

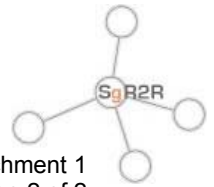
## **SgR2R.EPP API User Guide**

Version 1.0

This document is a concise guide to installing and using the SgR2R API. SgR2R is a customized version of the Neteka NeR2R (Neteka Extensible Registry-to-Registrar System) API, specifically designed for the use of Singapore Network Information Centre (SGNIC).

The SgR2R API is a set of Perl Libraries intended for Unix based operating systems. The API creates Extensible Provisioning Protocol (EPP) commands that can be used to communicate with the SGNIC Registry EPP Gateway Server.

Contrary to the previous experimental release of the API-Server setup, all EPP commands generated from the API need to be sent to the EPP gateway server using SSL. The task of establishing and maintaining a SSL connection to the gateway server is to be handled by another component. This component is referred to as the SSL Proxy in this document. The SSL Proxy is not included with the SgR2R.EPP API.



## Table of Contents

1. Prerequisites.....	3
2. Installing the SgR2R API Startup Kit .....	3
3. Using the SgR2R API .....	4
3.1 The SgR2R EPP implementation .....	4
3.1.1 Related Web Links on EPP .....	4
3.2 SSL Proxy.....	5
3.3 Logging in and maintaining the session .....	5
3.3.1 Maximum number of socket connections .....	5
3.3.2 Idle time-out and Absolute time-out.....	5
3.4 Calling the API.....	6
4. Available Functions.....	6
4.1 Session Management.....	6
4.1.1 EPPlogin.....	6
4.1.2 EPPlogout .....	6
4.1.3 EPPhello.....	7
4.2 Domain Operations.....	7
4.2.1 EPPdomaincheck .....	7
4.2.2 EPPdomaincreate .....	7
4.2.3 EPPdomaininfo.....	8
4.2.4 EPPdomainupdateadd .....	9
4.2.5 EPPdomainupdateremove .....	9
4.2.6 EPPdomainupdatechange.....	10
4.2.7 EPPdomaindelete.....	11
4.2.8 EPPdomaintransferrequest .....	12
4.2.9 EPPdomaintransferoperation .....	12
4.2.10 EPPdomaintransferstatus.....	13
4.2.11 EPPdomainrenew.....	13
4.2.12 EPPdomainupdatewhoisprofile .....	13
4.2.13 EPPdomaininfowhoisprofile.....	14
4.3 Contact Operations.....	15
4.3.1 EPPcontactcreate.....	15
4.3.2 EPPcontactinfo.....	15
4.3.3 EPPcontactupdatechange.....	16
4.3.4 EPPcontactdelete.....	17
4.4 Host Operations.....	17
4.4.1 EPPhostcreate .....	17
4.4.2 EPPhostinfo.....	18
4.4.3 EPPhostupdateadd .....	18
4.4.4 EPPhostupdateremove .....	18
4.4.5 EPPhostdelete.....	19
4.5 Batch Processing.....	19
Appendix A: Sample Configuration File .....	20
Appendix B: Glossary of SgR2R.EPP Commands .....	21
Appendix C: List of Global Error Code .....	22
Appendix D: Change History.....	23



## 1. Prerequisites

A Unix or Linux based operating system and Perl 5.6.

## 2. Installing the SgR2R API Startup Kit

The following packages are included in the SgR2R API Startup Kit.

- SGR2R::API::APIProtocol
- SGR2R::API::DomainUtil
- SGR2R::API::R2R\_API
- SGR2R::API::XML\_Builder\_for\_Request
- SGR2R::API::XML\_Parser\_for\_Response
- SGR2R::API::XML\_Checker
- SGR2R::API::Objects

The installation procedures of these packages follow Perl CPAN module standards. Just extract the files from the .gz file included, go into each module directories, and install the modules using the following commands:

```
perl Makefile.PL
make
make test
make install
```

### The Registrar.conf configuration file

The standard location of the configuration file Registrar.conf is: /neteka/SGR2R/conf/. If the location of the file is changed, please update the corresponding file path in the APIProtocol.pm and R2R\_API.pm packages and then re-install the modules.

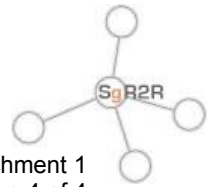
Contrary to the previous experimental releases of the API, the API should be configured to send commands to an SSL Proxy instead of to the Gateway server. Upon receiving the command, the SSL Proxy should forward the command to the gateway server through a secure SSL channel.

The SSL Proxy location is set in the configuration file at:

```
$R2R_Server = "sslproxy";
```

The API will establish a socket connection to the proxy and wait for the response.

Each registrar should obtain a Registrar ID and password from SGNIC. Please update the ID and password values in the configuration file in order to communicate with the server. Without a valid ID, the API client will not be able to communicate with the gateway server.



```
$REGISTRAR_ID = "TestID";  
$REGISTRAR_PASSWORD = "abcde";
```

### 3. Using the SgR2R API

This section describes how the SgR2R.EPP API could be used to perform domain transactions at the SgR2R.EPP Gateway.

#### 3.1 The SgR2R EPP implementation

The Extensible Provisioning Protocol (EPP) is a set of XML text protocol that permits multiple service providers to perform object provisioning operations using a shared central object repository. It provides a standard Internet domain name registration protocol for use between domain name registrars and registries, although the protocol can also be used with other applications and types of objects.

The SgR2R API is a simple application-programming interface to help Registrars to communicate with the SGNIC Registry. The API provides a simple interface that can be used to create EPP XML commands and process the XML response sent back from the EPP gateway server. Thus Registrars are shielded from the application implementation details of EPP XML parser and builder.

All API generated EPP commands need to be forwarded to the EPP gateway server with SSL. The API is designed to integrate easily with an independent SSL Proxy. All it does is create an EPP command, send the command to the SSL Proxy through a socket connection, wait for the response, and then parse the response back to the user.

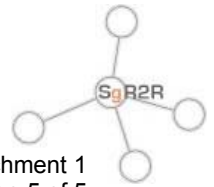
SgR2R implements the EPP connection-oriented operating mode. A registrar has to use the EPP login command to establish a secure session with the Registry server before any other commands can be issued. The EPP server transaction ID <srTRID>, together with the Registrar ID, is used to identify each Registrar client session.

The SgR2R EPP implementation can be divided into 4 different groups: Session management, Domain object operation, Contact object operation, and Host object operation. The commands transfer, renew and check are not implemented for the contact and host objects since these operations are deemed meaningless in the SgR2R system.

##### 3.1.1 Related Web Links on EPP

For a detailed description of the EPP protocol, please refer to the following web sites:

1. IETF Provisioning Registry Protocol (provreg) Workgroup:  
<http://www.ietf.org/html.charters/provreg-charter.html>
2. EPP: <http://www.ietf.org/internet-drafts/draft-ietf-provreg-epp-06.txt>



3. EPP Contact Mapping:  
<http://www.ietf.org/internet-drafts/draft-ietf-provreg-epp-contact-04.txt>
4. EPP Domain Mapping:  
<http://www.ietf.org/internet-drafts/draft-ietf-provreg-epp-domain-04.txt>
5. EPP Host Mapping:  
<http://www.ietf.org/internet-drafts/draft-ietf-provreg-epp-host-04.txt>

## 3.2 SSL Proxy

There are 2 options for registrars to acquire SSL connectivity to the EPP gateway server.

1. Neteka provides a SSL Proxy that is available separately for a cost.
2. Registrars can write their own SSL Proxy using standard SSL request-response handshakes.

The specification of SSL can be found at:

<http://wp.netscape.com/eng/ssl3/draft302.txt>.

Open Source SSL source code and documentation is also available with the openssl package, which can be found at

[www.openssl.org](http://www.openssl.org).

Perl can also be used to write the SSL Proxy. The package Net::SSLeay is recommended:

[http://www.cpan.org/authors/id/SAMPO/Net\\_SSLeay.pm-1.18.tar.gz](http://www.cpan.org/authors/id/SAMPO/Net_SSLeay.pm-1.18.tar.gz)

## 3.3 Logging in and maintaining the session

Once the EPP gateway server receives the EPPlogin() command, if the validation is successful, it will send a success response back to the SSL Proxy and maintain the current SSL session. The client can then send additional commands to the proxy/server. The EPPlogout() command can be used to instruct the EPP gateway server to close the current session.

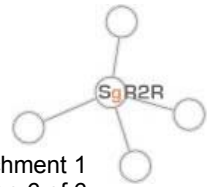
### 3.3.1 Maximum number of socket connections

There are a maximum number of socket connections allowed for each registrar ID. If the number is exceeded, the next login request will be rejected.

### 3.3.2 Idle time-out and Absolute time-out

Each session has a time to live (TTL) of 10 minutes (i.e., The session will be closed once it has been inactive for 10 minutes. This interval is configurable by SGNIC). This is referred to as the Idle Timeout. To keep the connection alive, issue the command EPPhello() before time out occurs.

All sessions will be terminated after 24 hours of continuous connection (Absolute Timeout), regardless of the use of EPPhello() or other commands..



### 3.4 Calling the API

The SgR2R API can be used to check the availability of a domain name in any language or symbol, obtain the pricing for a particular domain name, and obtain information for a domain name, register a domain name as well as to provided suggested alternatives to a queried domain name. A comprehensive list of the available functions in the SgR2R API is included in the Section 4: Available Functions. The following routine is an example to query the availability of the domain name: "company.com.sg" using the command EPPdomaincheck()

```
use R2R_API;

$domain = "company.com.sg";
my $reply = SGR2R::API::R2R_API::EPPdomaincheck($domain);

if ($reply->{RCODE} eq "1"){
    print "Domain is available";
}
elsif ($reply->{RCODE} eq "0"){
    print "Domain already taken";
}
```

## 4. Available Functions

The following session provides description of each command available in the API for performing domain transactions with the SgR2R.EPP Gateway.

### 4.1 Session Management

Session Management commands allow the registrar to login to or logout off the Gateway as well as to perform "keep alive" alerts for active sessions.

#### 4.1.1 EPPlogin

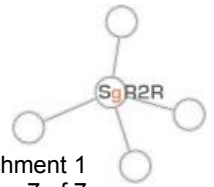
Issue this command to establish the connection session with the server.

```
(\%ref) = EPPlogin()
$ref    -> {RCODE} = 1   ; login successful
         -> {RCODE} = -1  ; error
$ref    -> {MSG}   ; error messages
```

#### 4.1.2 EPPlogout

Issue this command to close the SSL connection.

```
(\%ref) = EPPlogout()
$ref    -> {RCODE} = 1   ; logout successful
         -> {RCODE} = -1  ; error
$ref    -> {MSG}   ; error messages
```



### 4.1.3 EPPhello

A client can request a greeting message from the server to ensure connection is alive. This command is to be used for keeping the SSL sessions alive.

```
(\%ref) = EPPhello()  
$ref      -> {MSG}          ; Server alive message
```

## 4.2 Domain Operations

Domain Operation commands are used for domain oriented transactions such as domain creation, update, transfer or status change.

### 4.2.1 EPPdomaincheck

The Domain check function is used to determine the availability of a domain name for registration. The domain check function does not automatically create or hold the domain name if the domain is available.

```
(\%ref) = EPPdomaincheck ($domain)  
$ref      -> {RCODE} = 0 ; domain unavailable  
           {RCODE} = 1 ; domain available  
           {RCODE} = -1 ; error  
$ref      -> {MSG}          ; error messages
```

### 4.2.2 EPPdomaincreate

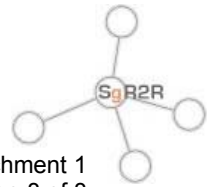
The Domain Create function is used to initiate the registration of a domain name. Domain name, Contact information as well as Name Server information should be provided. The registrant ID is a required field. All other contact IDs can be left blank, however the requirements of each TLD will dictate the appropriateness of the request.

The domaininfo hash is used to store the domain information. To update the whois profile information for each contact, please use the EPPdomainupdate whoisprofile function after the creation of the domain.

Pragmatic Approval Codes: 1 = Yes (forced pragmatic by registrar)  
2 = Registry Determined

Parking Flag: 1 = Yes, 2 = No

```
(\%ref) = EPPdomaincreate(\%domaininfo)  
; Domain Information: %domaininfo  
; Domain Name: NAME  
; Term (No. of yrs): TERM  
; Pragmatic Approval:  
PRAGMATICAPPROVAL  
; Parking Flag: PARKING  
; Registrant ID: OWNERID  
; Admin Contact: ADMINID  
; Tech Contact: TECHID  
; Billing Contact: BILLID  
; Agent Contact: AGENTID
```



```

; Owner Parent Org. ID: OWNERPARENTID
; Admin Parent Org. ID: ADMINPARENTID
; Tech Parent Org. ID: TECHPARENTID
; Billing Parent Org. ID: BILLPARENTID
; Agent Parent Org. ID: AGENTPARENTID
; Name Server: NS1
; Name Server: NS2
; etc....
; Up to 13 Name Servers are allowed
; new domain ID
$ref -> {ID}
$ref -> {RCODE} = 1 ; success
           {RCODE} = 0 ; domain name not available for registration
           {RCODE} = 2 ; domain application in queue for approval
           {RCODE} = -1 ; error
           {RCODE} = -2 ; domain length too long
           {RCODE} = -3 ; contact ID invalid
           {RCODE} = -4 ; name server name invalid
           {RCODE} = -5 ; domain name rejected
           {RCODE} = -6 ; Invalid Singapore Personal ID number
           {RCODE} = -7 ; This Contact ID has to have a valid RCB
ID, or an agent ID with a valid Singapore personal ID or RCB ID has to be
present
           {RCODE} = -8 ; invalid TLD for domain name
           {RCODE} = -9 ; registration term cannot be empty
           {RCODE} = -10 ; RCB number is required
           {RCODE} = -11 ; Organization name does not match name
in RCB database for this RCBID
           {RCODE} = -12 ; Full registered address required
           {RCODE} = -13 ; Full mailing address required
           {RCODE} = -14 ; Singapore registered address required
           {RCODE} = -15 ; Singapore mailing address required
           {RCODE} = -16 ; Phone number required
           {RCODE} = -17 ; Fax number required
           {RCODE} = -18 ; Pager/Mobile number required
           {RCODE} = -19 ; Registration term period exceeds limit
$ref -> {MSG} ; error messages

```

### 4.2.3 EPPdomaininfo

The Obtain Domain Information function is used to retrieve domain information. To retrieve the whois profile information for each contact, please use the EPPdomaininfowhoisprofile function.

Domain Status IDs: ok 1, hold 2, inactive 3, suspended 4, pendingDelete 5, pendingTransfer 6, pendingVerification 7

(\%ref) = EPPdomaininfo(\$domain\_name)

```

$ref -> {ID} ; Domain ID
$ref -> {NAME} ; Domain name
$ref -> {OWNERID} ; Registrant ID
$ref -> {ADMINID} ; Admin Contact ID
$ref -> {TECHID} ; Tech Contact ID
$ref -> {BILLID} ; Billing Contact ID

```



```

$ref -> {AGENTID} ; Agent Contact ID
$ref -> {OWNERPARENTID}; Owner(Registrant) Parent Organization ID
$ref -> {ADMINPARENTID}; Admin Parent Organization ID
$ref -> {TECHPARENTID} ; Tech Parent Organization ID
$ref -> {BILLPARENTID} ; Billing Parent Organization ID
$ref -> {AGENTPARENTID}; Agent Parent Organization ID
$ref -> {TERM} ; term of domain name purchased, in years
$ref -> {PARKING} ; parking flag
$ref -> {CRDATE} ; date of purchase
$ref -> {EXDATE} ; expiry date
$ref -> {STATUS} ; Status ID
$ref -> {NS1} ; Primary Name Server Domain
$ref -> {NS2,...etc} ; Secondary Name Server(s) Domain(s)
; .. up to 13 name servers

$ref -> {RCODE} = 1 ; success
; {RCODE} = -1 ; error
; {RCODE} = 0 ; No domain information available
$ref -> {MSG} ; error messages

```

#### 4.2.4 EPPdomainupdateadd

The Update Domain Add function is used to add information to a registered domain name. You may add Name servers for a domain name with this command. To create a new hostname (which could be used as a Name Server), please use the EPPhostcreate command.

Note: Duplicate status id or name server cannot be added.

```

(\%ref) = EPPdomainupdateadd(\%info)
; Domain Name: NAME;
; Optional Information: %info:
; Name Server 1: NS1
; Name Server 2: NS2
; etc....
; Up to 13 Name Servers is allowed

$ref -> {RCODE} = 1 ; success
; {RCODE} = -1 ; error
; {RCODE} = -4 ; Must provide a minimum number of name
servers
; {RCODE} = -5 ; name server name invalid
$ref -> {MSG} ; error messages

```

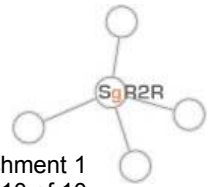
#### 4.2.5 EPPdomainupdateremove

The Update Domain Remove function is used to delete information from a registered domain name.

```

(\%ref) = EPPdomainupdateremove(\%info)
; Domain Name: NAME;
; Optional Information: %info:
; Name Server 1: NS1
; Name Server 2: NS2
; etc....

```



```

; Up to 13 Name Servers is allowed
$ref    -> {RCODE} = 1 ; success
          {RCODE} = -1 ; error
          {RCODE} = -3 ; No such name server for this domain
          {RCODE} = -4 ; Must provide a minimum number of name
servers
$ref    -> {MSG} ; error messages

```

#### 4.2.6 EPPdomainupdatechange

The Update Domain Change function is used to update the registrant information for a registered domain name.

STATUS code:

Status Description	Code
Active	1
Inactive	2
Pending Registrar Transfer	3
Hold: Court Order	4
Hold: Registrant Request	5
Hold: Breach of Contract	6
Hold: Registrar Request	7
Delete: Lame Domain	8
Delete: Court Order	9
Delete: Registrant Request	10
Delete: Breach of Contract	11
Delete: Registrar Request	12
Expired	13
Deleted	14

(\%ref) = EPPdomainupdatechange(\%info)

```

; Domain Name:      NAME
; Optional Information in %info:
; New Registrant ID: OWNERID
; Admin Contact:    ADMINID
; Tech Contact:     TECHID
; Billing Contact:   BILLID
; Agent Contact:    AGENTID
; Owner Parent Org. ID: OWNERPARENTID
; Admin Parent Org. ID: ADMINPARENTID
; Tech Parent Org. ID: TECHPARENTID
; Billing Parent Org. ID: BILLPARENTID
; Agent Parent Org. ID: AGENTPARENTID
; Parking Flag:     PARKING
; New Status:       STATUS

```



```

; Status Change Reason:
STATUSREASON
$ref -> {RCODE} = 1 ; success
        {RCODE} = -1 ; error
        {RCODE} = -2 ; Invalid contact ID
        {RCODE} = -3 ; Invalid status code
        {RCODE} = -6 ; Invalid Singapore Personal ID number
        {RCODE} = -7 ; This Contact ID has to have a valid RCB
ID, or an agent ID with a valid Singapore personal ID or RCB ID has to be
present
        {RCODE} = -10 ; RCB number is required
        {RCODE} = -11 ; Organization name does not match name
in RCB database for this RCBID
        {RCODE} = -12 ; Full registered address required
        {RCODE} = -13 ; Full mailing address required
        {RCODE} = -14 ; Singapore registered address required
        {RCODE} = -15 ; Singapore mailing address required
        {RCODE} = -16 ; Phone number required
        {RCODE} = -17 ; Fax number required
        {RCODE} = -18 ; Pager/Mobile number required
$ref -> {MSG} ; error messages

```

#### 4.2.7 EPPdomaindelete

The Delete Domain function is used to delete a registered domain name. Upon successful processing of the command, the domain will be placed under the Delete status code listed below and will be purged from the registry database after the deletion escrow period.

Status Description	Code
Delete: Lame Domain	8
Delete: Court Order	9
Delete: Registrant Request	10
Delete: Breach of Contract	11
Delete: Registrar Request	12
Deleted	14

(\%ref) = EPPdomaindelete(\%info)

```

; Domain Information: %info
; Domain Name: NAME
; Domain Status Code: STATUS
; Reason: STATUSREASON
$ref -> {RCODE} = 0 ; domain name does not exist
        {RCODE} = 1 ; success
        {RCODE} = -1 ; error
        {RCODE} = -2 ; fail – this domain has association with
other objects
        {RCODE} = -3 ; Invalid status code
        {RCODE} = -12 ; Status change disallowed
$ref -> {MSG} ; error messages

```



#### 4.2.8 EPPdomaintransferrequest

The Domain transfer request function is used to request transfer of a registered domain to the registrar issuing the command. The password field is the password of the registrant. The 5 contact IDs are the ID's for the domain in the new registrar. NEWOWNERID is a required field. The parent organization ID of the 5 contacts can also be provided.

(\%ref) = EPPdomaintransferrequest(\%dominfo)

```

; Domain Information: %dominfo
; Domain Name:      NAME;
; Years Added:      TERM;
; Password:         PASSWORD;
; New Owner ID:     NEWOWNERID;
; New Admin ID:     NEWADMINID;
; New Tech ID:      NEWTECHID;
; New Billing ID:    NEWBILLID;
; New Agent ID:     NEWAGENTID;
; New Owner Parent Org ID:
NEWOWNERPARENTID;
; New Admin Parent Org ID:
NEWADMINPARENTID;
; New Tech Parent Org ID:
NEWTECHPARENTID;
; New Billing Parent Org ID:
NEWBILLPARENTID;
; New Agent Parent Org ID:
NEWAGENTPARENTID;

$ref    -> {RCODE} = 1 ; success
          {RCODE} = -1 ; error
          {RCODE} = 0 ; domain name does not exist
          {RCODE} = -3 ; invalid contact ID
          {RCODE} = -4 ; invalid password

$ref    -> {TRSTATUS} ; transfer status
$ref    -> {MSG}      ; error messages

```

#### 4.2.9 EPPdomaintransferoperation

The Domain transfer operation function is used to operate on a transfer request of a domain. There are three different operations that can be issued: Approve, Reject, Cancel. Approve and reject operations can only be issued by the current sponsoring registrar of the domain. Only the registrar that requested the transfer can issue the cancel operation.

Transfer Operation Code: Approve 1, Reject 2, Cancel 3

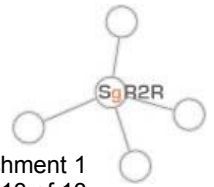
(\%ref) = EPPdomaintransferoperation(\%dominfo)

```

; Domain Information: %dominfo
; Domain Name:      NAME;
; Operation:        OPERATION;

$ref    -> {RCODE} = 1 ; success
          {RCODE} = -1 ; error

```



```

{RCODE} = 0 ; No transfer request outstanding for this
            domain
$ref    -> {RCODE} = -2 ; operation code invalid
        -> {MSG}      ; error messages

```

#### 4.2.10 EPPdomaintransferstatus

The Domain transfer status function is used to request the transfer status of a registered domain. This command should be used by the winning registrar to check if a domain have already been successfully transferred to them or have been rejected by the current registrar. If so, they should update their own database to reflect the change as well as to inform their customer about the event.

Transfer Status Code: Pending 1, Rejected 2

```

(\%ref) = EPPdomaintransferstatus($domainname)
$ref    -> {RCODE} = 0 ; No transfer request outstanding for this
            domain
            {RCODE} = 1 ; success
            {RCODE} = -1 ; error
$ref    -> {TRSTATUS} ; transfer status
$ref    -> {REID}      ; requesting registrar ID
$ref    -> {REDATE}   ; transfer request date
$ref    -> {ACID}     ; ID of registrar to accept request
$ref    -> {ACDATE}   ; deadline for accepting request
$ref    -> {MSG}      ; error messages

```

#### 4.2.11 EPPdomainrenew

The Domain Renew function is used to extend the terms of a domain. Domain periods are specified in number of years.

Note: Date format: DD-MMM-YYYY, e.g. 24-DEC-2002

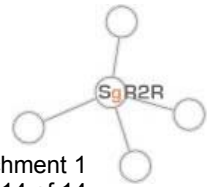
```

(\%ref) = EPPdomainrenew(\%dominfo)
            ; Domain Information: %dominfo
            ; Domain name:      NAME;
            ; Current exp date:  CUREXPIREDATE;
            ; Period to be added: TERM;
$ref    -> {RCODE} = 0 ; invalid domain name
            {RCODE} = 1 ; success
            {RCODE} = -1 ; error
            {RCODE} = -2 ; invalid expiry date format
            {RCODE} = -3 ; invalid term period
            {RCODE} = -4 ; new expiry date beyond allowable limit
$ref    -> {EXPIREDATE} ; new expiry date for domain
$ref    -> {MSG}      ; error messages

```

#### 4.2.12 EPPdomainupdatewhoisprofile

The Domain Update Whoisprofile function is used to update the whois profile for a particular contact for a domain. The contact type must be provided.



Contact type: Registrant 1, admin 2, tech 3, billing 4, agent 5  
Whois profile flag: 1 = Yes, 2 = No

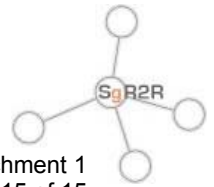
```
(\%ref) = EPPdomainupdatewhoisprofile( \%profile )
; Domain Information: %profile
; Domain Name:      NAME
; Contact type:     CONTACTTYPE
; Address1 Flag:    ADDRESS1FLAG
; Address2 Flag:    ADDRESS2FLAG
; Phone1 Flag:      PHONE1FLAG
; Phone2 Flag:      PHONE2FLAG
; Phone3 Flag:      PHONE3FLAG
; Email Flag:       EMAILFLAG
$ref    -> {RCODE} = 1 ; success
          {RCODE} = 2 ; success, but some of the flags are
mandatory and cannot be changed
          {RCODE} = 0 ; invalid domain name
          {RCODE} = -1 ; error
          {RCODE} = -2 ; invalid contact type
          {RCODE} = -3 ; invalid flag
$ref    -> {MSG}      ; error messages
```

#### 4.2.13 EPPdomaininfowhoisprofile

The EPPdomaininfowhoisprofile function is used to retrieve the whois profile information for a particular contact for a domain. The contact type must be provided.

Contact type: Registrant 1, admin 2, tech 3, billing 4, agent 5  
Whois profile flag: 1 = Yes, 2 = No

```
(\%ref) = EPPdomaininfowhoisprofile( \%info )
; Domain Information: %info
; Domain Name:      NAME
; Contact type:     CONTACTTYPE
$ref    -> {ADDRESS1FLAG} ; Address1 Flag
$ref    -> {ADDRESS2FLAG} ; Address2 Flag
$ref    -> {PHONE1FLAG}   ; Phone1 Flag
$ref    -> {PHONE2FLAG}   ; Phone2 Flag
$ref    -> {PHONE3FLAG}   ; Phone3 Flag
$ref    -> {EMAILFLAG}    ; Email Flag
$ref    -> {RCODE} = 1 ; success
          {RCODE} = 0 ; invalid domain name
          {RCODE} = -2 ; invalid contact type
          {RCODE} = -1 ; error
$ref    -> {MSG}      ; error messages
```



### 4.3 Contact Operations

The Contact Operations commands allow you to create, modify and update contacts. Both organizational contacts as well as personal contacts could be created using the same set of commands.

#### 4.3.1 EPPcontactcreate

The Create Contact function is used to create new contacts. The server will return a new contact ID in the response. For a company, input the ORGNAME field, for a personal contact, input the firstname and lastname fields. Note that these are mutually exclusive fields, meaning that if you entered the ORGNAME, you cannot have FIRSTNAME and LASTNAME. If you wish to create a parent organization for an individual, you must create a contact ID for each entity and then specify the relationship using the domain operation commands.

Contact type: Company: 1, Personal: 2

Country code: Use ISO3166 two character code

```

(\%ref) = EPPcontactcreate(\%contact) ; the required fields in %contact include
; Organization name: ORGNAME
; First name: FIRSTNAME
; Last name: LASTNAME
; Contact type: TYPE
; RCB ID: RCBID
; Personal ID: PERID
; Password: PASSWORD
; Street Address 1a: STREET1A
; Street Address 1b: STREET1B
; Street Address 1c: STREET1C
; Country 1: COUNTRY1
; Postal Code 1: POSTALCODE1
; Street Address 2a: STREET2A
; Street Address 2b: STREET2B
; Street Address 2c: STREET2C
; Country 2: COUNTRY2
; Postal Code 2: POSTALCODE2
; Telephone: PHONE1
; Fax Number: PHONE2
; Pager/Mobile Number: PHONE3
; Email: EMAIL

$ref -> {RCODE} = -1 ; error
      {RCODE} = 1 ; success
      {RCODE} = -2 ; Incorrect parent organization code

$ref -> {MSG} ; error message
$ref -> {CONTACTID} ; ID of the new contact

```

#### 4.3.2 EPPcontactinfo

The Obtain Contact Information function retrieves the contact information based on a particular Contact ID. If the contact is an organization, the ORGNAME and RCBID (if a SG local company) will be returned. If the contact is a person, the FIRSTNAME, LASTNAME, and PERID will be returned.



```
(\%ref) = EPPcontactinfo($contactid)
$ref    -> {FIRSTNAME}    ; first name
$ref    -> {LASTNAME}     ; last name
$ref    -> {ORGNAME}      ; organization name
$ref    -> {TYPE}         ; type
$ref    -> {RCBID}        ; RCB Number
$ref    -> {PERID}        ; Personal ID
$ref    -> {PASSWORD}     ; Password
$ref    -> {STREET1A}     ; Street address 1a
$ref    -> {STREET1B}     ; Street address 1b
$ref    -> {STREET1C}     ; Street address 1c
$ref    -> {COUNTRY1}    ; Country 1
$ref    -> {POSTALCODE1} ; Postal Code 1
$ref    -> {STREET2A}     ; Street address 2a
$ref    -> {STREET2B}     ; Street address 2b
$ref    -> {STREET2C}     ; Street address 2c
$ref    -> {COUNTRY2}    ; Country 2
$ref    -> {POSTALCODE2} ; Postal Code 2
$ref    -> {PHONE1}      ; Phone number 1
$ref    -> {PHONE2}      ; Fax number
$ref    -> {PHONE3}      ; Pager/Mobile number
$ref    -> {EMAIL}       ; Email
$ref    -> {RCODE} = 0   ; No contact info available
$ref    -> {RCODE} = 1   ; success
$ref    -> {RCODE} = -1  ; error
$ref    -> {MSG}         ; error messages
```

### 4.3.3 EPPcontactupdatechange

The Update Contact Change function is used to update contacts information.

(\%ref) = EPPcontactupdatechange(\%contact); the required fields in %contact include

```
; Contact ID: CONTACTID
; Organization name: ORGNAME
; First name: FIRSTNAME
; Last name: LASTNAME
; RCB ID: RCBID
; Personal ID: PERID
; Password: PASSWORD
; Street Address 1a: STREET1A
; Street Address 1b: STREET1B
; Street Address 1c: STREET1C
; Country 1: COUNTRY1
; Postal Code 1: POSTALCODE1
; Street Address 2a: STREET2A
; Street Address 2b: STREET2B
; Street Address 2c: STREET2C
; Country 2: COUNTRY2
; Postal Code 2: POSTALCODE2
; Telephone: PHONE1
; Fax Number: PHONE2
; Pager/Mobile Number: PHONE3
```



```

; Email: EMAIL
$ref -> {RCODE} = 0 ; Contact ID does not exist
      {RCODE} = 1 ; success
      {RCODE} = -1 ; error
$ref -> {MSG} ; error message

```

#### 4.3.4 EPPcontactdelete

The Delete Contact function is used to delete a contact. This operation will fail if a contact still has outstanding domains assigned to it.

(\%ref) = EPPcontactdelete(\$contactid)

```

$ref -> {RCODE} = 0 ; contact ID does not exist
      {RCODE} = 1 ; success
      {RCODE} = -1 ; error
      {RCODE} = -2 ; fail – contact has association with other
                   objects
$ref -> {MSG} ; error message

```

### 4.4 Host Operations

Host Operations are concerned with creating hostnames under a particular domain name. The Host Operations allow the creation, modification and deletion of hostnames. If you wish to set a hostname as a name server for a domain name, you should use the EPPdomainupdate commands.

#### 4.4.1 EPPhostcreate

The Create host function is used to create new hostnames (name servers to be used for domain names). Note that a .SG host name can only be created if the corresponding .SG domain has been created, and that authenticity that the domain is currently sponsored by the requesting registrar is established.

The DOMAINID field contains the SGNIC handle for the domain of this host name. If the host's parent domain ID is a pragmatic domain under revision, the host will be automatically deleted by the system if the pragmatic domain application is rejected.

```

(\%ref) = EPPhostcreate(\%host) ; the required fields in %contact include
                                ; Name: NAME
                                ; Parent Domain ID: PARENTDOMAINID
                                ; IP Address 1: IP1
                                ; IP Address 2: IP2
                                ; IP Address 3: IP3
                                ; ... up to 13 IP addresses
$ref -> {ID} ; new host ID
$ref -> {RCODE} = 0 ; host name already in use
      {RCODE} = 1 ; success
      {RCODE} = 2 ; host created. Parent domain ID refers to a
                   pragmatic domain
      {RCODE} = -1 ; error

```



```

{RCODE} = -2 ; This registrar is not authorized to create a
             host for this domain
{RCODE} = -3 ; Invalid parent domain ID
{RCODE} = -4 ; Out-of-zone host name does not require
             parent domain ID
{RCODE} = -5 ; Parent domain name does not match host
             name
$ref      -> {MSG} ; error message

```

#### 4.4.2 EPPhostinfo

The host info function is used to retrieve information about a host.

```

(\%ref) = EPPhostinfo($hostname)
$ref    -> {NAME} ; host name
$ref    -> {PARENTDOMAINID} ; Parent Domain ID
$ref    -> {IP1} ; IP Address 1
$ref    -> {IP2} ; IP Address 2
$ref    -> {IP3} ; IP Address 3
             ; ... up to 13 IP addresses
$ref    -> {RCODE} = 0 ; host name does not exist
             {RCODE} = 1 ; success
             {RCODE} = -1 ; error
$ref    -> {MSG} ; error messages

```

#### 4.4.3 EPPhostupdateadd

The host update add function is used to add IP addresses to a host. (a maximum of 13 IP addresses may be included for each host)

```

(\%ref) = EPPhostupdateadd(\%host) ; the fields in %host include
             ; Hostname: NAME
             ; IP Address 1: IP1
             ; IP Address 2: IP2
             ; IP Address 3: IP3
             ; ... up to 13 IP addresses
$ref    -> {RCODE} = 0 ; host name does not exist
             {RCODE} = 1 ; success
             {RCODE} = -1 ; error
             {RCODE} = -2 ; No more IP address can be added to this
             host
$ref    -> {MSG} ; error message

```

#### 4.4.4 EPPhostupdateremove

The host update remove function is used to remove IP addresses from a host.

```

(\%ref) = EPPhostupdateremove(\%host) ; the fields in %host include
             ; Hostname: NAME
             ; IP Address 1: IP1
             ; IP Address 2: IP2
             ; IP Address 3: IP3
             ; ... up to 13 IP addresses
$ref    -> {RCODE} = 0 ; host name does not exist

```



```
{RCODE} = 1 ; success
{RCODE} = -1 ; error
{RCODE} = -2 ; No such IP address(es) associated with
              this host
$ref      -> {MSG} ; error message
```

#### 4.4.5 EPPhostdelete

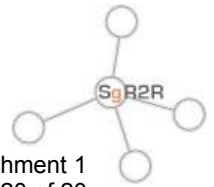
The host delete function is used to delete a host.

```
(\%ref) = EPPhostdelete($hostname)
$ref      -> {RCODE} = 0 ; host name does not exist
           {RCODE} = 1 ; success
           {RCODE} = -1 ; error
           {RCODE} = -2 ; fail - host has association with other
                         objects
$ref      -> {MSG} ; error message
```

### 4.5 Batch Processing

The EPPbatch command could be used to send a batch of EPP commands to the server. The command takes an input file (file name \$inputfile) with a batch of EPP XML commands and output the responses to an output file (filename \$outputfile). The commands in the input file should be well-formed XML EPP formatted commands. (NOT the SgR2R.EPP API function calls)

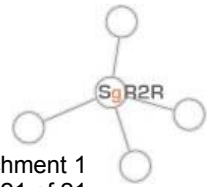
```
EPPbatch($inputfile, $outputfile)
```



## Appendix A: Sample Configuration File

This is a sample of the registrar.conf file. This file MUST be stored under the directory: /Neteka/R2R/conf/ for the SgR2R API to function properly.

```
# Copyright(C) 2002 Neteka Inc.
#
# Registrar.conf
#
#   Created: Feb 15, 2002
#   Created by:
#       David Leung (david@neteka.com)
#       Jimmy Lam (jimmy@neteka.com)
#       Henry Tong (henry@neteka.com)
#
# Description:
# - SGR2R Registrar Package Configuration file
#
#
# General Variable
#
# DO NOT MODIFY THIS SECTION
#
$_Protocol = "R2R";           #Protocol Name
$_Version = "1.0";          #Version Number
#
# SSL Proxy information
#
$R2R_Server = "sslproxy";
$R2R_Port = "123";
$_SecretKey = "R2RServe";
#
# Registrar ID
#
$REGISTRAR_ID = "RRID-ADM965920";
$REGISTRAR_PASSWORD = "neteka";
```



## Appendix B: Glossary of SgR2R.EPP Commands

### Session Management

- 4.1.1 EPPlogin
- 4.1.2 EPPlogout
- 4.1.3 EPPhello

### Domain Operations

- 4.2.1 EPPdomaincheck
- 4.2.2 EPPdomaincreate
- 4.2.3 EPPdomaininfo
- 4.2.4 EPPdomainupdateadd
- 4.2.5 EPPdomainupdateremove
- 4.2.6 EPPdomainupdatechange
- 4.2.7 EPPdomaindelete
- 4.2.8 EPPdomaintransferrequest
- 4.2.9 EPPdomaintransferoperation
- 4.2.10 EPPdomaintransferstatus
- 4.2.11 EPPdomainrenew
- 4.2.12 EPPdomainupdatewhoisprofile
- 4.2.13 EPPdomaininfowhoisprofile

### Contact Operations

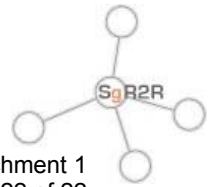
- 4.3.1 EPPcontactcreate
- 4.3.2 EPPcontactinfo
- 4.3.3 EPPcontactupdatechange
- 4.3.4 EPPcontactdelete

### Host Operations

- 4.4.1 EPPhostcreate
- 4.4.2 EPPhostinfo
- 4.4.3 EPPhostupdateadd
- 4.4.4 EPPhostupdateremove
- 4.4.5 EPPhostdelete

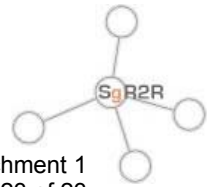
### Batch Processing

- 4.5 EPPbatch



## Appendix C: List of Global Error Code

- 100 – Transaction fail. Not enough credit in registrar account
- 101 – Domain name translation error
- 102 – Registrar status error
- 103 – TLD accreditation error
- 104 – Current domain status prohibit this action
- 105 – This domain status change is not allowed



## Appendix D: Change History

V1.0 Add package SGR2R::API::XML\_Checker