System Design

Specification

For

RESTful WHOIS

CNNIC

Revision Sheet

Release No.	Date	Revision Description
1.0	2014.02.22	Initial
1.1	2014.04.15	Redesign system
1.2	2014.12.18	Modify for update API: modify architecture diagram, add update module description.
1.3	2015.04.15	Add class diagram for search strategy

Table of contents

1. Overview

RESTful WHOIS is an implementation of RDAP (Registration Data Access Protocol), and it is used to retrieve registration information from registries using RESTful (HTTP+JSON) web access patterns.

1.1. Architecture

Architecture of RESTful WHOIS is shown as Figure 1.1-1.

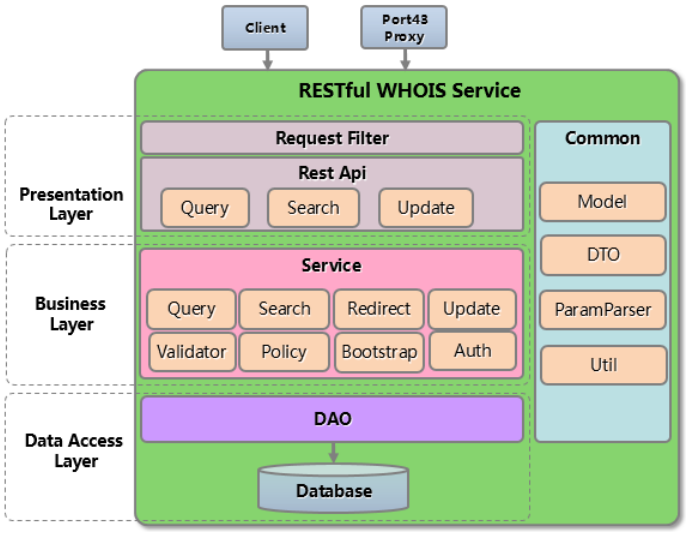

Figure 1.1-1 RESTful WHOIS Architecture

1.1.1. Overview of Modules

1. Presentation Layer
   a) Request Filter: limit rate, handle requests with non-HTTP GET method.
   b) Rest API: query, search and update API.
2. Business Layer
   a) Service module
      i. Query: query registration data by exactly matching.
      ii. Search: search registration data.
      iii. Redirect: redirect service for non-authoritative objects of this registry, see [RFC 7480].
      iv. Update: update API, used to update RESTful WHOIS data.
      v. Auth: authentication and authorization.
      vi. Policy Control: used to hide some columns of objects according to registry's policy.

vii. Bootstrap: synchronize ‘Redirect Table’ from IANA’s Registry, ‘Redirect Table’ is used in Redirect module.
viii. Validator: validate query parameters.

3. Data Access Layer
   a) DAO: access data from data source.
   b) Database: MYSQL database.
4. Common module
   a) Model: object models corresponding to Object Class in JSON response, see [RFC 7483].
   b) DTO: data transfer object, used to transmit data in update API.
   c) ParamParser: parse query parameters.
   d) Util: validate query parameters, and convert data format.

1.1.2. Related Outer Object

1. Client: the client of the RESTful WHOIS Server.
2. Port43 Proxy: a proxy which will translate port 43 WHOIS queries ([RFC3912]) into RESTful queries.
3. IANA registry: ‘Redirect Table’ for Redirect service is synchronized from IANA registry, see [RFC 7480].

1.1.3. Query Process

For illustrative purposes, a domain query process by a logined client is shown as following Figure 1.1.3-1.

1. The client sends a domain query request ‘/domain/cnnic.cn’ to RESTful WHOIS Server.
2. Request Filter check whether query frequency of this client exceeds the system limit.
3. Call Rest API.
4. Rest API parses the request URI and obtains query type - ‘domain’, and call Query module.
5. Query module checks the validity of request parameter.
6. If query parameter is valid, then Query module calls Data Access module.
7. Query module calls Policy Controller to remove some columns from the result according to policy. See section 2.2.3.
8. Query module calls Auth module, Auth module removes columns which is not accessible to this client. See section 2.2.2.2.
9. Query module formats result data to JSON by calling Util module.
10. Rest API writes JSON response to client

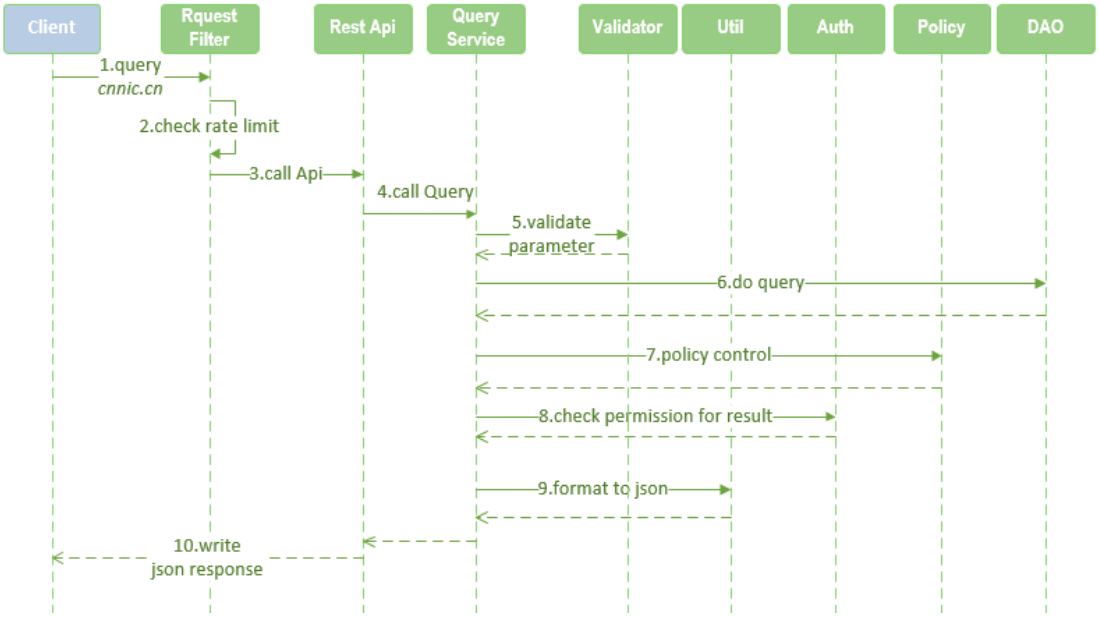

Figure 1.1.3-1 Domain query process

2.Modules

2.1. Presentation Layer

Presentation layer is designed to display the user interface and manage user interaction.

2.1.1. Request Filter

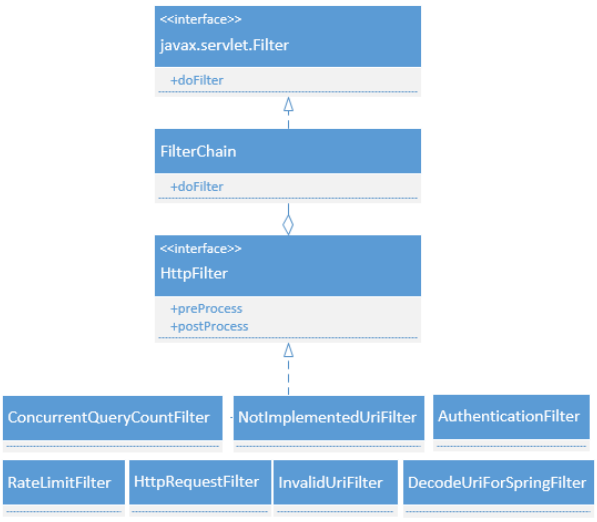

Figure 2.1.1-1 Request Filter class

Description of classes:

1. Javax.servlet.Filter: servlet filter interface. Request Filter module depends on servlet filter.
2. FilterChain: contains all filters in RESTful WHOIS Server, and invokes them to filter request.
3. HttpFilter: filter interface for all HTTP filters in RESTful WHOIS. It has two methods to be implemented:
   a) preProcess: this is called before process service.
   b) postProcess: this is called after process service.
4. HttpFilter implementation
   a) RateLimitFilter: prevents highly frequent queries, it returns 429 error to client if query interval is shorter than the configured interval.
   b) ConcurrentQueryCountFilter：controls the concurrent query count.
   c) HttpRequestFilter: handles requests with non-HTTP GET method, and it will return error to client.
   d) AuthenticationFilter: authenticates, and gets authentication information from client.
   e) InvalidUriFilter: handles invalid URI.
   f) DecodeUriForSpringFilter：decodes URI for spring MVC.
   g) NotImplementedUriFilter：handles the not-implemented URI.

2.1.2. Rest API

There are 2 types of Rest API:
1. Query and Search API
2. Update API

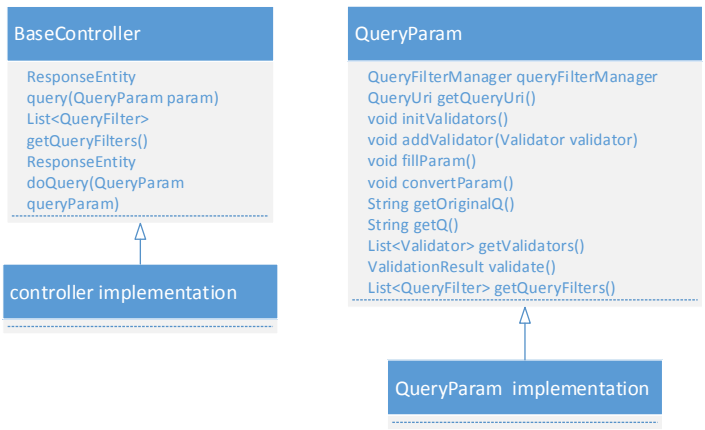

Figure 2.1.2-1 Query and Search API class

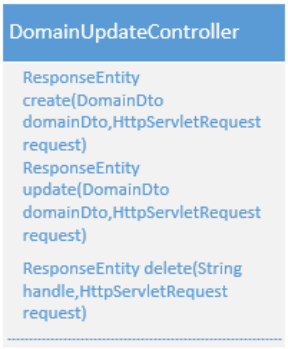

Figure 2.1.2-2 Domain update API class

Description of class:

1. BaseController: base controller for query/search service. It has a template method for querying, and the method 'doQuery' need to be implemented by subclass.
2. QueryParam: base query parameter bean.
3. Update API:
   a) DomainUpdateController
   b) NameserverUpdateController
   c) EntityUpdateController
   d) AutnumUpdateController
   e) NetworkUpdateController

Query process:

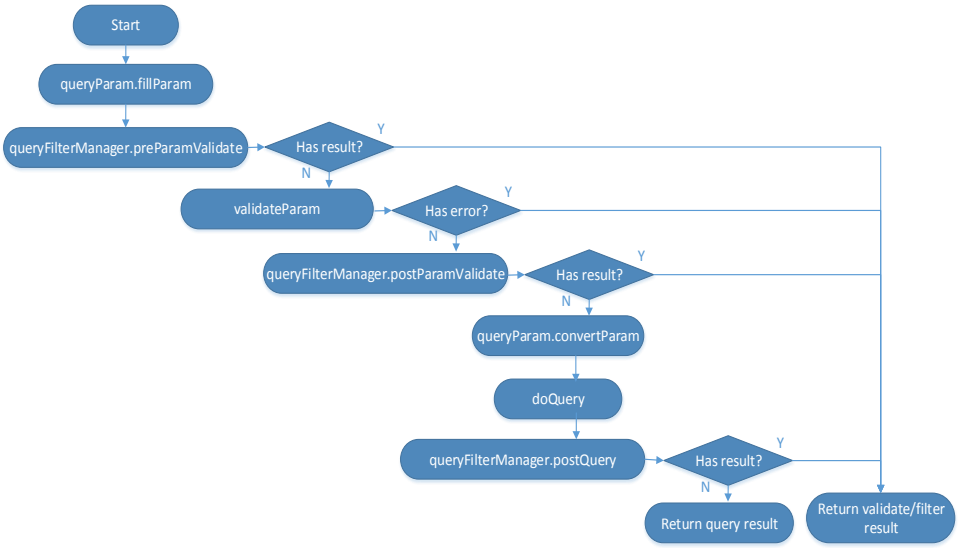

Figure 2.1.2-3 Query process

2.2. Business Layer

2.2.1. Query

Query service is designed to lookup registration objects.

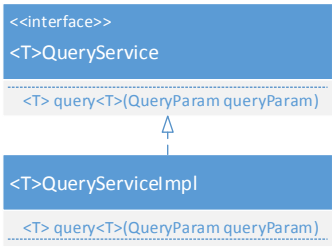

Figure 2.2.1-1 Query Service class

Every model type has its query service interface and implementation, called <T>QueryService and <T>QueryServiceImpl, 'T' is model type name.

2.2.2. Search

Search service class diagram is shown as following figure.

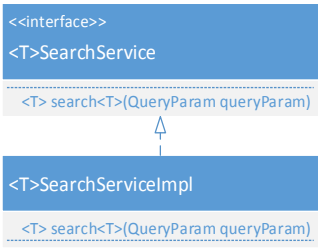

Figure 2.2.2-1 Search service class

Every model type has its search service interface and implementation, called <T>SearchService and <T>SearchServiceImpl, 'T' is model type name.

The search process is different betweeenn different search type, and it is handled by search strategy:

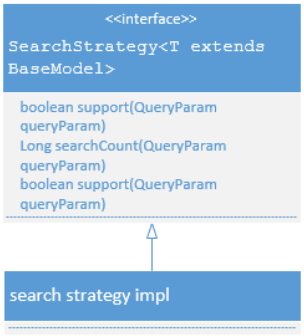

Figure 2.2.2-2 Search strategy class

2.2.3. Update

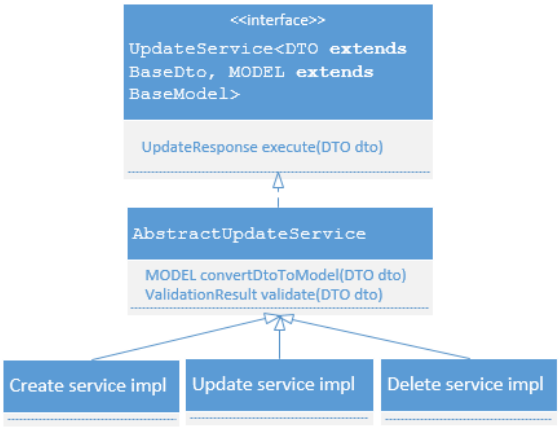

Figure 2.2.3-1 Update service class

Every model type has its update service implementation.

2.2.4. Redirect

Redirect module is designed to query authoritative information for non-authoritative query. This is done by querying a 'Redirect Table', which is synchronized from IANA registry periodically, see [RFC 7480].

Redirecting is done by an implementation of HttpFilter.

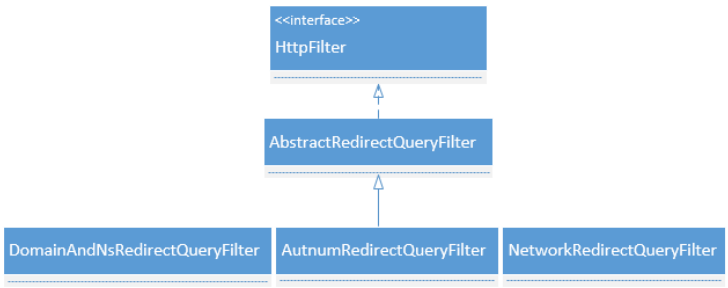

Figure 2.2.4-1 Redirect class

2.2.5. Auth

Auth module is used for authentication and authorization.

1. **Authentication**

   Authentication module has a default implementation by using HTTP Basic + TLS, see [RFC 7481].

   It is done by AuthenticationFilter, an implementation of HttpFilter.

2. **Authorization**

   Authorization module is designed to control access to resources.

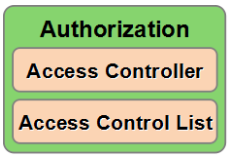

Figure 2.2.5-1 Authorization module

RESTful WHOIS uses Access Control List (ACL) for authorization, ACL is a permission control model, see [RFC4949].

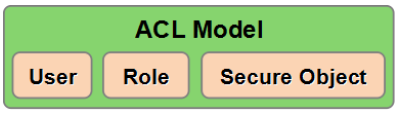

Figure 2.2.5-2 ACL model

1. User: authenticated user.
2. Role: role of user.
3. Secure Object: resource to be protected. There are two kinds of Secure Object:
   a) Registration Object: per registration object.
   b) Uri: protected Uri, such as '/domains', '/nameservers'.

Access Controller is a module to Control access for Secure Object when query or search. Non-authorized access will be forbidden.

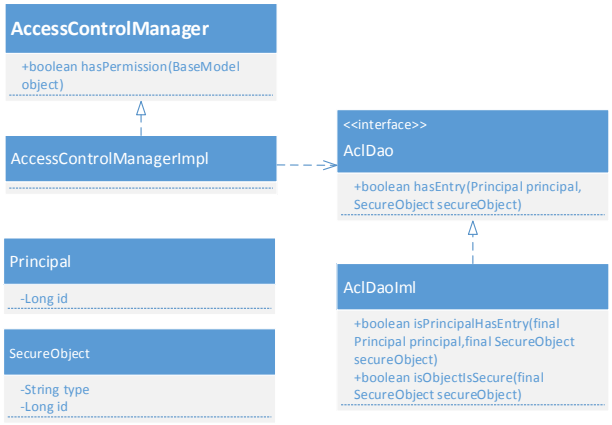

Figure 2.2.7.2-3 Access Controller class

Class description:

1. Principal: client's identity.
2. SecureObject: object that needs access control.
3. AccessControlManager: access control interface.
4. AccessControlManagerImpl: default implementation of AccessControlManager. All Registration object is readable (or accessible) for all users by default. An authorization entry [user-role-secure Object] will be added when permission control is needed. Users who have no entry for this secure object will be forbidden to access this object.
5. AclDao: used to load ACL entry.
6. AclDaoImpl: default implementation of AclDao, by loading ACL entry from Database.

2.2.6. Policy Control

Policy control is used to hide some columns of objects according to registry's policy. Policy control is done by an implementation of HttpFilter.

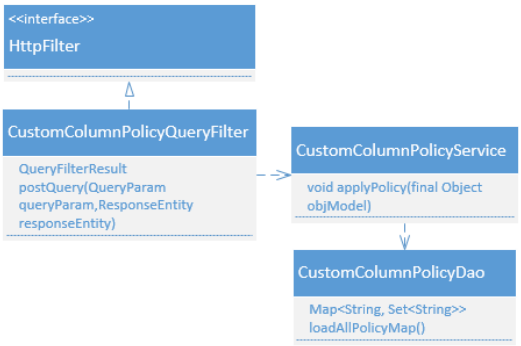

Figure 2.2.6-1 Policy Control class

2.2.7. Validator

Validator is used to validate query parameters. All validators are drived from the 'Validator' interface. There are 11 pre-defined validators.

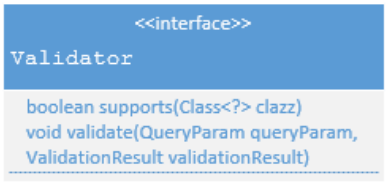

Figure 2.2.7-1 Validator interface

2.2.8. Bootstrap

The Redirect module depends on a 'Redirect table', and bootstrap module is designed to synchronize this Redirect table from IANA periodically.

2.3. Data Access Layer

Data Access Layer is designed to access data from data source.

2.3.1. Data Access

Data Access is aimed at making it easy to work with different data sources.

Data Access module has a default implementation for MYSQL database accessing, this implementation can be replaced by writing custom Data Access implementation and providing database SQL dialect for specified database (see

section 4.1).

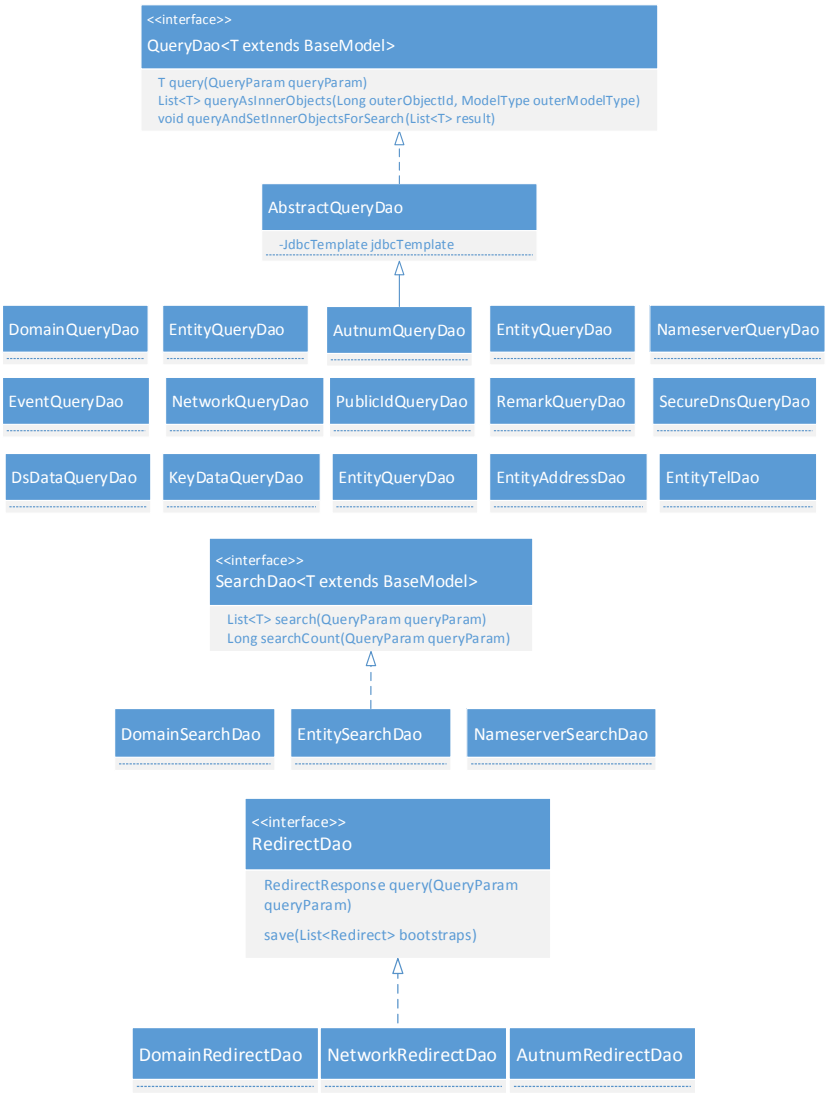

Figure 2.3.1-1 Data Access class

Class description:

1. QueryDao: query DAO interface.
2. SearchDao: search DAO interface
3. RedirectDao: redirect DAO interface.

2.4. Common Modules

2.4.1. Model

Model class is shown as following figure. Some joined-objects are not included.

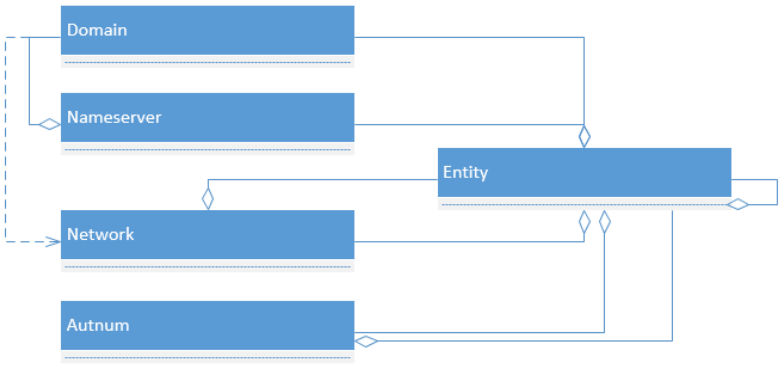

Figure 2.4.1-1 Model class

2.4.2. DTO

Data transfer object, used to transmit data in update API.

2.4.3. ParamParser

Parse query parameters from query request.

2.4.4. Util

Util module contains validation and data conversion tools.

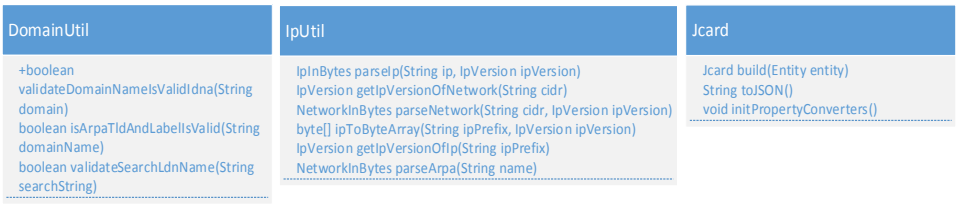

Figure 2.4.4-1 Util class

1. DomainUtil: validate a-label/u-label domains according to [RFC1035] and [RFC5891].
2. IpUtil: validate ipv4/ipv6/arpa; convert ip/arpa from/to byte value.
3. Jcard: format entity information to Jcard (see [RFC 7483]) format.

2.5. Related Outer Object

Related outer object to RESTful WHOIS.

2.5.1. Client

Client send query/search requests to RESTful WHOIS server, and get corresponding responses.

1. Http Client: Client that can direct access to RESTful WHOIS Server using HTTP1.1, for example: using Linux command ‘wget’ or ‘curl’ to send requests.
2. Port43 Client: Client that can access WHOIS service via port 43 conforming to [RFC3912].

2.5.2. Port43 Proxy

Port43 Proxy is a user interface which is able to accept client queries via port 43, conforming to [RFC3912].

Port43 Proxy transmits port43 queries to HTTP queries that can be accepted by Request Filter (see section 2.1.2).

Class Diagram: https://github.com/cnnic/rdap/wiki/proxy43-class--diagram.

2.5.3. IANA Registry

IANA maintains bootstrap’s information in its registry (ref [RFC 7480]), and ‘Redirect Table’ in RESTful WHOIS use this information to handle redirect query.

3. Extension

3.1. Custom Column Extension

Query object for domain/nameserver/entity/as number/ip can has dynamic columns.

1. Database support: these tables has an extension column ‘CUSTOM_PROPERTIES’, which is in JSON format and is used to store dynamic column key-value.
2. Code support: The base model of all object - ‘BaseModel’ has a ‘map’ type property called ‘customProperties’, which is used to maintain dynamic column data in key-value format.

3.2. Other Extension

Please ref Customize and Develop.

4. References

[RFC 7483] JSON Responses for the Registration Data Access Protocol (RDAP)
[RFC 7482] Registration Data Access Protocol (RDAP) Query Format
[RFC 7480] HTTP Usage in the Registration Data Access Protocol (RDAP)
[RFC 7481] Security Services for the Registration Data Access Protocol (RDAP)
[RFC3912] WHOIS Protocol Specification
[RFC1035] Domain names - implementation and specification
[RFC5891] Internationalized Domain Names in Applications (IDNA)
[RFC4949] Internet Security Glossary, Version 2