

# **Global Digital Format Registry**

## **Technology Platform**

Version 0.2

January 4, 2007



# Table of Contents

<b>1. Document Purpose</b>	<b>1</b>
<b>2. System Implementation Narrative</b>	<b>1</b>
<b>3. Technology Platform Model</b>	<b>2</b>
3.1. Model	2
3.2. Description	2
3.3. Alternate Storage Solutions	3
3.3.1. Using a RDBMS backing store	4
3.3.2. Façade an existing service such as PRONOM	4
3.3.3. Design Benefits	5
<b>4. References</b>	<b>6</b>



---

## **1. Document Purpose**

This document contains the technology selection portion of the architecture for the Global Digital Format Registry. It implements the high level requirements described by Abrams [1] and the naming standard proposed by Young [2, 3], and the corresponding analysis model proposed by Stanescu [5].

This document implements the Technology Viewpoint of the RM-ODP model. For more information, see Stanescu [5].

---

## **2. System Implementation Narrative**

The technology platform proposed here clearly underscores the basic requirement of using open source components, capable of independent investigation and reusing existing mature platforms, protocols and services, which minimizes the size of the development effort paid for by this project.

---

## 3. Technology Platform Model

### 3.1.Model

The following diagram identifies the main components needed by the service implementation. Additional components and capabilities are listed below.

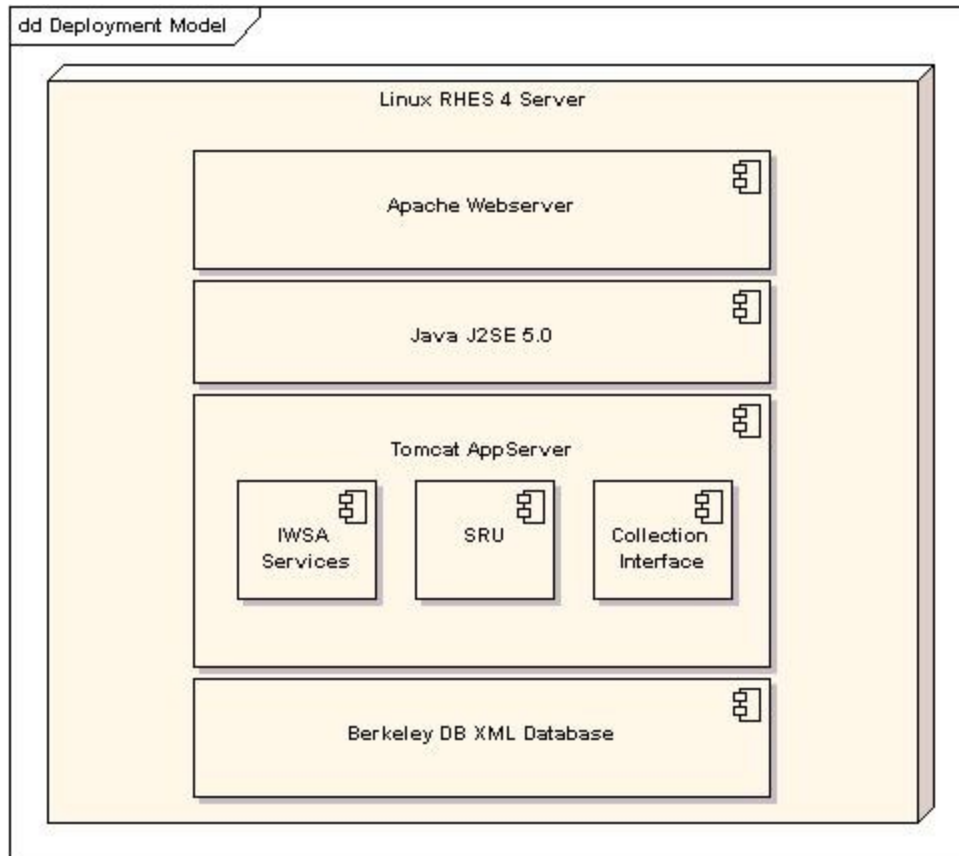


Figure 1: Deployment Model

### 3.2.Description

The preferred deployment environment is a Linux RedHat Enterprise Server 4.0, running the 2.6.9 kernel, either in 32- or 64-bit configuration. Additional Linux platforms running similar configurations will likely be sufficient.

The IWSA service stack uses Apache as its front end, to accomplish two goals:

1. increased security, stability and configurability

2. use the plugin capability to add seamless authentication, authorization and token passing for single signon, as well as independently vary the implementation of different protocols (e.g.: Project Liberty Alliance, Shibboleth) supporting these capabilities

The currently preferred version for Apache is 2.1 [TODO: verify].

The behind Apache, the Tomcat application server will provide the environment necessary to execute IWSA service servlets, XML manipulation via XSL stylesheets, the SRU/SRW capabilities and the connection to the database. The Java VM is the JSDE 5.0 version. The preferred Tomcat version is 5.5.9. Additional components necessary to parse XML, to implement the Collection interface, etc., are documented below:

Component	Description
<b>SRU/SRW</b>	This platform selected Ralph LeVan's SRU/SRW server implementation. Current version is 2.0. The component will be included in the basic installation.
<b>Axis</b>	Webservices platform from the Apache Project
<b>Ant</b>	Built tool
<b>Log4j</b>	Logging
<b>Xalan</b>	XSLT processor
<b>Saxon8</b>	XML processing
<b>CQL to XQuery</b>	Home-grown parser capable of translating CQL statements into valid XPath statements, also using the document schema to properly use field types into comparisons
<b>CQL parser</b>	Mike Taylor's cql-java implementation

Lastly, the database implementation uses a native XML database library implementation from Oracle (formerly SleepyCat), the Berkeley DB XML database. Built on top of the high-performance Berkeley DB (BDB) implementation widely deployed as LDAP backend stores, it is capable of storing large number of XML documents, either in whole or "shredded" into elements, indexing them for efficient retrieval (i.e.: apparent constant time, most likely lg(n) time). Additional capabilities introduced by the database are:

Component	Description
<b>XQuery</b>	XQuery is to XML what SQL is to database tables. Based largely on XPath. Currently supporting the 2.0 version of the spec.
<b>XPath</b>	XPath is a language for finding information in an XML document. XPath is used to navigate through elements and attributes in an XML document. Currently supporting the 1.0 version of the spec.
<b>XML validation</b>	Pathan parser is used to handle XML and validate against the XSD/DTD schema.

### **3.3.Alternate Storage Solutions**

Two types of extensions were envisioned when the design emerged:

3. replace the proposed XML database with a RDBMS
4. delegate to an existing system such as PRONOM [6]

While a native XML database solution fits the data processing paradigm better than other types of databases, there are many reasons why one would choose a relational database as a backing store: for example, an existing database which needs a standard interface to communicate with the networked world.

Furthermore, existing services can be simply adapted into this design so that an existing registry service acquires all the standard IWSA services by following three simple steps. As seen in the design document [7], the backing store is completely hidden behind an interface, therefore an alternate technology could be substituted by implementing three aspects:

1. 6 methods in Collection Interface to create, add, read, update, delete and search collection records
2. 1 method to translate from the standard CQL query to SQL or other native search language
3. mapping the internal storage attribute schema to the standard IWSA record schemas

The sections below illustrate the effort needed to implement these two storage alternatives.

### **3.3.1.Using a RDBMS backing store**

To substitute a RDBMS database, a software developer would have to implement necessary code to “shred” and “reassemble” the XML into the specific database tables and columns. Importantly, however, the implementer does *not* need to validate the XML data because precondition checking is already implemented by the base package. Only the mapping to the proprietary database schema is necessary to be implemented.

The search query must be mapped from CQL to a valid SQL statement matching the data schema. While possible, it is not trivial to generically map the two search languages and, more particularly, it isn't easily extended to arbitrary database schemas. Such a mapping would likely be quite particular to specific technologies (eg: Oracle vs. Sybase vs. MySQL) and particular schemas.

### **3.3.2.Façade an existing service such as PRONOM**

PRONOM provides many of the data elements needed by the GDFR registry. Hence, it should be possible to make it a first-class member of the GDFR network with minimal development. Such a feat would be accomplish either by the method described above, by simply writing software to read and write to the database and plug it directly under the software stack offered by GDFR. Alternatively, the same interface can be implemented to façade existing services, if such services exist, and essentially delegate to a service instead of a database manager.

The same three steps apply. An implementer must be able to read, write and search records by delegating to a service when requested by the GDFR components. Again, only 6 methods must be implemented. Secondly, the query must be mapped from CQL to whatever search language used by

PRONOM services. Thirdly, and last, the records returned from the service or needed by the service must be cross-walked to match the standard IWSA record schemas.

### **3.3.3.Design Benefits**

This simple implementation solution isolates database developers from the needs of the standard IWSA-Reg registry services.

An implementer would only worry about mapping database records to the standard record schemas and without any other development would make a database participate in standard registry exchanges such as OAI harvesting, SRU/SRW processing, SRU/SRW Update processing, Atom/RSS notifications, and UI client processing.

---

## 4. References

- [1] Abrams, Stephen. 2006. [http://cweb.oclc.org/cmsd/projects/GDFR/plans/Proposal\\_PUBLIC.rtf](http://cweb.oclc.org/cmsd/projects/GDFR/plans/Proposal_PUBLIC.rtf).
- [2] Young, Jeff. *Interoperable Web Systems Architecture (ISWA)*. <http://cweb.oclc.org/ArchitectureAndStandards/EnterpriseArchitectureServices/SOA/IWSA.html>. For public users: [https://collaborate.oclc.org/wiki/gdfr/index.php/Drafts:Interoperable\\_Web\\_Systems\\_Architecture\\_%28IWSA%29](https://collaborate.oclc.org/wiki/gdfr/index.php/Drafts:Interoperable_Web_Systems_Architecture_%28IWSA%29).
- [3] Young, Jeff. *Interoperable Web Systems Architecture – Registry Extension (ISWA-Reg)*. [http://cweb.oclc.org/ArchitectureAndStandards/EnterpriseArchitectureServices/SOA/IWSA\\_Registry.html](http://cweb.oclc.org/ArchitectureAndStandards/EnterpriseArchitectureServices/SOA/IWSA_Registry.html). For public users: [https://collaborate.oclc.org/wiki/gdfr/index.php/Drafts:Interoperable\\_Web\\_Systems\\_Architecture\\_-\\_Registry\\_Extension\\_%28IWSA-Reg%29](https://collaborate.oclc.org/wiki/gdfr/index.php/Drafts:Interoperable_Web_Systems_Architecture_-_Registry_Extension_%28IWSA-Reg%29).
- [4] LOCKSS. Lots of Copies Keep Stuff Safe. <http://www.eecs.harvard.edu/~mema/publications/SOSP2003.pdf>.
- [5] Stanescu, Andreas. *GDFR Analysis Model*. <http://cweb.oclc.org/cmsd/projects/GDFR/designs/GDFR-analysis-model.doc>. For public users: <https://collaborate.oclc.org/wiki/gdfr/images/8/8b/GDFR-analysis-model.pdf>.
- [6] PRONOM. *The Technical Registry*. <http://www.nationalarchives.gov.uk/pronom/> .
- [7] Stanescu, Andreas. *GDFR Registry Node System Design*. [http://cweb.oclc.org/cmsd/projects/GDFR/designs/Registry\\_Node\\_System\\_Design.doc](http://cweb.oclc.org/cmsd/projects/GDFR/designs/Registry_Node_System_Design.doc). For public users: [https://collaborate.oclc.org/wiki/gdfr/images/b/ba/Registry\\_Node\\_System\\_Design.pdf](https://collaborate.oclc.org/wiki/gdfr/images/b/ba/Registry_Node_System_Design.pdf).