benutzer über die Teilsysteme liegen.

Ein Punkt, in dem sich beide Varianten u.U. wesentlich unterscheiden, ist das Rechte- und Benutzermanagement. Der aktuelle Ansatz sieht vor, dass das Rechte- und Benutzermanagement in der Anwendungsschicht realisiert und der Zugriff auf beide Systeme jeweils mit Robot-Usern[24] durchgeführt wird. In der Variante mit transparentem iRODS könnte das Fedora Rechte- und Benutzermanagement verwendet werden, um das System abzusichern. Fedora würde dann die Interaktion mit dem iRODS-Server mit einem Robot-User (mit Admin-Rechten) realisieren. Bei der Interaktion sowohl über Fedora, als auch über iRODS müsste ein gemeinsames bzw. synchronisiertes Rechte- und Benutzermanagement entwickelt und realisiert werden.

Jenseits der Fragen des Benutzer- und Rechtemanagements besteht die Konfiguration des Systems aus den folgenden Komponenten:

- Einem Fedora-Server.[25]
- Mindestens einem, sinnvollerweise aber mehreren[26] iRODS-Servern, die zusammen eine iRODS-Zone bilden und sich einen iCAT (Metadaten-Katalog) teilen. Zu-

[24] Ein spezieller Benutzer mit weitreichenden (ggf. Admin-) Rechten.
[25] Bei einer Föderation oder einem Server-Cluster auch aus mehreren.
[26] Wegen Ausfallsicherheit und Lastverteilung

dem sollten mehrere Speicher-Ressourcen eingebunden sein, um Datensicherheit garantieren zu können.

- Dem `wissgridservice`-Modul, welches zentral oder verteilt betrieben (deployed) werden kann. Die Installation sollte jeweils beim iRODS-Server erfolgen, um bei einem erforderlichen Streaming eines Daten-Objektes, z.B. für die Indexierung, keine zusätzliche Transferlast zu verursachen. Der Dienst wird bei jeder Interaktion über iRODS aufgerufen.
- Dem Davis WebDAV Modul, das zentral beim Fedora-Server eingesetzt wird.
- Weiteren, optionalen Modulen, z.B. dem Fedora Generic Search Service (GSearch), dem OAI-PMH[27]- oder OAI-ORE[28]-Provider, dem Fedora Resource Index oder Fedora Resource Index Search Service.
- Auf iRODS-Seite kann das Cheshire 3 Information Framework installiert werden, um iRODS mit Schnittstellen für SRU[29] oder OAI-PMH auszustatten. Bei der Evaluierung des Frameworks hat sich jedoch dessen starke Abhängigkeit von einer konkreten Python-Version und der Version des eingesetzten iRODS-Servers als Belastung erwiesen, so dass ein genereller Einsatz derzeit nicht empfohlen werden kann.

Bei dieser Integrationsvariante wird die Bereitstellung und der Zugriff auf Daten-Objekte meist über Fedora stattfinden, der Ingest und Update von Daten-Objekten wird aber vornehmlich über den iRODS-Server durchgeführt.

3.1.1 Interaktion über Fedora (Managed u. Referenced Content)

Wenn die zu speichernden Daten-Objekte dem Fedora System über seine REST- bzw. SOAP-basierte Schnittstelle physisch (als Dateien) übergeben werden und die Speicherung auf iRODS über die Fedora Storage Ebene erfolgt (siehe gestrichelter Ablauf in Abb. 6), so gehören die so erzeugten Datastreams zur Kontrollgruppe Managed Content.

Zudem besteht die Möglichkeit, *externe* Daten-Objekte aus Fedora heraus zu referenzieren (Referenced Content). Hierbei ist zu berücksichtigen, dass diese referenzierten Datastreams außerhalb der Kontrolle von Fedora (und iRODS) gespeichert werden und dass das Löschen des Daten-Objektes von Fedora nicht bemerkt werden kann (zu Löschen von Datastreams siehe auch Seite 11). Umgekehrt hat das Löschen über Fedora keinen Einfluss auf das referenzierte Daten-Objekt, es ist weiterhin unter seiner alten Adresse (URL) erreichbar.

Wenn dieses Verhalten nicht den Erwartungen entspricht, sollte das Hinzufügen eines Datastreams entweder über Fedoras Schnittstellen (mit physische übergabe des Daten-Objektes) oder direkt über iRODS, unter Anwendung des Callbacks erfolgen. Nur so befindet sich das Daten-Objekt unter der Kontrolle des Repositories und kann darüber

[27] Open Archives Initiative - Protocol for Metadata Harvesting
[28] Open Archives Initiative - Object Reuse and Exchange
[29] Search and Retrieve via URL

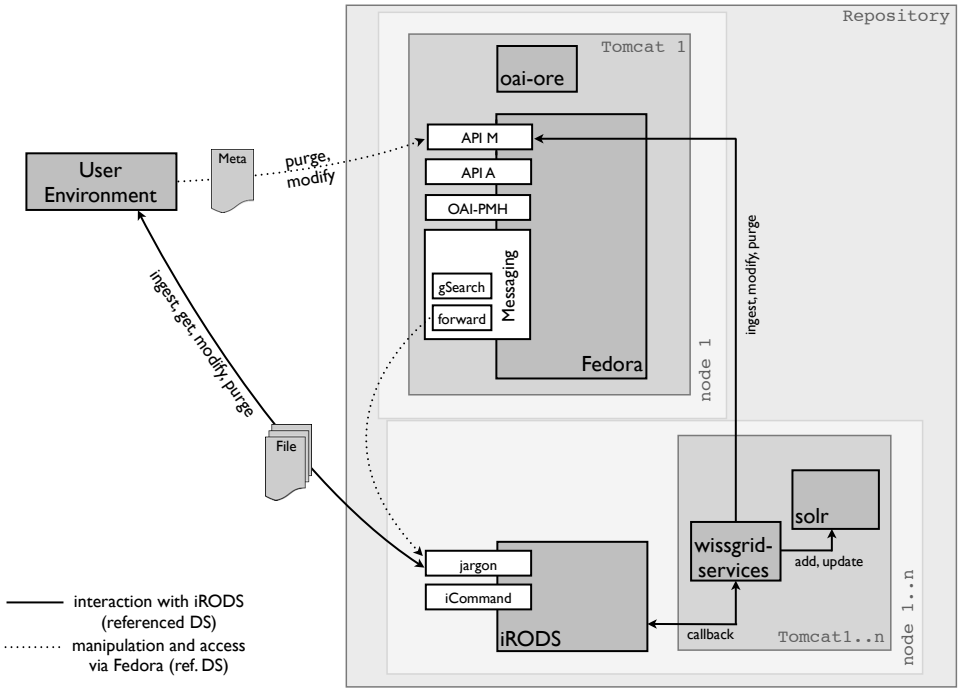


Abbildung 6: Beide Interaktionsvariante bei Profil B

manipuliert werden. Ein Löschen über iRODS würde an Fedora zurückgemeldet (Callback) und ein Löschen über Fedora würde an iRODS weitergeleitet (Forward).

3.1.2 Interaktion über iRODS (Referenced Content)

Beim Einfügen von Objekten oder deren Manipulation über iRODS, muss Fedora per Callback über den Sachverhalt informiert und aktualisiert werden (siehe durchgezogener Ablauf in Abb. 6). Der Callback übergibt das Ereignis an den Fedora-Server und dieser kann die erforderlichen Reaktionen veranlassen (z.B. FDO erzeugen, Datastream hinzufügen oder löschen, ...). Der Callback erfolgt über die Zwischenkomponente `wissgrid-service`. In dieser Komponente wird auch die Versionierung, die Indexierung, das Setzen von Metadaten auf iRODS-Elementen, sowie der eigentliche Service-Aufruf beim Fedora-Server realisiert bzw. angestossen.

Neue Daten-Objekte werden über Solr indexiert, Manipulationen und auch das Löschen werden in den Index eingepflegt. Bei Versionierung wird eine Kopie in das Versionsverzeichnis gelegt. Beim Löschen einer aktuellen Version wird die nächstältere Version ermittelt und diese wird zur aktuellen Version.

Ein typischer Ablauf sieht so aus, dass ein iCommand ausgeführt wird, welcher einen Aufruf beim iRODS-System veranlasst. Im Rahmen der Verarbeitung wird eine Regel (ggf. auch mehrere) aktiviert. Eine Regel kann weitere Regeln aktivieren und einen oder mehrere Microservices aufrufen. Ein Microservice nimmt dann die eigentliche Verarbeitung vor und delegiert einen Teil der Verarbeitung durch entsprechende Service-Aufrufe

an die `wissgridservice`-Komponente.

Beispiel: Mit Aufruf von `iput a.txt testdir/b.txt` wird ein iRODS-iCommand zum Ingest der Datei `a.txt` im aktuellen, lokalen Arbeitsverzeichnis abgesetzt, der die Speicherung der Datei, unter dem Namen `b.txt` im iRODS-Verzeichnis `testdir/` unter dem aktuellen iRODS-Arbeitsverzeichnis bewirkt. Infolge dieses Speicherns wird die Rule `acPostProcForPut` aktiviert. Diese (Post-)Rule ruft zunächst den Microservice `msiSysChksumDataObj` auf, welcher eine Checksummenberechnung vornimmt und delegiert die Weiterverarbeitung anschließend an den Microservice `msiWissGridCallbackForPutObject`, welcher den "Object-Path", d.h. den logischen[30] Speicherungsort für das Datenobjekt ermittelt. Der letztgenannte Microservice ruft dann die Methode `addDS` (add-Datastream) des `wissgridservice`-Modules auf. Hier wird zunächst der Objekt-Typ festgestellt (anhand eines Abgleichs des Verzeichnisses `testdir/` mit den Angaben in der Property-Datei), in Abhängigkeit davon werden `AVU` Metadaten mit dem iRODS-Objekt verbunden, die Versionierung und Indexierung realisiert, und letztendlich wird ein Service-Aufruf (hier z.B. `addDS`) auf dem Fedora System vorgenommen.

Nachfolgend wird anhand des vorausgegangenen Beispiels beschrieben, wie das `wissgridservice`-Modul die verschiedenen Objekt-Typen (siehe Kapitel 2.6, Seite 16) behandelt:

- Für Single-Objects wird ein FDO erzeugt und das Daten-Objekt `b.txt` wird diesem als einzelner Datastream hinzugefügt.

  Versionierung: Wenn in der Property-Datei Versionierung aktiviert wurde, wird eine Kopie des Datastreams `b.txt` mit Erweiterung `._v0` in ein Versionsverzeichnis[31] kopiert. Wenn das Daten-Objekt später überschrieben wird, so wird eine neue Version (eine Kopie) mit Erweiterung `._v1` ins Versionsverzeichnis gelegt, usw. Sollte die aktuelle Version gelöscht werden, so wird das Daten-Objekt `testdir/b.txt`, welches der aktuellen Version entspricht, durch die nächstältere Version ersetzt (`testdir/_version_/b.txt._v0 -> testdir/b.txt`). Wenn damit die letzte Version gelöscht wurde, also keine weitere Version mehr existiert, so wird das FDO gelöscht.

- Für Recursive-Objects wird das Daten-Objekt `b.txt` dem FDO, das mit dem Verzeichnis `testdir/` verbundenen ist, als Datastream hinzugefügt. Die Versionierung erfolgt wie zuvor beschrieben. Bei der Erstellung eines neuen (inneren) Verzeichnisses wird automatisch ein FDO erstellt, welches auf Fedora-Seite per RDF mit dem übergeordneten (Verzeichnis-) FDO in Beziehung (`fedora:isMemberOfCollection`) gesetzt wird.

- Für Multi-Objects wird mit dem Ingest der Beschreibungsdatei `b.txt` ein FDO erzeugt und alle in der Datei referenzierten Daten-Objekte werden dem FDO als Datastream hinzugefügt. Die Versionierung erfolgt wie zuvor beschrieben. Derzeit

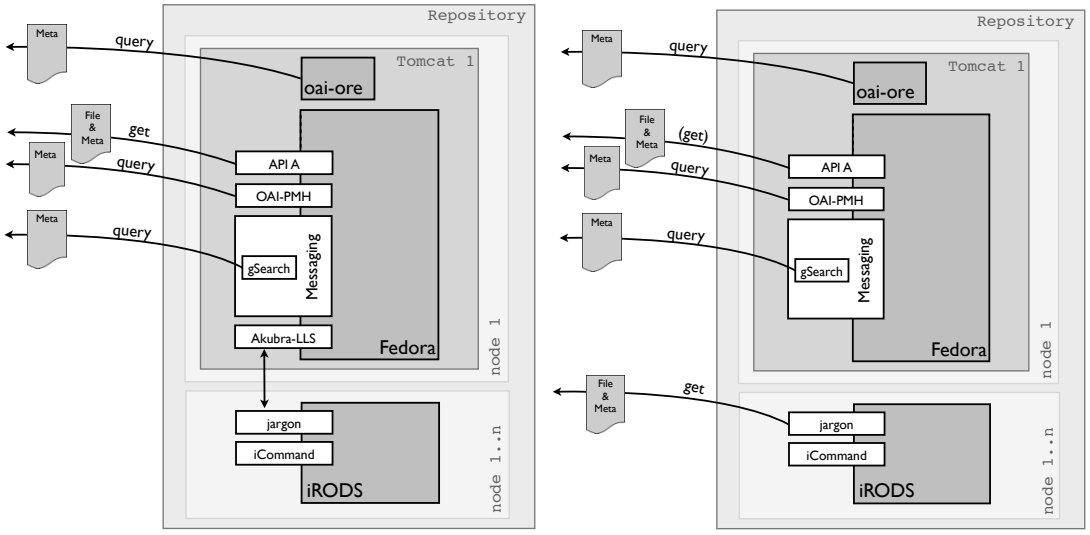
[30] Durch die Unterscheidung von logischen und physischen Speicherungsorten ist es iRODS möglich, z.B. die Anzahl der max. zulässigen Dateien pro Verzeichnis (für Nutzer transparent) zu beschränken oder Daten-Objekte auf unterschiedlichen Speicher-Ressourcen abzulegen, ohne dass das im "Object-Path" offensichtlich würde.

[31] Jedes Verzeichnis mit Daten-Objekten enthält ein Versionsverzeichnis, z.B. `testdir/_version_/`. Der Name `_version_` ist über die Property-Datei konfigurierbar.

werden BagIt, METS und OAI-ORE als Formate für die Beschreibungsdatei unterstützt, wobei die Auflösung über eine Namenskonvention erfolgt. Bei der Verarbeitung der Beschreibungsdatei wird erwartet, dass alle zum Objekt gehörenden Daten-Objekte im iRODS-System erreichbar sind.

Erwähnenswert ist, dass das Überschreiben von Beschreibungsdateien keinen Einfluss hat, es wird ignoriert und führt weder zu einem neuen FDO noch zum Update des bestehenden. Wenn die alte Beschreibungsdatei gelöscht und dieselbe erneut eingefügt wird, führt dies jedoch zu einem neuen FDO, weil keine Verbindung zwischen der neuen und der alten Beschreibungsdatei besteht bzw. feststellbar ist (durch das Löschen sind auch die zugehörigen `AVUs` gelöscht, welche die Zuordnung ermöglichen). Wenn die Versionierung von Beschreibungsdateien erlaubt wäre, so wäre nicht klar, wie dann mit bestehenden Datastreams umgegangen werden sollte. Ist die neue Version ein Update oder eine komplette Neubeschreibung des FDOs? Gehören bestehende Datastreams noch zur neuen Version, auch wenn sie nicht mehr in der Beschreibungsdatei enthalten sind oder sind sie aus dem FDO zu löschen? Insbesondere verursacht das Löschen oder Überschreiben von Beschreibungsdateien viel Aufwand, es müssten alle alten Zuordnungen (`AVUs`) gelöscht und neue gesetzt werden, so dass diese Möglichkeit derzeit nicht berücksichtigt wird.

Das Manipulieren und Löschen des FDO kann bei Multi-Objects somit nur über Fedora (über dessen Messaging) vorgenommen werden. Manipulationen an Daten-Objekten sind auch über iRODS möglich, d.h. Überschreiben und somit Hinzufügen einer neuen Version, sowie Löschen einer Version. Das Hinzufügen eines neuen Datastreams kann nur über Fedora (Managed Content) erfolgen, weil es kein iRODS-Element gibt, welches als Gegenstück zum FDO dient. Über iRODS besteht somit keine direkte bzw. automatische Zuordnungsmöglichkeit eines Datastreams zu einem FDO.



a) Zugriff auf Managed Content b) Zugriff auf Referenced Content

Abbildung 7: Zugriffsmöglichkeiten

3.2 Integrationsvariante: Repositorien als Archiv-Backend für das Grid

Wie in der Einleitung bereits angedeutet, kann das System bei dieser Konfiguration entweder nur aus einem einzelnen iRODS-Server bestehen oder aus der Kombination von Fedora und iRODS. Der erste Fall wird nicht weiter besprochen, da dazu keine Veränderungen am iRODS-System erforderlich sind (iRODS Standard Distribution). Wenn die im AP3 entwickelte Versionierung auch für diesen Fall eingesetzt werden soll, so müsste die `wissgridservice`-Komponente entsprechend angepasst werden. Aktuell nimmt diese Komponente die Abbildung von iRODS Verzeichnissen und Daten-Objekten auf die entsprechenden Fedora Gegenstücke vor (vgl. Objekt-Typen 2.6 auf Seite 16) und realisiert die Versionierung. Da in dem Szenario, in dem iRODS ohne Fedora betrieben wird, keine PIDs oder DsIds existieren, muss eine andere Abbildung von zusammengehörigen Objekten gefunden werden. Versionen werden bei der kombinierten Konfiguration so abgebildet, dass alle zu einem Datastream gehörenden Versionen dieselbe PID und DsId haben und über eine versions-bezogene Erweiterung des Dateinamens oder auch ein `AVU` mit zugehörigem Timestamp unterschieden werden können.

Somit wird nur die Konfiguration beschrieben, bei der dem iRODS-Server ein Fedora-Server zur Metadaten-Verwaltung vorgelagert wird. Die Konfiguration ist im Prinzip dieselbe wie bei Profil B (*Daten-Grid als Storage für Repositorien*), jedoch kann es hier sinnvoll sein, die Interaktion mit dem Fedora-System über entsprechende XACML-Policies zu beschränken. Durch das Verbieten von Ingests und Manipulationen über Fedora erreicht man, das ausschließlich Referenced Content (durch iRODS kontrolliert) im System vorliegt.

Bei diesem Anwendungsprofil wird die Interaktion vornehmlich über iRODS stattfinden, Fedora kann als Metadaten-Katalog für iRODS betrachtet werden und erweitert zudem das eingeschränkte Schnittstellenangebot von iRODS. Auch in Kombination mit dem Fedora-Server bleibt die Kernanforderung dieses Profils die Unterstützung von Grid-Workflows, d.h. die effiziente Einbettung der Daten in Grid-Jobs.

3.2.1 Interaktion über iRODS

Ein wesentliches Merkmal dieser Integrationsvariante ist, dass ausschließlich Referenced Content verwaltet wird, weil nur dieser direkt in Grid-Workflows eingebunden werden kann. Die Interaktion über iRODS entspricht dabei den Ausführungen in Kapitel 3.1.2 auf Seite 20. Der Bezug von Managed Content über Fedora wäre für die betreffenden Anwendungsfälle (Grid-Workflows) wenig komfortabel und wird nicht weiter betrachtet.

3.2.2 Interaktion über Fedora (Referenced Content)

Der Zugriff über Fedora dient dem Auffinden von Objekten und dem Daten-Export. Für den Export, d.h. den Zugriff auf die iRODS-Objekte wird die Davis WebDAV-Schnittstelle für das iRODS-System eingesetzt. Die Datastream Location, die zu jedem Datastream gespeichert wird, enthält die Davis-URL, über welche das Objekt per HTTP bezogen wer-

den kann. Für die Authentifizierung über die WebDAV-Schnittstelle wird der Authentifizierungsmechanismus von iRODS verwendet, das bedeutet, dass man sich mit seinen iRODS-Benutzerdaten authentifiziert.

4 Technische Umsetzung

In diesem Unterkapitel werden Hinweise auf die eingesetzten Programmiersprachen gegeben, es werden die Mechanismen zum Zugangsschutz aufgezeigt, die Log-Files aufgelistet und kurze Anmerkungen zur Performance gemacht.

4.1 Programmiersprachen

Wie einleitend erwähnt werden hier Informationen zu den verwendeten Programmiersprachen und Technologien gegeben, um darüber eine Entscheidung zu erleichtern, ob die Systeme in die bestehende Anwendungslandschaft passen, wie mit den Systemen interagiert werden kann, wie Manipulationen vorgenommen werden können und ob entsprechendes Know-How vorhanden ist.

- iRODS
  - C für Microservices, d.h. für interne Routinen, welche die Funktionalität des Servers ausmachen
  - Jargon API, bereitgestellte Java API zur Interaktion mit dem iRODS-System
  - Es besteht die Möglichkeit, Micro-Services in Python zu schreiben und es gibt eine Python-API zur Interaktion mit dem iRODS-System
  - Proprietäre Rule-Language, text-basiert Sammlung von Rule-Definitionen
- Fedora
  - java-basierter Server
  - REST- und SOAP-basierte Web-Services zur Interaktion mit dem System. Bereitgestellte APIs:
    - API M - Management API zur Manipulation und
    - API A - Access API für den Zugriff
- Davis WebDAV
  - java-basierte Webanwendung
- `wissgridservice`-Modul
  - java-basierter SOAP Web-Service
- GSearch
  - java-basierter REST- und SOAP Web-Services
- Pazpar2
  - c-basierte Anwendung
- Basic OAI-PMH Provider

  - java-basiertes Fedora Server Plug-In

- ORE Provider
  - java-basierte Webanwendung

4.2 Zugangsschutz der Systeme (AuthN/AuthZ)

- iRODS
  - Benutzer/Passwort oder
  - Grid Security Infrastructure (GSI) oder
  - Kerberos in Entwicklung
  - ACL für Autorisierung
- Fedora
  - Benutzer/Passwort
  - Basiert auf Java Authentication and Authorization Service (JAAS)
  - Fedora Security Layer (FeSL) für Authentifizierung und Autorisierung - für letzteres noch in Entwicklung
  - XACML-policy-basierter Mechanismus zur Autorisierung
- Davis WebDAV
  - Nutzt den iRODS-Mechanismus

Aktuell wird lediglich mit Admin- bzw. Robot-Usern, auf Basis des Benutzername/Passwort-Mechanismus gearbeitet. Ziel einer Weiterentwicklung bzgl. der Sicherheitsaspekte könnte eine Erweiterung des Systems dahingehend sein, dass eine Eins-zu-Eins Abbildung von Fedora- und iRODS-Benutzern realisiert wird. Dies kann durch eine gemeinsame oder aber getrennte und synchronisierte Rechte- und Benutzerverwaltung erfolgen. Eine Alternative wäre die Integration von GSI in Fedora, iRODS unterstützt dies bereits. Mit GSI würde zugleich die Frage der Art der Rechte- und Benutzerverwaltungen und deren Synchronisation beantwortet.

4.3 Log files

- iRODS:
  - WissGrid-spezifisch: `$IRODS_HOME/server/bin/wissgrid.log`
  - allgemein: `$IRODS_HOME/server/log/rodsLog.<yyyy>.<mm>.<dd>`
  - empfangene Nachrichten: `$IRODS_HOME/server/bin/RECV.log`
  - gesendete Nachrichten: `$IRODS_HOME/server/bin/SENT.log`
- Fedora:

  - allgemeines Log: `$FEDORA_HOME/server/logs/fedora.log`

- Tomcat: für Fedora, Solr/GSearch[32] und `wissgridservice`
  - allgemeines Log: `$CATALINA_HOME/logs/catalina.out`
- Apache HTTP Server: (für Pazpar2 und Proxy-Modul)
  - abhängig von der Einstellung des Systems und des Webservers. Wahrscheinlich in: `/var/log/apache2/error.log`

4.4 Performance und Skalierbarkeit

Durch die direkte Interaktion mit dem iRODS System, d.h. durch Verwendung der Kommandozeilen-Tools (iCommands), wird der Datentransfer automatisch optimiert, indem Datenpakete auf mehrere Threads und Socket-Verbindungen verteilt und parallel übertragen werden.

Eine Möglichkeit der Performanceverbesserung auf Fedora-Seite bestünde in der Errichtung eines Server-Cluster, was jedoch aktuell nicht Teil der Entwicklung ist und somit hier nicht weiter diskutiert werden soll.

Eine potenzielle Schwachstelle in Bezug auf Performance bildet die Kopplung der beiden Systeme und insbesondere deren Synchronisation. Der iRODS-Poundtest, bei dem Daten-Objekte in unterschiedlicher Anzahl und Größe an iRODS übergeben werden, hat jedoch gezeigt, dass die Performance beim Ingest nur unwesentlich sinkt.

Wesentlich für die Gesamtperformance ist, dass der Tomcat-Server ausreichend Speicher zugewiesen bekommt (vgl. 5.1 auf Seite 5.1).

[32] Es besteht die Möglichkeit eine separate Log-Datei zu erstellen.

5 Installation, Anpassungsmöglichkeiten und Benutzung

Die eigentliche Installation wird in der `README` (siehe Anhang A Seite 40) beschrieben. Die folgenden Ausführungen skizzieren den Installationsprozess daher nur grob.

Bei einer Testinstallation ist es empfehlenswert alle Komponenten (iRODS, Fedora, ...) auf einem Host zu installieren. Es ist aber auch möglich und sinnvoll, eine verteilte Installation vorzunehmen. Die lokale Installation erfolgt entsprechend der Beschreibung in der README. Für die verteilte Installation muss der Installer auf allen Hosts ausgeführt werden, es werden jedoch nur die Installationsskripte der Komponenten, die auf dem betreffenden Host von Interesse sind ausgeführt. Bei der verteilten Installation sind zudem die in Abschnitt 5.5 beschriebenen Anpassungen an den Konfigurationsdateien zu beachten.

Der Aufruf `java -jar WissGrid-D3.5.3-grid-repository-fedora-irods-installer.jar` startet den Installationsprozess. Das verwendete Installationstool schreibt die zum Aufbau des Repository notwendigen Pakete in ein temporäres Installations-Verzeichnis, aus welchem heraus anschließend die Installation der Teilkomponenten erfolgt. Bei Verteilung auf verschiedenen Hosts erfolgt dieser schritt dann jeweils nur für die Teilkomponenten.

Wichtig: Da Abhängigkeiten zwischen den zu installierenden Komponenten bestehen, ist die Installationsreihenfolge unbedingt, wie in der `README` angegeben, einzuhalten. So ist es für die Kompilierung von iRODS bspw. erforderlich, dass das `wissgridservice`-Modul installiert und seine Laufzeitumgebung (z.B. Tomcat) gestartet wurde, damit beim Bauen des iRODS-Servers Source-Code für die Interaktion mit dem Service-Modul (Stubs) generiert[33] werden kann, was nur bei laufendem Service möglich ist.

5.1 Tomcat Konfiguration

Um beim Indexieren mit Solr einen `OutOfMemory Error: PermGen Space` Fehler zu vermeiden, muss die Tomcat-Instanz, die Solr beherbergt, ausreichend Heap-Speicher zugewiesen bekommen.

Die Zuweisung erfolgt, indem beim Tomcat-Start die Umgebungsvariable `JAVA_OPTS` bspw. wie folgt gesetzt ist (`Xms` setzt die Startgröße und `Xmx` die maximale Größe des Heap):

```
JAVA_OPTS="-server -Xms768M -Xmx1024M -XX:MaxPermSize=768M"
```

Wenn nun sehr große Dateien übertragen und indexiert werden, d.h. Solr immer mehr Speicher belegt, ohne das zwischenzeitlich Speicher freigegeben werden kann, scheitert die Übertragung und Tomcat kann u.U. nachhaltig in seiner Funktion beeinträchtigt sein. Um dies zu verhindern, sollte man sich über die erwarteten Transfergrößen bewusst sein, und die Funktion in einer Testumgebung überprüfen. Letztendlich gibt auch diese Untersuchung keine absolute Sicherheit, denn man kann über die Systemauslastung nur spekulieren.

[33] Die Source-Code Erzeugung erfolgt auf Basis der Web Service Beschreibung (WSDL) des `wissgrid-service`-Moduls

5.2 iRODS Installation

Die iRODS Installation teilt sich in die Installation des iRODS-Servers (siehe `README`, Schritt 4.2.2) und dessen Anpassung für WissGrid (siehe `README`, Schritt 4.2.4). Wenn man eine bestehende Server-Installation weiterverwenden will, entfällt der erste Schritt.

Nach erfolgreicher Installation, kann der iRODS-Server über ein Konsolen-Fenster mit `irodsctl start` gestartet werden. Für erste Schritte siehe: 5.7 Seite 33.

5.3 Fedora Installation

`WissGrid-D3.5.3-grid-repository-fedora-irods-installer.jar` ist ein ausführbares Java-Archiv, über welches die Fedora-Installation vorgenommen wird. Die einzelnen Schritte sind in der `README` aufgeführt.

5.4 Installation sonstiger Komponenten

`wissgridservice`-Modul Installation

Die Installation erfolgt durch Ablage des zugehörigen Web-Archives (WAR-Datei) in das Deployment-Verzeichnis des Tomcat-Servers (`$CATALINA_HOME/webapps/`). Ob Versionierung aktiviert sein soll; welche iRODS-Verzeichnisse mit welchen Objekt-Typen assoziiert werden sollen (vgl. Objekt-Typen in 2.6 auf Seite 16); ob Solr, d.h. Indexierung genutzt werden soll; unter welchen URIs Dienste wie Davis WebDAV, Solr oder Fedora erreichbar sind - dies und anderes kann durch Modifikation der Property-Datei `$CATALINA_HOME/webapps/wissgridservices/WEB-INF/wissgridservices.properties` festgelegt werden (siehe auch 5.5 auf Seite 30).

gSOAP Installation

gSOAP wird im Rahmen der iRODS-Installation installiert. Für weitere Hinweise sei auf die `README` oder auf die Ausführungen auf der iRODS Web-Seite (Web Services As Micro Services) verwiesen.

Davis WebDAV Installation

Davis WebDAVwird in einem Jetty Servlet-Container installiert ausgeliefert. Die Installation besteht im wesentlichen aus dem setzen von Umgebungsvariablen, dem bereitstellen eines Zertifikates und der Konfiguration. Zu Details bezüglich der Installation wird auf die Online-Quellen verwiesen.

GSearch, Pazpar2, OAI-Provider und ORE-Provider Installation

Alle diese Anwendungen (bis auf Pazpar2) sind als WAR-Datei in Apache Tomcat zu integrieren. Für detaillierten Information bzgl. der Installation wird auch hier auf die Online-Quellen verwiesen.

5.5 Anpassungsmöglichkeiten

In diesem Abschnitt werden Anpassungsmöglichkeiten aufgezeigt. Diese sind auch zu beachten, wenn von der Standardinstallation abgewichen wird, z.B. verteilte Installation; nicht `rods` als iRODS Admin Benutzer; oder nicht `fedoraAdmin` als Fedora Admin Benutzer.

5.5.1 wissgridservice-Komponente

`$CATALINA_HOME/webapps/wissgridservices/WEB-INF/wissgridservices.properties` legt viele Attribute fest und ermöglicht darüber zugleich die Anpassung des Systems. Einzelne Attribute werden in der folgenden Auflistung mit einer Kurzbeschreibung aufgeführt. Im Falle einer verteilten Installation sind insbesondere die Attribute `IRODS_<...>`, `DAVISURL`, `SOLR_URL`, `FEDORA_URL` und `FEDORA_WSDL_URL` anzupassen.

- RECURSIVE_OBJECT_DIRS: Durch Semikolon separierte Liste von iRODS-Verzeichnissen. Dateien die in diese Verzeichnis gelegt werden, sollen vom Objekt-Typ Recursive-Object sein (siehe 2.6 Seite 16) und das Mapping soll entsprechend erfolgen.
- SINGLE_OBJECT_DIRS: Wie zuvor, jedoch sollen die Dateien entsprechend dem Objekt-Typ Single-Object behandelt werden.
- MULTI_OBJECT_DIRS: Wie zuvor, jedoch Objekt-Typ Multi-Object.
- BAGIT_DESCRIPTION_FILE: Legt die Endung einer BagIt-Beschreibungsdatei fest und definiert damit die zugehörige Namenskonvention.
- METS_DESCRIPTION_FILE: Wie zuvor, jedoch für METS-Beschreibungsdateien.
- OAI_ORE_DESCRIPTION_FILE: Wie zuvor, jedoch für OAI-ORE-Beschreibungsdateien.
- IRODS_<...>: Zum setzen von iRODS-Verbindungsdaten.
- ZONE: Zum setzen der iRODS-Zone.
- OBJECT_STORE: Legt den iRODS-Pfad für den Fedora Object-Store fest. In diesem Pfad speichert Fedora die FOXMLs als Managed Content (über das Lowlevel-Storage-Modul).
- DATASTREAM_STORE: Wie zuvor, jedoch für Datastreams.
- DAVISURL: Die URL unter der die Davis WebDAV erreichbar ist.
- FEDORASCHEMA: Setzt das Fedora-Schema, welches als Prefix von Fedora PIDs oder Beziehungs-Prädikaten verwendet wird. Fedora spricht hier auch vom Fedora-Namespace.
- AVU_ATTR_NAME_<...>: Legt verschiedene AVU-Attributnamen fest.

- AVU_ATTR_VALUE_<...>: Setzt für einzelne AVUs Werte.
- IRODS_VERSIONING: Legt fest, ob versioniert werden soll. Diese Eigenschaft kann nur global gesetzt werden.
- VERSIONSUBPATH: Setzt den Namen des Verzeichnisses, in dem die Versionen abgelegt werden.
- VERSION_MARK: Setzt das Suffix, das hervorhebt, das es sich bei einem Daten-Objekt um eine Version handelt. Dem Suffix wird system-seitig zudem eine Versionsnummer angehängt.
- SOLR_INDEXING: Legt fest, ob irods-seitig indexiert werden soll.
- SOLR_URL: Die URL, unter welcher der Solr-Server erreichbar ist (irods-seitige Indexierung).
- FEDORA_FORCE: Legt fest, ob auf Fedora-Seite das Löschen erzwungen werden soll, auch wenn dadurch Abhängigkeiten gebrochen werden.
- FEDORA_URL: Die URL unter welcher der Fedora-Server erreichbar ist.
- FEDORA_WSDL_URL: URL unter welcher die WSDL zur Service-API-M erreichbar ist.

Wenn sich bspw. die URLs oder Zugangsdaten zu iRODS oder Fedora im Betrieb ändern, dann sind die Änderungen auf die entsprechenden Attributen zu übertragen und die Web-Anwendung `wissgridservice` muss neu gestartet werden.

5.5.2 iRODS

Zentrale Stelle zur Anpassung des iRODS Systems sind die Rule-Definitionen, wobei Änderungen an Rules sofort wirksam werden. Für WissGrid sind hier die beiden Rule-Definitionsdateien `wissgrid.re` (neue Rule-Language) und `wissgrid.irb` (alte Rule-Language) in `server/config/reConfigs/` von Interesse. Wenn man die Mail-Adresse des iRODS Admin ändern will, so muss man diese in beiden Dateien ändern (am Ende). Auch wenn man die Default Resource ändert, muss man dies in den besagten Dateien nachziehen (ebenfalls am Ende).

Wenn die `wissgridservice`-Komponente nicht lokal beim iRODS-Server installiert wird (was jedoch zu empfehlen ist, siehe auflistung zur System-Konfiguration in 3.1 auf Seite 19), dann muss vor der Kompilierung des iRODS-Servers, die `wissgridservice`-URL in der Datei `<Installations-Pfad>/res/irodssChangeset/do.sh` angepasst werden (anpassen der URL: http://localhost:8080/wissgridservices/apim?wsdl). Die Anpassung der besagten Datei muss vor dem Bau (Kompilierung) des iRODS Servers erfolgen, da hiervon die Generierung von Code zur Kommunikation mit der `wissgridservice`-Komponente abhängt und ansonsten scheitert. Auch wenn sich die URL der `wissgridservice`-Komponente im Betrieb ändern sollte, muss vorgenannte Änderung durchgeführt werden und iRODS muss neu gestartet werden.

5.5.3 Fedora

Wenn Fedora nicht auf dem selben Host wie iRODS installiert wird oder die Verbindungsdaten geändert wurden, dann sind die Verbindungs- und Benutzerdaten (Admin) der Beans `objectstoreIrodsaccount` und `datastreamstoreIrodsAccount` in der Konfigurations-Datei `<FEDORA_HOME>/server/config/akubra-llstore.xml` anzupassen. Wenn sich die Verbindungs- oder Benutzerdaten im Betrieb ändern, dann muss die Änderung, wie zuvor beschrieben nachgezogen werden und die Fedora Web-Anwendung muss neu gestartet werden. Um die Integrität zwischen Fedora und iRODS weiterhin sicherzustellen, darf sich dadurch nicht die iRODS-Zone ändern (selbe iCAT) .

5.6 Systeme starten

Beim erstmaligen Starten des Systems ist es wichtig, dass iRODS vor Fedora (d.h. Tomcat) gestartet wird, da Fedora in diesem Zeitpunkt seine Standard-Komponenten (z.B. Default Content-Model oder Default Service-Definition) speichert. Wenn der iRODS-Server nicht läuft, scheitert der Fedora-Start.

iRODS: Der Server kann wie folgt gestartet, gestoppt oder neu gestartet werden:

```
$IRODS_HOME/irodsctl {start  | stop  | restart}
```

Davis WebDAV: Starten oder Stoppen des Servers erfolgt mit:

```
/opt/davis/davis/bin/jetty.sh {start  | stop  | restart}
```

Fedora, Solr und `wissgridservice`: Durch Starten oder Stoppen des Tomcat-Servers in dem sie residieren:

```
$CATALINA_HOME/bin/{startup.sh  | shutdown.sh}
```

5.7 Erste Schritte bei der Benutzung

Dieser Abschnitt gibt erste Hinweise zur Nutzung der System und ihrer Schnittstellen, und soll den Einstieg etwas erleichtern.

Fedora:

Es besteht die Möglichkeit das Fedora-System über seine Web-Schnittstellen, die Admin-Konsole oder von der Kommandozeile aus zu erkunden.

Web-Schnittstellen[34]:

- `http://localhost/fedora`

[34] URL kann variieren, abhängig vom gesetzten Kontext-Pfad bei deren Installation und der Einstellung des Apache HTTP Server Proxy-Moduls.

- `http://localhost/fedora/admin`

Admin-Konsole:

- `$FEDORA_HOME/client/bin/fedora-admin.sh`

Aufrufe von der Kommandozeile (Rest-API)[35]:

- neues Objekt erzeugen mit:

```
curl -request POST "http://user:password@localhost/
  fedora/objects/test:1?label=testwas"
```

- Datastream laden:

```
curl -request GET "http://user:password@localhost/
  fedora/objects/test:1/datastreams/MyDS/content"
    -o my.pdf
```

iRODS iCommands

Das iRODS-System kann über die iCommand Kommandozeilen-Tools getestet werden. Diese sind im Verzeichnis `$IRODS_HOME/clients/icommands/bin/` zu finden.

- Daten-Objekt übertragen (Ingest): `iput einFile.txt a.txt`
- Kollektionen/Verzeichnisse anlegen: `imkdir a`
- Objekte im aktuellen Arbeitsverzeichnis auflisten: `ils`
- Hilfe: `iput -h`

Relative Pfadangaben beziehen sich immer auf das `irodsCwd` (current working directory), welches in der `irodsEnv`-Datei gesetzt wurde. Die Datei liegt nach der iRODS-Installation im Verzeichnis `~/.irods/.irodsEnv`.

Davis WebDAV

Das Davis WebDAV Modul kann z.B. über den Browser getestet werden.

```
<host>:<port>/
<host>:<port>/<zone>/home/<user>/
```

GSearch

Nach der Konfiguration erreichbar unter:

`http://localhost/gsearch/rest`[40]

Pazpar2

Diese Anwendung ist erreichbar unter:

`http://localhost/pazpar2/search.pz2`[40]

[35] Ebenfalls abhängig vom Kontext-Pfad.

5.8 Migration

Wenn eine bestehende iRODS-Installation weiterverwendet werden soll, so sollen u.U. auch Daten in Fedora (nachträglich) registriert werden, was durch die Rules in der Datei `<Installations-Pfad>/res/irods/subsequentFedoraIngest.r` unterstützt wird.

Die nachträgliche Registrierung ist auch von Bedeutung für iRODS-Pfade, zu denen in der Property-Datei erst nachträglich eine Abbildung zwischen Verzeichniszweig und Objekt-Typ festgelegt wird (siehe Seite 17).

Die nachträgliche Registrierung (manuell angestoßener Callback) erfolgt durch folgenden Aufruf:

```
irule -F subsequentFedoraIngest.r
        "'/tempZone/home/rods/testColl'" "null"
```

Dem Aufruf wird die zu berücksichtigende iRODS-Kollektion (ohne "/" am Ende) als Parameter übergeben. Der Ingest erfolgt dann in Abhängigkeit des Objekt-Typs (Single-Object, Recursive-Object oder Multi-Object). Wenn die Kollektion nicht in einer der Pfad-Listen (`wissgridservices.properties`) zu den Objekt-Typen, direkt oder als Sub-Pfad enthalten ist, erfolgt keine Registrierung (kein Ingest). Die Ausführung erfolgt verzögert, was den Server beim Scheduling seiner Aufgaben unterstützt. Den Status der Ausführung kann man mit `iqstat` oder `iqstat <id>` anzeigen lassen.

6 Versionsinformationen

In diesem Abschnitt werden Informationen zu Systemanforderungen und den vorgenommenen Anpassungen an Fedora und iRODS gegeben. Zudem werden die Unterschiede zur Vorgängerversion aufgezeigt und auf weitergehenden Entwicklungsbedarf und Schwachstellen hingewiesen.

6.1 Minimale Systemanforderungen

Da sich iRODS nur auf linux-basierten Systemen installieren lässt, wird derzeit nur Linux und OS X unterstützt. Zudem erfordert Fedora eine Java Runtime Environment (JRE) in der Version 1.6.x, und die Installation ist nur unter Tomcat 6.x möglich, weil Fedora mit Tomcat 7.x derzeit nicht zuverlässig startet. Zudem muss Tomcat mit einem ausreichend großen Heap gestartet werden (siehe 5.1 auf Seite 28).

6.2 Anpassungen an Fedora und iRODS

- Im Rahmen der Installation wird der iRODS-Server (Auswahl zwischen Version 2.5 und 3.0) zunächst unverändert installiert und nachträglich für WissGrid angepasst. Dies wird über Installations-Skripte gesteuert. Bei der Anpassung werden neue Microservices und angepasste Rules, sowie einige Änderungen an der Konfiguration, in den iRODS-Server eingebracht. Zudem wird im Rahmen der Installation gSOAP installiert, um Microservices als Web-Service Clients realisieren zu können.
- `WissGrid-D3.5.3-grid-repository-fedora-irods-installer.jar` (Fedora-Installer) enthält statt des Default Akubra-FS Plug-In (File-System), das iRODS-Akubra Plug-In. Zudem wurden einige Veränderungen an der Konfiguration vorgenommen.

6.3 Unterschiede zur Vorgängerversion v0.4.0

- Verbesserte Konfigurationsmöglichkeiten (`wissgridservices.properties`)
- Unterstützung aller drei Objekt-Typen
- Durchgehende Unterstützung von Versionierung über alle Objekt-Typen
- Schnittstellen: SRW/SRU, OAI-ORE, OAI-PMH, verteilte Suche
- Einsatz des Apache HTTP Server und des Apache Proxy Moduls

6.4 Entwicklungsbedarf

Sinnvolle Erweiterungen wären:

- Verbesserte Unterstützung für AuthN/AuthZ.

- Modularisierung, d.h. Umstellung auf OSGi und damit die Möglichkeit der Änderung der Konfiguration, Austausch von Systemteilen (Bundles bzw. Modulen), usw. zur Laufzeit. Es wurde bereits mit der Umstellung begonnen.

6.5 Bekannte Probleme/Schwächen

Im aktuellen Stand der Entwicklung sind die folgende Probleme bekannt :

- Die Installation ist nur unter Linux und OS X möglich.
- Derzeit wird auf iRODS-, wie auch Fedora-Seite, lediglich mit Robot- bzw. Admin-Usern gearbeitet und das Rechte- und Benutzermanagement muss in der Anwendungsschicht realisiert werden.
- Mit der eingesetzte Solr-Version (1.4.1) zeigten sich Probleme bei der Indexierung von PDF-Dateien. Die Ursache liegt vermutlich in der PDFBox-Implementierung, welche die Solr einsetzt.
- Wenn Tomcat nicht ausreichend Heap zugeteilt bekommt, dann kann ein großer Ingest scheitern (bspw. 10 x 500MB gleichzeitig). Das Problem ist weniger Tomcat, als vielmehr Solr. Da Solr jedoch von den Tomcat-Resourecn abhängig ist, bleibt es ein Konfigurationsproblem von Tomcat. Was ausreichend Heap bedeutet, ist vom Anwendungskontext abhängig und muss im Zweifelsfall in einer Testumgebung ermittelt werden.

7 Entwickler-Informationen

Die Informationen in diesem Abschnitt sollen Entwicklern helfen das System zu erschließen und passende Einstiegspunkte für die Entwicklung aufzeigen.

7.1 Aufrufkette

Die Callback Implementierung besteht derzeit aus nachfolgend aufgeführten Regeln, von diesen aufgerufenen Microservices, dem `wissgridservice`-Modul und seinen Methoden, an welche die Microservices die Weiterverarbeitung delegieren.

Beim Ingest über iRODS bspw. feuert die Rule acPostProcForPut , welche einen Aufruf des Microservice msiWissGridCallbackForPutObject veranlasst. Dieser ruft seinerseits die `wissgridservice`-Methode addDS auf.

- Regeln (iRODS-Rules), die bei entsprechenden Aktionen aktiviert werden:
  1. **acPostProcForPut** - Nach Einfügen eines Datenobjekt
  2. acPostProcForCopy - Nach Kopieren eines Datenobjektes
  3. acDataDeletePolicy - *Vor* Löschung eines Datenobjektes
  4. acPostProcForCollCreate - Nach Erzeugung einer Kollektion (iRODS-Verzeichnis)
  5. acPreprocForRmColl - Vor Löschung einer Kollektion
  6. acPostProcForRmColl - Nach Löschung einer Kollektion
- Microservices (C-Routinen), welche von vorgenannten Regeln aufgerufen werden. Die Microservices ermitteln den Objekt-Pfad oder den Kollektions-Pfad und einzelne Metadaten, wie PID oder DSID:
  1. **msiWissGridCallbackForPutObject**
  2. msiWissGridCallbackForCopyObject
  3. msiWissGridCallbackForDeleteObject
  4. msiWissGridCallbackForCollCreate
  5. msiWissGridCallbackForRmColl
- Von den Microservices verwendete Service-Aufrufe (`wissgridservice`):
  1. **addDS** - Neuen Datastream hinzufügen, Indexieren und Versionieren
  2. purgeDS - Datastream entfernen, Index anpassen und Version löschen)
  3. ingestFDO - Fedora Digital Object erzeugen
  4. purgeFDO - Fedora Digital Object löschen
  5. addRS - Beziehung zwischen Objekten herstellen
  6. purgeRS - Beziehung entfernen

7.2 Einstiegspunkte

7.2.1 Callback

Die zuvor genannten Rules, Microservices und `wissgridservice`-Methoden sind die idealen Einstiegspunkte, um die Aufrufkette des Callbacks zu verstehen. Rules nach neuer Rule-Language[36] sind in der Datei `$IRODS_HOME/server/config/reConfig/wissgrid.re` deklariert, nach alter Rule-Language in der Datei `wissgrid.irb`.

Im Verzeichnis `$IRODS_HOME/modules/webservices/microservices/src/wissgrid` liegen WissGrid-spezifischen Microservices.

Der Quellcode zum `wissgridservice`-Modul ist im zugehörigen Web-Archiv enthalten, wobei die Klasse `de.unigoettingen.sub.wissgridservices.callback.impl.WissGridAPIImpl.java`, in welcher die Service-Methoden deklariert sind, Ausgang der Untersuchung sein sollte.

[36] Mit dem iRODS Release-Wechsel von Version 2.5 auf 3.0 wurde eine neue Rule-Language mit wesentlich übersichtlicherer Syntax eingeführt.

8 Lizenzen

Die Entwicklung wird unter Apache Lizenz, Version 2.0 gestellt:

Nachfolgend werden die Lizenzen von eingesetzten Bibliotheken und zu installierender Software aufgelistet.

- Fedora Commons Repository: Apache v. 2.0
  http://www.fedora-commons.org/software/licenses
- iRODS: BSD
  https://www.irods.org/index.php/License
- Jargon-API: BSD
  https://www.irods.org/index.php/Jargon_License
- gSOAP: GPL (GNU Public License)

Anhang A

README

Die nachfolgend aufgeführte Installationsbeschreibung (README bzw. WissGrid-D3.5.3-grid-repository-fedora-irods-README.pdf) ist auch auf der WissGrid Webseite abrufbar unter: D-3.5.3, http://www.wissgrid.de/publikationen/deliverables/wp3.html.

Inhaltsverzeichnis

Installationsanleitung

1 Hinweise

Diese Installationsanleitung (README) liegt nach der Installation im Verzeichnis *doc/* des Installationsverzeichnisses, und sie ist zudem auf der WissGrid Web-Seite abrufbar unter: D-3.5.3 - `http://www.wissgrid.de/publikationen/deliverables/wp3.html`.

- Wenn das Datenbankmanagementsystem, z.B. Postgres, im Rahmen der iRODS-Installation geladen und installiert wird, dann ist die Installation des iRODS-Servers

sehr zeitaufwändig (>30 Minuten). Bei Nutzung einer vorliegende Datenbank-Installation (Postgres, mysql, ...) verläuft die Installation wesentlich schneller

- Bei der Installation der Komponenten ist die angegebene Reihenfolge einzuhalten, weil z.B. die Web-Service Komponente (wissgridservices) installiert und gestartet sein muss, damit bei der iRODS Installation, Code für Service-Aufrufe generiert werden kann.
- Das System wurde unter Tomcat 6.x getestet, unter Tomcat 7.x gab es Probleme mit Fedora (für weiteres wird auf die Fedora-Webseite verwiesen). Im folgenden wird die Installation mit Tomcat 6.x beschrieben, wobei Tomcat ausreichend Heap-Speicher zugewiesen werden muss.
- Für den Einsatz des Apache HTTP Server genügt es, wenn man diesen vom Linux Distributions Repository installiert. Dabei muss man beachten, dass 'mod_proxy' und 'mod_proxy_ajp' zusätzlich installiert werden.

2 Voraussetzungen

1. Mindestens 1024MB Arbeitsspeicher.
2. Java JRE in Version 1.6.x
3. Tomcat in Version 6.x
4. In Konsole/Terminal testen, ob die Entwicklungstools 'g++', 'yacc', 'flex' auf dem vorliegenden System installiert sind, ggf. müssen diese nach-installiert werden (ohne diese Tools scheitert die Installation).

   Test, z.B. durch Abfragen der Version:

   g++ -v
   yacc -V
   flex -V

3 Überblick

Die Installation erfolgt in zwei Etappen:

1. Ausführen des WissGrid-Installationsprogramms, welches einzelne Parameter abfragt und die Installationskomponenten (wissgridservices, Fedora, iRODS, Änderungs-/Erweiterungsdateien für iRODS, gSOAP, Solr) in ein Installations Verzeichnis kopiert (siehe 4.1).

   Dieses Verzeichnis wird nachfolgend Bezeichnet mit: `Installationsverzeichnis` oder `Installations-Pfad`.

2. Anschließend müssen die Installationskomponenten (iRODS, Fedora, ...) installiert werden. Dieser Schritt wird im wesentlichen über Shell-Skripte gesteuert (siehe 4.2).

   Bei den Verzeichnissen, in welche diese Komponenten installiert werden, wird von Home-Verzeichnis oder Installationspfad der Komponente gesprochen (`IRODS_HOME` oder `IRODS-Installationspfad`).

Bei einer Testinstallation ist es empfehlenswert alle Komponenten (iRODS, Fedora, ...) auf einem Host zu installieren. Es ist aber auch möglich und sinnvoll, eine verteilte Installation vorzunehmen. Die lokale Installation erfolgt entsprechend den Ausführungen in dieser README. Für die verteilte Installation muss der Installer auf allen Hosts ausgeführt werden, wobei nur die Installationsskripte der Komponenten, die auf dem betreffenden Host von Interesse sind, ausgeführt werden. Bei der verteilten Installation sind zudem die in Abschnitt 5.5 der Dokumentation[1] beschriebenen Anpassungen an den Konfigurationsdateien zu beachten.

4 Installation

Zunächst sind die folgenden Umgebungsvariablen zu setzen. Dies ist nicht nur für den Betrieb wichtig, sondern vereinfacht auch die Installation, da über die Umgebungsvariablen auch die Pfade in den zugehörigen Eingabemasken des Installationsprogramms korrekt vorbelegt werden.

- export IRODS_HOME=<iRODS-Installationspfad>
- export FEDORA_HOME=<Fedora-Installationspfad>
- export SOLR_HOME=<Solr-Installationspfad>
- export CATALINA_HOME=<Tomcat-Installationspfad>
- export JAVA_OPTS="-server -Xms768M -Xmx1024M -XX:MaxPermSize=768M"
- PATH=$PATH:$IRODS_HOME/clients/icommands/bin:
  $IRODS_HOME/

Es bietet sich an, die Umgebungsvariablen in der Datei `~/.profile` zu setzen, womit sie auch nach einem erneuten Systemstart korrekt gesetzt sind (Ausdrücke in <..> müssen auf lokale Gegebenheiten angepasst werden, z.B. `export IRODS_HOME=/opt/iRODS`). Damit die Änderungen auch übernommen werden, ist der Aufruf `source ~/.profile` auf der Kommandozeilen erforderlich.

4.1 WissGrid-Installationsprogramm

Beim Auslieferungspaket WissGrid_D3.5.3-grid-repository-fedora-irods-installer.jar handelt es sich um ein ausführbares Java-Archiv, welches den ersten o.g. Installationsschritt durchführt. Der Aufruf erfolgt durch:

[1] siehe: <Installationspfad>/doc/WissGrid-D3.5.3-grid-repository-fedora-irods-Dokumentation.pdf

```
java -jar WissGrid_D3.5.3-grid-repository-fedora-irods-installer.jar
```

Der Installer umfasst die folgenden Schritte (bzw. Seiten):

1. Allgemeine Infoseite
2. Informationen (diese README)
3. Lizenzseite und Bestätigung
4. Auswahl eines temporären Installationsverzeichnisses (<Installations-Pfad>)
5. Benutzerdaten zu iRODS (u.a. `IRODS_HOME`)
6. Benutzerdaten zu Fedora (u.a. `FEDORA_HOME`)
7. Sonstige Benutzerdaten (u.a. `Tomcat_HOME, SOLR_HOME`)
8. Fortschrittsanzeige zur Installation
9. Informationen zum weiteren Vorgehen
10. Abschlussseite

Nach Abschluss des Installationsprogramms, ist im Installationsverzeichnis (gemäß Angabe in Schritt 4) folgende Struktur vorhanden, wobei diese README auch im Unterverzeichnis 'doc/' zu finden ist:

<**Installations-Pfad**>/
- Lizenz.txt
- **doc**/
  - WissGrid_D3.5.3-grid-repository-fedora-irods-**README**.pdf (diese)
  - WissGrid_D3.5.3-grid-repository-fedora-irods-**Dokumentation**.pdf
- **licenses**/
  - Lizenztexte
- **res**/
  - irods2.5.tgz
  - irods3.0.tgz
  - wissgridservices.war
  - fcrepo-installer_WissGrid-AP3-Deliverable3.5v0.5.0.jar
  - **irodsChangeset**/
    - do.sh
    - **ERA**/
      - info.txt
    - gsoap_2.8.0.zip
    - gsoap.sh
    - irods_wissgridservices_MS.c
    - irods_wissgridservices_MS.h

Makefile
microservices.header
microservices.table
subsequentFedoraIngest.r
testsetup.sh
tomcat-users.xml
webservices/
info.txt
wissgrid.irb
wissgrid.re
scripts/
cleanUpIrods.sh
config.sh
fedora.sh
irods.sh
irodsChanges.sh
solr.sh
wissgridservices.sh
solr/
...
solr_sources/
...
wissgridservices.war

Im Verzeichnis `res/` liegen die Installationsprogramme für iRODS und Fedora, sowie die WAR-Datei der Service-Komponente.

Die Anpassungsdateien für iRODS, die gSOAP-Distribution und Shell-Skripte zum Steuern der gSOAP-Installation (`gsoap.sh`) und generieren von Code für Service-Aufrufe (`do.sh`) liegen im Verzeichnis `res/irodsChangeset/`.

4.2 Installationskomponenten

Die folgenden Schritte sind auszuführen, nachdem das WissGrid Installationsprogramm erfolgreich abgeschlossen wurde. Die einzelnen Schritte werden von Skripten gesteuert, wobei jeweils Hinweise zum aktuellen und den anschließenden Schritten gegeben werden. Begonnen wird mit dem Skript `<Installations-Pfad>/res/scripts/setEnv`.

1. Umgebungsvariablen setzen

   Wenn die Umgebungsvariablen noch nicht gesetzt wurden (vgl. 4), dann kann man das mit dem Skript `<Installations-Pfad>/res/scripts/setEnv` nachholen. Das Skript setzt die Umgebungsvariablen (`IRODS_HOME, FEDORA_HOME, CATALINA_HOME, SOLR_HOME`), wie sie im WissGrid-Installationsprogramm angegeben wurden.

   Aufruf: `source setEnv`

2. iRODS-Installation

Hinweis: Bei dieser Auslieferung handelt es sich um eine prototypische Entwicklung, deren Eignung für den Produktivbetrieb nicht gewährleistet werden kann.

Wenn man bereits einen iRODS-Server installiert hat, können die WissGrid-betreffenden Änderungen auch daran vorgenommen werden (siehe 4. iRODS Veränderungen). In diesem Fall kann dieser Schritt (2. iRODS-Installation) vollständig übersprungen werden.

(a) Shell-Skript `<Installations-Pfad>/res/scripts/irods.sh` aufrufen und Ausgabe bzw. Hinweise beachten (siehe 6.1).
Aufruf: `./irods.sh`

(b) optional: Wenn man mehrere iRODS-Accounts oder -Server verwenden will, kann man dafür verschiedene irodsEnv-Dateien[2] anlegen, z.B. `.irodsEnv_user1` für Benutzer `user1`. Ein Wechsel zwischen den Accounts ist dann durch Setzen der Variable `irodsEnvFile=~/.irods/.irodsEnv_user1` möglich. Anschliessend ist ein Aufruf von `iinit` erforderlich, um das für die iCommands hinterlegte Passwort anzupassen. Die in dem besagten Verzeichnis liegende Standard Env-Datei kann als Muster verwendet werden.

Am Ende dieses Installationsschritts wird man aufgefordert, folgende Dateien zu Editieren:

(a) In `$IRODS_HOME/server/config/irodsHost` muss das Mapping von `localhost` auf die zugehörige `<aktuelle-IP-Adresse>` festgelegt werden. Das geschieht, indem am Ende folgende Zeile hinzugefügt wird, wobei `<local-ip-address>` durch die aktuelle IP-Adresse zu ersetzen ist):

```
localhost <local-ip-address>
```

(b) In der Datei `$IRODS_HOME/server/config/server.config` muss dem Attribut `reRuleSet wissgrid` hinzugefüpgt werden, so dass in der betreffenden Zeile mindestens folgendes steht:

```
reRuleSet wissgrid,core.
```

(c) In der Datei `$IRODS_HOME/config/config.mk` muss dem Attribut `MODULE` die Eigenschaft `webservices` und `ERA` hinzugefügt werden, so dass in der betreffenden Zeile folgendes steht:

```
MODULES= webservices properties ERA
```

3. Installation der Web-Service-Komponente

   Shell-Skript `<Installations-Pfad>/res/scripts/wissgridservice.sh` aufrufen und Hinweise beachten.

   Aufruf: `./wissgridservice.sh`

4. iRODS Veränderungen für WissGrid vornehmen

   Shell-Skript `<Installations-Pfad>/res/scripts/irodsChanges.sh` aufrufen und Hinweise beachten.

   Aufruf: `./irodsChanges.sh`

[2] Enthält Verbindungsdaten, Benutzername und Default-Einstellungen, wie z.B. dem aktuellen Arbeitsverzeichnis auf iRODS.

5. Fedora-Installation

   Shell-Skript `<Installations-Pfad>/res/scripts/fedora.sh` aufrufen und Hinweise beachten (zur Fedora-Installation siehe 6.2).

   Aufruf: `./fedora.sh`

6. Solr-Installation

   Für die Solr-Installation werden zwei alternative Wege beschrieben. 6a beschreibt eine schnelle und ebenso einfache Installation, welche durch ein Shell-Skript unterstützt wird und im wesentlichen eine Solr-Instanz anlegt und in Tomcat registriert. Bei der zweiten Alternative, welche 6b beschreibt, wird eine Solr-Instanz aus den Solr-Sourcen erstellt, nachdem diese auf die WissGrid-Bedürfnisse angepasst wurden. Dieser Weg wäre für eine bestehende Solr-Instanz zu beschreiten - jedoch sei auch an dieser Stelle nochmals auf den prototypischen Charakter der Entwicklung hingewiesen.

   (a) Solr-Demo-Instanz anlegen

   Shell-Skript `<Installations-Pfad>/res/scripts/solr.sh` aufrufen und Hinweise beachten.

   Aufruf: `./solr.sh`

   (b) Solr-Sourcen anpassen, Solr bauen und in Tomcat registrieren

   i. Laden der Solr-Distribution und anschliessend entpacken. Der Entpackungsort wird im folgenden mit `<SOLR_DIR>` bezeichnet.
   Download (in Version 1.4.1): http://lucene.apache.org/solr/

   ii. `<Installations-Pfad>/res/solr/ContentStreamBase.java` nach `<SOLR_DIR>/src/common/org/apache/solr/common/util/` kopieren.

   iii. `<Install.-Pfad>/res/solr/jargon-core-2.4.1-SNAPSHOT.jar` nach `<SOLR_DIR>/lib/` kopieren.

   iv. Solr bauen, d.h. in `<SOLR_DIR>/` folgende Aufrufe absetzen

   ```
   ant clean
   ant dist
   ```

   v. `SOLR_HOME` (Kontext-Pfad) anlegen und anpassen

   Dieses Verzeichnis (z.B. `/opt/solr/testinstance/solr`) bildet das Kontext-Verzeichnis für die Solr-Webanwendung. Hier werden Konfigurationsdateien abgelegt und die eigentlichen Index-Daten gespeichert. Das Verzeichnis `<SOLR_HOME>` sollte nicht mit dem Installationspfad `<SOLR_DIR>` verwechselt werden.

   - `<SOLR_DIR>/example/solr/` nach `/opt/solr/testinstance/` kopieren.
   - `<SOLR_DIR>/dist/apache-solr-1.4.1.war` nach `<SOLR_HOME>` kopieren.

   vi. `<SOLR_HOME>/conf/solrconfig.xml` anpassen

   - `<dataDir>`-Element auf `<SOLR_HOME>/data` setzen, z.B.:

```
<dataDir>
   ${solr.data.dir:/opt/solr/testinstance/solr/data}
</dataDir>
```

Hinweis:

Wenn Solr nicht startet (Test mit Aufruf `http://localhost:8080/solr` und in der Tomcat Log-Datei (`<tomcat>/logs/catalina.out`) darauf hingewiesen wird, dass `<SOLR_HOME>` nicht gefunden werden kann, dann sollte die Erweiterung der `JAVA_OPTS` Umgebungsvariablen helfen:

```
export JAVA_OPTS="$JAVA_OPTS
   -Dsolr.solr.home=/opt/solr/testinstance/solr"
```

In `~/.profile` aufnehmen und Änderung mit `source ~/.profile` aktivieren.

- Element `<requestHandler name="/update/extract"...>` editieren und folgendes ändern bzw. ersetzen:

```
<lst name="defaults">
   <str name="fmap.content">attr_content</str>
   <str name="lowernames">true</str>
   <str name="uprefix">attr_</str>
  ...
```

Hinweis:

- `fmap.content` bildet Content auf ein dynamisches Feld `attr_content` ab.
- `uprefix` hänget Elemente den Zusatz `attr_` vornan, wenn sie nicht direkt einem Feld zugeordnet sind. Damit werden sie vom dynamischen Feld `attr_*` behandelt.

vii. `<SOLR_HOME>/conf/schema.xml` anpassen

- Folgende Elemente bei den Feld-Definitionen hinzufügen:

```
<!- wissgrid-fields ->
<field name="pid" type="text" indexed="true"
     stored="true" required="false"/>
<field name="dsid" type="text" indexed="true"
     stored="true" required="false" />
<field name="timestamp" type="text" indexed="true"
     stored="true" required="false" />
<field name="dslocation" type="text" indexed="true"
     stored="true" required="false" />
```

- Das dynamische Feld `attr_*` bestimmt, wie mit nicht explizit definierten Feldern umgegangen werden soll, z.B. mit Content (siehe 6(b)vi).

```
<dynamicField name="attr_*" type="textgen"
     indexed="true" stored="false"
```

```
multiValued=‘‘true‘‘/>
```

Hier wird der Content bspw. indexiert, nicht aber gespeichert. Der Content ist damit nicht über den Index verfügbar.

viii. Tomcat Context Fragment erstellen

Dazu wird eine Datei `<SOLR_HOME>/temp/solr.xml` erstellt:

```
<?xml version='1.0' encoding='utf-8'?>
<Context debug='0' crossContext='true'
   docBase=
     '/opt/solr/testinstance/solr/apache-solr-1.4.1.war'>
  <Environment name=‘solr/home' type=‘java.lang.String‘
    value='/opt/solr/testinstance/solr' override='true'/>
</Context>
```

In 6(b)x wird Solr über diese Datei auf Tomcat installiert.

ix. Bibliotheken kopieren:

- `<SOLR_HOME>/lib/` anlegen
- Kopieren aller Dateien aus `<SOLR_DIR>/contrib/extraction/lib/` nach `<SOLR_HOME>/lib/`
- Kopieren der Datei `<SOLR_DIR>/dist/apache-solr-cell-<vers.>.jar` nach `<SOLR_HOME>/lib/`

x. Installieren von Solr in Tomcat

`<SOLR_HOME>/temp/solr.xml` nach `<tomcat>/conf/Catalina/localhost/` kopieren (siehe 6(b)viii). Tomcat nimmt diese Datei automatisch auf und Installiert darüber Solr. Bei erfolgreicher Installation wird die Datei automatisch gelöscht.

xi. Die Installation kann nun mit `http://localhost:8080/solr` getestet werden.

7. Davis-Installation

(a) Gemäß:

- `https://projects.arcs.org.au/trac/davis/wiki/HowTo/Install`
- `https://projects.arcs.org.au/trac/davis/wiki/HowTo/Configuration`

Die Installation des WissGrid-Repository ist hier abgeschlossen. Nachfolgend wird die Installation von Fedora-Plug-Ins und zusätzlichen Diensten beschrieben, welche den Funktionsumfang des Repository erweitern und deren Installation daher empfohlen wird.

4.3 Zusätzliche Komponenten

1. **ORE Provider**

Diese Komponente kann unter folgender Adresse heruntergeladen werden:

http://sourceforge.net/projects/oreprovider/files/oreprovider/oreprovider%200.4%20Beta%202/oreprovider-0.4b2.war/download

Die offizielle Online Dokumentation findet man unter:

http://oreprovider.sourceforge.net/examples.html

Es handelt sich um eine WAR-Datei, die zur Installation in das webapp-Verzeichnis des Tomcat-Servers gelegt werden muss. Die Konfiguration erfolgt über die Property-Datei $CATALINA_HOME/webapps/oreprovider/WEB-INF/classes/ore.properties, in welcher folgende Einstellungen vorgenommen werden müssen:

- `fedora.server = http://localhost:8080/fedora`
- `rem.autocreation = true`

2. **OAI Provider**

Die Komponente kann unter folgender Adresse Herunterladen werden:

http://downloads.sourceforge.net/fedora-commons/oaiprovider-1.2.2.zip

Die offizielle Online Dokumentation findet man unter:

https://wiki.duraspace.org/display/FCSVCS/OAI+Provider+Service+1.2.2

Vor der Installation der Komponente muss eine Datenbank angelegt und ein Benutzer mit vollen Zugriffsrechten darauf erstellt werden. Hier beispielhaft für PostgreSQL beschrieben:

- Erstellung der Datenbank *oaiPMH*:
  `/path/to/postgres/pgsql/bin/createdb -h localhost oaiPMH`
- Definition eines Benutzers *oaiAdmin*, Kennwort *oaiAdmin*: (kein *Superuser*!)
  `/path/to/postgres/pgsql/bin/createuser -h localhost oaiAdmin`
- In `/path/to/postgres/pgsql/data/pg_hba.conf` kann nun das Recht zum Zugriff auf die Datenbank gesetzt werden, indem folgende Zeile hinzugefügt wird:
  `host oaiPMH ¨oaiAdmin¨ ¨IP-ADRESSE¨ ¨255.255.255.255¨ md5.`
- Danach erstellt man eine Verbindung zum *PostgreSQL*-Server und setzt das Kennwort für den Benutzer *oaiAdmin*:
  `/path/to/postgres/pgsql/psql -h localhost -d template1 -E`
  `template1=# ALTER USER ¨oaiAdmin¨ PASSWORD ´oaiAdmin´`
- Um zu prüfen, ob das Kennwort gesetzt wurde, dient das folgende SELECT, wobei das Kennwort MD5-kodiert angezeigt werden sollte:
  `SELECT usename,password FROM pg_shadow;`

Nach der Intergration der WAR-Datei in Apache Tomcat müssen folgende Werte in $CATALINA_HOME/webapps/oaiprovider/WEB-INF/classes/proai.properties geändert werden (hier mit beispielhaften Werten):

- `proai.cacheBaseDir` = `/path/to/tomcat/webapps/oaiprovider/cache`
- `proai.sessionBaseDir` = `/path/to/tomcat/oaiprovider/sessions`

- proai.schemaDir = /path/to/tomcat/webapps/oaiprovider/schemas
- proai.db.url = jdbc:postgresql://localhost/oaiPMH
- proai.driverClassName = org.postgresql.Driver
- proai.db.username = oaiAdmin
- proai.db.password = oaiAdmin
- org.postgresql.Driver.ddlConverter = proai.util.
  PostgresDDLConverter
- org.postgresql.Driver.backslashIsEscape = true
- driver.fedora.identify = http://localhost:8080/oaiprovider
  /Identify.xml
- driver.fedora.md.format = oai_doc
- driver.fedora.md.format.oai_dc.dissType = info:fedora/*/oai_dc
- driver.fedora.md.format.oai_dc.about.dissType = info:fedora/*/oai_dc
- driver.fedora.md.format.oai_dc.about.dissType = info:fedora/*/DC

Es ist zu erkennen, dass PostgreSQL als Datenbank gewählt wurde. Bei Verwendung eines anderen Datenbankserver sind die Einstellungen entsprechend zu ändern.

Nun muss die Datei Identify.xml in $CATALINA_HOME/webapps/oaiprovider erstellt werden. Ein Beispiel dazu ist im Abschnitt 6.3 angegeben.

3. **Fedora GSearch**

   GSearch kann unter folgender Adresse geladen werden:
   http://sourceforge.net/projects/fedora-commons/files/services/3.1/genericsearch-2.2.zip/download

   Die zugehörige Dokumentation ist zu finden unter:

   https://wiki.duraspace.org/display/FCSVCS/Generic+Search+Service+2.2

   Auch bei GSearch sind Anpassungen bzw. Einstellung vorzunehmen. Nachfolgende Auflistung zeigt einen Teil der Konfigurationsmöglichkeiten, eine ausführliche Beschreibung folgt im Abschnitt 6.4:

   (a) Bezüglich des Apache Tomcat Servers, unterhalb des Verzeichnisses $CATALINA_HOME/webapps/fedoragsearch/WEB-INF/classes/:

   - log4j.xml
   - Alle demo...-Dateien in config/rest/ wurden umbenannt. Die Namen werden später in der Property-Datei korrigiert. Auch in diesen Dateien soll die Zeile CONFIGPATH/rest/demoCommon.xslt mit dem richtigen Pfad und Dateinamen ersetzt werden.
   - config/fedoragsearch.properties
   - config/index/INDEXNAME/index.properties
   - config/index/INDEXNAME/indexInfo.xml
   - config/repositroy/REPOSITROYNAME/repositroy.properties
   - config/repositroy/REPOSITROYNAME/repositroyInfo.xml

- `config/updater/UPDATERNAME/update.properties`

(b) Beim Apache Solr Server unter `$SOLR_HOME`[3]:

- Ordner `data/gsearchindex`
- `conf/solrConfig.xml`
- `conf/schema.xml`

(c) Die Konfigurationsdatei `fedora.fcfg` (Fedora Common Konfigurationsdatei) unter `$FEDORA_HOME/server/config` muss überarbeitet werden.

(d) Es wird empfohlen, alle Solr- und Lucene-Bibliotheken durch aktuelleren zu ersetzen (durch Versionen ab 2.9.x bzw. 1.4.x). Zudem müssen dann auch die Versionsbezeichner in folgenden Dateien angepasst werden:

- `library.properties` unter `$CATALINE_HOME/webapps/fedoragsearch/WEB-INF/lib`
- `runRESTClient.sh` und `runSOAPClient.sh` unter `$CATALINA_HOME/webapps/fedoragsearch/client`

4. **Pazpar2**

Es wird bei dieser Installation empfohlen, den Anweisung der Linux Distribution zu folgen. Die Anwendung kann unter folgender Adresse herunterladen:

`http://ftp.indexdata.dk/pub/pazpar2`

Die offizille Online Dokumentation findet man unter:

`http://www.indexdata.com/pazpar2/doc/`

`http://www.indexdata.com/pazpar2/doc/pazpar2.pdf`

Vor der Installation sind aktuelle Versionen der Bibliotheken `libyaz4` und `libyazpp` zu laden und installieren.

Zur Konfiguration müssen folgende Dateien angepasst und/oder bestimmte Konfigurationsdateien aktivieren[4] werden. Ausführliche Beispiele und Anweisungen sind im Abschnitt 6.5 zu finden:

- `/etc/pazpar2/server.xml`
- `/etc/pazpar2/services-available/default.xml`
- `/etc/pazpar2/services-available/solr.xml`
- Erstellen Symbolischer Links von beiden letzten Dateien in `/etc/pazpar2/services-enabled`

5. **Apache HTTP Server**[2]

Zum Beispiel für Debian als '*root*':

```
$ apt-get install apache2 apache2-common
```

Das Ziel ist, eine einheitliche URL-Adressierung für alle installierten Komponenten zu erreichen. Aus diesem Grund sollen einige "Module" nach der Installation vom Server aktiviert werden, wie `mod_proxy`, `mod_proxy_ajp` und `mod_rewerite`.[2] Ein Konfigurationsbeispiel für Debian ist im Abschnitt 6.6 zu finden.

[3] Hier gehen wir davon aus, dass Sie nur einen Instanz von **Apache Solr Server** installiert haben. Eine Alternative Einleitung wird noch im Abschnitt 6.4 beschrieben.

[4] Gemäß der Anweisung von Linux Distribution

5 Mögliche Probleme bei der Installation

1. **Postgres und unixODBC:** Das iRODS Installations-Skript lädt bei entsprechender Auswahl den Postgres-Server und unixODBC Bibliothek nach. Bei Test Installationen war die hinterlegte Adresse wiederholt nicht erreichbar, so dass die Installation abbrach. Um das Problem zu lösen, können die Dateien `postgresql-9.0.3.tar.gz` und `unixODBC-2.2.12.tar.gz` manuell geladen werden.

   Damit die geladenen Sourcen gefunden werden können, muss von der Muster-Datei `$IRODS_HOME/config/installPostgres.config.template` eine Kopie angelegt werden (`installPostgres.config` im selben Verzeichnis), und es sind folgende Angaben, entsprechend den lokaler Gegebenheiten, anzupassen.

   ```
   # (Ziel) Pfad für Postgres-Sourcen
   $POSTGRES_SRC_DIR = '/opt/postgres';

   # Postgres (Ziel) Installationsverzeichnis
   $POSTGRES_HOME = '/opt/postgres/pgsql';

   # Wo liegen die Dateien (die oben geladenen)
   $DOWNLOAD_DIR = '/home/userXYZ/Downloads/postgres';

   $POSTGRES_SOURCE = 'postgresql-9.0.3.tar'; # bzw. .tar.gz
   $ODBC_SOURCE = 'unixODBC-2.2.12.tar'; # bzw. .tar.gz
   ```

2. **Berechtigungen:** Sollten beim Start der Server oder im Betrieb Schwierigkeiten auftreten, so könnte dies durch fehlende Schreib-Rechte des Benutzers verursacht werden. Z.B. schreibt iRODS, wenn nicht anders konfiguriert, in ein Verzeichnis '`$IRODS_HOME/Vault/`' (Standard iRODS-Ressourcen). Wenn die Rechte zum Schreiben in diesem Verzeichnis fehlen, verursacht das einen Fehler.

3. Ein gescheiterter Ingest oder die Fehlfunktion des Tomcat-Servers im Rahmen eines Ingests, mit der Fehlermeldung `OutOfMemory Error: PermGen Space` in der Tomcat Log Datei `catalina.out`, kann seine Ursache darin haben, dass Tomcat nicht mit ausreichend Heap-Speicher gestartet wurde.

   Die Zuweisung des Heaps erfolgt beim Tomcat-Start durch Setzen der Umgebungsvariable `JAVA_OPTS`. Nachfolgend ist eine beispielhafte Konfiguration angegeben, wobei zu prüfen ist, ob diese für die eigenen Anwendungsfälle ausreichend ist (`Xms` setzt die Startgröße und `Xmx` die maximale Größe des Heap).

   ```
   JAVA_OPTS="-server -Xms768M -Xmx1024M -XX:MaxPermSize=768M"
   ```

6 Eingabehinweise

6.1 iRODS - Installationsangaben

Im wesentlichen sollten die Standardwerte gewählt werden. Wenn die erste Aufforderung mit “no“ beantwortet wird, müssen lediglich der Zielpfad für die neue Postgres-Instanz und ein Postgres Passwort für den Benutzer angegeben werden. Die folgende Ausgabe ist verkürzt, der Installer liefert weitere Informationen.

```
Include additional prompts for advanced settings [no]?
Build an iRODS server [yes]?
Make this Server ICAT-Enabled [yes]?
iRODS zone name [tempZone]?
iRODS login name [rods]?
Password [rods]?
Download and build a new Postgres DBMS [yes]?
New Postgres directory? /path/to/postgres          #Eingabe
New database login name [user]?
Password? aPassword                                # Eingabe
PostgreSQL version [postgresql-9.0.3.tar.gz]?
ODBC version [unixODBC-2.2.12.tar.gz]?
Port [5432]?
Include GSI [no]? yes                              # Eingabe, wenn
                                                   # GSI erwünscht
GLOBUS_LOCATION? /opt/globus                       # Eingabe, gemäß den
                                                   # Gegebenheiten
GSI Install Type to use? gcc64dbg                  # Eingabe
Save configuration (irods.config) [yes]?           # abgelegt in:
                                                   # $IRODS_HOME/config/
Start iRODS build [yes]?
...
```

6.2 Fedora Installationsangaben

Auch hier werden im wesentlichen Standardwerte verwendet und der Installer liefert ebenfalls weitergehende Informationen.

```
Installation type
Enter a value ==> custom                                  # Eingabe

Fedora home directory
Enter a value [default is $FEDORA_HOME] ==>

Fedora administrator password
Enter a value ==> fedoraAdmin                             # Eingabe

Fedora server host
Enter a value [default is localhost] ==>                  # Auf default belassen,
                                                          # sonst stimmt die iRODS
                                                          # Konfiguration nicht mehr

Fedora application server context
Enter a value [default is fedora] ==>                     # Auf default belassen,
                                                          # sonst stimmt die iRODS
                                                          # Konfiguration nicht mehr

Authentication requirement for API-A
Enter a value [default is false] ==>

SSL availability
Enter a value [default is true] ==> false                 # Eingabe, SSL wird
                                                          # für erste Tests nicht
                                                          # benötigt

Servlet engine
Enter a value [default is included] ==> existingTomcat    # Eingabe. Existiert
                                                          # eine Tomcat Instanz?


Tomcat home directory
Enter a value ==> /path/to/tomcat/                        # Eingabe. Wird mit
                                                          # $CATALINA_HOME
                                                          # automatisch gesetzt

Tomcat HTTP port
Enter a value [default is 8080] ==>

Tomcat shutdown port
Enter a value [default is 8005] ==>

Database
Enter a value ==> included                                # Eingabe. Das Einbinden
                                                          # einer existierenden DB
                                                          # verkürzt die Installation

Enable FeSL AuthN
Enter a value [default is true] ==>
```

```
Enable FeSL AuthZ (Experimental Feature)
Enter a value [default is false] ==>          # Alternative ist AuthZ
                                              # über XACML

Policy enforcement enabled
Enter a value [default is true] ==>           # Um Rechte festlegen zu
                                              # können. Definitionen
                                              # liegen in:
                                              # '$FEDORA_HOME/
                                              # data/../../default/'


Low Level Storage
Enter a value [default is akubra-fs] ==>      # Es wird trotzdem das
                                              # iRODS-Akubra Modul
                                              # installiert

Enable Resource Index
Enter a value [default is false] ==>          # Für erste Tests nicht benötigt,
                                              # verwaltet RDFs in Triplestore

Enable Messaging
Enter a value [default is false] ==>          # Für erste Tests nicht benötigt,
                                              # externe Systeme lassen sich
                                              # hierüber integrieren.

Deploy local services and demos
Enter a value [default is true] ==> false     # Eingabe
```

6.3 OAI Provider - Beispiel für `Identify.xml`

```
<Identify>
     <repositoryName>WissGrid Aggregator</repositoryName>
     <baseURL/>
     <protocolVersion>2.0</protocolVersion>
     <adminEmail>mailto:samadi@sub.uni-goettingen.de</adminEmail>
     <earliestDatestamp>2011-01-01T00:00:00Z</earliestDatestamp>
     <deletedRecord>no</deletedRecord>
     <granularity>YYYY-MM-DDThh:mm:ssZ</granularity>
     <description>

        <oai-identifier xsi:schemaLocation="¨http://www.openarchives.org/OAI/2.0
         /oai-identifier http://www.openarchives.org/OAI/2.0/oai-identifier.xsd¨">

            <scheme>oai</scheme>
            <repositoryIdentifier>wissgrid.org</repositoryIdentifier>
            <delimiter>:</delimiter>
            <sampleIdentifier>oai:wissgrid:2</sampleIdentifier>

        </oai-identifier

     </description>

</Identify>
```

6.4 Fedora GSearch - Ausführliche Einstellungen

- Beispiel für `log4j.xml`:

```
<appender name="FILEOUT" class="org.apache.log4j.FileAppender">
     <param name="File"
      value="/path/to/fedora/server/logs/gsearch.log"/>
     ...
<logger name="dk.defxws.fgslucene" additivity="true">
     <level value="DEBUGLEVELL" />
     <appender-ref ref="FILEOUT"/>
</logger>
<logger name="dk.defxws.fgssolr" additivity="true">
     <level value="DEBUGLEVELS" />
     <appender-ref ref="FILEOUT"/>
</logger>

<root>
     <level value="DEBUGLEVELS" />
     <appender-ref ref="FIELOUT"/>
</root>
```

- In `$CATALINA_HOME/webapps/fedroagsearch/WEB-INF/classes/` sieht die Ordnerstruktur wie folgt aus:

```
config/
        |-index/SolrIndex/conf
        |-repositroy/WissGridRepo
        |-rest/css
        |-updater/BasicUpdaters
```

- Beispiel für `config/fedoragsearch.properties`:

```
fedoragsearch.soapBase =                          http://localhost:8080/gsearch/services
fedoragsearch.soapUser =                          fedoraAdmin
fedoragsearch.soapPass =                          fedoraAdmin
fedoragsearch.deployFile =                        /path/to/tomcat/webapps/fedoragsearch/
                                                   WEB-INF/classes/config/deploy.wsdd

#rest
fedoragsearch.defaultNoXslt =                     copyXml
fedoragsearch.defaultUpdateIndexRestXslt =        UpdateIndexToHtml
fedoragsearch.defaultGfindObjectsRestXslt =       GfindObjectsToHtml
fedoragsearch.defaultBrowseIndexRestXslt =        BrowseIndexToHtml
fedoragsearch.defaultGetRepositoryInfoRestXslt =  GetRepositoryInfoToHtml
fedoragsearch.defaultGetIndexInfoRestXslt =       GetIndexInfoToHtml
```

```
#resultPage
fedoragsearch.maxPageSize =                          100
fedoragsearch.defaultBrowseIndexTermPageSize =       25
fedoragsearch.defaultGfindObjectsHitPageSize =       10
fedoragsearch.defaultGfindObjectsSnippetsMax =       3
fedoragsearch.defaultGfindObjectsFieldMaxLength =    150
fedoragsearch.mimeTypes =                            text/plain text/html application/pdf
fedoragsearch.repositoryNames =                      WissGridRepo
fedoragsearch.indexNames =                           SolrIndex
fedoragsearch.updaterNames =                         BasicUpdaters
```

- Beispiel für `config/index/SolrIndex/index.properties`:

```
fgsindex.indexName =                        SolrIndex
fgsindex.operationsImpl =                   dk.defxws.fgssolr.OperationsImpl

fgsindex.defaultUpdateIndexDocXslt =        FoxmlToSolr
fgsindex.defaultUpdateIndexResultXslt =     updateIndexToResultPage
fgsindex.defaultGfindObjectsResultXslt =    gfindObjectsToResultPage
fgsindex.defaultBrowseIndexResultXslt =     browseIndexToResultPage
fgsindex.defaultGetIndexInfoResultXslt =    copyXml

fgsindex.indexBase =                        http://localhost:8080/solr/GSearch
fgsindex.indexDir =                         /path/to/solr/cores/gsearch/data/index
fgsindex.analyzer =                         org.apache.lucene.analysis.standard.StandardAnalyzer

fgsindex.defaultQueryFields =               dc.description dc.title dc.creator dc.identifier
                                            dc.subject dc.date dc.type dc.coverage dc.publisher
                                            dc.contributor dc.source pid PID dsid version
                                            timestamp dslocation

fgsindex.defaultSortFields =                PID,AUTO,true
fgsindex.untokenizedFields =                fgs.contentModel fgs.ownerId PID
fgsindex.mergFactor =                       10
fgsindex.maxBufferedDocs =                  10

fgsindex.snippetBegin =                     <span class=\"highlight\">
fgsindex.snippetEnd =                       </span>
```

- Beispiel für `config/index/SolrIndex/index.properties`:

```
fgsrepository.repositoryName =                       WissGridRepo
fgsrepository.fedoraSoap =                           http://localhost:8080/fedora/services
fgsrepository.fedoraUser =                           fedoraAdmin
fgsrepository.fedoraPass =                           fedoraAdmin
fgsrepository.fedoraObjectDir =                      /path/to/iRODS/Vault/home/rods/objects
fgsrepository.fedoraVersion =                        3.5

fgsrepository.defaultGetRepositoryInfoResultXslt =   copyXml
fgsrepository.trustStorePath =                       /path/to/fedora/server/truststore
fgsrepository.trustStorePass =                       changeit
```

- Beispiel für `config/rest/BrowseIndexToHtml`:

```
<xsl:variable name="EQCHAR">
    <xsl:choose>
```

```
<xsl:when test="$INDEXNAME = 'ZebraIndex'">=</xsl:when>
...
     <xsl:text> </xsl:text>Index name:
      <select name="indexName">
       <xsl:choose>
        <xsl:when test="$INDEXNAME='SolrIndex'">
         <option value="SolrIndex" selected="true">SolrIndex
          </option>
        </xsl:when>
        <xsl:otherwise>
         <option value="SolrIndex" selected="true">SolrIndex
          </option>
        </xsl:otherwise>
       </xsl:choose>
      </select>
     <xsl:text> </xsl:text>restXslt:
      <select name="restXslt">
       <option value="BrowseIndexToHtml">BrowseIndexToHtml
        </option>
       <option value="copyXml">no transformation</option>
       ...
```

- Beispiel für `config/updater/BasicUpdaters/updater.properties`:

```
java.naming.factory.initial =   org.apache.activemq.jndi.ActiveMQInitialContextFactory
java.naming.provider.url =      tcp://localhost:61616
connection.factory.name =       ConnectionFactory
topic.fedoraAPIM =              fedora.apim.update
client.id =                     gsearch
```

- `schema.xml` muss angepasst werden, damit nach den neuen Indexkategorien indiziert werden kann:

```
<field name="pid" type="string" indexed="true" stored="true" required="true" />
<field name="dsid" type="string" indexed="true" stored="true" required="true" />
<field name="version" type="string" indexed="true" stored="true" required="true" />
<field name="dslocation" type="string" indexed="true" stored="true" required="true" />
```

- Wenn man bereits eine **Apache Solr Server** Instanz installiert hat, kann diese über die MultiCore-Funktion für den Einsatz mit **GSearch** umkonfigurieren. Dafür sind folgende Schritte vorzunehmen:
  - Unter `$SOLR_HOME/cores/<CORENAMEN>/conf` sind Konfigurationsdateien zu erstellen, z.B. `$SOLR_HOME/cores/gsearch/conf`.
  - Alle alten Konfigurationsdateien sind von `$SOLR_HOME/conf` in den neuen Konfigurationsordner zu kopieren.
  - Es ist noch eine Datei `server.xml`, unter `$SOLR_HOME` zu erstellen, z.B.:

```
<?xml version="1.0" encoding="UTF-8" ?>
```

```
<solr presitent=¨true¨>
    <cores adminPath=¨amdin/cores¨>
        <core name=¨GSeach¨ intanceDir=¨cores/gsearch¨>
        <core name=¨iRODS¨ intanceDir=¨cores/irods¨>
    </cores>
</solr>
```

- Letztendlich müssen nochfolgende Dateien überarbeitet werden:
  - `$CATALINA_HOME/webapps/wissgridservices/WEB-INF/wissgrid.properties`. Die Adresse der **SOLR_URL** anpassen.
  - `$CATALINA_HOME/webapps/fedoragsearch/WEB-INF/classes/config/index /SolrIndex/index.properties`. Hier müssen die Adresswerte (URLs) von **fgsindex.indexBase** und **fgsindex.indexDir** angepasst werden.

6.5 Pazpar2 - Einstellungen mit Solr für Debian

- Beispiel für `/etc/pazpar2/server.xml`:

```
<?xml version="1.0" encoding="UTF-8"?>
<pazpar2 xmlns="http://www.indexdata.com/pazpar2/1.0">
     <threads number="50"/>
     <server>
        <listen port="8004"/>
        <include src="services-enabled/*.xml"/>
     </server>
</pazpar2>
```

- Beispiel für `/etc/pazpar2/services-enabled/default.xml`:

```
<timeout session="280" z3950_operation="240" z3950_session="360"/>
...
<icu_chain id="relevance" locale="de">
...
<icu_chain id="sort" locale="de">
...
<icu_chain id="mergekey" locale="de">
...
<icu_chain id="facet" locale="de">
...
```

- Beispiel für `/etc/pazpar2/services-enabled/solr.xml`:

```
...
     <timeout session="210" z3950_operation="180" z3950_session="240"/>
     ...
```

```
<set target="IP-ADRESSE:8080/solr/GSearch/" name="pz:name"
 value="solrGSearch"/>
<set target="IP-ADRESSE:8080/solr/iRODS/" name="pz:name"
 value="solriRODS"/>
...
<set name="pz:sru" value="solr"/>
<set name="pz:sru_version" value="1.4.1"/>
<set name="pz:maxrecs" value="500"/>
...
<set name="pz:cclmap:term" value="s=al t=r"/>
<set name="pz:cclmap:fulltext" value="s=al t=r"/>
<set name="pz:elements" value="*"/>
<set name="pz:nativesyntax" value="xml;utf-8"/>
<set name="pz:xslt" value="solr-pz2.xsl"/>
...
<!- Wissgrid special fields ->
<set name="pid" value="1=pid s=al t=r"/>
<set name="dsid" value="1=dsid s=al t=r"/>
<set name="version" value="1=version s=al t=r"/>
<set name="timestamp" value="1=timestamp s=al t=r"/>
<set name="dslocation" value="1=dslocation s=al t=r"/>
...
<icu_chain id="relevance" locale="de">
...
<icu_chain id="sort" locale="de">
...
<icu_chain id="mergekey" locale="de">
...
<!- wissgrid specified metadata: start->
<metadata name="pid" brief="yes" termlist="yes"/>
<metadata name="dsid"/>
<metadata name="version"/>
<metadata name="timestamp"/>
<metadata name="dslocation"/>
...
```

6.6 Apache HTTP Server - Debian Proxy Einstellungen

- Beipiel für die Konfigurationsdatei des vorinstallierten default Virtual Host:
  ProxyPass `/tomcat/` **http://localhost:8080/**
  ProxyPassReverse `/tomcat/` **http://localhost:8080/**

  ProxyPass `/manager/html` **http://localhost:8080/manager/html**
  ProxyPassReverse `/manager/html` **http://localhost:8080/manager/html**

  ProxyPass `/manager/status` **http://localhost:8080/manager/status**
  ProxyPassReverse `/manager/status` **http://localhost:8080/manager/status**

  ProxyPass `/fedora` **http://localhost:8080/fedora**
  ProxyPassReverse `/fedora` **http://localhost:8080/fedora**

  ProxyPass `/fedora/describe` **http://localhost:8080/fedora/describe**
  ProxyPassReverse `/fedora/describe` **http://localhost:8080/fedora/describe**

  ProxyPass `/fedora/objects` **http://localhost:8080/fedora/objects**
  ProxyPassReverse `/fedora/objects` **http://localhost:8080/fedora/objects**

  ProxyPass `/fedora/services` **http://localhost:8080/fedora/services**
  ProxyPassReverse `/fedora/services` **http://localhost:8080/fedora/services**

  ProxyPass `/fedoragsearch` **http://localhost:8080/fedoragsearch**
  ProxyPassReverse `/fedoragsearch` **http://localhost:8080/fedoragsearch**

  ProxyPass `/fedoragsearch/rest` **http://localhost:8080/fedoragsearch/rest**
  ProxyPassReverse `/fedoragsearch/rest` **http://localhost:8080/fedoragsearch/rest**

  ProxyPass `/fedoragsearch/services` **http://localhost:8080/fedoragsearch/services**
  ProxyPassReverse `/fedoragsearch/services` **http://localhost:8080/fedoragsearch/services**

  ProxyPass `/solr` **http://localhost:8080/solr**
  ProxyPassReverse `/solr` **http://localhost:8080/solr**

  ProxyPass `/solr/GSearch` **http://localhost:8080/solr/GSearch**
  ProxyPassReverse `/solr/GSearch` **http://localhost:8080/solr/GSearch**

  ProxyPass `/solr/iRODS` **http://localhost:8080/solr/iRODS**
  ProxyPassReverse `/solr/iRODS` **http://localhost:8080/solr/iRODS**

  ProxyPass `/oaiprovider` **http://localhost:8080/oaiprovider**
  ProxyPassReverse `/oaiprovider` **http://localhost:8080/oaiprovider**

  ProxyPass `/oreprovider` **http://localhost:8080/oreprovider**

ProxyPassReverse `/oreprovider` **http://localhost:8080/oreprovider**

ProxyPass `/wissgridservices` **http://localhost:8080/wissgridservices**
ProxyPassReverse `/wissgridservices` **http://localhost:8080/wissgridservices**

ProxyPass `/wissgridservices/apim` **http://localhost:8080/wissgridservices/apim**
ProxyPassReverse `/wissgridservices/apim` **http://localhost:8080/wissgridservices/apim**

...

- Für dieses Beispiel müssen noch folgende Konfigurationsdateien angepasst werden:
  - `$CATALINA_HOME/webapps/fedoragsearch/WEB-INF/classes/config/fedoragsearch.properties`
  - `$CATALINA_HOME/webapps/fedoragsearch/WEB-INF/classes/config/index/SolrIndex/index.properties`
  - `$CATALINA_HOME/webapps/fedoragsearch/WEB-INF/classes/config/repository/WissGridRepo/repository.properties`
  - `$CATALINA_HOME/webapps/wissgridservices/WEB-INF/wissgridservices.properties`
  - `$CATALINA_HOME/webapps/oreprovider/WEB-INF/classes/ore.properties`
  - `$CATALINA_HOME/webapps/oaiprovider/WEB-INF/classes/proai.properties`
  - `/etc/pazpar2/services-available/solr.xml`