Global Digital Format Registry Analysis Model

Version 2.0

October 1, 2007

Table of Contents

1. Document Purpose

This is document contains the analysis model for the Global Digital Format Registry. It uses the high level requirements described by Abrams [1] and the naming standard proposed by Young [3, 4].

The GDFR is an extension of a general registry in that it contains registry records encoded in the GDFR data schema and in that it is an instance of a Distributed Registry. Therefore, the use cases are written for the general registry, explicitly pointing out specializations requested by any Distributed Registry and particularly by GDFR (if any).

The theoretical background used by this author is based on the approach described by Jacobson [2]. Every analysis model contains three parts:

a) Use case model

b) Interface descriptions

c) Problem domain model

2. System Narrative

The system implementation of the Global Digital Format Registry proposed in this document attempts to balance data distribution and system management by creating a network of loosely coupled and cooperating registry nodes. The registry nodes are capable of managing more than one collection of records, in this case digital format description records, deployed either locally or in a hosted mode. At the same time, interoperability with existing research workflows and authentication systems was fundamentally important for the system architecture, as well as it was the need to reuse existing components, interfaces and protocols already present in the open source marketplace.

Registry nodes are said to be software deployed on network accessible machines. Nodes manage (or host) collections of metadata or content records, such as digital format records, bibliographic records, photographs or music streams. Access to collections can be authenticated or public and all collections offer a standard list of services at three different levels: registry, collection and item level. The registry software can be deployed in a standalone fashion or configured to augment an existing network of cooperating nodes, by simply deploying the software necessary to acquire, synchronize and maintain integrity of their contents.

When part of a network, registry nodes can act as data entry points, presumably being integrated into some local record creation workflow, or simply as mirrors, capable of storing and searching local copies of the records. This distinction was introduced to lower the barrier of entry for those institutions or individuals wanting to contribute to the permanence of the registry records but did not need the added complexity of a full data entry registry node.

Registry nodes participating in a network are also capable of being configured and having their configuration changed by administrators from other similar nodes. Registry configuration, as well as remote administration, are governed by policy records, some of which being enforced across the network, such as synchronization deadlines, while others being subject of local restrictions, such as authorization and access control.

Collections will have the option of keeping versions of the records or just simply updating the records in place. When a collection is distributed across a network, all the nodes follow the same policy. In the case of GDFR, all the versions of the records will be maintained and searches can be executed against record histories or only against current versions. All collections, including GDFR, will maintain administrative data for each record, including (but not restricted to) the user making the change, the type of change, date and time of the change, and other similar information about the actions performed against the records.

Entering records in the GDFR collection can take two forms: vetted and non-vetted. The vetting workflow is considered outside the scope of the GDFR collection since it is most likely already implemented locally in software and workflows. However, the collection is capable of storing, replicating and searching the vetting activity from such an external source, tightly coupled with the version of record it defines. Once the local vetting process concludes, the record creators will have access to GDFR services to import all the documentation and the final version of the record in the GDFR collection network.

Similarly, non-vetted records can be created via the interfaces provided by the collection, but the designers' expectation is that relatively few institutions will be creating or updating digital format records, relative to the much larger group capable of deploying mirrors, to ensure permanent access to these records.

3. Requirements

3.1. High-level requirements

The following requirements come from the original proposal document [1] and subsequent discussion between the Harvard and OCLC project teams.

3.1.1. Service requirements

The Global Digital Format Registry (GDFR) will provide sustainable distributed services to store, discover, and deliver representation information about digital formats.

The Global Digital Format Registry will provide services for:

- The centrally-organized collection of format representation information
- The distributed storage, discovery, and delivery of that information

The most significant architectural aspect of the GDFR is its distributed nature.

To meet its aims, the project will define a common data model and network protocol by which multiple independent, but cooperating registries can communicate with each other and synchronize their holdings. The set of registries that participate in this shared enterprise is referred to as the "GDFR network."

New format representation information can be introduced into the network by any node in one of two modes: vetted and non-vetted.

The GDFR encompasses four categories of services:

- maintenance services, to add, review, update, and store format representation information
- administrative services, to manage the local administration of a GDFR node
- synchronization services, to manage inter-nodal communication
- end-user services, for discovery and delivery of format representation information to human and automated agents

The behavior under scale of these services in the software reference implementation for a network node will meet or exceed established project metrics.

The human interfaces for the maintenance, administrative, and end-user services will permit language localization.

All human interfaces will conform to W3C accessibility guidelines.

3.1.2. Data requirements

[…] format typing is fundamental to the use, interchange, and preservation of all digital assets.

The Global Digital Format Registry [will be] populated with representation information for a significant number of digital formats in most common contemporary use.

All representation information is tagged in a manner indicating its submitting agent and level of review.

Non-vetted information is immediately propagated through the network without any technical review. Vetted representation information is subject to an editorial process to ensure its authenticity and technical veracity prior to being propagated.

The relationship between vetted and non-vetted representation information with respect to technical review is thus similar to that between the IETF and vendor/personal trees of the Internet Assigned Numbers Authority (IANA) MIME type registry.

The GDFR will provide a persistent and unambiguous means of identifying a registered format and binding that identity to significant descriptive, administrative, and technical information about the format.

At the minimum, there will be provision for retaining the following data:

- Canonical and variant format names (variant names will include, but not be limited to, MIME types, PRONOM identifiers, and LC FDD identifiers)
- Nominal external signature(s) (e.g. file extension(s))
- Unique internal signature(s) (e.g., "magic number")
- Format author, IPR holder, and maintenance agency
- Authoritative specification document(s) , including bibliographic citations, public identifiers (actionable and non-actionable), and actual documents in the public domain or for which explicit permission to copy/distribute has been received from the rights holder
- Ontological classification
- Relationships to other formats (e.g., subtype-of, new-version-of, can-be-encapsulated-by)
- Links to systems, services, and tools that support the format as an input or output
- Format grammars specified in typed formal notations
- Preservation risk assessments specified in typed formal notations

Complete change history will be kept for all of these properties. It will be possible to recover the state of a given format's representation information for any prior point in time.

3.1.3. Process requirements

The veracity attributed to non-vetted information is based solely on the reputation of the submitting agent.

This review process will enlist the participation of international experts in a manner similar to the Internet Engineering Task Force (IETF) Internet Standards process.

The following sections identify requirements discovered during the process of defining use cases.

One [or more] nodes in the network [are] designated as the root nodes, with administrative responsibility to coordinate registration of new top-level nodes.

For vetted information, the relationship between the editorial process and the root nodes is thus similar to that between the Internet Engineering Steering Group (IESG) and IANA in the current MIME type registration process.

Editorial process - representation information submitted to the GDFR under the vetting mode is subject to appropriate technical review prior to being released for propagation on the GDFR network.

3.2. Additional requirements

3.2.1. Requirements for Identifiers

The GDFR identifier namespace MUST be large enough to accommodate millions of records (i.e.: a string of 32 bytes is sufficient).

The identifiers MUST be easy to create and have almost no potential of collisions.

Globally unique identifiers MUST be easily created by multiple nodes in a distributed registry setting, without checking for duplicates.

3.2.2. Versioning of Records

The set of information defined for each conceptual entity in the GDFR data model is referred to as a “record.”

All records MUST be versioned with a timestamp to second granularity.

In general, registry instances MUST be able to configure the number of versions supported, from only one, to a set number, to all versions.

Particularly, GDFR nodes MUST be configured to retain all versions of all the records.

Record locking MUST not be required, because there is very little chance of two users updating the same record at exactly the same.

For update, the most recent version of a record SHOULD be obtained. However, this is done best-effort because there is always the chance the record may have changed between the time it was obtained and the time it was updated.

3.2.3. Synchronization of Nodes

The protocol used to synchronize holdings MUST be impervious to data attacks, including accidental and malicious updates to format records.

The protocol used to synchronize holdings MUST be secure in order to prevent rogue registry nodes from impersonating nodes capable of introducing new and updated records.

The protocol used to synchronize holdings MUST be secure in order to prevent registry nodes from leaking format records without prior consent.

The nodes MUST verify the condition of the holdings and if they are damaged, the nodes MUST synchronize with other nodes to receive the correct version of the damaged records.

Nodes mirroring the registry records MUST ensure records duplication within 1 (one) day of update.

Each record MUST be replicated at most by 20 mirrors and at least by 6 (if so many exist in the network).

3.2.4. Registry Configuration Policies

In general, registries MUST be able to configure how many prior versions of records are kept, subject to a local policy.

In distributed registries (such as GDFR), all nodes MUST follow the same policy (i.e.: GDFR nodes will keep all versions of records).

Access controls for internal users MUST be configurable and subject to a local policy.

Access controls for external users MUST be configurable and subject to a local policy.

Synchronization frequency MUST be configurable and subject to a local policy.

4. Domain Model

A Domain Model represents the concepts of the domain of discourse and creates a common vocabulary to describe the problem solved by the system. The model shows the objects of the domain (concepts, entities), their relationships and their attributes. Typically, objects shown here translate into actual design classes, but most often they are represented by more than one class, either by a component or an entire subsystem.

Below is the diagram of the semantic object space of typical registries:

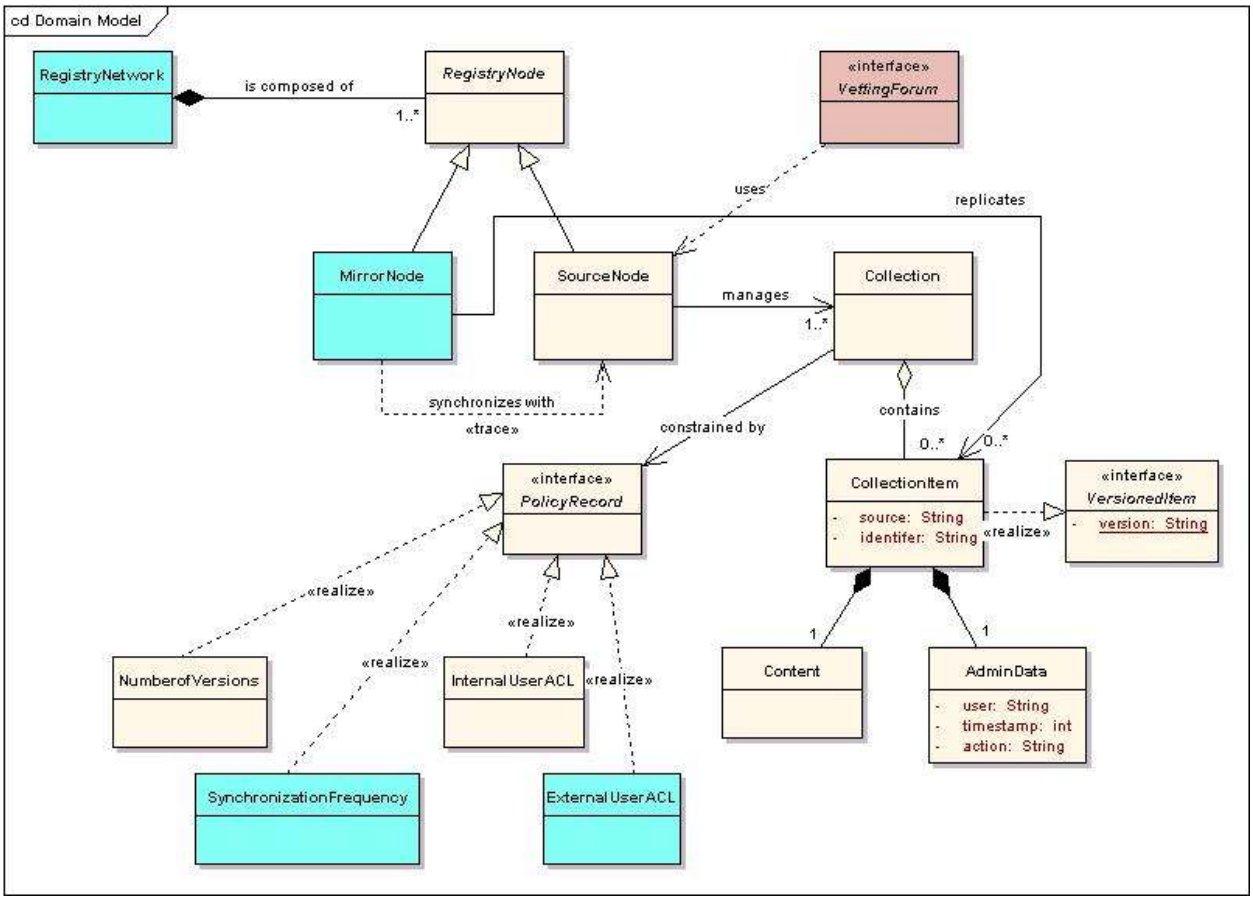

Figure 1: Registry domain model

The neutral-colored symbols represent the object model for a single-instance registry. In this case, a registry node can only be implemented by a source node. A source node is capable of introducing records into the registry (create, add, update, import, export, notify) in addition to the more generic read capabilities (search, read, list-services).

Source nodes manage collections, which contain versioned collection items and are constrained by policy records. Collection items are comprised of two parts, each of which being able to be obtained independently: the content record itself, the admin data related to that version of the record (who updated it, when, why, etc.). Every version of the record can be requested by users and retrieved by the system.

The green symbols represent objects introduced in the model by the distribution requirements. In this case, many registry nodes comprise a registry network. Registry nodes can be implemented by another type, a mirror node, which implements the more generic read-only capabilities. Additionally, a synchronization feature is supported by all registry nodes. Lastly, distributed node management policies such as external users' access control lists and synchronization frequency further constrain the registry nodes managing the collections.

Lastly, a GDFR-specific extension is the vetting forum. The architecture assumes any third party software to be such a discussion and approval forum. Once the vetting community agrees upon a version of a record, it is submitted to a source node, alongside its approval & vetting metadata, and simply enters the system just like any other registry record.

Looking deeper into the data model, the GDFR domain model introduces concepts such as Format, Agent, Process and Event. The set of these domain objects represent the GDFR data dictionary and its definition helps users and implementers alike in discussing their relationships. Specifically, the GDFR domain model is graphically represented as follows:

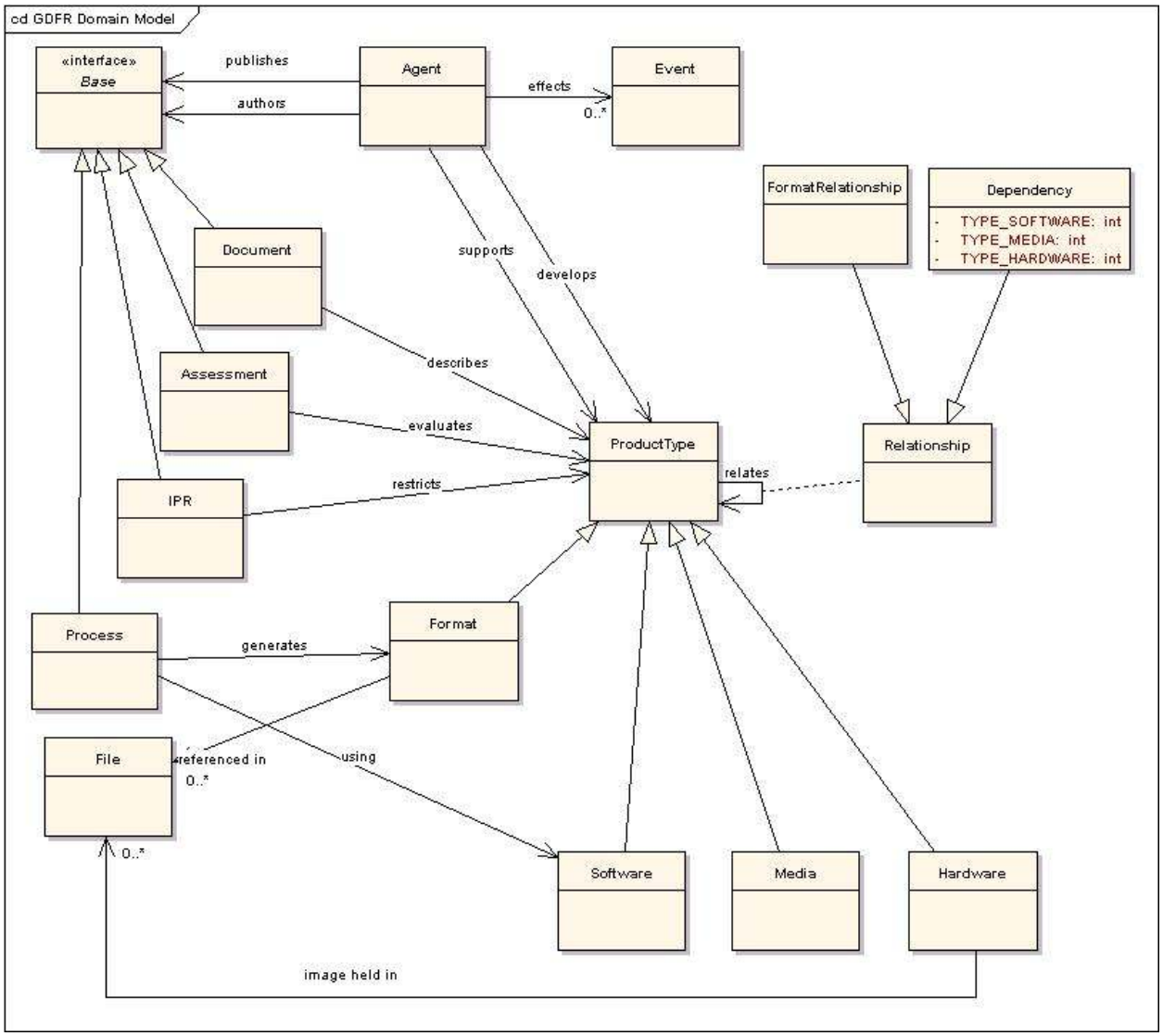

Figure 2: GDFR domain model

To interpret the diagram, one would read the relationships in the following manner:

- `Software` *is_a* `ProductType`
- `Media` *is_a* `ProductType`
- A `Relationship` *relates* one `ProductType` to another `ProductType`
- A `Process` *generates* one or more `Format(s)`

An implementation of this model would combine the individual domain object management (create, add, update, delete, search) with necessary management of the relationships between these objects. The exact details of implementation are documented in the accompanying design document.

5. Use case model

As shown by Jacboson [1]: use case models employ actors and use cases to represent a functional view of the requirements, from a user perspective as well as from the point of view of external systems with which the system described here will interact. The two concepts, actors and use cases, define what is outside the system (actors) and what the system should perform (use cases).

5.1. Actors

a) Registry User – any person or system on the Internet or a local network. Such an actor is allowed to use GDFR end-user services to discover GDFR registry nodes, search through the registry records and retrieve registry records.

b) Registry Node – an instance of the GDFR Registry. This actor is allowed to export its contents to, and import new contents from, other registry nodes, a process known as synchronization. The system must recognize such an actor via some authentication & authorization protocol.

c) Registry Editor – a person recognized by a specific registry node. An extension to the Registry User, this actor is allowed to use GDFR maintenance services to add and edit registry records into a registry node.

d) Registry Administrator – a person recognized by a specific registry node. This actor is allowed to use GDFR administrative services to create and configure registry nodes, in addition to discovering other registry nodes. This actor has no privileges to access data in the registry nodes. Additional restrictions apply (as detailed above).

5.2. Use cases

The following diagram shows the functional view of the proposed system.

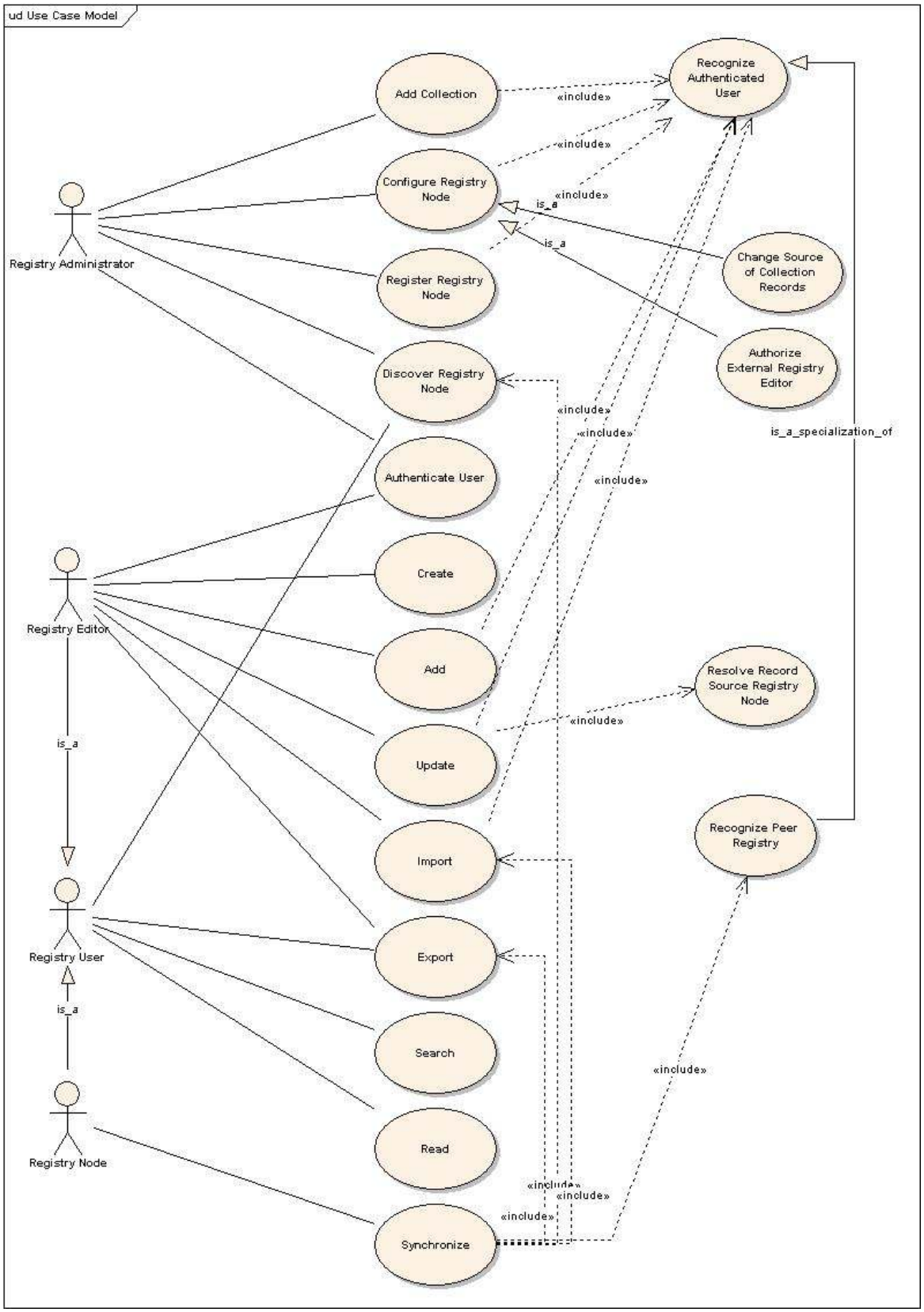

Figure 3: Use Case model

As shown, the following use cases were identified:

- Search Collection Records
- Read Collection Record
- Create Collection Record
- Add Collection Record
- Update Collection Record
- Import Collection Records
- Export Collection Records
- Synchronize Collection Records
- Register Registry Node
- Add Collection
- Discover Registry Node
- Recognize Peer Registry
- Authenticate User
- Recognize Authenticated User
- Configure Registry Node
    - Change Source of Collection Record
    - Authorize External Collection Editor
    - Configure Synchronization Frequency
    - Configure Retained Number of Versions
- Resolve Record Source Registry Node

A Registry User is allowed to discover registries, search for records and retrieve records.

In addition, a Registry Editor is allowed to add, update and export registry records, but only after it authenticates and it is recognized by the registry note on which the functions are requested.

Similarly, in addition to the Registry User capabilities, a Registry Node is capable of importing and exporting registry records, or, combined to synchronize holdings across the nodes. This process is subject to protections to avoid data and services attacks and to protect the integrity of the records. Furthermore, all these requests are permitted after the registry nodes recognize each other.

A Registry Administrator is a special user capable of configuring registry nodes, for example to change the source of a given record history, to let nodes know about other neighbor nodes, to allow Registry Editors from one registry to update records in another, and other similar functions. These functions are executed only after the user is recognized by the local registry node.

5.2.1. Search Registry Records

Use Case ID	Reg-1	
Description	Find records using a known query language and protocol. The result is a list of record abstracts (identifiers of records, short description), not actual records.	
Actors	Registry User	
Assumptions	The client software or the user knows the query language and has some familiarity with the data in the collection	
	The client software or the user knows the URL naming pattern used to invoke the registry services.	
	The user has discovered a registry node and has obtained a URL handle to it.	
	The user is authorized to execute this use case.	
	Distributed Registry Extension	The registry node has obtained a relatively up-to-date copy of collection records originated in all other registry nodes.
Pre-conditions	The query statement is valid.	
Primary functional path	Search local repository indexes of the registry node.	
	Format the result set.	
Primary result	Return a list (possibly empty if no matches) of abstracts of collection records, containing its known identifiers and some unique & relevant identifying attributes.	
Post-conditions	True	
Exceptional path	A system error has occurred.	

5.2.2. Read Collection Record

Use Case ID	Reg-2	
Description	Read one record. There are two distinct outputs: 1) the record itself; 2) the admin data related to the record. The arguments supplied by the request must identify which aspect of the record is requested.	
Actors	Registry User	
Assumptions	The client software or the user has some familiarity with the data in the collection.	
	The client software or the user knows the URL naming pattern used to invoke the registry services.	
	The user has discovered a registry node and has obtained a URL handle to it.	
	The client software or the user has obtained a valid identifier of the format record, presumably by previously invoking the Search use case, or by some other off-line means.	
	The user is authorized to execute this use case.	
	Distributed Registry Extension	The registry node has obtained a relatively up-to-date copy of collection records originated in all other registry nodes.
Pre-conditions	The record identified by the given identifier exists.	
Primary functional path	Read the arguments and determine whether the data or admin portions of the record are to be returned.	
	Retrieve the record from the registry node local repository.	
	Format the record either using a default schema or as requested by the user.	
Primary result	Return the format record.	
Post-conditions	True	
Exceptional path	A system error has occurred.	

5.2.3. Create Collection Record

Use Case ID	Reg-3
Description	Create and return to the user an empty collection record. This is the recommended means to obtain a new, valid identifier for a new record.
Actors	Registry Editor
Assumptions	The client software or the user knows the URL naming pattern used to invoke the registry services.
	The user has discovered a registry node and has obtained a URL handle to it.
	The user is authorized to execute this use case.
Pre-conditions	The user is recognized and is allowed to execute this feature (invoke use case Reg-14).
Primary functional path	Create a valid identifier.
	Format the record either using a default schema or as requested by the user.
Primary result	An empty collection record with a valid identifier.
Post-conditions	True.
Exceptional path	A system error has occurred.

5.2.4. Add Collection Record

Use Case ID	Reg-4	
Description	Add a new record.	
Actors	Registry Editor	
Assumptions	The client software or the user must know the format of the data in the collection.	
	The client software or the user knows the URL naming pattern used to invoke the registry services.	
	The user has discovered a registry node and has obtained a URL handle to it.	
	The client software or the user has obtained a valid identifier of the format record, presumably by previously invoking the Create use case, or by some other off-line means.	
	The user is authorized to execute this use case.	
Pre-conditions	The user is recognized and is allowed to execute this feature (invoke use case Reg-14).	
	The identifier given to the record is valid and does not already exist.	
	The record contains all the required data elements, in the proper format.	
Primary functional path	Retrieve from the system an empty record with a correct identifier (invoke Reg-3).	
	Fill in all the data attributes, as required by the record schema.	
	Submit the add request to the registry node local repository.	
	Add the record to the registry node local repository.	
	Add administrative record data about this record and this action to the corresponding admin collection.	
	Distributed Registry Extension	Each Mirror Registry must configure itself to monitor and synchronize changes for this particular record (invoke Reg-8).
Primary result	void	
Post-conditions	Record is added to the local repository.	
	Attributes are indexed.	

Exceptional path	The registry node becomes the Source Registry of the record in question.
	A system error has occurred.

There is a semantic limitation built into this use case. If a record for, let's say PDF 1.4 already exists, the system cannot properly detect the duplication, if a new record is added. We should consider a number of attribute validation rules to attempt to detect the duplication, but ultimately, it's the Registry Editor's responsibility to maintain unique format records in the collection.

5.2.5. Update Collection Record

Use Case ID	Reg-5	
Description	Update an existing record.	
Actors	Registry Editor	
Assumptions	The client software or the user must know the format of the data in the collection.	
	The client software or the user knows the URL naming pattern used to invoke the registry services.	
	The user has discovered a registry node and has obtained a URL handle to it.	
	The client software or the user has obtained a recent version of the format record, presumably by previously invoking the Read use case, or by some other off-line means.	
	The user is authorized to execute this use case.	
Pre-conditions	The user is recognized and is allowed to execute this feature (invoke use case Reg-14).	
	The identifier given to the record is valid and already exists.	
	The record contains all the required data elements, in the proper format.	
Primary functional path	Find the record either by searching (invoke Reg-1) or by resolving a previously saved identifier.	
	Read the previous version of the record (invoke Reg-2).	
	Fill in all the data attributes, as required by the record schema.	
	Submit the update request to the registry node local repository.	
	The system reads the previous version of the record.	
	Preserve read-only attributes (i.e.: existing identifiers, pointer to Source Registry, etc.) and make all other user-requested updates to the record. Future extension – to resolve possible update sequence issues, the system can attempt to merge the updates with the most recent version. There could be issues with conflicting updates that must resolved, however.	
	Distributed Registry Extension	Identify the Source Registry by searching the local database for the record, retrieve the public ID and find the registry node base URL in the mirror collection.
		If the record originated in a different

		node, authenticate to that node.
		Once authenticated, proxy the update request to the Source Node.
		Add a new version of the record in the Source Registry.
		If the record is local, add administrative record data about this record and this action to the corresponding admin metadata.
	Single-node Registry Extension	Add a new version of the record in the registry node local repository.
		Add administrative record data about this record and this action to the corresponding admin collection.
		Clear prior versions of the record as requested by the local policy.
Primary result	void	
Post-conditions	Record is correctly updated in the Source Node repository.	
	Attributes are indexed.	
	The Mirror Nodes, including the local repository, are informed of the update and go about synchronizing records (invoke Reg-8).	
Exceptional path	The Source Registry is not available (save the record locally in a “holding pen”, the attempt the update later).	

There are two main reasons for the proposed processing path:

- accuracy, in order to prevent garbage data being introduced into the network
- inherent lag time in distributing the updates to all the nodes in the network

If the record is updated locally, it becomes very easy to introduce versions of the record that are invalid or have incompatible updates, either from malicious intent or simply operator error. Additionally, a rogue node can make the system thrash by making many, many updates, each of which requiring distribution of updates to peers, essentially creating a denial-of-service attack.

If, instead, there is only one source of the record, then the nodes can agree on the sequence of the updates without using locks. Furthermore, the history of the record is kept in only one place and then duplicated across the network for persistence.

The problem then shifts to:

- changing the source of any given record
- allowing external users to make updates locally to a registry node

In both cases, these are problems of network configuration, resolved by the Configure Registry use case below, Reg-11, and its extensions. As noted in the requirements, the policy to allow such changes are matters of local choices, therefore subject to limitations.

5.2.6. Import Collection Records

Use Case ID	Reg-6
Description	Import new records, presumably held in other data sources or registries. It is a batch execution of multiple add or updates. By default, this use case is the recipient of records generated by the Export use case (Reg-7).
Actors	Registry Editor, Registry Node
Assumptions	The client software or the user must know the format of the data in the collection.
	The client software or the user has obtained collection records, presumably from a different data source, and has converted them to the format required by this system.
	The client software knows the URL naming pattern used to invoke the registry services.
	The user has discovered a registry node and has obtained a URL handle to it.
	The user is authorized to execute this use case.
Pre-conditions	The records contain all the required data elements, in the proper format.
	The records may contain identifiers. If so, they may NOT be sourced by the local registry instance.
	The records contain provenance information, describing the source of the record.
Primary functional path	For each record in the input argument list:
	If new record, add an identifier to the record by using Create Collection Record use case (Reg-3). Else, update the current version of the locally held record with the new data.
	Add or updates records in the local database as necessary.
Primary result	void
Post-conditions	Records are now added or updated in the local repository.
	Attributes are indexed.
Exceptional path	A system error has occurred.

See the notes on semantic limitations described by the Add Collection Record use case.

5.2.7. Export Collection Records

Use Case ID	Reg-7
Description	Export records using a known query language and protocol. The return set may contain earlier versions of the records, depending on the query sent to the service.
Actors	Registry User
Assumptions	The client software or the user knows the query language and has some familiarity with the data in the collection.
	The client software or the user knows the URL naming pattern used to invoke the registry services.
	The user has discovered a registry node and has obtained a URL handle to it.
	The user is authorized to execute this use case.
Pre-conditions	The query statement is valid.
Primary functional path	Search local repository indexes of the registry node (invoke use case Reg-1).
	For each abstract in the result set, read the record.
	For each record, read only the current version and corresponding admin data
	Format the result set.
Primary result	Return a list (possibly empty if no matches) of collection records and associated admin data.
Post-conditions	True
Exceptional path	A system error has occurred.

5.2.8. Synchronize Collection Records TODO: finish this!

Use Case ID	Reg-8
Description	Distributed Registry Extension. This use ensures that nodes have most recent copies of a particular record, as published by the Source Registry. This way, new registry nodes become part of the network in a trustworthy manner. Additionally, a secondary path of this use case ensures that collection records are not damaged and if they are, they get corrected.
Actors	Registry Node
Triggers	A configured timer, given the correct Synchronization Policy.
Assumptions	The Registry Node software must know the format of the data in the collection.
	The Registry Node software knows the URL naming pattern used to invoke the registry services.
	The Registry Node has discovered the Source Registry for updates for a particular record.
	The Registry Node understands the data verification & correction algorithm.
	The user is authorized to execute this use case.
Pre-conditions	The Registry Administrator has configured the Registry Node to synchronize with Source Registries for specific collection records.
	The Registry Administrator has configured the Registry Node to verify holdings with other Mirror Nodes for specific registry records.
Primary functional path	Exchange credentials with the Source Registry to be properly recognized as a user.
	Obtain the history of the record since the last request. (invoke Reg-9).
	Add the new versions of the record and admin data records to the local repository.
Secondary functional path	Verify that the records held in local repository have not been damaged by validating the content with the configured set of peer Mirror Nodes.
Primary result	void
Post-conditions	Records are now duplicated in the local repository.
	Attributes are indexed.
Exceptional path	A system error has occurred.
Secondary	The verification algorithm cannot overwhelmingly determine which version is

exceptional path	correct, if 2 or more versions of the same record are found in cooperating Mirror Nodes. Registry Administrators must then correct the situation.

5.2.9. Register Registry Node

Use Case ID	Reg-9
Description	Distributed Registry Extension. This action identifies a new registry node into the network. Once registered, nodes automatically recognize each other recognize and authorize users of each other to seamlessly update records held in peer nodes and transfer updates from one another using synchronization. The registration is held in a special collection and only the local administrator is allowed to add records to it.
Actors	Registry Administrator
Triggers	A new node requests to be added to the network.
Assumptions	The registry administrator managing the network registration has a process to verify the accuracy of the request.
	The requester has generated a PGP public and private key.
	The user is authorized to execute this use case.
Pre-conditions	The Registry Administrator has determined, working with the requestor, a unique generic name for the new node, which will become the high-order string of all identifiers generated by the new node.
	The Registry Administrator has received the PGP public key for the node from the requestor.
	The Registry Administrator has received the base URL for the node from the requestor.
Primary functional path	Registry Administrator creates a new record in the network registration collection for the new node.
	The new node is registered with DNS, to be used by a well-known internet domain round-robin resolution.
Primary result	void
Post-conditions	A new record is added to the network registration collection.
Exceptional path	A system error has occurred.

5.2.10. Add Collection

Use Case ID	Reg-10	
Description	Add a new collection.	
Actors	Registry Editor	
Assumptions	The client software or the user must know the format of the data in the Collections collection and provide the format of the data in the collection to be created.	
	The client software or the user knows the URL naming pattern used to invoke the registry services.	
	The user has discovered a registry node and has obtained a URL handle to it.	
	The client software or the user has obtained a valid identifier for the collection, presumably by previously invoking the Create Collection use case, or by some other off-line means.	
	The user is authorized to execute this use case.	
Pre-conditions	The user is recognized and is allowed to execute this feature (invoke use case Reg-14).	
	The identifier given to the collection profile is valid and does not already exist.	
	The collection profile contains all the required data elements, in the proper format.	
Primary functional path	Retrieve from the system an empty record with a correct identifier (invoke Reg-3).	
	Fill in all the data attributes, as required by the collection profile schema.	
	Submit the add request to the registry node local repository.	
	Create a new database to hold the collection records, add necessary indexes, policy records and data restrictions.	
	Add the profile record to the Collections collection in the local repository.	
	Add administrative record data about this profile record and this action to the corresponding admin collection.	
	Distributed Registry Extension	Notify known registry nodes of the newly added collection.
		Each Mirror Registry must configure itself to monitor and synchronize changes for particular records from this collection (invoke Reg-8).
Primary result	void	

Post-conditions	Record is added to the local repository.
	A new database is created in the local repository.
	Indexes are created for the attributes specified in the collection profile.
	The registry node can now be assigned as Source Registry for records in this collection.
Exceptional path	A system error has occurred.

5.2.11. Discover Registry Node

Use Case ID	Reg-11
Description	Discover registry nodes.
Actors	Anyone
Assumptions	The registry collection exposes an HTML interface, which is widely available through all major search engines.
	The user has access to an Internet search engine.
Pre-conditions	None
Primary functional path	User searches for GDFR registry nodes.
	One of the node's public HTML website is found.
	The user selects the one to use.
	Alternatively, users can be directed to the local node by some other means (links from portals, well-known local websites, etc.)
Secondary functional path	User requests a well-known internet domain.
	DNS returns one of the known nodes.
	The user connects to the one returned.
Primary result	void
Post-conditions	True
Exceptional path	A system error has occurred.

Two aspects still to be defined is how to relate all the GDFR registry nodes and how does a new node become part of the network. Ideally, such as process would not create a unique "root" node, through which all registry registration requests go through. The rest of the model clearly eliminates such a single point of failure and the same concept should be applied here.

One proposal would create a specialized collection, holding registration records for all other GDFR registry nodes. In particular, this would be necessary in order to distribute policy records to all other registry nodes to recognize future requests from a peer node. The problem is that when following the model of a Source Node, it creates the dreaded "root" node.

The LOCKSS model defines the mode through which a node becomes a peer node. The defining feature of this process is that each node must spend resources to become and remain a peer node (to avoid free-loading) and to avoid detailed schemes of peer authentication via private keys. Such a mode should be investigated further.

5.2.12. Recognize Peer Registry

Use Case ID	Reg-12
Description	Distributed Registry Extension Recognize any given request by a node in the GDFR network.
Actors	any
Assumptions	The registry node has been previously configured as a peer node and authorization has been granted via policy records.
	The token presented is valid, not expired and purportedly held by a Registry Node.
Pre-conditions	Registry node has been previously authenticated.
Primary functional path	Verify presented token.
Primary result	The caller has been recognized as a Registry Node.
Post-conditions	True
Exceptional path	The token is found to be expired or invalid. The caller is redirected to Reg-13.
Exceptional path 2	A system error has occurred.

This use case is a specialization of Reg-14, where the caller purports to be a Registry Node.

5.2.13. Authenticate User

Use Case ID	Reg-13	
Description	Verify credentials passed by a user. Credentials can take many forms including: userid and password for local users, SAML assertions signed by PGP keys for remote single signon, session tokens generated upon successful verification of other credentials. Typically, a simpler, with a shorter life span, but still secure token is returned to the user. This mechanism ensures that regular verification is fast, since authentication can be quite slow for a high-transaction system. Additionally, users can be authenticated in one location but recognized by many others, as it is required by a distributed registry.	
Actors	Any	
Assumptions	The user has been previously configured in an authentication system.	
	The credentials presented are valid, not expired and purportedly of a user.	
Pre-conditions	User has obtained credentials by some other means – userid/password registration, digital certificates, biometrics, etc.	
	Distributed Registry Extension	A registry node is capable of sending a SAML assertion on behalf of the user to another node.
Primary functional path	Verify presented credentials.	
	Generate a SAML assertion for the current user and sign it with the node private PGP key.	
	Distributed Registry Extension	The Registry Node's PGP public key is obtained from the registry node registration collection.
		The given SAML assertion signature is verified using the PGP public key of the registry node.
		The SAML assertion is accepted.
Primary result	User has been authenticated and a SAML assertion has been generated and signed or accepted as such, and stored.	
Post-conditions	True	
Exceptional path	The credentials are found to be expired or invalid. No further recourse exists.	
Exceptional path 2	A system error has occurred.	

5.2.14. Recognize Authenticated User

Use Case ID	Reg-14	
Description	Recognize and authorize the initiator of any given request.	
Actors	Any	
Assumptions	The user has been previously configured and authorization has been granted via policy records.	
	The token presented is valid, not expired and purportedly of a user.	
Pre-conditions	User has been previously authenticated.	
	Distributed Registry Extension	Role assignment for external users has been previously configured when the Node was registered (Reg-15.2)
Primary functional path	Verify presented token. Verify user is authorized to initiate the requested action.	
Primary result	User has been recognized.	
Post-conditions	True	
Exceptional path	The token is found to be expired or invalid. The caller is redirected to Reg-13.	
Exceptional path 2	A system error has occurred.	

5.2.15. Configure Registry Node

This abstract use case is a catch-all to a number of configuration parameters available to each registry node.

5.2.15.1. Change Source of Collection Records

Use Case ID	Reg-15.1	
Description	This use case is a particular implementation of Reg-15, where the policy record being created and exchanged is of type Record Source. In this case, the policy is stored in the registry nodes registration collection.	
Actors	Registry Administrator	
Assumptions	The user has been previously configured and authorization has been granted via policy records.	
	The user is authorized to execute this use case.	
Pre-conditions		
	Distributed Registry Extension	The registry nodes registration collection is replicated across all notes in the network.
Primary functional path	Find the registration record and replace the base URL with the new URL.	
Primary result	The collection record is changed.	
Post-conditions	From this point forward, all records originated from the original source can be updated in the new node. Furthermore, no new records could contain the original source identifier.	
Exceptional path	A system error has occurred.	

5.2.15.2. Authorize External Registry Editor

Use Case ID	Reg-15.2
Description	Distributed Registry Extension This use case is a particular implementation of Reg-15, where the policy record being created and exchanged is of type Authorize Registry Editor. In this case, the policy is stored in the access control role assignment policy collection. This policy selects the role name in the SAML authentication assertion, which must be Registry Editor and assigns its user the same role locally.
Actors	Registry Administrator
Assumptions	The user has been previously configured and authorization has been granted via policy records.
	All GDFR nodes have exactly 4 roles: Registry User (mapped to Anonymous), Registry Administrator, Registry Editor and Registry Node. The assertion must contain the string Registry Editor. The only role that is mapped across nodes is Registry Editor.
	The user is authorized to execute this use case.
Pre-conditions	Reg-9 (register registry node) has been previously executed.
Primary functional path	(For GDFR, there is nothing to do. Because the roles are exactly the same in each node, the authorization is done automatically by the node registration use case Reg-9.)
Primary result	True
Post-conditions	The external users with the role Registry Editor are allowed to update local records.
Exceptional path	A system error has occurred.

5.2.15.3. Configure Synchronization Frequency

Use Case ID	Reg-15.3
Description	Distributed Registry Extension This use case is a particular implementation of Reg-15, where the policy record being created and exchanged is of type Synchronization Frequency.
Actors	None
Assumptions	This policy is implemented as a configuration property and it is identical across all GDFR nodes. No update is possible.
	The user is authorized to execute this use case.
Pre-conditions	None
Primary functional path	None.
Primary result	True
Post-conditions	The registry node knows how often to synchronize records from other collections.
Exceptional path	A system error has occurred.

5.2.16. Resolve Record Source Registry Node

Use Case ID	Reg-16
Description	Given an identifier, determine the URL of the registry where the record has originally been created.
Actors	Anyone
Assumptions	Reg-9 has been executed.
Pre-conditions	
Primary functional path	Find the record in the registry node registration collection which matches the base of the given identifier.
Primary result	The proper registration record has been located.
Post-conditions	True
Exceptional path	A system error has occurred.

6. Component Architecture

The proposed architecture allows clear separation of concerns between registry nodes, the infrastructure replicating the records, and the security infrastructure necessary to make the registry network function coherently. This architecture has a number of benefits, not found in other peer-to-peer networks:

- Allows the project to build components as needed, while having in place a fully functional system
- Allows the project to substitute registry implementations at will
- Allows the hosting institutions to choose their level of participation in the network authentication scheme, while still using local systems for authenticating their individuals
- Allows the project to build a secure replication infrastructure based on proven models such as LOCKSS [5].
- Allows the project to add a common authentication “blanket” over the participating nodes, to let external administrators to update records and configure local registries, based on locally defined policies.

6.1. Distribution model

Graphically, a network of registries, collaborating for redundancy and integrity, can be described as follows:

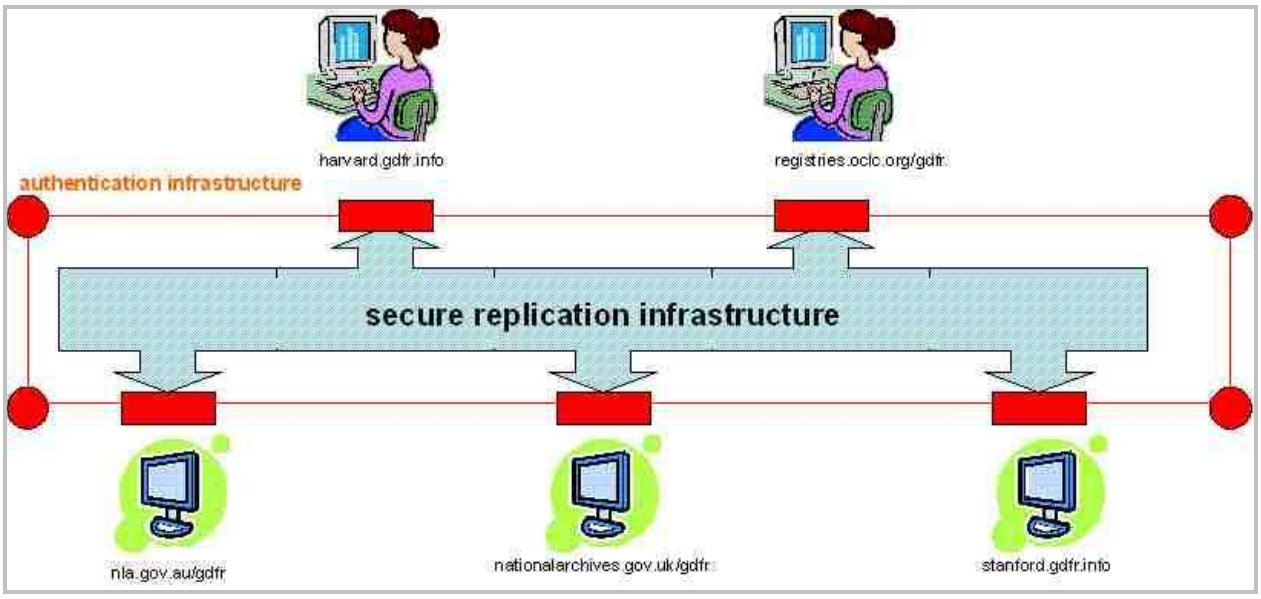

Figure 4: Distribution mode diagram

In this example, there are two Source Registries (harvard.gdfr.info and registries.oclc.org/gdfr) and three Mirror Nodes: nla.gov.au/gdfr, nationalarchives.gov.uk/gdfr and stanford.gdfr.info).

This deployment shows that while the domain gdfr.info is reserved for collaborating registries, this model does not require that all nodes be part of the same domain. Furthermore, a hosted registry system like the one planned by OCLC may contain more than one registry, one of which, the gdfr node, being part of the replication network.

The common authentication “blanket” will be designed to federate existing authentication systems, while being able to exchange role and permission information between the nodes, necessary to perform some of the use cases described earlier.

In more detail, the Source Registries and Mirror Nodes will cooperate as follows:

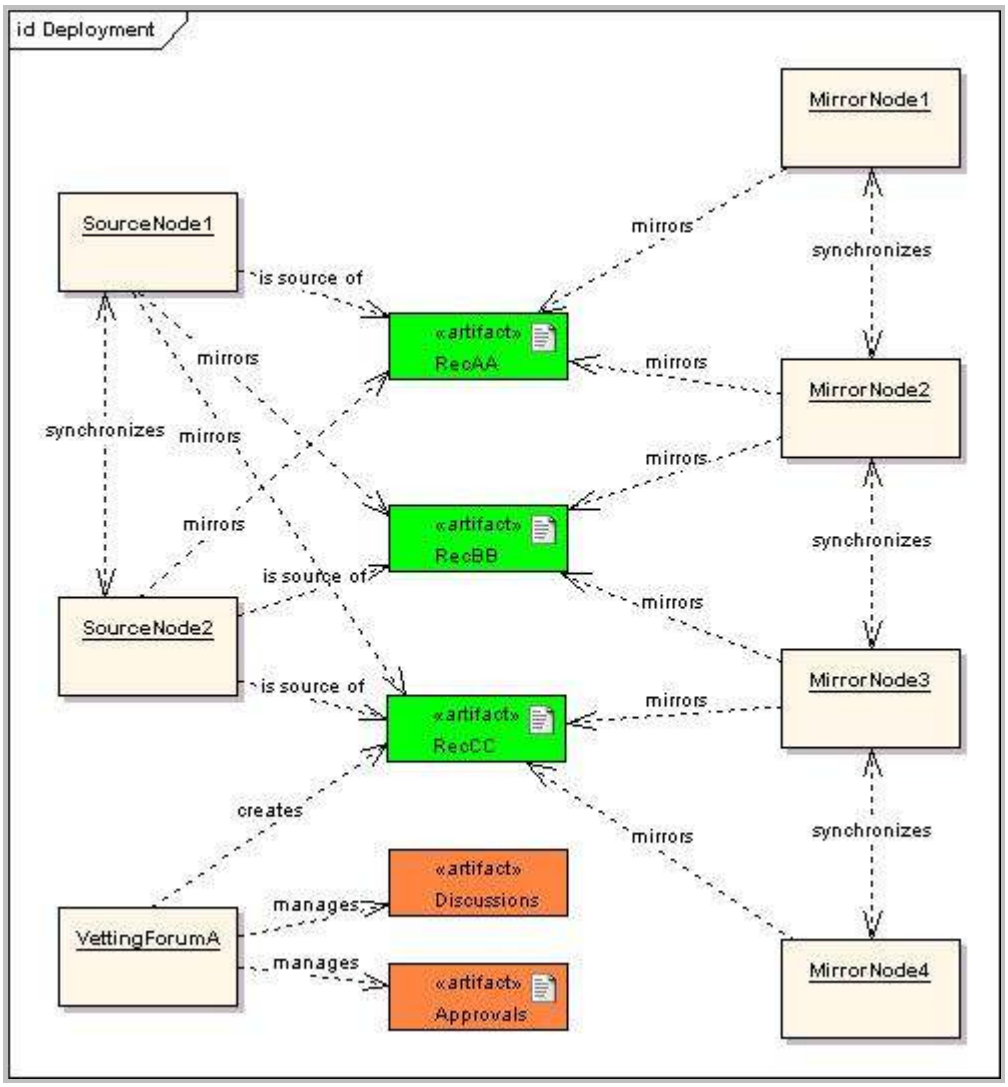

Figure 5: Example deployment model

Assuming that:

- SourceNode1 publishes the registry record identified as RecAA
- SourceNode2 publishes the registry record identified as RecBB,
- Vetting ForumA captures a number of discussions, approvals and early drafts of a record, and ultimately uses SourceNode2 (either hosted locally, or using an existing Source Registry) to publish the registry record identified as RecCC

… then:

- both Source Registries, SourceNode1 and SourceNode2 mirror each others records, in their entirety
    - SourceNode1 will have the master copy of RecAA and backup copies of RecBB and RecCC

    - SourceNode1 will have the master copy of RecBB and RecCC and backup copies of RecAA
- the vetting forum can use whatever software they choose to hold the preliminary discussions, records and drafts, and then decide to publish the final version of the registry record, along with a transcript of all the vetting metadata
- Mirror Nodes will choose which records they backup; in this example:
    - MirrorNode1 has backup copies of RecAA
    - MirrorNode2 has backup copies of RecAA and RecBB
    - MirrorNode3 has backup copies of RecBB and RecCC
    - MirrorNode4 has backup copies of RecCC
    - MirrorNode1, MirrorNode2, SourceNode1 and SourceNode2 collaborate to ensure accuracy of RecAA
    - MirrorNode2, MirrorNode3, SourceNode1 and SourceNode2 collaborate to ensure accuracy of RecBB
    - MirrorNode3, MirrorNode4, SourceNode1 and SourceNode2 collaborate to ensure accuracy of RecCC

6.2. Distribution components

There are a number of fundamental requirements implemented by this architecture:

- Nodes should operate on their own, without the need of the network, for most operations
- A search executed against one node should return exactly the same thing as if executed against a mirror node, including record history, identifiers, owner, version IDs, etc.
- Collections of identical schemas should collect mirrored data and merge it into the local store
- Update requests are only processed through the initial source node, then replicated across the network
- Mirrors should not be used to replicate records, only source nodes. While not enforced by the harvester service, this is implemented as a system policy.

7. Service interfaces

In general, as the use case definition illustrated, registries offer the following standard services[1] [6]:

Business function	Description	Candidate Access Method
Discover	Find relevant registry and access instructions.	UI; human readable url; OpenURL
Search	Returns registry entries based on query	UI; SRU/SRW; Opensearch
Lookup (resolve)	Returns registry entries based on one high order identifier	UI; human readable url; OpenURL
Export	Transfer all or part of registry entries to specified destination	FTP, OAI or browser function, encoded in XML
Import	Transfer registry entries or data elements into registry	FTP, OAI or browser function, encoded in XML
Update	Create , delete or change registry entities	UI; SRU Update; batch script
Subscribe/notify	Request notification, updates or data from registry on a periodic basis	UI; RSS, Atom

As shown by Young [3,4], registry nodes are comprised of and manage different collections, and so standard capabilities can be abstracted for all collections, including one such as the format records of GDFR.

The services supported by the GDFR collection are listed below. There are three categories of services: registry level, collection level and item level.

7.1. Registry level services

Service	URI	Description	Access Method
List available services	**info:rfa/IWSA/svc_id/listServices**	Return a list of Service descriptions available for this Registry (the list shown here).	XML response over HTTP; UI

[1] The acronyms used are: FTP – file transfer protocol, OAI – Open Archives Initiative, REST - Representational State Transfer, RSS – Really Simple Syndication, SOAP – Simple Object Access Protocol, UI – user interface, SRU – Search Retrieve using URL, SRW – Search Retrieve Web Service (SOAP), ZING – SRU related update.

Display record	**info:rfa/IWSA/svc_id/display**	Get a UI view of info:iwsa/localhost/Collections/Collections.	HTML over HTTP
Get record	**info:rfa/IWSA/svc_id/content**	Get the Content of info:iwsa/localhost/Collections/Collections.	XML over HTTP
Get admin data for record	**info:rfa/IWSA/svc_id/adminData**	Get the AdminData of info:iwsa/localhost/Collections/Collections.	XML over HTTP

7.2. Collection level services

Service	URI	Description	Access Method
List services	**info:rfa/IWSA/svc_id/listServices**	Return a list of Service descriptions available for this Collection (the list shown here).	XML response over HTTP; UI
Display record	**info:rfa/IWSA/svc_id/display**	Get a UI view of info:iwsa/localhost/Collections/gdfr.	HTML over HTTP
Get record	**info:rfa/IWSA/svc_id/content**	Get the Content of info:iwsa/localhost/Collections/gdfr.	XML over HTTP
Get admin data for record	**info:rfa/IWSA/svc_id/adminData**	Get the AdminData of info:iwsa/localhost/Collections/gdfr.	XML over HTTP
Search	**info:rfa/IWSA/svc_id/sru**	SRU BaseURL (search within a Collection)	XML over HTTP
Process update record	**info:rfa/IWSA/svc_id/formProcessor**	Process an HTML form (w/submit, preview, delete, cancel)	HTML over HTTP
Update record	**info:rfa/IWSA/svc_id/update**	SRU Update BaseURL	XML over HTTP
Create record	**info:rfa/IWSA/svc_id/create**	Create an item (w/edit)	HTML over HTTP
Import records	**info:rfa/IWSA/svc_id/import**	Import records (w/update)	XML over HTTP
Export records	**info:rfa/IWSA/svc_id/oai**	OAI BaseURL	XML over HTTP
Notify	**info:rfa/IWSA/svc_id/rss**	RSS feed (related to GetUIView)	XML over HTTP

Notify	**info:rfa/IWSA/svc_id/atom**	Atom feed (related to GetUIView)	XML over HTTP
Synchronize	**info:rfa/IWSA/svc_id/lockss**	LOCKSS synchronization	Binary
Authenticate	**info:rfa/IWSA/svc_id/authenticate**	Authenticates the {Credentials} (userID/password, certificates). If authentication fails, it sends the user to a configured location where they can try again. If authentication is successful, it invokes an authorization method interface and then sends that result alongside the userID to another interface where it can be recorded in a session mechanism. Lastly, it invokes a "next" method interface to process the "next" parameter (descriptor).	XML over HTTP, HTML over HTTP

7.3. Item level services

Service	URI	Description	Access Method
List services	**info:rfa/IWSA/svc_id/listServices**	Return a list of Service descriptions available for this item (the list shown here).	XML response over HTTP; UI
Display record	**info:rfa/IWSA/svc_id/display**	Get a UI view of info:iwsa/localhost/gdfr/{Item}.	HTML over HTTP
Get record	**info:rfa/IWSA/svc_id/content**	Get the Content of info:iwsa/localhost/gdfr/{Item}.	XML over HTTP
Get admin data for record	**info:rfa/IWSA/svc_id/adminData**	Get the AdminData of info:iwsa/localhost/gdfr/{Item}.	XML over HTTP
Edit record	**info:rfa/IWSA/svc_id/edit**	Present Content in an HTML FORM for editing	HTML over HTTP
Crossswalk record	**info:rfa/IWSA/svc_id/crosswalk**	Transform Content into an alternate format	XML over HTTP
Get record history	**info:rfa/IWSA/svc_id/history**	Display a history of Content changes using AdminData.	XML over HTTP
Compare records	**info:rfa/IWSA/svc_id/diff**	Display a diff between two historical versions of the Content using AdminData.	XML over HTTP

8. Data schemas

The standard services listed above take advantage of existing protocols and schemas already developed and deployed. The schemas employed by this design are:

- SRU/SRW [7]
- HTML [8]
- SRU Record Update [9]
- LOCKSS [5]
- GDFR record schema [10, 11]
- Shibboleth [12]
- SAML [13]
- XCAML [14]
- WSDL [15]

9. References

[1] Abrams, Stephen. 2006. http://cweb.oclc.org/cmsd/projects/GDFR/plans/Proposal_PUBLIC.rtf.

[2] Jacobson, Ivar, Christerson M., Jonsson P., Overgaard G. *Object Oriented Software Engineering.* ACM Press, Addison Wesley, Harlow, Englad. 1996.

[3] 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.

[4] Young, Jeff. *Interoperable Web Systems Architecture – Registry Extension (ISWA-Reg).* 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.

[5] LOCKSS. Lots of Copies Keep Stuff Safe. http://www.eecs.harvard.edu/~mema/publications/SOSP2003.pdf.

[6] Hamparian, Don. *Registry Architecture.* Presentation to SAT Meeting on Sept. 6, 2006. http://cweb.oclc.org/ArchitectureAndStandards/EnterpriseArchitectureServices/SOA/SAT_Registry_Architecture.ppt. OCLC will publish a version for public consumption shortly.

[7] SRU (Search/Retrieve via URL). http://www.loc.gov/standards/sru/

[8] HTML 4.01 Specification. http://www.w3.org/TR/REC-html40/

[9] SRU Record Update. http://srw.cheshire3.org/docs/update/

[10] GDFR record schema. https://collaborate.oclc.org/wiki/gdfr/index.php/Draft:Data_Model

[11] GDFR identifiers schema. https://collaborate.oclc.org/wiki/gdfr/index.php/Draft:Identifiers

[12] Shibboleth. http://shibboleth.internet2.edu/

[13] SAML (Security Assertion Markup Language). http://www.oasis-open.org/committees/security/

[14] XCAML (eXtensible Access Control Markup Language). http://www.oasis-open.org/committees/xacml/

[15] WSDL (Web Service Definition Language). http://www.w3.org/TR/wsdl