Reference Manual

For

RESTful WHOIS

CNNIC

Revision Sheet

Release No.	Date	Revision Description
1.0	2014.10.12	Initial
1.1	2014.11.02	Add API description
1.2	2015.03.08	Modify for update API
1.3	2015.04.17	Add description about system extension

Table of Contents

1. Introduction

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. RESTful WHOIS Module

The project is written in JAVA. Modules:

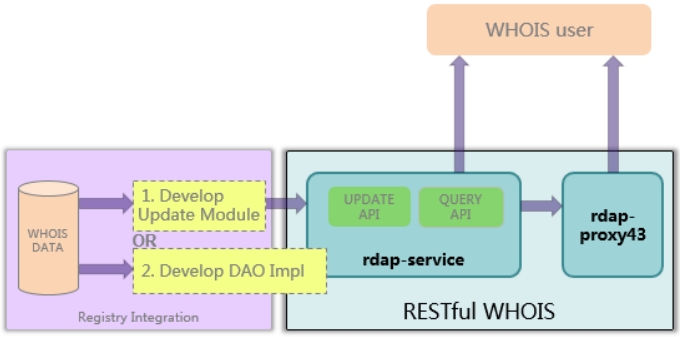

Figure 1.1-1 RESTful WHOIS Modules

1. rdap-service

   The rdap-service is a HTTP service, can be deployed in SERVLET container, such as TOMCAT or JETTY. It loads data from MYSQL database by default.

   It has two kinds of API:

   1) Query API

      Registrar user can query WHOIS data from this API.

   2) Update API

      Registry can update WHOIS data from this API.

   There are two approaches for registries to implement:

   1) Update RESTful WHOIS service data via data update API.
   2) Develop DAO module of rdap-service to access the database of the registry.

2. rdap-proxy43

The rdap-proxy43 is a proxy which will translate port 43 WHOIS queries into RESTful queries. It can query the RESTful WHOIS Service, and return the plain text results via PORT 43 back to clients.

2. Install

2.1. Supported Operating System

Tested operating environment:

1) Red Hat Enterprise Linux Server release 5.3
2) CentOS release 5.7
3) Win7
4) Win8
5) OS X 10.8.4.

2.2. rdap-service

1. Install JDK7, or higher version. (Skip this step if already installed)

2. Install MYSQL5, or higher version. (Skip this step if already installed)

   Create user 'rdap' and grant privilege.

   ($RDAP_SERVER_IP must change to RESTful WHOIS server IP, $MYSQL_PASSWORD change to 'rdap' user's password)

```
GRANT ALL PRIVILEGES ON *.* TO 'rdap'@'$RESTful_WHOIS_SERVER_IP' IDENTIFIED BY '$MYSQL_PASSWORD';
FLUSH PRIVILEGES;
```

   More details please refer here.

3. Install TOMCAT7, or higher version. (Skip this step if already installed)

   Download and Install TOMCAT7 or higher version, and HTTP port use default port 8080 (see here if use other ports).
   Installed TOMCAT root folder called '$TOMCAT_HOME', which contains folders: bin, conf, lib, webapps, etc.

4. Get RESTful WHOIS war file.
   There are two methods to get war file:
   - You can get war file building by JDK7.
   - Or Build war file from source

5. Deploy war to TOMCAT.
    - Create folder 'rdap' in directory $TOMCAT_HOME/webapps/
    - Unzip war file to $TOMCAT_HOME/webapps/rdap/
    - Edit database configuration file jdbc.properties
    - Edit global configuration file rdap.properties

6. Initialize database.

    This step will create database named 'rdap', and you can insert test data into it.

    This step will use database info in jdbc.properties you have configured before.

    WARN: this step will DROP database of 'jdbc.url.dbName' if it is existing, and then recreate it.

```
cd $TOMCAT_HOME/webapps/rdap/WEB-INF/classes
CLASSPATH=.:$CLASSPATH #in windows this command can be ignored
java -Djava.ext.dirs=../lib org.restfulwhois.rdap.init.Init initschema          #DROP database 'jdbc.url.dbName', and recreate it, and create table.
```

7. Start up TOMCAT.

    - Start up TOMCAT

```
[in Linux/OS X, open a shell and execute command:]
    cd $TOMCAT_HOME        #$TOMCAT_HOME must be replaced by real directory
    bin/startup.sh
[in Windows, open command prompt window and execute command:]
    cd $TOMCAT_HOME/bin        #$TOMCAT_HOME must be replaced by real directory
    startup.bat
```

    - Test if it is ok

```
    curl -H Accept:application/rdap+json http://$RESTful_WHOIS_SERVER_IP:$RESTful_WHOIS_SERVER_PORT/rdap/autnum/2100
```

    It's ok if response contains 'rdapConformance'.

2.3. rdap-proxy43

1. Get executable jar file 'rdap-proxy43-jar-with-dependencies.jar'.

    There are two methods to get this file:

    - Get from here for jar build with JDK7.
    - Build from source

2. Copy rdap-proxy43-jar-with-dependencies.jar to proxy43 install directory, and we call it $PROXY43_INSTALL_DIR.

3. Download configuration file "proxy43.properties" from here, and copy it to $PROXY43_INSTALL_DIR.

   You can edit this file for production use, see here

4. Start up. **Must use root user**.

   - Start up

```
[in Linux/OS X, open a shell and execute command:]
    cd $PROXY43_INSTALL_DIR #$PROXY43_INSTALL_DIR must be replaced by real directory
    nohup java -jar rdap-proxy43-jar-with-dependencies.jar start &
[in Windows, open command prompt window and execute command:]
    cd $PROXY43_INSTALL_DIR #$PROXY43_INSTALL_DIR must be replaced by real directory
    java -jar rdap-proxy43-jar-with-dependencies.jar start
```

   - Test if it is ok

     Run jwhois command, in Linux/OS X for example:

```
  whois -h $PROXY43_HOST cnnic.cn #$PROXY43_HOST must be replaced by real proxy43 host
```

     It's ok if response contains 'rdapConformance'.

   - Shutdown

```
cd $PROXY43_INSTALL_DIR #$PROXY43_INSTALL_DIR must be replaced by real directory
java -jar rdap-proxy43-jar-with-dependencies.jar shutdown
```

3. Configuration

3.1. Database Configuration

$TOMCAT_HOME/webapps/rdap/WEB-INF/classes/jdbc.properties

For test purpose, the first 4 properties - 'jdbc.url.hostPort', 'jdbc.url.dbName', 'jdbc.username', 'jdbc.password' must be configured.

(Lines start with '#' are comments)

```
#jdbc URL host and port.
#MUST change $MYSQL_HOST_OR_IP to MYSQL host or ip
jdbc.url.hostPort=jdbc:MYSQL://$MYSQL_HOST_OR_IP:3306/
#database name
jdbc.url.dbName=rdap
#value change to jdbc username
jdbc.username=**
#value change to jdbc password
jdbc.password=**
#jdbc URL params
jdbc.url.params=useUnicode=true&characterEncoding=UTF-8
# jdbc driver class name
jdbc.driverClassName=com.MYSQL.jdbc.Driver
#jdbc max pool size
jdbc.maxPoolSize=100
#jdbc min pool size
jdbc.minPoolSize=3
```

3.2. Common Configuration

$TOMCAT_HOME/webapps/rdap/WEB-INF/classes/rdap.properties

For test purpose, the first 3 properties - 'localServiceUrl','inTlds','notInTlds' must be configured.

(Lines start with ‘#’ are comments)

```
#local service URL, without scheme. This value is used in redirect service,
#to check if redirect URL is local service URL, and ignore the redirect
#if is local service URL.
localServiceUrl=http://rdap.restfulwhois.org
#puny name of tlds in this registry, splited by ';'.Only in this list can
#be query.
inTlds=cn;xn--fiqs8s;arpa
#tlds not in this registry, splited by ';'.
#tlds in this list can not be query,and will query redirect instead.
#NOT-IN-TLDS has higher priority than IN-TLDs.
notInTlds=edu.cn
#max size for search.
maxsizeSearch=5
#batch size for search.
batchsizeSearch=100
```

```
#minSecondsAccessInterval for anonymous request. <=0 means no limit.
minSecondsAccessIntervalAnonymous=-1
#minSecondsAccessInterval for authenticated request. <=0 means no limit.
minSecondsAccessIntervalAuthed=-1
#max concurrent query count. 0 means no limit. Should less than web container's max threads.
maxConcurrentCount=0
#ipWhiteListForAccessInterval.proxy43'ip may put into this list.
ipWhiteListForAccessInterval=127.0.0.1;
#Requests from these IPs can be handled, and others will return 403 error.
ipWhiteListForUpdateApi=127.0.0.1;0:0:0:0:0:0:0:1
#not implemented uri,splited by ';'
#if this valid values    not null ,it must be a combination of '/help' '/domains' '/domain/'
#'/entity/' '/entities' '/nameserver/' '/nameservers' '/autnum/' '/ip/', splited by ';' .
# else the valid values is null
notImplementedUri=
#custom property prefix, NIC name is recommended.
customPropertyPrefix=cnnic_
```

3.3. Bootstrap Configuration

$TOMCAT_HOME/webapps/rdap/WEB-INF/classes/bootstrap.properties

This is used for bootstrap service in RESTful WHOIS service. It will synchronize data from IANA periodically.

(Lines start with '#' are comments)

```
#IANA bootstrap registry base URL
bootstrapRegistryBaseUrl=
#bootstrap URI for domain
bootstrapRegistryUriForDomain=domain.jsp
#bootstrap URI for as
bootstrapRegistryUriForAs=as.jsp
#bootstrap URI for ipv4
bootstrapRegistryUriForIpv4=ipv4.jsp
#bootstrap URI for ipv6
bootstrapRegistryUriForIpv6=ipv6.jsp
#CronExpression
cron.bootstrap=0 0 0 1 1 ?
```

(lease refer Quartz cron expression for CronExpression)

4. Query API

- Only support HTTP 'GET' method
- Media type must be 'application/rdap+json' or ‘application/json’, in 'Accept' header.
- URI and parameters must be encoded in UTF-8.
- Unknown parameters will be ignored.
- Support HTTP BASIC authentication. When using BASIC authentication, HTTPS must be used.
- Response is in JSON format. See 'Response Code' section for Response code.

Examples:

- Request without authentication:

```
Request URL: http://rdap.restfulwhois.org/entity/et-1
Request Method: GET
Accept: application/rdap+json
```

- Request with HTTP BASIC authentication:

```
Request URL: https://rdap.restfulwhois.org/entity/et-1
Request Method:GET
Accept:application/rdap+json
Authorization: BASIC $BASE64_ENCODED_USERNAME_PASSWORD
```

($BASE64_ENCODED_USERNAME_PASSWORD must be replaced by base64 encoded "username:password" string. Certificate for rdap.restfulwhois.org is here)

4.1. Response Code

Response Code	Description
200	Ok
404	Not found for query
400	Request URI is invalid, or parameter is invalid
422	Unsupported search string : more than one '*' in queries, or a query starts with '*', or a query is '*'
415	Request with invalid media type: 'Accept' header doesn’t contain

Response Code	Description
	'application/rdap+json' or 'application/json'
401	Unauthorized: authentication failed
403	Forbidden: the query object is forbidden for this client
405	Method Not Allowed. Only 'GET' method is allowed
500	Internal Server Error
301	Moved Permanently, client should request for the URL by 'Location' in Response header
429	Too Many Requests, client should wait for a moment before query
509	Bandwidth Limit Exceeded: server is busy, client should try later

4.2. API

1. Query Entity

URI: /entity/<handle>
Example: http://rdap.restfulwhois.org/entity/et-1

Click here for the response body.

2. Query Nameserver

URI: /nameserver/<nameserver name>
Example: http://rdap.restfulwhois.org/nameserver/xn--1-dr6av31f.xn--0zwm56d.xn--fiqs8s

Click here for the response body.

3. Query Domain

URI: /domain/<domain name>
Example: http://rdap.restfulwhois.org/domain/cnnic.cn

Click here for the response body.

4. Query IP Network

URI: /ip/<IP address> or ip/<CIDR prefix>/<CIDR length>

Example:
http://rdap.restfulwhois.org/ip/218.0.0.3
http://rdap.restfulwhois.org/ip/218.241.0.0/30

Click here for the response body.

5. Query Autonomous System Number

URI: /autnum/<autonomous system number>
Example: http://rdap.restfulwhois.org/autnum/1

Click here for the response body.

6. Query Help

URI: /help
Example: http://rdap.restfulwhois.org/help

Click here for the response body.

7. Search Domain

URI:
/domains?name=<domain search pattern>
/domains?nsLdhName=<ns ldhName search pattern>
/domains?nsIp=<ns IP>
Example:
http://rdap.restfulwhois.org/domains?name=c*
http://rdap.restfulwhois.org/domains?nsLdhName=ns1.host*.cn
http://rdap.restfulwhois.org/domains?nsIp=218.241.111.96

Click here for the response body.

8. Search Nameserver

URI: /nameservers?name=<nameserver search pattern>
URI: /nameservers?ip=<IP network search pattern>

Example:
http://rdap.restfulwhois.org/nameservers?name=n*cn
http://rdap.restfulwhois.org/nameservers?ip=218.241.111.96

Click here for the response body.

9. Search Entity

URI: /entities?fn=<entity name search pattern>
URI: /entities?handle=<entity handle search pattern>

Example:
http://rdap.restfulwhois.org/entities?fn=Jo*n
http://rdap.restfulwhois.org/entities?handle=et*

Click here for the response body.

5. Update API

The RESTful WHOIS data can be updated by update API.

- All Update API prefix: /u/
- Content type must be 'application/rdap+json' or 'application/json', in 'Content-Type' header.
- URI and parameters must be encoded in UTF-8.
- Unknown parameters will be ignored.
- Security consideration: Update API supports IP authentication. Only the IP in the white list is allowed to be requested.
- Request and Response body is in JSON format.
- About 'handle': only contains ASCII chars or "- _".
- Max length of columns: for 'handle' value is 100, all others are 255 if not specified in the following tables.

5.1. Common Request Format

1. Create

- HTTP METHOD: POST
- URI: /u/{objectType}
  objectType: domain, nameserver, ip, autnum, entity
- CONTENT TYPE: 'application/rdap+json' or 'application/json'
- BODY: JSON formatted key-value parameters.

2. Update

- HTTP METHOD: PUT
- URI: /u/{objectType}/{handle}
  handle: object handle
- CONTENT TYPE: 'application/rdap+json' or 'application/json'
- BODY: the same with 'create'

3. Delete

- HTTP METHOD: DELETE
- URI: /u/{objectType}/{handle}
  handle: object handle

5.2. Common Response Format

HTTP status code	Service code	Body	Description
200		{"handle":"xxx"}	Success response
not 200		{"handle":"domain-1", "errorCode":400, "subErrorCode":4002, "description":["Property can’t be empty:ldhName"]}	Failure response

5.3. Response Code

HTTP status code	Service code	Description
200		Success response
400	4001	Request data is not valid JSON or has invalid date type
400	4002	Property can't be empty
400	4003	Property exceed max length
400	4007	Property must be valid date
400	4008	Property value is not valid
400	4009	Unrecognized request URI
400	40010	Property value must between [start, end]

403	4031	Forbidden
404	4041	Object not found with handle
409	4091	Object already exist for handle
405		Method not allowed
415		Unsupported media type
500		Internal server error

5.4. Common Request Body Parameter

Request body parameters are used for CREATE and UPDATE request.

5.1.1. Common Parameter

All update API can have these parameters:

Name	**Type**	**Length/ Range**	**Not empty**	**Description**
handle	string	1-100	Y	Registry-unique identifiers of a referenced object. Should be ASCII and '-'/'_'.
entities	array		N	Arrays of inner-entity object
status	array		N	Status array, each status length must be [0-20]. For example: [“validated”,”redacted”]
remarks	array		N	Arrays of remark object
links	array		N	Arrays of link object

events	array		N	Arrays of event object
lang	string	0-64	N	Language Identifier, for example: "en"
port43	string	0-4096	N	Port 43 WHOIS Server
customProperties	object		N	For example: {"customKey1":"value1","customKey2":"value2" }

1. Common Inner-object

Name	Type	Length/Range	Not empty	Description
handle	string	1-100	Y	Object handle. Non-exist handle will be ignored.

2. Inner-entity

Name	Type	Length/Range	Not empty	Description
handle	string	1-100	Y	Entity handle. Non-exist handle will be ignored.
roles	array		N	For example: [“registrant”, “administror”]

3. “remark” or “notice”

Name	Type	Not empty	Description
title	string	N	Title of the object
description	array	N	Each description length must be [0-2048]
links	array	N	Arrays of link object

4. “link”

Name	Type	Not empty	Description
value	string	N	[0-2048]. For example: "http://example.com/context_uri"
rel	string	N	For example: "self"
href	string	N	For example: "http://example.com/target_uri"
hreflang	array	N	For example: ["en", "ch"]
title	string	N	http://tools.ietf.org/html/rfc5988#section-5
media	string	N	For example: "screen"
type	string	N	For example: "application/json"

5. “publicId”

Name	Type	Not empty	Description
type	string	N	A string denoting the type of public identifier
identifier	string	N	A public identifier of the type denoted by 'type'

6. ”event”

Name	Type	Not empty	Description
eventAction	string	Y	A string denoting the reason for the event
eventActor	string	N	Denoting the actor responsible for the

			event
eventDate	string	Y	UTC date time. Format example: 2015-01-01T01:01:01Z
links	array	N	Arrays of link object

5.1.2. Domain

Name	Type	Not empty	Description
ldhName	string	Y	Puny name of domain. Can't contain last '.' of domain. Must be lowercased.
unicodeName	string	N	[0,1024]. Unicode name of domain. If is ASCII domain then it is the same with ldhName
variants	array	N	Array of variant object
nameservers	array	N	Arrays of inner-object object
secureDNS	object	N	secureDNS object
publicIds	array	N	Arrays of publicId, for example: [{"type":"IANA Registrar ID", "identifier":"1"}]
type	string	N	"dnr" for DNR domain, or "arpa" for ARPA domain
networkHandle	string	N	Network handle for ARPA domain. This value will be ignored if network not exist.

1. ”variant”

Name	Type	Not empty	Description
relation	array	N	Array of relation string. for example: ["registered", "conjoined"]
idnTable	string	N	The name of the IDN table of code points
variantNames	array	N	Array of variant name object

2. ”variantName”

Name	Type	Not empty	Description
ldhName	string	N	Variant's ldhName. Can't contain last '.' of domain. Must be lowercased.
unicodeName	string	N	Variant's Unicode Name

3. ”secureDNS”

Name	Type	Not empty	Description
zoneSigned	boolean	N	True if the zone has been signed, false otherwise.
delegationSigned	boolean	N	Boolean true if there are DS records in the parent, false otherwise.
maxSigLife	int	N	The signature life time in seconds will be used when creating the RRSIG DS record
dsData	array	N	Array of dsData object
keyData	array	N	Array of keyData object

4. ”dsData”

Name	Type	Not empty	Description
keyTag	int	Y	The key tag field of a DNS DS record as specified by RFC 4034
algorithm	int	Y	The algorithm field of a DNS DS record as described by RFC 4034
digest	string	Y	[0-2048]. The digest field of a DNS DS record as specified by RFC 4034
digestType	int	Y	The digest type field of a DNS DS record as specified by RFC 4034
links	array	N	Arrays of link object
events	array	N	Arrays of event object

5. ”keyData”

Name	Type	Not empty	Description
flags	int	Y	The flags field value in the DNSKEY record as specified by RFC 4034
protocol	int	Y	The protocol field value of the DNSKEY record as specified by RFC 4034
publicKey	string	N	The public key in the DNSKEY record as specified by RFC 4034
algorithm	int	Y	The algorithm field of a DNSKEY record as specified by RFC 4034
links	array	N	Arrays of link object

events	array	N	Arrays of event object

5.1.3. Nameserver

Name	Type	Not empty	Description
ldhName	string	Y	Puny name of nameserver. Can't contain last '.' of domain. Must be lowercased.
unicodeName	string	N	[0,1024]. If is ASCII nameserver then it is the same with ldhName
ipAddresses	object	N	ipAddresses object

1. ”ipAddresses”

Name	Type	Not empty	Description
ipList	array	N	Arrays of IP. IP can be v4 or v6. For example: [“218.1.1.1”, “2001:db8::”]

5.1.4. Entity

Name	Type	Length/Range	Not empty	Description
fn	string		Y	Entity name
kind	string		N	http://tools.ietf.org/html/rfc6350#section-6.1.4
email	string		N	Email

title	string		N	http://tools.ietf.org/html/rfc6350#section-6.6.1
org	string		N	org
url	string	0-4096	N	http://tools.ietf.org/html/rfc6350#section-6.7.8
addresses	array		N	Array of address object
telephones	array		N	Array of telephone telephones
publicIds	array		N	The same with domain

1. ”address”

Name	Type	Not empty	Description
pref	string	N	http://tools.ietf.org/html/rfc6350#section-5.3
types	string	N	Multiple type strings separated by ';'.http://tools.ietf.org/html/rfc6350#section-5.6
postbox	string	N	Postbox
extendedAddress	string	N	The extended address
streetAddress	string	N	Street address
locality	string	N	The locality. For example: city
Region	string	N	The region. For example:state or province
Postalcode	string	N	The postal code
Country	string	N	The country name

2. ”telephone”

Name	Type	Not empty	Description
pref	string	N	http://tools.ietf.org/html/rfc6350#section-5.3
types	string	N	String of type for multiple http://tools.ietf.org/html/rfc6350#section-5.6, separated by';'
number	string	Y	telephone number
extNumber	string	N	extended number

5.1.5. Network

Name	Type	Not empty	Description
startAddress	string	Y	The starting number in the block of network
endAddress	string	Y	The ending number in the block of network
ipVersion	string	N	'v4' or 'v6'. This value will not affect the real type for startAddress and endAddress.
name	string	N	An identifier assigned to the network registration by the registration holder
type	string	N	A string containing an RIR-specific classification of the network
country	string	N	A string containing the two-character country code of the network

parentHandle	string	N	Parent network of this network registration
cidr	string	Y	Formatted network used to generate self link for query. http://tools.ietf.org/html/rfc4632. For example: 92.168.99.0/24

5.1.6. As Number

Name	Type	Not empty	Description
startAutnum	string	Y	The starting number in the block of autonomous system numbers
endAddress	string	Y	The ending number in the block of autonomous system numbers
name	string	N	An identifier assigned to the autnum registration by the registration holder
type	string	N	A string containing an RIR-specific classification of the autnum
country	string	N	A string containing the name of the 2 character country code of the autnum

6. Proxy43

- Only support JWHOIS client
- Response is plain text

Usage:

```
whois [type] [parameter]
    type: query type. Values are: ""(empty), nameserver, entity, as, domains, nameservers, entities.
    parameter: query parameter.
```

6.1. API

1. Query IP

```
whois {IP}
    type: empty
    parameter: IP address
```

examples:

```
    whois 218.241.111.44
    whois 218.241.111.44/8
    whois 3000:0DB8:0000:0000:0000:0000:1428:0000/128
```

2. Query Domain

```
whois {domain name}
    type: empty
    parameter: domain name, not IP formatted
```

examples:

```
    whois cnnic.cn
```

3. Search Domain

```
whois domains {domain name search pattern}
    type: domains
    parameter: domain name search pattern
```

examples:

```
    whois domains cnnic*.cn
```

4. Search Domain by Nameserver

```
whois domains nsLdhName={nameserver name search pattern}
    type: domains
    parameter: nameserver name search pattern
```

examples:

```
    whois domains nsLdhName=ns.cnnic*.cn
```

5. Search Domain by IP

```
whois domains nsIp={IP}
    type: domains
    parameter: IP of domain's nameserver
```

examples:

```
    whois domains nsIp=218.241.111.44
```

6. Query Nameserver

```
whois nameserver {nameserver name}
    type: nameserver
    parameter: nameserver name
```

examples:

```
    whois nameserver ns.cnnic.cn
```

7. Search Nameserver by Name

```
whois nameservers {nameserver name search pattern}
    type: nameservers
    parameter: nameserver name search pattern
```

examples:

```
    whois nameservers ns*.cn
```

8. Search Nameserver by IP

```
whois nameservers ip={IP of nameserver}
    type: nameservers
    parameter: IP of nameserver
```

examples:

```
    whois nameservers ip=218.241.111.44
```

9. Query As Number

```
whois as {as number}
    type: as
    parameter: as number
```

examples:

```
    whois as 2345
```

10. Query Entity

```
whois entity {entity handle}
    type: entity
    parameter: entity handle
```

examples:

```
    whois entity handle_of_an_entity
```

11. Search Entity by Entity Name

```
whois entities fn={entity name search pattern}
    type: entities
    parameter: entity name search pattern
```

examples:

```
whois entities fn=John*
```

12. Search Entity by Entity Handle

```
whois entities fn={entity handle search pattern}
    type: entities
    parameter: entity handle search pattern
```

examples:

```
whois entities handle=handle_of_John*
```

7. Customization and Development

7.1. Use Registry's Database

Registry can modify code to use its own database and schema, instead of updating date from API periodically.

Steps:

1. If database is not MYSQL, you should change database driver:
    - Modify pom.xml, remove MYSQL dependency. Then add your database dependency.
    - Modify database configuration
2. Modify JAVA code of DAO implementation.

    Modify all DAO implementation JAVA class, such as DomainQueryDaoImpl:

    - Change DAO implementation according to your own database schema.
    - If you need more properties than default, you can add your custom properties .

7.2. Function Customization

RESTful WHOIS server supports all 6 query functions and 3 search functions defined in RFC4782.

- IP network query
- autonomous system number query
- domain query

- nameserver query
- entity query
- help query
- domain search
- nameserver search
- entity search

You can disable some of these functions by adding the function URI to 'notImplementedUri' property in rdap.properties.

7.3. Validator Customization

Query/search parameter are validated before query.
You can modify validation logic by add/remove/modify validators.
All validators extend from Validator.java, such as DomainNameValidator.

7.4. Enable/Disable Access Control

Access Control is done in AccessControlQueryFilter.
You can enable/disable access control by configuring 'accessControlQueryFilter' in queryFilter.

7.5. Enable/Disable Redirect

Redirect is done in *RedirectQueryFilter.
You can enable/disable redirect by configuring these filters in queryFilter.

7.6. Add Custom Feature

You can add custom features by adding queryFilter.

7.7. VCARD Extension

We use Jcard to convert VCARD to JSON.

Jcard convert property by property, using JcardPropertyConverter.
If you need to show more vcard information, you can add your custom converter:

- Add your own implementation, extending JcardPropertyConverter.
- Register this class to Jcard by adding it to converters in Jcard.java.

8. Other

Other information can be found in project wiki: https://github.com/cnnic/rdap/wiki.