RDAP Reference Manual

For

Registry

Reference Manual for Version 0.3

CNNIC

Table of Contents

[1 Introduction 2](#_Toc427655683)

[1.1 RDAP Modules 2](#_Toc427655684)

[1.1.1 RDAP-service 2](#_Toc427655685)

[1.1.2 RDAP-proxy43 3](#_Toc427655686)

[1.1.3 Registry Data Integration 3](#_Toc427655687)

[2 Install 3](#_Toc427655688)

[2.1 Supported Operating Systems 3](#_Toc427655689)

[2.2 RDAP-service 3](#_Toc427655690)

[2.3 RDAP-proxy43 5](#_Toc427655691)

[3 Registry Data Integration 6](#_Toc427655692)

[3.1 Update API for default MySQL database 6](#_Toc427655693)

[3.1.1 Introduction 6](#_Toc427655695)

[3.1.2 Common Request Format 6](#_Toc427655696)

[3.1.2.1 Create 6](#_Toc427655697)

[3.1.2.2 Update 7](#_Toc427655698)

[3.1.2.3 Delete 7](#_Toc427655699)

[3.1.3 Common Response Format 7](#_Toc427655700)

[3.1.4 Response Code 7](#_Toc427655701)

[3.1.5 Request Body Parameters 8](#_Toc427655702)

[3.1.5.1 Common Parameter 8](#_Toc427655703)

[3.1.5.2 domain 11](#_Toc427655704)

[3.1.5.3 nameserver 15](#_Toc427655705)

[3.1.5.4 entity 15](#_Toc427655706)

[3.1.5.5 network 17](#_Toc427655707)

[3.1.5.6 as number 18](#_Toc427655708)

[3.2 Use Registry’s Database 18](#_Toc427655710)

[4 Customization and Development 22](#_Toc427655711)

[4.1 Function Customization 22](#_Toc427655713)

[4.2 Add custom properties values 22](#_Toc427655714)

[4.3 Validator Customization 23](#_Toc427655715)

[4.4 Enable/Disable Access Control 23](#_Toc427655716)

[4.5 Enable/Disable Redirect 24](#_Toc427655717)

[4.6 Add Custom Features 25](#_Toc427655718)

[4.7 VCARD Extension 25](#_Toc427655719)

[5 Other 26](#_Toc427655720)

**Document Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason for Changes** | **Version** |
| Dan Zhou | 2015-7-31 | Creation | 0.1 |
| Dan Zhou | 2015-8-11 | Adjustments according to review comments | 0.2 |
| Dan Zhou | 2015-8-17 | Details modification | 0.3 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Introduction

This project is a starting point for registries to build restful WHOIS service so that they need not to start from scratch.

RDAP: Registration Data Access Protocol. The term ‘RDAP’ will be used as the project name in following sections.

## RDAP Modules

The project is written in JAVA. Following figure shows the Modules:

Figure 1.1-1 RDAP Modules

### RDAP-service

RDAP-service is a HTTP service, which 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 can query WHOIS data from this API. Please refer to [Introduction on how to use Query API](https://github.com/cnnic/rdap/wiki/Query-Api) for detailed information.

1. Update API

Registry can update WHOIS data from this API.

### RDAP-proxy43

It is a TCP 43 port service. Please refer to [Introduction on how to use Proxy43 API](https://github.com/cnnic/rdap/wiki/Proxy43-Api) for detailed information.

### Registry Data Integration

There are two methods for registries to provide WHOIS data:

1. Use default MySQL database

It needs update RDAP service data via data update API from registry’s own database. Please see section 3.1 for more information.

1. Use registry’s own database

It needs provide specified database SQL statement and modify the data access module. Please see section 3.2 for more information.

# Install

## Supported Operating Systems

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.

## RDAP-service

1. Install [JDK7](http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html), or higher version. (Skip this step if already installed)
2. Install [Mysql5](http://dev.mysql.com/downloads/mysql), or higher version. (Skip this step if already installed)

Create user 'rdap' and grant privilege: $RDAP\_SERVER\_IP must be changed to rdap server IP, and $MYSQL\_PASSWORD be changed to 'rdap' user's password.

GRANT ALL PRIVILEGES ON \*.\* TO 'rdap'@'$RDAP\_SERVER\_IP' IDENTIFIED BY '$MYSQL\_PASSWORD';

FLUSH PRIVILEGES;

More information please refers to [MySQL GRANT Syntax](http://dev.mysql.com/doc/refman/5.1/en/grant.html).

1. Install [Tomcat7](http://tomcat.apache.org/download-70.cgi), or higher version. (Skip this step if already installed)

HTTP port use default port 8080. If you want to use other ports, please refer to [How to use Tomcat](http://tomcat.apache.org/tomcat-7.0-doc/RUNNING.txt). Installed Tomcat root folder called '$TOMCAT\_HOME', which contains folders: bin, conf, lib, webapps, etc.

1. Get RDAP war file

There are two methods to get RDAP war file:

* Get [war file](https://github.com/cnnic/rdap/raw/dev/rdap-service/build/rdap-service-1.0.war) building by JDK7
* [Build war file from source](https://github.com/cnnic/rdap/wiki/%5Binstall%5DBuild-war-file-from-source)

1. Deploy RDAP war to tomcat

* Create folder 'rdap' in dir $TOMCAT\_HOME/webapps/
* Unzip RDAP war file to $TOMCAT\_HOME/webapps/rdap/
* Edit database configuration file [jdbc.properties](https://github.com/cnnic/rdap/wiki/jdbc.properties)
* Edit global configuration file [rdap.properties](https://github.com/cnnic/rdap/wiki/rdap.properties)

1. Init database

Use database info in jdbc.properties you have configured in previous step, and create database named 'rdap', you can insert test data into 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

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

1. 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 dir

bin/startup.sh

[in Windows, open command prompt window and execute command:]

cd $TOMCAT\_HOME/bin #$TOMCAT\_HOME must be replaced by real dir

startup.bat

* Test if it is ok

curl -H Accept:application/rdap+json http://$RDAP\_SERVER\_IP:$RDAP\_SERVER\_PORT/rdap/autnum/2100

It's ok if response contains 'rdapConformance'.

## RDAP-proxy43

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

There are two methods to get this jar file:

* Get [jar file](https://github.com/cnnic/rdap/raw/dev/rdap-proxy43/build/rdap-proxy43-jar-with-dependencies.jar) building by JDK7.
* [Build jar file from source](https://github.com/cnnic/rdap/wiki/Proxy43-install:build-from-source)

1. Copy rdap-proxy43-jar-with-dependencies.jar to proxy43 install directory, and we call it $PROXY43\_INSTALL\_DIR.
2. Download configuration file ‘[proxy43.properties](https://raw.githubusercontent.com/cnnic/rdap/dev/rdap-proxy43/src/main/resources/proxy43.properties)’, and copy it to $PROXY43\_INSTALL\_DIR.
3. You can edit ‘proxy43.properties’ for production use, please refer to [How to configure Proxy43](https://github.com/cnnic/rdap/wiki/proxy43.properties).
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 dir

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 dir

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 dir

java -jar rdap-proxy43-jar-with-dependencies.jar shutdown

# Registry Data Integration

## Update API for default MySQL database

If registry uses default MySQL database, it needs update RDAP service data via data update API from registry’s own database.



### Introduction

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

### Common Request Format

#### 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

#### Update

* HTTP METHOD: PUT
* URI: /u/{objectType}/{handle}

handle: object handle

* CONTENT TYPE: same with 'create'
* BODY: same with 'create'

#### Delete

* HTTP METHOD: DELETE
* URI: /u/{objectType}/{handle}

handle: object handle

### 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 |

### 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. |

### Request Body Parameters

Request body parameters are used for CREATE and UPDATE request.

#### 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 | An array of inner-entity objects. |
| status | array | 0-20 | N | An array of status, each status length must be [0-20]. e.g: [“validated”,”redacted”]. |
| remarks | array |  | N | An array of remark objects. |
| links | array |  | N | An array of link objects. |
| events | array |  | N | An array of event objects. |
| lang | string | 0-64 | N | Language Identifier, e.g: "en". |
| port43 | string | 0-4096 | N | Port 43 WHOIS Server. |
| customProperties | object |  | N | e.g: {"customKey1":"value1","customKey2":"value2"} |

##### inner-object

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

##### 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 | e.g: [“registrant”, “administrative”] |

##### 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 | An array of link objects. |

##### link

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

##### publicId

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Not empty** | **Description** |
| type | string | N | A string denoting type of the public identifier. |
| identifier | string | N | A public identifier of the type denoted by “type”. |

##### event

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Not empty** | **Description** |
| eventAction | string | Y | A string denoting the reason for the event. |
| eventActor | string | N | A string denoting the actor responsible for the event. |
| eventDate | string | Y | UTC date time. Format example: 2015-01-01T01:01:01Z |
| links | array | N | An array of link objects. |

#### 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 | An array of variant objects. |
| nameservers | array | N | An array of nameserver objects. |
| secureDNS | object | N | secureDNS object. |
| publicIds | array | N | An array of publicId objects, e.g: [{"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. It will be ignored if network does not exist. |

##### variant

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Not empty** | **Description** |
| relation | array | N | An array of relation strings. e.g: ["registered", "conjoined"] |
| idnTable | string | N | Name of the IDN table of code points. |
| variantNames | array | N | An array of variant name objects. |

##### 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. |

##### secureDNS

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Not empty** | **Description** |
| zoneSigned | boolean | N | True if the zone has been signed, false otherwise. |
| delegationSigned | boolean | N | 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 | An array of dsData objects. |
| keyData | array | N | An array of keyData objects. |

##### 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 field of a DNS DS record as specified by [RFC 4034]. |
| links | array | N | An array of link objects. |
| events | array | N | An array of event objects. |

##### 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 | An array of link objects. |
| events | array | N | An array of event objects. |

#### 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. |

##### ipAddresses

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Not empty** | **Description** |
| ipList | array | N | An array of IP. IP can be v4 or v6. e.g: [“218.1.1.1”, “2001:db8::”] |

#### 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 | An array of address objects. |
| telephones | array |  | N | An array of telephone objects. |
| publicIds | array |  | N | The same with domain. |

##### 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. e.g: city. |
| region | string | N | The region. e.g:state or province. |
| postalcode | string | N | The postal code. |
| country | string | N | The country name. |

##### 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. |

#### 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>. e.g: 92.168.99.0/24 |

#### 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 two-character country code of the autnum. |

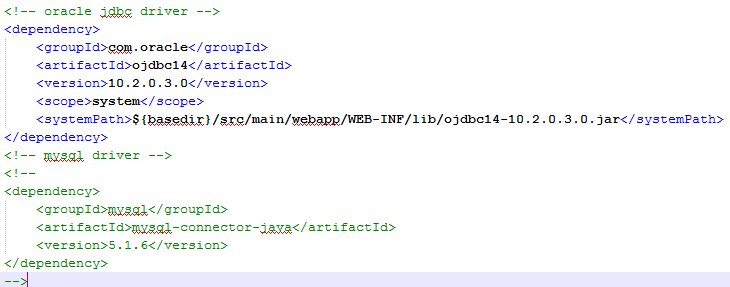


## Use Registry’s Database

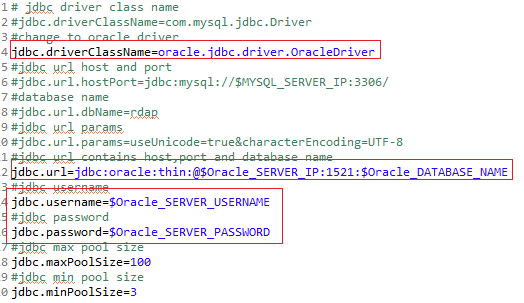
Instead of updating data from update API periodically, a registry can use its own database. It is easy to implement local database by providing specified database SQL statement and modifying the data access module. Take Oracle as an example, following are the steps:

1. Change database driver

* Modify pom.xml, remove MySQL dependency. Then add the Oracle database dependency.



* Modify database configuration in jdbc.properties



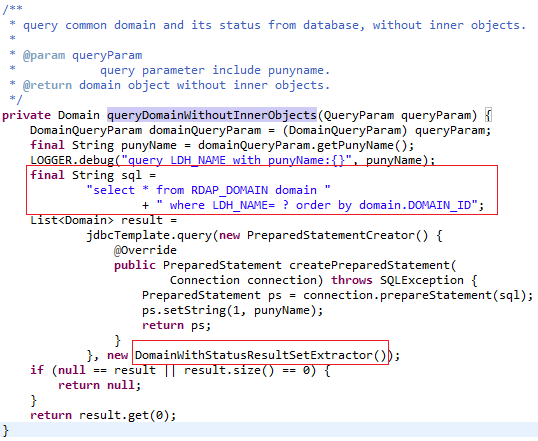
* Based on the modification in the previous step, you need to modify data source configuration in spring-serviceContext.xml.



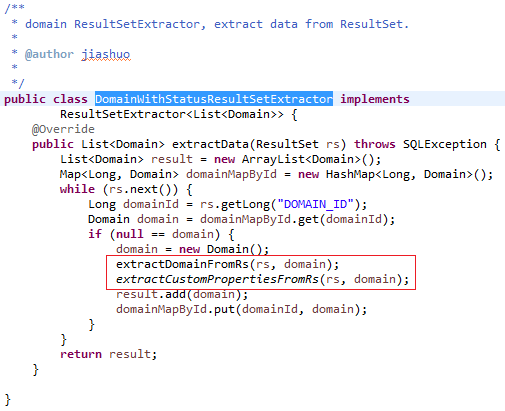
1. Modify DAO implementation

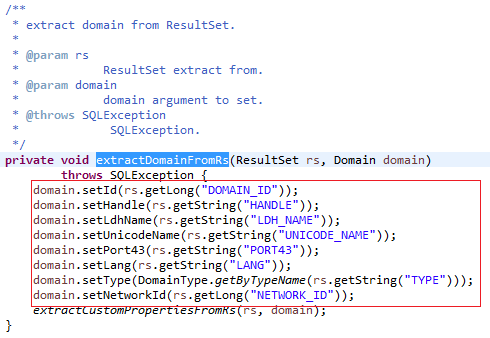
Change DAO implementation based on your own database schema. Take DomainQueryDaoImpl.java as an example, following are the steps:

* Modify the SQL statement to customize the query.



* Modify the implementation of extracting data from ResultSet to the domain object.





# Customization and Development

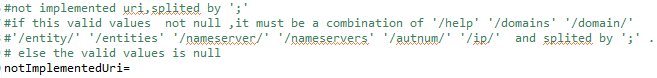


## Function Customization

RDAP server supports all 6 query functions and 3 search functions defined in draft-ietf-weirds-rdap-query-15.

1. IP network query
2. autonomous system number query
3. domain query
4. nameserver query
5. entity query
6. help query
7. domain search
8. nameserver search
9. entity search

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

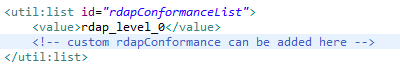


## Add custom properties values

If you need more properties than default, you can add your own custom properties (key-value pairs) to query response.

All models, such as domain, are extending from BaseModel class. The BaseModel has a MAP type field ‘customProperties’, which is used for custom key-value. For example, if you want to add custom ‘createTime’ key-value to Domain model, you need to:

1. Add your own RDAP conformance entry to spring-initData-rdapConformance.xml, which will be displayed in 'rdapConformance' of query JSON response.



1. Modify 'customPropertyPrefix' configuration in rdap.properties which is the configuration item for prefix of custom columns. NIC name is a suggested prefix value.



**Note**: custom property's name SHOULD conform to: ALPHA \*( ALPHA / DIGIT / "\_" )

## Validator Customization

Query/search parameters are validated before query.

You can modify validation logic by add/remove/modify validators.

All validators extend from [Validator.java](https://github.com/cnnic/rdap/blob/master/rdap-service/src/main/java/org/restfulwhois/rdap/common/validation/Validator.java), such as [DomainNameValidator](https://github.com/cnnic/rdap/blob/master/rdap-service/src/main/java/org/restfulwhois/rdap/core/domain/validator/DomainNameValidator.java).

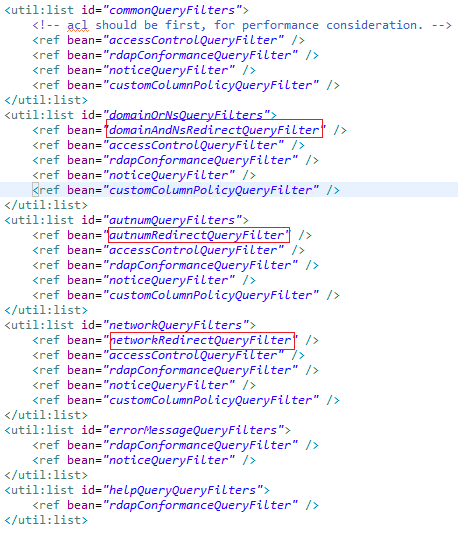
## Enable/Disable Access Control

Access control is done by AccessControlQueryFilter. You can enable/disable access control by adding/removing 'accessControlQueryFilter' in spring-queryFilter.xml.



## Enable/Disable Redirect

Redirect is done by \*RedirectQueryFilter. You can enable/disable redirect by adding/removing these filters in spring-queryFilter.xml.



## Add Custom Features

You can add custom features by adding [queryFilter](https://github.com/cnnic/rdap/wiki/query%20filter).

## VCARD Extension

[Jcard](https://github.com/cnnic/rdap/blob/dev/rdap-service/src/main/java/org/restfulwhois/rdap/core/entity/model/jcard/Jcard.java) is used to convert VCARD to JSON. [JcardPropertyConverter](https://github.com/cnnic/rdap/blob/master/rdap-service/src/main/java/org/restfulwhois/rdap/core/entity/model/jcard/JcardPropertyConverter.java) is used to convert a certain property of Entity, and set it to corresponding VCARD property. If you need to show more vcard information, you can add your custom converter:

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

# Other

Please refer to project wiki for further information: <https://github.com/cnnic/rdap/wiki>.