

RIM-Lite Documentation

Last update: Dec 15,2006

RIM-Lite Executive Summary

Recent evolution of the ADL-Registry (ADL-R) project and its growing community of users has resulted in an opportunity to energize and enable third party developers who wish to automate interactions between the Registry and various content repositories and authoring tools. The development and deployment of a standardized and extensible set of communication services, that would also meet the DTIC hosting requirements, would enable these third party developers to automate these functions in the short term for the operational ADL-R. This set of services, which can be thought of as convenient ways for developers to automate interaction with ADL-R, is named RIM (Registry Interface Mechanism), and the initial version, primarily designed to run at DTIC, will be named RIM-Lite. It will incorporate what is needed for the DTIC environment; however, generalized use for copies of the ADL-R operated in other environments will require further refinements.

The development of RIM-Lite will encourage repository software developers, interface developers, and various types of search aggregation mechanisms, to more easily integrate ADL-R into their applications, e.g., automatically registering an object in ADL-R as soon

as it is deposited into a given repository. At the current state of development, anyone wishing to do that would have to understand a great many details of the ADL-R configuration as it is hosted at DTIC. RIM-Lite will manage these details and present a simpler picture to third party developers and that picture will remain unchanged even if the Registry and the environment in which it is hosted change over time.

1 RIM-Lite Requirements

1.1 Problem

The code for ADL-Registry has reached a level of stability and maturity that allows us to focus on the interaction of it with other applications. The original interface mechanism for interaction with the registry was through the use of a well defined HTTP interface directly into the Registry Engine. This approach needs a serious review due to the execution characteristics inside DTIC and a series of specific community requirements expressed by many application developers.

In the spirit of reuse and autonomy for the ADL-R community, we would like to keep the core registry as generic as possible and allow for high configurability at an upper layer. In the same spirit we understand the advantages of common aggregated behaviors to be exposed to all applications communicating with the registry. We believe though that this behavior should be grouped and included independently of the actual registry in an intermediate layer. The characteristics of this layer are to be determined by a particular set of Environmental constraints and application use cases that we shall use to generate a list of Design Requirements.

1.2 Environmental Constraints

The ADL-Registry is hosted by DTIC, which has imposed certain requirements on the design of the Registry. These include:

- virus checking;

- prevention of Denial of Service attacks;
- LDAP authentication; and
- Authentication of third-party applications.

1.2.1 Virus Checking

As part of the requirements to interact with any server hosted at DTIC, all the submissions to these servers must be scanned for viruses and in the future passed through a XML firewall. The virus checking must be applied prior to any real contact or access to the data inside the registry. DTIC has an internal separate commercial solution which provides virus checking services. This virus checker could introduce significant delays into the submission process. While no significant delays are reported. The RIM-Lite should deal with this potential delays and keep the client experience stable.

1.2.2 Denial of Service (DoS) Attacks

There is the chance that by allowing multiple applications to talk directly to the Registry that malicious attacks or malfunctioning code could create Denial of Service attacks and disrupt the system. In order to prevent this a mechanism must be designed to uniquely identify every application communicating with the RIM-Lite and try to avoid DoS attacks.

1.2.3 User Authentication

DTIC requires all users to authenticate using a valid LDAP account. This authentication is enabled through the use of an LDAP authentication proxy that receives a user and a

password and returns in turn a user handle that is used to compute the user rights in the Registry.

The LDAP authentication and user handle retrieval is to be performed at the RIM-Lite in order to maintain the registry authentication scheme cleanly based on user handles and to allow for other RIM interfaces to provide alternate authentication methods to the same registry. This would be useful for other implementations of the registry.

1.2.4 Application Rights

Most standalone (non portal) clients will deploy or integrate many applications with the ADL Registry through the RIM Lite. We must have a way to uniquely identify and calculate the rights for each of these applications and therefore both log and regulate their interaction with the RIM Lite and the registry. These applications could be registration and update agents or whole LCMS implementations. Hence the importance to identify them and potentially differentiate the interaction with them.

1.3 Use Cases

The following use cases illustrate both RIM-Lite functionality and operation characteristics.

1.3.1 Registry Administration

In order to provide important information for the Registry Administration and helpdesk support. The ADL RIM Lite must be able to provide important metrics information that

should allow the Registry administrators to both evaluate registry performance and diagnose problems. While the registry already keeps metrics these are generic and will have no specific information to link these metrics to particular clients or applications. The RIM Lite should be able to both expose these metrics and collect and expose more specific client and application metrics.

1.3.2 Automated Query

Automated interaction with the applications are intended to be enabled via the RIM lite. Thanks to the presence of well defined ADL RIM Lite interfaces and well defined serialization types and transformation methods, applications will be able to produce automatic queries and require predefined responses using particular method invocations. While RIM Lite will initially only interact with portals and some applications the infrastructure used to provide this interaction should scale to accommodate future interactions, methods and types.

An example would be to allow an LCMS or Microsoft office tools to directly query the registry and expose content objects discovered there to the users. This will directly integrate the Registry as a local resource to both applications and users.

1.3.3 Automated Submission

Automated submissions performed by update agents and registration agents integrated into content management and rendering applications should both be endorsed and regulated. The canonical use case for this is agents that monitor content objects inside a particular LCMS and generate registry insertion operations based on updated

information. These registration operations can then be submitted directly to the registry making this process transparent to users publishing or updating courses inside a LCMS.

The same is applicable to registration modules appended to content creation tools such as the Microsoft Office environment where registry publication is either an automatic operation or a simple export procedure. The RIM Lite will not only allow us to identify and differentiate these applications but also provide them with stable interfaces and behaviors that can impose particular regulations without affecting the core Registry operation.

1.3.4 Communication with Registry of Registries,

RIM Lite will allow the registry to communicate with other Registry of Registries by providing special methods that enable the serialization of metadata for aggregation to upper layers. It will also provide the environment to enable the deployment of distributed query mechanisms for those registries that are not willing to federate through aggregation. The RIM Lite is the perfect place to provide specific data processing for RoR integration, aggregation and query propagation without affecting the core registry operations and keeping it lean and fast for the local community use.

2 Design Requirements

2.1 Metrics on registry and RIM-Lite performance

RIM-Lite must provide access to the Generic registry metrics offered through the registry and also keep generic metrics based on the application granularity level. These metrics include:

- Number of successful submissions per application
- Total number of successful submissions
- Number of queries per application and in total
- Number of unsuccessful submissions per application and in total
- Maximum, Minimum and Average query time
- Maximum, Minimum and Average processing time
- Maximum, Minimum and Average Submission size
- Total Number of Items in the registry and submission and query totals

2.2 XSLT Transformations

XSLT transformations are needed to provide a common set of modified messages from the RIM-Lite to its clients. The RIM-Lite, unlike the portal, does not define these transformations to fit any particular single user or Portal feature but rather to accommodate common transformation needs such as enhanced error descriptions, basic result formatting and traditional pagination behavior. RIM-Lite will implement these

transformations using the TM3 model that we describe next and at least initially will only operate using XSLTs for these very generic use cases

- Enhanced Error messages
- Initial Result grouping and representation with basic pagination
- Enhanced Status representation
- Hookups for Registry of Registry Aggregation serialization for Upper layer federation (this feature will be expanded by CNRI for their research)

2.2.1 RIM-Lite XSLT transformation and TM3

Type Methodology Mapping Mechanism or TM3 is a CNRI proposed model by which a registry with all the supported types and methods to map from a particular type to another is kept by a particular community. Clients to the RIM-Lite can therefore express a requirement to obtain a particular response in a defined Type. These types are restricted to a single method association for the case of the RIM-Lite but may be expanded to include multiple methods in the future. The methods in TM3 for the RIM-Lite will primarily consist of XSL transformations and therefore point directly to XSLTs.

It is important to notice that both Types and Methods are identified by persistent actionable identifiers that can then be used to resolve either the type description or schema or the actual method implementation. This allows for multiple registry to rely on the same TM3 instance to operate over a set of responses and in the future metadata instances and content objects. It is assumed that a vetting process for the inclusion of

future methods and types will be put in place at a later date as part of a general ADL –R policy.

2.3 Enhanced error descriptions

RIM-Lite must provide enhanced error description rendering of its results to provide human-readable error messages. The error descriptions must be customizable on a per-community basis.

2.4 Non-UTF8 compliance

RIM-Lite must provide a pre-submission validation process to reject any submissions that have any non-valid UTF8 characters. This will help RIM-Lite to both operate and expose a standardized encoding standard for all data contained in the Registry or the RIM-Lite.

2.5 Error notification by email

Errors produced during pre-submission validation in the RIM-Lite must be emailed to the email associated with each application in the same spirit of the original Registry operation guidelines. These guidelines expressed the need to report all validation status information about lengthy (over 10 MB) submissions and all transaction errors asynchronously via email. This will allow third-party applications to monitor submissions on a push mode in which the registry notifies the applications as errors are detected.

2.6 HTTP POST Interface

RIM-Lite must provide an HTTP POST API for all functionality. RIM-Lite must provide and HTTP GET API for search queries.

Future versions of RIM-Lite may expose other APIs, such as SOAP or XML-RPC, but these are out of scope for the current work.

2.7 State

RIM-Lite is not required to provide any caching or state services to individual applications or users. Possible state based behavior and operations are beyond the scope of the RIM-Lite.

2.8 XML Responses

The RIM-Lite must generate all responses in XML, according to the XML schema accompanying this document and described below.

Some responses will be generated in html for the case of the generic query responses used to provide a direct rendering of results to clients without portal interaction.

2.9 Verbose Logging

RIM-Lite must maintain verbose logging of the operations and only expose it to the RIM-Lite administrators.

3 RIM-Lite

3.1 Client Authentication

As it was mentioned before, every client communicating with the RIM lite will have a n explicit identifier used to authenticate it into the system. This identifier called Client Access Key or CAK is in effect a Handle that uniquely identifies each client. Every client that would like to communicate with the RIM-Lite should register with the helpdesk and provide it with the clients execution IP. This IP is stored inside the CAK and also provided to DTIC in order to allow the particular IP to communicate with the RIM-Lite. Every client is then required to provide this CAK for each request to the RIM-Lite. The RIM-Lite compares the CAK's IP hitw he requests originating IP and uses this information to provide a basic simple validation of the requesting client. The RIM-Lite also uses this ID to log all operations form a particular client in a given period of time and apply appropriate restrictions to prevent a DoS from occurring.

3.2 Client application rights

For the time being all third-party application authentication and rights computation will be pursued in the context of the CAK value fields. The following operations shall be associated with every application in regards to its relationship with the registry :

- Q Query
- S Submit
- T Transaction check

- M Metrics collection
- P Priviledged

Where the particular application rights constraint its access to a particular service from the RIM-Lite and override the user rights associated with it. Each application needs to provide additionally a particular user for submission and status check requests. The Priviledged right allows the application to behave as an aggregator for the future RoR operations.

3.3 RIM-Lite Error Handling

RIM Lite will provide enhanced error description and processing. This error description will be the result of a transformation on top of the actual set of generic errors produced by the registry and will therefore be used to accommodate a common glossary for these errors to be kept in the particular RIM-Lite deployment.

3.4 RIM-Lite Schemas

The RIM Lite will operate based on the currently existing Registry schemas present at:

ADL-R Transaction: <http://hdl.cordra.net/2000.2.1/ADL-R-Reg-T>

ADL-R Status: <http://hdl.cordra.net/2000.2.1/adlregistry-transaction-status>

ADL-R Query Response : <http://hdl.cordra.net/2000.2.1/adlregistry-query-response>

ADL LOM schema at : <http://hdl.cordra.net/2000.2/adlreg-lom>

This document will be updated to include the links to the new RIM-Lite schema extensions of the afore mentioned documents if these are needed during development.

3.5 Current Registry HTTP POST interface

As of version 1.6.1 third party applications could still talk directly to the registry either directly in the case of the Practice Registry or through the Portal in the case of the production. This HTTP interface description is expressed in terms of the practice registry. For operational registry specifics please refer to the Help Desk Team headed by Bill Redeen . Parameters will be the same but the actual action address will be different.

All HTTP interactions have the following parameters:

enctype	multipart/form-data
method	post
action	http://38.100.138.145:8080/CORDRAWeb/processrequest

Interactions use the following optional input parameters:

Parameter	Parameter Name	Description
Description		
LDAP User Id	userid	
LDAP Password	password	
User email	email	
Submission Batchfile	batchfile	the File that follows the ADLReg Transaction schema
Registry Handle	registry	The registry that you would like to talk to
Command	command	
Transaction Id	transactionID	the Transaction ID that was assigned to your submission and that was returned to your

		application either synchronously as an XML response or asynchronously as and email if your submission required asynchronous validation and therefore got a validation ID.
Validation Id	validationID	The Validation ID assigned to your submission when it was determined that it was to big to process synchronously.
Page Number	page	Upon query the responses from the registry are grouped into pages of which the total number is indicated in the totalPages element on the query response present in the query response schema and that refers to the total number of pages produced by a query (http://hdl.cordra.net/2000.2.1/adlregistry-query-response). The page parameter allows you to request an specific page of these results.

3.6 Submission

The submission process is used to submit all operations and transactions to the registry.

Input Parameter Description	Parameter Name	Parameter Value
LDAP User Id	userid	
LDAP Password	password	
User Email	email	
Submission	batchfile	
Registry Handle	registry	4444/registry
Command	command	processtransaction

All submission returns are exposed and represented through the registry status schema at

<http://hdl.cordra.net/2000.2.1/adlregistry-transaction-status>.

3.7 Batch Status

Parameter Description	Parameter Name	Parameter Default Value
------------------------------	-----------------------	--------------------------------

Corporation for National Research Initiatives

15

last update: 2006-12-15

LDAP User Id	userid	
LDAP Password	password	
Transaction Id	transactionid	
Registry Handle	registry	4444/registry
Command	command	gettransactionstatus

3.8 Validation Status

Parameter Description	Parameter Name	Parameter Default Value
LDAP User Id	userid	
LDAP Password	password	
Validation Id	validationid	
Registry Handle	registry	4444/registry
Command	command	getvalidationstatus

3.9 Search Request

Parameter Description	Parameter Name	Parameter Default Value
Search Query	textsearch	
Page Number	page	0
Registry Handle	registry	4444/registry
Command	command	searchrequest

3.10 RIM-Lite HTTP POST interface

This is the http interface for the RIM-Lite, please notice the additions in blue and the (op) term that means that a particular parameter is optional.

enctype: multipart/form-data

method: post

action:

3.10.1 Submission

The submission process is used to submit all operations and transactions to the registry.

Input Parameters:

Parameter Description	Parameter Name	Parameter Default Value
LDAP User Id	userid	
LDAP Password	password	
User Email	email	
Submission	batchfile	
Registry Handle	registry	4444/registry
Command	command	processtransaction
Client Authentication Key	cak	4444.app/12345
(op) Response Type	rtype	4444.tm3.types/1

All submission returns are exposed and represented through the ADL-RIM registry status schema that will be a variation of the core Registry schema located at <http://hdl.cordra.net/2000.2.1/adregistry-transaction-status>.

3.10.2 Batch Status

Parameter Description	Parameter Name	Parameter Default Value
LDAP User Id	userid	
LDAP Password	password	
Transaction Id	transactionID	
Registry Handle	registry	4444/registry
Command	command	gettransactionstatus
Client Authentication Key	cak	4444.app/12345
(op) Response Type	rtype	4444.tm3.types/2

3.10.3 Validation Status

Parameter Description	Parameter Name	Parameter Default Value
LDAP User Id	userid	
LDAP Password	password	
Validation Id	validationID	

Registry Handle	registry	100.3/registry
Command	command	getvalidationstatus
Client Authentication Key	cak	4444.app/12345
(op) Response Type	rtype	4444.tm3.types/3

3.10.4 Search Request

Parameter Description	Parameter Name	Parameter Default Value
Search Query	textsearch	
Page Number	page	0
Registry Handle	registry	100.3/registry
Command	command	searchrequest
Client Authentication Key	cak	4444.app/12345
(op) Response Type	rtype	4444.tm3.types/4

3.10.5 Metrics

Parameter Description	Parameter Name	Parameter Values
Operation	operation	{submission; status; validation;

		search;info }
metric	metric	{max_run_time; min_run_time;avg_run_time;num_fail; num_suc; num_ops; info_tot_items}
Registry Handle	registry	100.3/registry
Client Authentication Key	cak	4444.app/12345
(op) Response Type	rtype	4444.tm3.types/5

4 Testing Plan

RIM-Lite will be considered complete when it fulfills the characteristics mentioned in this document. Additionally in order to verify that the RIM-Lite delivers all the same functionality of the registry plus the additional metrics and DTIC policy enforcements it will be necessary to extend the Test Harness to include RIM-Lite tests.

The test harness should therefore be able to test both an ADL-R registry and associated RIM-Lite having the RIM-Lite validation be an option .

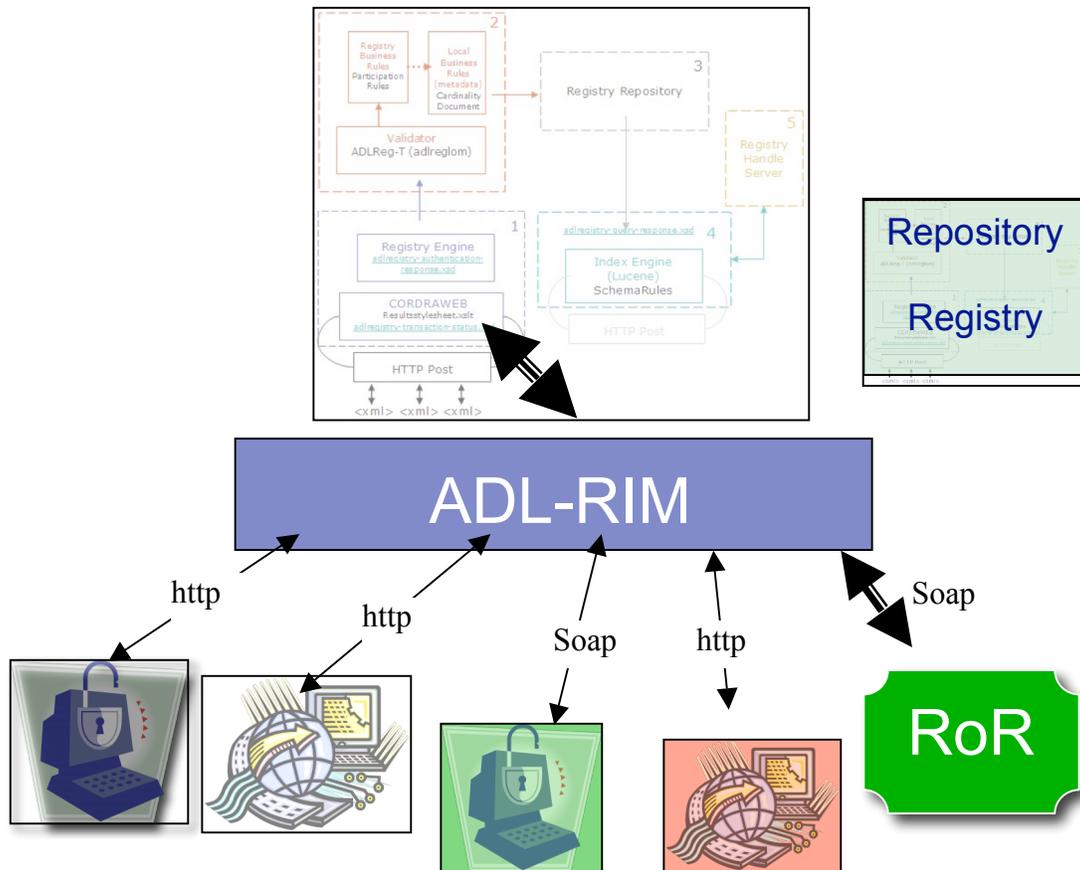
The test harness should also be fit with a new set of test cases and an official client access key.

As part of the test strategy CTC will create a new set of test cases to be used with the current test harness. These tests will be incorporated into this document as they become available. Additionally a generic test portal will be created by CTC and CNRI to both expose the RIM Lite functionality and test the RIM Lite scalability. This test portal will have a very generic and simple interface that will allow all the operations described in this document to be performed. It will not provide any of the functionality pursued by the official ADL-R portal or any of its future features.

Design details about the new test harness cases and the generic Portal will be incorporated into this document as they become available.

5 Internal Design

As it has been mentioned before the RIM-Lite is a completely independent module that operates on top of the core Registry as depicted in the following picture



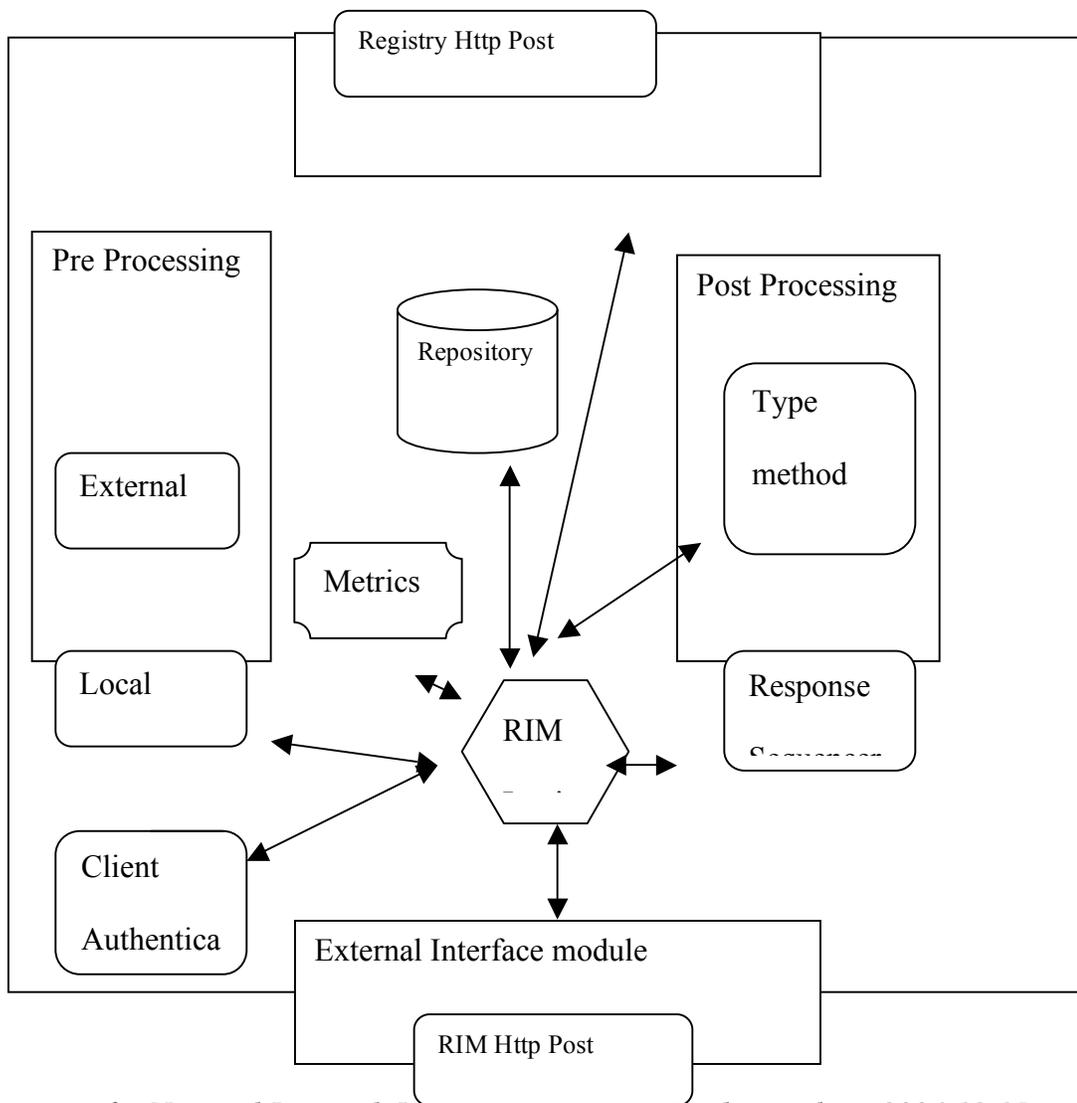
The RIM-Lite itself is intended to operate as a gateway between the different applications, portals, registries of registries and the Core Registry. The actual interface between the RIM-Lite and the exterior is a modular construct that allows it to use either Http Rest like services, SOAP or any other packaging protocol selected for web services.

RIM-Lite will initially deliver only REST like http based interaction and no other Web services implementation .

Internally the RIM-Lite provides the services mentioned before using the components showcased in the next section and following the generic data workflow introduced later.

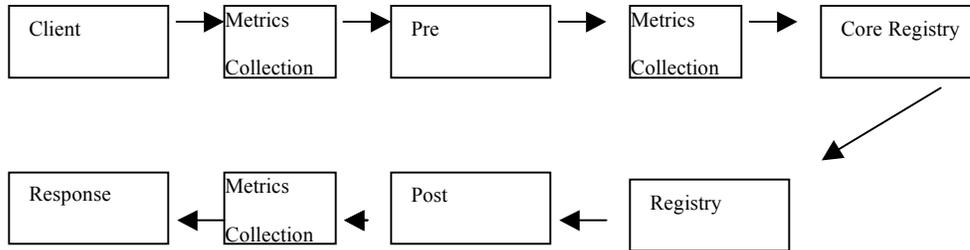
5.1 RIM-Lite Components

This graph showcases generic RIM-Lite components



5.2 General RIM-Lite Workflow

The following workflow shows the proposed generic workflow for all operations



5.3 RIM-Lite Implementation

RIM-Lite follows a modular paradigm in its implementation. Each module emphasizes on different functionality and has a well defined API for inter-module communication. The following are the different modules identified at this stage of implementation. The teams implementing the modules are stated in the parenthesis next to the module name below.

- (1) Service (CNRI)
- (2) Pre process (CNRI)
- (3) Process (CNRI)
- (4) Post Process (CNRI, CTC)
- (5) Gateway (CNRI)

(6) Access (CNRI)

(7) TM3 (CNRI)

(8) Repository (CNRI)

5.3.1 Service

This module performs well defined services for the RIM-Lite users. Please note that the implementation of each service is totally isolated from the definition of the service API. This loosely coupled behavior is to provide extensibility for integrating new services (or disintegrating existing services).

For each service request, the process work flow of Pre process -> Process -> Post Process is sequenced. More details about this work flow steps are described below.

The services offered are

- **Submission:** This service enables the user (a portal, or end user, or automated client) to submit the registry transaction batches. The submission summary (either a validation identifier, or transaction identifier, or a set of validation errors) is returned depending on the scenario. The exact formatting of the summary depends on the type of request parameters passed.

- Search: This service enables the user to submit a search query to the RIM-Lite. The query results are returned back. The exact formatting of the results depends on the type of request parameters passed.
- Transaction Status Retrieval: This service responds with the transaction status for the request transaction identifier. If the request parameters passed including user identifier/password, or transaction identifier, or registry handle are incorrect, an appropriate error message is returned back. Once again, the exact formatting of the status depends on the type of request parameters passed.
- Validation Status Retrieval: This service responds with the validation status for the request validation identifier. If the request parameters passed including user identifier/password, or validation identifier, or registry handle are incorrect, an appropriate error message is returned back. Once again, the exact formatting of the status depends on the type of request parameters passed.
- Metadata Retrieval: This service responds with the retrieval of the metadata instance corresponding to the metadata instance identifier passed as a request parameter. In the case of ADL-R, this would be LOM. Once again, the exact formatting of the status depends on the type of request parameters passed.
- Metadata Instance Information Retrieval: This service responds with the retrieval of the metadata instance information corresponding to the metadata instance identifier passed as a request parameter. In the case of ADL-R, this would return the user, group, and additional provenance of the metadata instance. Once again,

the exact formatting of the status depends on the type of request parameters passed.

- Metrics Evaluation: This service responds with some metrics and log summary.

The exact metrics provided by RIM-Lite are explained in the above sections.

5.3.2 Pre Process

This module performs the preprocessing for the services provided by RIM-Lite. Note that some of the services may not need pre processing. Some of the pre processing features are evaluating the business rules for validation, executing policies, storing metrics etc. The exact details of the pre processing may only be made available at the end of the implementation.

5.3.3 Process

This module performs the processing for the services provided. This includes accessing the core registry through the access module, and routing the response to the post process module.

5.3.4 Post Process

This module performs the post processing for the services provided. This includes formatting the response, elaborating the error response etc.

5.3.5 Gateway

This module interacts with the user. Once again, the user may be a portal, an end user, or an automated client. The gateway receives the request and its set of parameters, identifies the service requested and choreographs the process workflow for the service requested. It is important to note that for each communication protocol, there will be a different Gateway implementation. For RIM-Lite, the Gateway is implemented for HTTP. However, the modularity of the design allows for introducing many other communication protocols.

5.3.6 Access

This module interacts with the core registry. The module identifies the type of communication protocol, by noting the version of the registry. Therefore, the introduction of a new communication protocol at the core registry level, makes the RIM-Lite transparent to other modules. For Registry version 1.6.1, the access module is loaded with HTTP client.

5.3.7 TM3

The TM3 module is used to implement the Type method mapping. The conceptual description of the TM3 component is described in the sections above. The details of the implementation will be made available upon the implementation.

5.3.8 Repository

The Repository module is used to store the different logs and metrics. The implementation details, including the API, will be made available at the end of the implementation.

The different modules described above (barring TM3 and Repository) are depicted in the class diagram below. It is however important to note that the complexity of the class diagram may overwhelm the reader lacking adequate knowledge in Object Oriented Design, and UML. The reader is also encouraged to go through the Java Documentation simultaneously when perusing the implementation details.

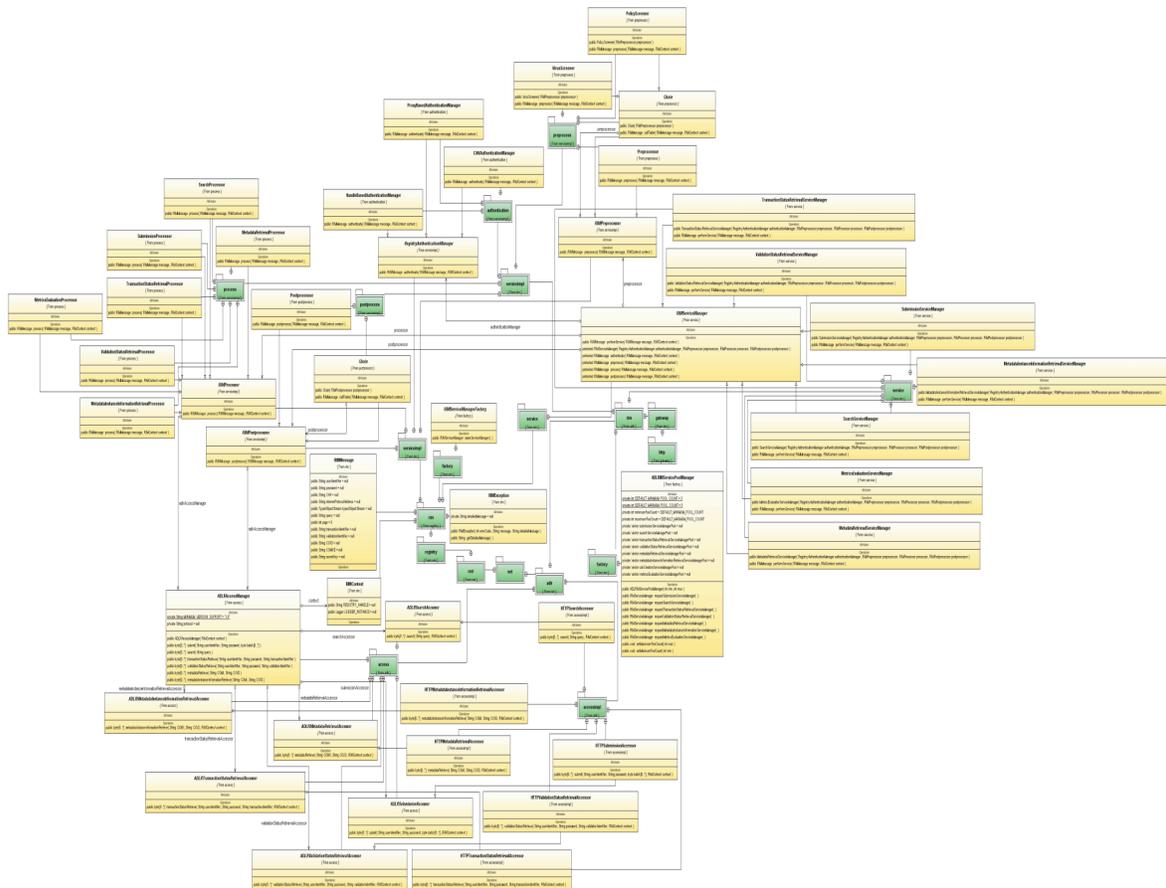


Figure 1: Detailed Class Diagram of RIM-Lite API (designed by CNRI)

The magnified version of the above class diagram may be found at the following URL:

[Detailed Class Diagram](#)

5.4 Error Codes and Descriptions:

Core Registry Errors: The errors resulted in the process of the core registry following a service request, are post processed by the RIM-Lite Post Processor module. This is to provide the user with detailed information about the root cause of the error. The error codes of the core registry are added to this document as an Appendix.

RIM-Lite Errors: In addition to the core registry errors, RIM-Lite introduces new error codes and descriptions. The details of the error codes and descriptions will be made available at the end of the RIM-Lite implementation.

5.5 Appendix: Core Registry Error Codes and Description

ERROR CODE NAME	ERROR CODE	DESCRIPTION
SUCCESS_CODE	0	
FAILURE_CODE	1	
NO_OPERATION_CODE	1	

MISSING_CONFIGURATION_VALUES	101	Missing Configuration Values
MISSING_REGISTRY_CONTEXT	102	Registry Configuration Context Missing
INVALID_REPOSITORY_CLIENT_SPECIFIED	103	Invalid Repository Client Specified in Configuration
UNKNOWN_URL_SCHEME	104	Index Engine URL Scheme (http or https) not found
HTTP_CLIENT_LOADING_FAILURE	105	HTTP Client Loading Failure
ERROR_CREATING_REGISTRY_LOG	106	Error creating Registry Log directory
PRINCIPAL_AUTHENTICATION_ERROR	501	Principal Authentication

		Error
UNKNOWN_PRINCIPAL_ERROR	502	Unknown Principal
MISSING_INFORMATION_FOR_AUTHENTICATION	503	Missing Information for Authentication
INCORRECT_REG_AUTH_CONFIGURATION	504	Incorrect registry authentication configuration.
UNKNOWN_AUTHENTICATION_ERROR	1000	Unknown Authentication Error
OPERATION_NOT_AUTHORIZED	1001	Operation Not Authorized
UNAUTHORIZED_GROUP	1002	User group not authorized to perform operation on registry
INSUFFICIENT_RIGHTS	1003	Insufficient Rights to

		perform request operation
ERROR_CREATING_REGISTRY_ADMIN_PRIVATEKEY	1004	Error Creating Registry Admin Privatekey
REGISTRY_ADMIN_HANDLE_READ_ERROR	1501	Registry Administration Handle Read Error
REGISTRY_ADMIN_HANDLE_WRITE_ERROR	1502	Registry Administration Handle Write Error
CORRUPTED_REGISTRY_ADMIN_HANDLE	1503	Corrupted Registry Administration Handle
FAILED_INITIALIZING_ADMINISTRATION_CLIENTS	1504	Failed Initializing Administration Naming

		Authority List
ADMINISTRATION_RIGHTS_MISSING	1505	Missing Administration Rights
REQUEST_NOT_SUPPORTED	2001	Request Not Supported
INVALID_INPUT_PARAMETERS	2002	Invalid Input Parameters for the request
INVALID_METADATA_NAMESPACE	2501	Invalid Namespace encountered for the Metadata.
VALIDATION_NOT_PROCESSED	2502	Validation not completely processed
INVALID_REGISTRY_METADATA	2503	Invalid Registry Metadata
CORRUPTED_CO_IDENTIFIER	2504	Corrupted CO Identifier (Handle

		Identifier)
MISSING_CORDRA_METADATA	2505	Missing CORDRA Metadata
UNKNOWN_VALIDATION_ERROR	3000	Unknown Validation Error
SEARCH_RESPONSE_FAILURE	3001	Search Response Failure
ERROR_CREATING_MIRROR_LOG	3201	Error creating Mirror Log
UNKNOWN_MIRROR_PROCESSING_ERROR	3300	Unknown Mirror Request Processing Error
NULL_BATCH_FOUND	6001	Null Batch Found
TIMESTAMP_FORMAT_UNKNOWN	6002	Unknown timestamp format
ERROR_JAVA_SERIALIZING_METADATA	6003	Error Java Serializing the Metadata
ERROR_JAVA_DESERIALIZING_METADATA	6004	Error Java

		Deserializing the Metadata
MISSING_VALIDATION_ID_ERROR	6005	Missing Validation ID
FIELD_NOT_SUPPORTED_FOR_QUERY	6006	Field not supported for query
ERROR_SERIALIZING_HANDLE	6501	Error serializing the Handle.
HANDLE_ALREADY_EXISTS	6502	handle already exists.
HANDLE_USED_IN_OTHER_CONTEXT	6503	handle already used in another content object context.
HANDLE_DOES_NOT_EXIST	6504	handle does not exist.
INCORRECT_HANDLE_TYPE	6505	Incorrect handle type.
ERROR_DELETING_HANDLE	6506	Error deleting the handle.

ERROR_RESOLVING_HANDLE	6507	Error resolving the handle.
ERROR_WRITING_HANDLE	6508	Error writing the handle.
ERROR_CREATING_HANDLE	6509	Error creating the handle.
ERROR_READING_HANDLE	6510	Error reading the handle.
ERROR_UPDATING_HANDLE	6511	Error updating the handle.
ERROR_DELETING_HANDLE_VALUES	6512	Error deleting the handle values.
ERROR_ADDING_HANDLE_VALUES	6513	Error adding the handle values.
ERROR_DESERIALIZING_HANDLE_ADMIN	6514	Error deserializing handle admin value
MISSING_EXPECTED_HANDLE_VALUES	6515	Missing Expected Handle

		Values
UNKNOWN_HANDLE_SYSTEM_ERROR	7000	Unknown Handle System Error
INSTANCE_ALREADY_EXISTS	7001	The content object metadata instance already exists in the store
INSTANCE_DOES_NOT_EXIST	7002	The content object metadata instance does not exist
ERROR_CONNECTING_REPOSITORY	7003	Error connecting to Repository
UNKNOWN_INDEX_ENGINE_HTTP_ERROR	8000	
KERNEL_PROCESS_CALL_FAILURE	8001	Failed to call kernel process
KERNEL_PROCESS_RESPONSE_FAILURE	8002	Failed to capture response from kernel process

MISSING_TRANSACTION_ID_ERROR	8003	Missing Transaction ID
PROCESS_INITIATED_IN_READ_MODE	8004	System Process Initiated in Read Mode
PROCESS_INITIATED_IN_WRITE_MODE	8005	System Process Initiated in Write Mode
INVALID_REGISTRY_OPERATION	8006	Invalid Registry Operation Specified
REGISTRY_OPERATIONS_NOT_DEFINED	8007	Registry Operations Not Defined
MISSING_CONTENT_OBJECT_HANDLE	8008	Missing Content Object Handle
INCORRECT_HANDLE_TYPES_IN_CORE	8009	Incorrect Handle Types in CORE Handle
INVALID_OPERATION_SPECIFIED	8010	Invalid Operation

		Specified
OPERATION_TIMESTAMP_IS_NOT_RECENT	8011	Requested Operation Timestamp is not recent
ERROR_CREATING_METADATA_BUFFERS	8012	Error creating metadata buffers
ERROR_RETRIEVING_CORE_ELEMENTS	8013	Error retrieving elements from CORE
UNKNOWN_SYSTEM_PROCESSING_ERROR	8500	Unknown System Processing Error
LDAP_SERVER_COMMUNICATION_ERROR	20501	
NULL_RESPONSE	20502	Null Response from LDAP Proxy
NULL_USERID	20503	Null Userid in Response from LDAP Proxy
NULL_STATUSCODE	20504	Null statusCode

		in Response from LDAP Proxy
INVALID_STATUSCODE	20505	Invalid statusCode in Response from LDAP Proxy
NULL_USERHANDLE	20506	Null userHandle in Response from LDAP Proxy
NULL_PRIVATEKEY	20507	Null privateKey in Response from LDAP Proxy
NULL_EMAIL	20508	Null email in Response from LDAP Proxy
ERROR_XML_RESPONSE_PARSING	20509	Error reading response from LDAP Proxy

INVALID_CONTENT_OBJECT_LOCATION_SPECIFIED	22501	Invalid Content Object Location Specified
MISSING_CONTENT_OBJECT_LOCATION	22502	Missing Content Object Location
BUSINESS_LOGIC_FAILURE	22701	
PARTICIPATION_RULES_NON_CONFORMITY	22702	Participation Rules Non Conformity. Data fields missing for the operation
SUBMITTER_USER_MISSING	22703	User Identifier Missing in the Batch
SUBMITTER_GROUP_MISSING	22704	Group Identifier Missing in the Batch
PARTICIPATION_RULES_PARSER_MISSING	22705	Participation Rules Parser not found

PARTICIPATION_RULES_LOADING_ERROR	22706	Participation Rules Loading Error
INCONSISTENT_USER_IDENTIFIERS	22707	Inconsistent userId specified in batch and submission
STATUS_READING_ERROR	22801	Validation status reading error
STATUS_WRITING_ERROR	22802	Validation status writing error
ERROR_XML_PARSING	24001	Error parsing the xml
SAX_XML_ERROR	24002	Error in the sax parser xml
ERROR_READING_XML_DOCUMENT	24003	The parser could not read the XML document
XML_DOCUMENT_ENCODING_ERROR	24004	The XML document has an unsupported

		encoding
XML_MISSING_INSTANCEID_ERROR	24005	The content object metadata instance is missing an instance identifier
XML_MISSING_REQUIRED_FIELD	24006	Missing required field(s)
XML_UNSUPPORTED_ENCODING	24007	
MISSING_TRANSFORMATION_ENTITIES	24008	Missing transtlet
XSL_TRANSFORMATION_ERROR	24009	Internal XSL Transformation Error.
INTERNAL_METADATA_VALIDATION_ERROR	24010	Internal Metadata Validation Error
XML_DATECONVERSION_ERROR	24011	XML Date Conversion Error
ERROR_LOADING_XSLT	24013	Error Loading

		XSLT
MISSING_TRANSACTION_EXTRACTOR	24014	Transaction Extractor is null. Apply Business Logic before Transformation.
INVALID_METADATA_NAMESPACE	24015	Invalid Namespace encountered for the Metadata.
TRANSACTION_STATUS_READING_ERROR	26001	Submission status reading error
TRANSACTION_ID_USED	26002	Transaction ID is already used
INVALID_TRANSACTION_ID_GENERATED	26003	Invalid Transaction ID generated