APPENDIX.D.5.1

Proposed RFC for SRRP

Stateless Registry-Registrar Protocol (SRRP) version 1.0

Abstract

The purpose of SRRP is to provide a stateless service for communications between the registrar and the registry. The design goal of the protocol is to provide a consistent service for a very high number of clients by not maintaining client state information on the server, and reduce the policy enforcements done by the protocol to a minimum.

The protocol describes the communications done between a registrar, normally acting on behalf of a registrant, and the registry. The registrar may preform operations such as creating comains, creating logical entities of name servers, assigning name servers to a domain, transferring a domain and querying a the server for information about name server entities, domains or the server itself.

Table of Contents

1.0 Introduction

The SRRP protocol is intended to fix shortcomings of the RRP protocol defined by NSI in RFC2832 by using a stateless "one shot" protocol model. The goals of the protocol is:

- Provide only the strictly required functionality
- Provide a completely stateless service
- Provide service to a very high number of concurrent clients
- Be implementation and performance friendly
- Provide complete idempotency to the client
- Share the complexity between the client and the server

2.0 Terminology

- The "request message" or "client request" is the message sendt from the client to the server, and consists of a one line "request header" and a multi line "request body".
- The "response message" or "server response" is always the response to a request message, and is sendt from the server to the client. It consists of a one line "response header" and possibly a "response body".
- "LWS", linear white space, is any combination of ASCII space (SP) and ASCII tabulator (TAB) characters.
- "CRLF" is one ASCII carriage return (CR) character followed by one ASCII line feed (LF) character.
- An "attribute/value pair" consists of a short textual string, termed "attribute", an ASCII '=' character, and another string, termed "value". The attribue/value pair is terminated by a CRLF sequence, and thus a line may only contain one attribute/value pair.
- The "client" is the registrars client software, and likewise the "server" is the registrys server software.
- An "object" is a set of attribute/value pairs that the server operates on. Currently there are two kinds of objects; domain objects and cluster objects.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [1].

3.0 Protocol model

The protocol is a simple one way client-server protocol using a textual format for easier debugging. A transaction is always initiated by the client, and the server must answer every valid request message with a response message containing a response code indicating the outcome of the client request. A well behaved client SHOULD wait for a response from the server before it issues a new request.

The messages should contain only printable ISO-8859-1 characters, ie. characters in the range 31-126 and 160-255, inclusive. Support for other characters sets or binary data are not supported in the current version of SRRP, but may be added later by using a character encoding.

The registrar is identified to the registry by an attribute/value pair in the request body, and authenticated by a similiar attribute/value pair. As the protocol does not itself provide any other security measures, the client MUST connect to the server using a secure, reliable communication method such as SSL [2] or an encrypted tunnel.

3.1 Protocol objects

The domain objects are a logical grouping of attribute/value pairs that are manipulated using SRRP.

3.1.1 Domain objects

The domain object is a collection of information defining a registered domain in the system. The domain object should contain the following attributes:

- Exactly one "registrar-id" attribute identifying the owner of the object.
- Exactly one "domain-name" attribute containing the name of the domain.
- Exactly one "expiry-date" attribute containing the expiry date for the registration. If the client does not specify a date, a system default should be used.
- Exactly one "status" attribute defining the current status of the object. This should be set to a system default if not specified by the client.
- Exactly one "domain-auth" attribute containing a registrar assigned password for this domain. This password should be used to authorize a domain transfer.
- Exactly one "cluster-id" attribute identifying a cluster object for this domain object.

3.1.2 Cluster objects *** er uklar ***

The cluster object is a collection of name server information. Both the name and the address of the name server is stored for every name server in the cluster. The name servers are stored in attributes starting with "nsi-" where i is any positive integer starting with one (1), possibly limited by the server. For instance, the first name server in a cluster object will have its IP address in the attribute "ns1-address" and its name in the attribute "ns1-name". This pair is termed the "name server entry".

The client should store the name servers in increments of one, as the server MAY choose to stop looking for name servers when it finds an empty name server entry.

The cluster object consists of any number of name server entries starting with "nsi-" where i is a positive integer starting with one (1) and increasing with increments of one (1) for every name server entry.

3.2 Request message format

The request message issued by a consists of a header line containing the command to performed, a command argument and the version number of the protocol. These fields are separated by one or more LWS characters, and the header line is terminated by one CRLF character sequence.

Following the header line, the client may add one or more lines of attribute/value pairs, the request body. While the protocol does not require the client to issue any attribue/value pairs, the authetication credentials are specified using attribute/value pairs in the request body, and these are required by every command currently specified. The order of the attribute/value pairs in the request body is arbitrary.

The request message is terminated by the ASCII end of file (EOF) character, and the server MUST disconnect from the client whenever it encounters EOF.

Example request message:

```
  ADD john.doe.name SRRP/1.0
  registrar-id=442885225
  registrar-auth=fo=43Ga axy
  domain-name=johnny.walker.name
  status=inactive
  cluster-id=752095231
```

Please note the usage of '=' and space characters in the registrar-auth attribute value. This is legal because there must only be exactly one attribute/value pair on every line, and everything from the first '=' up to the CRLF is considered part of the attribute value.

3.3 Response format

For every valid request message received from a client, the server MUST issue a response message starting with a one line header containing a valid response code and a short description of the response code, separated by one or more LWS characters and terminated by a CRLF sequence.

If the client request was completed successfully and the server needs to send additional information in the response message, it must send this information in one or more lines of attribute/value pairs contained in the response body. The response body is terminated by and EOF character, also marking the end of the response message. If the command failed, ie. the response code is a 3XX (Temporary failure) or 4XX (Permanent failure), the server MAY add one or more "text" attributes in the response body further describing the error condition.

The response body for a successful command MUST contain only attributes defined for that particular command. The order of the attributes in the response body is arbitrary with one exception: the order of the special "text" attribute is important as these are used for human readable data. The server MUST send the "text" attributes in the order they are stored or retrieved, and likewise the client MUST read them in the order received.

Example response message for a QUERY CLUSTER command:

```
  200 Command completed successfully
  ns1-address=192.168.4.5
  ns1-name=ns1.example.com
  ns2-address=192.168.4.6
  ns1-name=ns2.example.com
  ns3-address=10.10.56.11
  ns1-name=ns1.example.net
```

The response code of 200 indicates that the command did complete successfully, and the response body contains the data returned from the command, which is a set of attribute/value pairs.

Example response message for a QUERY DOMAIN command:

```
  200 Command completed successfully
  domain-name=example.com
  registrar-id=123456789
  expiry-date=2003-02-09
  created-date=2001-02-09
  cluster-id=987654321
  status=active
  text=Last-change: ADD DOMAIN
  text=Changed-date:2001-02-13 10:15:12 UTC
  text=Changed-by: registrar 123456789
```

This is a more complex response, containing both normal attributes and ordered "text" attributes. If the domain did not exist, the response would be a 401 Domain not registered, possibly with one or more "text" attributes giving a human readable explaination of the error.

3.4 Client requirements

The client MUST NOT make any assumptions about the length of the value pairs. The ordering of the attributes is irrelevant except for the "text" attribute where the client MUST keep the order.

3.5 Server requirements

The server SHOULD issue a response message to every well formed client request message. A client request message is considered well formed when it contains an initial header line consisting of three fields separated by one or more LWS characters, and the last value is recognized as a SRRP protocol version number. If the client request message is not well formed, the server MUST drop the connection immediately.

The server MUST answer the client request with a response message using the same version of SRRP as the client request message. If the server is unable to answer the request using the same protocol version as the client, it must issue a 413 Unsupported protocol version message.

4.0 SRRP commands

This section contains the commands defined for use in client request

messages and their expected response. All of these messages MUST contain a "registrar-id" attribute identifying the registrar issuing the command, and a "registrar-auth" authenticating the registrar. Clients may only view and/or change their own objects, and attempts to operate objects belonging to other registrars should result in a 411 Access denied error message.

Note that the ordering of the attribute/value pairs is not significant except for the "text" attributes.

4.1 CREATE

The create commands are used for adding an object to the registry. In the current release of SRRP, the "domain" and "cluster" object types are supported, containing a domain registration and a series of name server registrations, respectively.

4.1.1 CREATE DOMAIN

The CREATE DOMAIN command attempts to register the domain name contained in the "domain-name" attribute in the request body.

The request body MAY also contain any of the following attributes:

- Exactly one "expire-date" attribute giving the requested expiration date of the registration. For further description, see 4.2.1.
- Exactly one "cluster-id" attribute pointing to a cluster object containing the name servers for this domain. For further description, see 4.2.2.
- Exactly one "status" attribute giving the current status of the domain. For further description, see 4.2.3.
- Exactly one "domain-auth" attribute containing a registrar assigned password for this domain. For further description, see 4.2.5 and 4.8.1.
- Zero or more (possibly server limited) name server entries each consisting of the attributes "nsi-address" and "nsi-name" where i is a positive integer. For further description, see 3.1.2.

If the user specifies any name server entries, the server must attempt to create a cluster object for these. If successful, it MUST return the following attribute/value pairs:

- Exactly one "cluster-id" attribute containing the cluster ID of the newly created cluster object. The client must store this value as it is the only way of keeping track of the cluster object.
- Exactly one "expire-date" attribute containing the expire date for the domain.
- Exactly one "status" attribute containing the status of the domain.

Note that the server may limit the minimum and/or maximum number of nameservers the user is allowed to specify. The server should notice the client of any limitations on the number of name servers in the STATUS DEFAULTS response body.

If the client specifies both a "cluster-id" attribute and any number of name server entries, the server SHOULD ignore the name server entries and use the cluster ID.

If the cluster ID does not exist in the system, the response message should be a 402 Cluster not registered. If the expiration date is invalid, the response message should be a 405 Invalid expire date. If the "status" attribute contains an unknown status value, the response message should be a 404 Invalid attribute value. If the client specified too few or too many name servers, the server should respond with a 406 Invalid number of name servers error message. If the client attempts to register a domain which is blacklisted, the server should issue a 409 Blucked domain error message. If the client does not have the neccesary credit to register a domain, the response message should be a 410 Credit failure error.

Examples:

```
CREATE DOMAIN SRRP/1.0
registrar-id=123456789
registrar-auth=pass-phrase
domain-auth=domain-pass-phrase
cluster-id=987654321
domain-name=example.com
```

In this example, the registrar 123456789 adds the domain example.com using the default expiry date and status and a pre-defined cluster object.

```
CREATE DOMAIN SRRP/1.0
registrar-id=123456789
registrar-auth=pass-phrase
domain-name=example.com
domain-auth=domain-pass-phrase
ns1-address=114.72.91.131
ns1-name=ns1.example.com
ns2-address=114.72.91.132
ns2-name=ns2.example.com
```

Here, the registrar specifies two name servers in the request body. If the number of name servers (two) is valid, the response might look like this:

```
200 Command completed successfully
cluster-id=987654321
expire-date=2004-03-12
status=active
```

The client would now own the cluster object identified by 987654321 containing the two name servers ns1.example.com and ns2.example.com and their IP addresses.

4.1.2 CREATE CLUSTER

Cluster objects for name servers may be added by using the CREATE CLUSTER command. A number of name server entries each consisting of the attributes "nsi-address" and "nsi-name" where i is a positive integer. The minimum and/or maximum number of name server entries may be limited by the server, and the server should show these limits in the STATUS DEFAULTS response body. For further description of name server entries, see 3.1.2.

The server must create a cluster object for this client, and respond with a "cluster-id" attribute in the response body containing the ID of the newly created cluster object. The client must store the cluster ID as this is the only way of keeping track of the cluster object.

If the client specified too few or too many name servers, the server should respond with a 406 Invalid number of name servers error message.

Example:

```
  CREATE CLUSTER SRRP/1.0
  registrar-id=123456789
  registrar-auth=pass-phrase
  ns1-address=114.72.91.131
  ns1-name=ns1.example.com
  ns2-address=114.72.91.132
  ns2-name=ns2.example.com
  ns3-address=114.72.91.133
  ns3-name=ns3.example.com
```

A typical response message would look like this:

```
  200 Command completed successfully
  cluster-id=987654321
```

The cluster identified by the cluster ID 987654321 is now assigned to the registrar identified by the registrar ID 123456789, and contains three name servers.

4.2 SET

The SET command is functionally equivalent to the ADD command, except for that it wil overwrite any previous data contained in the attribute.

4.2.1 SET EXPIRE

If not specified, the expire date is set to a system default time, ie. a year after the registration was performed. However, the registrar may change the expire date himself by issuing an SET EXPIRE command with the domain in the "domain-name" attribute and the requested expiry date in the "expire-date" attribute. The previous expiry date of the domain object will be overwritten by the new one.

The value of the "expire-date" attribute should be the year month and day of the requested registration expire date, specified with a four digit year number, a two digit month number and a two digit day number, separated with ASCII '-' characters. The client MUST sppecify the expiry date in UTC (Universal Time Coordinated).

The system may have an upper limit of the length of an registration, and if the registrar attempts to set an expire date past this boundary, the server must respond with a 405 Invalid expire date error message.

Example:

```
  SET EXPIRE SRRP/1.0
  registrar-id=123456789
```

```
  registrar-auth=pass-phrase
  expire-date=2007-06-04
  domain-name=example.com
```

This will set the expire date of the domain lo.wang.name to June 2007.

4.2.2 SET CLUSTER

The SET CLUSTER combination will specify a cluster of nameservers, identified by the "cluster-id" attribute in the request body, for the domain object specified by the "domain-name" attribute.

If the domain object is unknown, the server must respond with a 401 Domain not registered error message. If the cluster object is unknown, the server must respond with a 402 Cluster not registered error message.

Example:

```
  SET CLUSTER SRRP/1.0
  registrar-id=123456789
  registrar-auth=pass-phrase
  cluster-id=987654321
  domain-name=example.com
```

Here the client will change the "cluster-id" attribute for the domain example.com to 987654321, if both the doman and cluster objects exists.

4.2.3 SET STATUS

The client may change the status of a domain object by using the SET STATUS command. The following values are valid:

- "inactive" signaling that the domain is not active.
- "active" signaling thet the domain is active.

Example:

```
  SET STATUS SRRP/1.0
  registrar-id=123456789
  registrar-auth=pass-phrase
  domain-name=example.com
  status=inactive
```

In this example, the client deactivates the domain "example.com" by setting its status to "inactive".

4.2.4 SET NAMESERVERS

The SET NAMESERVER command is used for changing all of the name server entries in a cluster object. The request body should contain exactly one "cluster-id" object identifying the cluster object and a number of name server entries defining the new name servers for the cluster object. The name server entries consists of the attributes "nsi-address" and "nsi-name" where i is a positive integer. The minimum andor maximum number of name server entries may be limited by the server, and the server should show these limits in the STATUS DEFAULTS response body. For further description of name server entries, see 3.1.2.

The new name server entries should completely replace all prevoius name server entries.

If the cluster ID does not exist in the system, the response message should be a 402 Cluster not registered. If the client specified too few or too many name servers, the server should respond with a 406 Invalid number of name servers error message.

Example:

```
SET NAMESERVERS SRRP/1.0
registrar-id=123456789
registrar-auth=pass-phrase
cluster-id=987654321
ns1-address=171.81.19.159
ns1-name=ns1.example.com
ns2-address=171.81.19.160
ns2-name=ns2.example.com
```

This will completely remove any name server entries from the cluster object in question, and replace them with the two name servers above.

4.2.5 SET PASSWORD

The client may change the domain password of a domain object by using the SET PASSWORD command. The new domain password should be given in the "domain-auth" attribute.

The putpose of the domain password is to authorize domain transfers between registrars. The transfer request message should contain the domain password for the requested domain, and the server should only perform the transfer when the password is correct.

Example:

```
SET PASSWORD SRRP/1.0
registrar-id=123456789
registrar-auth=pass-phrase
domain-name=example.com
domain-auth=domain-pass-phrase
```

4.5 DELETE

The DELETE command is used for deleting objects.

4.5.1 DELETE DOMAIN

The DELETE DOMAIN command will attempt to delete a domain object. The request body must contain exactly one "domain-name" attribute specifying the domain to be deleted.

If the domain object cannot be found, the server must respond with a 401 Domain not registered error message.

Example:

```
  DELETE DOMAIN SRRP/1.0
  registrar-id=123456789
  registrar-auth=pass-phrase
  domain-name=example.com
```

This will delete the domain example.com provided that the registrar attempting the operation has the proper authorization.

4.5.2 DELETE CLUSTER

The DELETE CLUSTER command will attempt to delete a cluster object. The request body must contain exactly one "cluster-id" attribute identifying the cluster object to be deleted.

If the cluster object cannot be found, the server must respond with a 402 Cluster not registered error message. If a cluster object which is in use by one or more active domain objects is attempted deleted, the server should return a 408 Removal not permitted error message. The client will have to assign another cluster ID to the domain objects using this cluster object, or set their status to "inactive" befor attempting the operation again.

Note that all the name server attribute groups contained withing the cluster object will be deleted too.

Example:

```
  DELETE DOMAIN SRRP/1.0
  registrar-id=123456789
  registrar-auth=pass-phrase
  cluster-id=987654321
```

This will delete the cluster object identified by the "cluster-id" attribute.

4.6 QUERY

The QUERY commands are used for fetching all the available information about an object.

4.6.1 QUERY DOMAIN

The QUERY DOMAIN command will attempt to retrieve some or all of the information for a domain. The request message must contain exactly one "domain-name" attribute giving the name of the domain object to query, and zero or more "get-specific" attributes naming the specific attributes to fetch.

If no "get-specific" attributes are present in the query, the server must return all available information for the domain object. If one or more "get-specific" are specified, the server must return the values of all the attributes named by the "get-specific" attributes or an error message.

If the server is unable to return required information, it must return a 301 Attribute temporarily unavailable. If one or more of

the "get-specific" attributes contains an unknown attribute, the server must return av 403 Invalid attribute. If the client attempts to query a domain which is not registered, the server must return a 401 Domain not registered.

Normally registrars should only be able to query their own domains, and attempts to query other registrars domains should result in a 411 Access denied error.

If there are no "get-specific" attributes in the query, the server MUST return at least the following information:

- The current registrar id in the "registrar-id" attribute.
- The domain name in the "domain-name" attribute.
- The expiry date in the "expiry-date" attribute.
- The current status of the domain in the "status" attribute.

If the server is unable to retrieve this information, it MUST respond to the client with a 301 Attribute temporarily unavailable, indicating the failure to retrieve required information about the domain.

The response to a query without any "get-specific" attributes SHOULD also contain the following information:

- The creation date of the domain in the "created-date" attribute.
- The cluster ID of the cluster object for this domain in the "cluster-id" attribute, if set.
- Any other relevant information about the domain contained in ordered "text" attributes.

Example query retrieving only the "expire-date" attribute:

```
QUERY DOMAIN SRRP/1.0
registrar-id=123456789
registrar-auth=pass-phrase
domain-name=example.com
get-specific=expire-date
```

This command should return the "expire-date" attribute for the domain natasha.vissaranovitsj.name, and a successful response might look like this:

```
200 Command completed successfully
expire-date=2004-12-20
```

Example query retrieving all the available information:

```
QUERY DOMAIN SRRP/1.0
registrar-id=123456789
registrar-auth=pass-phrase
domain-name=example.com
```

This command will try to retrieve all the information for the domain example.com. If it is successful, the output might look like this:

```
200 Command completed successfully
domain-name=natasha.vissaranovitsj.name
```

```
  registrar-id=123456789
  expire-date=2004-12-20
  created-date=2001-04-20
  cluster-id=987654321
  status=inactive
  text=Change: SET STATUS (to inactive)
  text=Changed-date:2001-04-03 12:46:01 UTC
  text=Changed-by: registrar 123456789
  text=Change: TRANSFER DOMAIN (from registrar 234567890)
  text=Changed-date:2001-02-13 10:15:12 UTC
  text=Changed-by: registrar 123456789
```

4.6.2 QUERY CLUSTER

The QUERY CLUSTER command is used for retrieving information about the name server entries of a cluster object. The request must contain exactly one "cluster-id" attribute identifying the cluster object.

The server must return all the name server entries in the cluster.

Example query:

```
  QUERY CLUSTER SRRP/1.0
  registrar-id=123456789
  registrar-auth=pass-phrase
  cluster-id=987654321
```

This request indicates that the client wants a list of the name servers in cluster object. The output could look like this:

```
  200 Command completed successfully
  ns1-address=128.39.19.168
  ns1-name=ns1.example.com
  ns2-address=128.39.19.169
  ns2-name=ns2.example.com
```

4.8 TRANSFER

The TRANSFER commands are used for handling the transfer of a domain from one registrar to another.

4.8.1 TRANSFER DOMAIN

This command is used for requesting a transfer of a domain belonging to another registrar to the requesting registrar. The request body must contain the requested domain name in the "domain-name" attribute and the domain password in the "domain-auth" attribute.

If the domain is not registered, the server should issue a 401 Domain not registered. If the domain password did not properly authorize the transfer, the sever should issue a 412 Authorization failed error message.

Note that information regarding domain transfers, such as domain passwords and notification about lost and obtained domains, is not handeled by SRRP. Out of band communications means should be used for this purpose.

Example:

```
TRANSFER DOMAIN SRRP/1.0
registrar-id=234567890
registrar-auth=pass-phrase
domain-name=example.com
domain-auth=domain-pass-phrase
```

Here, the domain example.com is requested transferred to the requesting registrar. If the domain password is correct, the domain server should immediately transfer the ownership of the domain to the requesting registrar.

4.9 STATUS

The STATUS commands gives information about the implementation and configuration of the server.

4.9.1 STATUS DEFAULTS

The STATUS DEFAULTS command is used for retrieving various default values, such as default status and default registration period, from the server.

The response body MUST contain the following attributes:

- The default status for new registrations in the "default-status" attribute
- The default registration period, in months, for new registrations in the "default-period" attribute.
- The maximum user definable registration period, zero (0) if unset or unlimited, in the "maximum-period" attribute.
- The default domain transfer response in the "transfer-default" attribute. Valid values are the ASCII strings "yes", "no" or "unset".
- The transfer timeout period in the "transfer-timeout" atribute. If this is set to zero (0), the feature is disabled and both this attribute and "transfer-default" SHOULD be ignored.
- The minimum number of name servers allowed in the "minimum-ns" attribute, zero (0) if unspecified.
- The maximum number of name servers allowed in the "maximum-ns" attribute, zero (0) if unspecified.

The server MAY add additional "text" attributes for returning server specific defaults. The client MUST NOT rely on these "text" attributes.

Example command:

```
STATUS DEFAULTS SRRP/1.0
registrar-id=123456789
registrar-auth=pass-phrase
```

A typical response message would look like this:

```
200 Command completed successfully
default-status=active
default-period=66
```

```
  maximum-period=120
  minimum-ns=2
  maximum-ns=8
```

4.9.2 STATUS SERVER

Clients may use a STATUS command with the SERVER argument to fetch informtaion about the server inplementation. The information is returned in one or more "text" attributes. If the server does not wish to return any information, it can do so by returning a 200 Command completed successfully and leave the response body empty.

The server MAY return information on a STATUS SERVER command, but the client MUST NOT rely on this information.

Example:

```
  STATUS SERVER SRRP/1.0
  registrar-id=123456789
  registrar-auth=pass-phrase
```

The response message may look like this:

```
  200 Command completed successfully
  text=Standard SRRP server version 1.0.4p3
  text=Compiled 2001-01-29 03:58:32 GMT+1
```

5.0 Response codes

5.1 Success codes (2xx)

There is only one success code, and it indicates unconditional success.

200 Command completed successfully
This response code indicates unconditional success when executing the requested command. It is the only success code.

5.2 Temporary error codes (3xx)

Temporary error codes indicates that the requested command could not be executed due to a temporary failure. The client MAY retry the command later.

300 Internal server error
The server suffered from a fatal internal error, and the client is adviced to notify the server administrator and retry the operation later. The server SHOULD present contact information in the error message, log the error and notify the server administrator.

301 Attribute temporarily unavailable
This error code indicates that the server was unable to return a mandatory attribute due to a temporary failure.

5.3 Permanent error codes (4xx)

Permanent error codes indicates that the requested command could not be executed due to a permanent failure. The client SHOULD NOT retry

the command.

400 Domain already registered
This error code signals that the client has attempted to register an object that is already in the registered.

401 Domain not registered
This indicates that the client attempted to operate on an object which is not registered.

402 Cluster not registered
This indicates that the client attempted to operate on an object which is not registered.

403 Invalid attribute
The request body contained one or more invalid attributes, indicating a client error or protocol mismatch.

404 Invalid attribute value
The request body contained one or more invalid attribute value. This could be a character string where the server expected a number, or an incomplete data string.

405 Invalid expire date
The client specified an expiry date which were either in the past or to far in the fututure.

406 Invalid number of name severs
The client specified either too few or too many name serverers.

407 Mandatory attribute missing
This indicates that a mandatory attribute was missing.

408 Removal not permitted
The client attempted to remove an entity which is required, for instance a cluster object which is in use by one or more domain objects.

409 Blocked domain
The domain which the client attempted to register was blacklisted by the server.

410 Credit failure
The client attempted to execute a command which he did not have the neccesary credit to preform.

411 Access denied
The client attempted to operate on an object he was not authorized to. The server should log such errors.

412 Auhorization failed
The authorization credentials specified by the client did not match, or the registrar ID was unknown.

413 Unsupported protocol version
The client specified a protocol which the server did not support, either too new or old.

6.0 ABNF Definition of SRRP

This is a formal definition of SRRP using standard ABNF as defined in [1].

6.1 Lexical definitions

```
SP = %x20          ; ASCII space
HT = %x09          ; ASCII horizontal tab
DOT = %x2e         ; ASCII "."
EOF = %x00         ; ASCII end of file
DASH = %x2d        ; ASCII "-"
SL = %x2d          ; ASCII "-"
EQ = %x3d          ; ASCII "="
CR = %x0D          ; ASCII carriage return
LF = %x0A          ; ASCII linefeed
LWS = SP / HT      ; linear white space
CRLF = CR LF       ; carriage return line feed sequence
UALPHA = %x41-5a     ; ASCII A-Z
LALPHA = %x61-7a     ; ASCII a-z
ALPHA = UALPHA / LAPLHA       ; ASCII a-z / A-Z
DIGIT = %x30-39               ; ASCII 0-9
PCHAR = ALPHA / DIGIT / DASH        ; prococol characters
UCHAR = %x20-%ff                    ; user charachters
```

6.2 Basic grammatical definitions

```
ip-address = 1*3DIGIT DOT 1*3DIGIT DOT 1*3DIGIT DOT 1*3DIGIT
protocol = "SRRP" SL version
version = main-version DOT sub-version
main-version = 1*DIGIT
sub-version = 1*DIGIT
date = year DASH month DASH day
year = 4DIGIT
month = 2DIGIT
day = 2DIGIT

response-header = success-header / tempfail-header / permfail-header
success-header = success-code LWS response-text
tempfail-header = temporary-fail-code LWS response-text
permfail-header = permanent-fail-code LWS response-text
success-code = "2" 2DIGIT
temporary-fail-code = "3" 2DIGIT
permanent-fail-code = "4" 2DIGIT
response-text = *PCHAR

standard-response-message = response-header [CRLF response-body]
response-body = 1*text-pair

error-response-message = (tempfail-header / permfail-header)
                         [CRLF response-body]
```

6.3 Attribute/value set definitions

```
attribut-value-pair = attribute EQ value CRLF
attribute = 1*PCHAR
value = *UCHAR
```

```
text-pair = text-attribte EQ text-value CRLF
text-attibute = "text"
text-value = *UCHAR

cluster-id-pair = cluster-id-attribute EQ cluster-id-value CRLF
cluster-id-attribute = "cluster-id"
cluster-id-value = 1*PCHAR

status-pair = status-attribute EQ status-value CRLF
status-attribute = "status"
status-value = "active" / "inactive"

registrar-id-pair = registrar-id-attribute EQ registrar-id-value CRLF
registrar-id-attribute = "registrar-id"
registrar-id-value = 1*PCHAR

registrar-auth-pair = registrar-auth-attribute EQ registrar-auth-value CRLF
registrar-auth-attribute = "registrar-auth"
registrar-auth-value = *UCHAR

expire-date-pair = expire-date-attribute EQ expire-date-value CRLF
expire-date-attribute = "expire-date"
expire-date-vaue = date

domain-name-pair = domain-name-attribute EQ domain-name-value CRLF
domain-name-attribute = "domain-name"
domain-name-value = 1*UCHAR

domain-auth-pair = domain-auth-attribute EQ domain-auth-value CRLF
domain-auth-attribute = "domain-auth"
domain-auth-value = *UCHAR

get-specific-pair = get-specific-attribute EQ get-specific-value CRLF
get-specific-attribute = "get-specific"
get-specific-value = 1*PCHAR

name-server-entry = ns-address-pair ns-name-pair
ns-address-pair = ns-address-attribute EQ ns-address-value CRLF
ns-address-attribut = "ns" 1*DIGIT "-address"
ns-address-value = ip-address
ns-name-pair = ns-name-attribute EQ ns-name-value CRLF
ns-name-attribut = "ns" 1*DIGIT "-name"
ns-name-value = UCHAR

registrar-auth-entry = registrar-id-pair registrar-auth-pair
```

6.3 Message difinition

```
message = (create / set / delete / query / transfer / status) EOF

create = create-domain / create-cluster
set = set-expire / set-cluster / set-status / set-nameservers
delete = delete-domain / delete-cluster
query = query-domain / query-cluster
transfer = transfer-request / transfer-response
status = status-defaults / status-server
```

```
create-domain = create-domain-request / create-domain-response
create-cluster = create-cluster-request / create-cluster-response
set-expire = set-expire-request / set-expire-response
set-cluster = set-cluster-request / set-cluster-response
set-status = set-status-request / set-status-response
set-nameservers = set-nameservers-request / set-nameservers-response
set-password = set-password-request / set-password-response
delete-domain = delete-domain-request / delete-domain-response
delete-cluster = delete-cluster-request / delete-cluster-response
query-domain = query-domain-request / query-domain-response
query-cluster = query-cluster-request / query-cluster-response
transfer-domain = transfer-domain-request / transfer-domain-response
status-defaults = status-defaults-request / status-default-response
status-server = status-server-request / status-server-response

; CREATE DOMAIN REQUEST
create-domain-request = create-domain-request-header CRLF
                        create-domain-request-body
create-domain-request-header = "CREATE" LWS "DOMAIN" LWS protocol
create-domain-request-body = registrar-auth-entry domain-name-pair
                             domain-auth-pair [expire-date-pair]
                             [status-pair] (cluster-id-pair /
                             *name-server-entry)

; CREATE DOMAIN RESPONSE
create-domain-response = create-domain-success / error-response-message
create-domain-success = success-header CRLF cluster-id-pair status-pair
                        expire-date-pair

; CREATE CLUSTER REQUEST
create-cluster-request = create-cluster-request-header CRLF
                         create-cluster-request-body
create-cluster-request-header = "CREATE LWS "CLUSTER" LWS protocol
create-cluster-request-body = registrar-auth-entry *name-server-entry

; CREATE CLUSTER RESPONSE
create-cluster-response = create-cluster-success / error-response-message
create-cluster-success = success-header CRLF cluster-id-pair

; SET EXPIRE REQUEST
set-expire-request = set-expire-request-header CRLF set-expire-request-body
set-expire-request-header = "SET" LWS "EXPIRE" LWS protocol
set-expire-request-body = registrar-auth-entry expire-date-pair
                          domain-name-pair

; SET EXPIRE RESPONSE
set-expire-response = standard-response

 SET CLUSTER REQUEST
set-cluster-request = set-cluster-request-header CRLF set-cluster-request-body
set-cluster-request-header = "SET" LWS "CLUSTER" LWS protocol
set-cluster-request-body = registrar-auth-entry cluster-id-pair
                           domain-name-pair

; SET CLUSTER RESPONSE
set-expire-response = standard-response
```

```
; SET STATUS REQUEST
set-status-request = set-status-request-header CRLF set-status-request-body
set-status-request-header = "SET" LWS "STATUS" LWS protocol
set-status-request-body = registrar-auth-entry domain-name-pair status-pair

; SET STATUS RESPONSE
set-expire-response = standard-response

; SET NAMESERVERS REQUEST
set-nameservers-request = set-nameservers-request-header CRLF
                          set-nameservers-request-body
set-nameservers-request-header = "SET" LWS "NAMESERVERS" LWS protocol
set-nameservers-request-body = registrar-auth-entry cluster-id-pair
                               *name-server-entry

; SET NAMESERVERS RESPONSE
set-expire-response = standard-response

; SET PASSWORD REQUEST
set-password-request = set-password-request-header CRLF
                       set-password-request-body
set-password-request-header = "SET" LWS "PASSWORD" LWS protocol
set-password-request-body = registrar-auth-entry domain-name-pair
                            domain-auth-pair

; SET PASSWORD RESPONSE
set-password-response = standard-response

; DELETE DOMAIN REQUEST
delete-domain-request = delete-domain-request-header CRLF
                        delete-domain-request-body
delete-domain-request-header = "DELETE" LWS "DOMAIN" LWS protocol
delete-domain-request-body = registrar-auth-entry domain-name-pair

; DELETE DOMAIN RESPONSE
delete-domain-response = standard-response

; DELETE CLUSTER REQUEST
delete-cluster-request = delete-cluster-request-header CRLF
                         delete-cluster-request-body
delete-cluster-request-header = "DELETE" LWS "CLUSTER" LWS protocol
delete-cluster-request-body = registrar-auth-entry cluster-id-pair

; DELETE CLUSTER RESPONSE
delete-domain-response = standard-response

; QUERY DOMAIN REQUEST
query-domain-request = query-domain-request-header CRLF
                       query-domain-request-body
query-domain-request-header = "QUERY" LWS "DOMAIN" LWS protocol
query-domain-request-body = registrar-auth-entry domain-name-pair
                            [get-specific-pair]

; QUERY DOMAIN RESPONSE
query-domain-response = full-domain-response / specific-domain-response /
                        error-response-message
```

```
full-domain-response = success-header CRLF *attribute-value-pair
specific-response = success-header CRLF 1attribute-value-pair

; QUERY CLUSTER REQUEST
query-cluster-request = query-cluster-request-header CRLF
                        query-cluster-request-body
query-cluster-request-header = "QUERY" LWS "CLUSTER" LWS protocol
query-cluster-request-body = registrar-auth-entry cluster-id-pair

; QUERY CLUSTER RESPONSE
query-cluster-response = standard-response

; TRANSFER DOMAIN REQUEST
transfer-request-request = transfer-request-request-header CRLF
                           transfer-request-request-body
transfer-request-request-header = "TRANSFER" LWS "DOMAIN" LWS protocol
transfer-request-request-body = registrar-auth-entry domain-name-pair
                                domain-auth-pair

; TRANSFER DOMAIN RESPONSE
transfer-request-response = standard-response

; STATUS DEFAULTS REQUEST
status-defaults-request = status-defaults-request-header CRLF
                          status-defaults-request-body
status-defaults-request-header = "STATUS" LWS "DEFAULTS" LWS protocol
status-defaults-request-body = registrar-auth-entry

; STATUS DEFAULTS RESPONSE
status-defaults-response = status-defaults-response-message /
                           standard-error-message
status-defaults-response-message = success-header CRLF
                                   status-defaults-response-body
status-defaults-response-body = default-status-pair / default-period-pair /
                                maximum-period / transfer-default / text-pair /
                                transfer-timeout / minimum-ns / maximum-ns
default-status-pair = default-status-attribute EQ default-status-value CRLF
default-status-attribute = "default-status"
default-status-value = "active" / "inactive"
default-period-pair = default-period-attribute EQ default-period-valie CRLF
default-period-attribute = "default-period"
default-period-value = 1*DIGIT
maximum-period-pair = maximum-period-attribute EQ maximum-period-pair CRLF
maximum-period-attribute = "maximum-period"
maximum-period-value = 1*DIGIT
transfer-default-pair = transfer-default-attribute EQ transfer-default-value
                        CRLF
transfer-default-attribute = "transfer-default"
transfer-default-value = "yes" / "no" / "unset"
minimum-ns-pair = minimum-ns-attribute EQ minimum-ns-value CRLF
minimum-ns-attribute = "minimum-ns"
minimum-ns-value = 1*DIGIT
maximum-ns-pair = maximum-ns-attribute EQ maximum-ns-value CRLF
maximum-ns-attribute = "minimum-ns"
maximum-ns-value = 1*DIGIT

; STATUS SERVER REQUEST
```

```
status-server-request = status-server-request-header CRLF
                        status-server-request-body
status-server-request-header = "STATUS" LWS "SERVER" LWS protocol
status-server-request-body = registrar-auth-entry

; STATUS SERVER RESPONSE
status-server-response = status-server-response-message /
                         standard-error-message
status-server-response-message = success-header CRLF
                                 status-server-response-body
status-server-response-body = *text-pair
```

7.0 RRP to SRRP mapping

As RRP is stateful, ie. requires the server to maintain state information for every connected client for as long as he is connected, it is impossible for a RRP client to talk directly to an SRRP server. The only way to allow for RRP clients to talk to SRRP servers, would be to use an RRP/SRRP gateway to maintain the state required by the RRP client, and issue SRRP messages for every RRP operation the client performs. This is, however, outside of the scope of this document.

8.0 References

[1] Bradner, S., "Key Words for Use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[2] A. Frier, P. Karlton, and P. Kocher, "The SSL 3.0 Protocol", Netscape Communications Corp., November 18, 1996.

[3] Crocker, D. (Editor) and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.