



HTTP API Documentation

Document version:
Date:

1.0
April 2010

Table of Contents

1. Introduction	4
2. Connecting	5
3. Domain Commands	8
3.1 Check domain	9
3.2 Info domain	10
3.3 Create domain	11
3.4 Update domain	15
3.5 Renew domain	19
3.6 Transfer domain	20
3.7 Transfercancel domain	24
3.8 Transferrespond domain	25
3.9 Trade domain	26
3.10 Delete domain	29
4. Contact Commands	30
4.1 Check contact	31
4.2 Info contact	32
4.3 Create contact	33
4.4 Update contact	35
4.5 Delete contact	37
5. API Response Codes	38
6. Appendix - PHP Examples	42

1. Introduction

This documentation is designed for people with a technical point of view. The intention of this documentation is to give some insight into the inner workings of the HTTP API.

However for a full implementation of the API technical knowledge of a programming language like PHP, Java, VB etc. is a requirement.

Also understanding of the concepts of JSON and HTTP requests as a whole would be required. More information on those topics can be found at the following websites.

JSON:

- [JSON.org \(http://www.json.org/\)](http://www.json.org/)
- [Wikipedia.org \(http://en.wikipedia.org/wiki/JSON\)](http://en.wikipedia.org/wiki/JSON)

HTTP:

- [Wikipedia.org \(http://nl.wikipedia.org/wiki/Hypertext_Transfer_Protocol\)](http://nl.wikipedia.org/wiki/Hypertext_Transfer_Protocol)

cURL:

- [Wikipedia.org \(http://en.wikipedia.org/wiki/CURL\)](http://en.wikipedia.org/wiki/CURL)
- [cURL Homepage \(http://curl.haxx.se/\)](http://curl.haxx.se/)

cURL is commonly used in many languages to establish HTTP connections with external sources. Communication with the HTTP API is done by sending POST requests to the API URL. All other types of HTTP requests will be denied.

2. Connecting

Connection

To connect to the HTTP API a HTTP POST request has to be made at the API location.

The API is available on two different locations. A production environment and a test environment.

Production: <https://httpapi.yoursrs.com/v1/>

Test: <https://httpapi.realtimeregister-ote.com/v1/>

Versioning

Versioning of the API takes place inside the API location URL. When a newer version of the API is being released the old API will continue to be supported for a few more months.

JSON

The body of each POST request should be entirely JSON encoded.
For further documentation about JSON please visit: <http://json.org/>

The API responses will be JSON encoded as well.

Requests

The API expects requests to be made using a http POST method.

The body of the POST request will need to be a JSON object containing at least the following basic format:

```
{
  "login_handle": "<login_handle>",
  "login_pass": "<login_pass>"
}
```

This will be sufficient for requests that do not support additional parameters like check, info and delete requests. However most requests will require more parameters to be provided. For more information on the parameters to be used for each specific command please check the corresponding command under either "Domain Commands" or "Contact Commands".

Each request made uses its own resource based URL.

The URL syntax is:

```
https://<api_url>/<version>/<resource_type>/<resource_handle>/<resource_action>
```

Breakdown

api_url	The URL used to connect to the http_api of Realtime Register. Either the test or live environment URL should be used here.
version	The version of the http-api to be used. Format is v + the version number, like v1
resource_type	The type of resource your command will target. Can be either "domains" or "contacts".
resource_handle	The handle of the resource the command will target.
resource_action	The action to perform on the resource.

The type should be chosen depending on the handle you'd like to work with.

If for example you'd like to know if a certain domain name is available a check domains command is what you'd be looking for.

The URL for such a command would look similar to:

```
https://httpapi.yoursrs.com/v1/domains/example.com/check
```

A list of all available actions for each type of resource can be found under Domain Commands and Contact Commands.

DateTime format

Please note that the DateTime format used in requests and responses is in unix timestamp format. For more information on the unix timestamp format, visit http://en.wikipedia.org/wiki/Unix_timestamp

Parameters

Most commands require additional parameters to be supplied with the request. For example a domain create command requires nameserver data and contact handles.

Those parameters should be supplied inside the JSON body of the POST request. A list of all available parameters for each request can be found below.

Responses

The API responses are formatted as JSON Objects. The general layout of those responses is similar for each type of request.

```
{
  "command": "<resource_type>:<resource_action>",
  "handle": "<resource_handle>",
  "code": "<resonse_code>",
  "msg": "<response_message>",
  "error": [
    "<error_message_1>",
    "<error_message_2>"
  ],
  "params": {
    "<key>": "<value>",
    "<key>": "<value>"
  },
  "svTRID": "<server_transaction_id>",
  "response": {
    "<key>": "<value>",
    "<key>": "<value>"
  }
}
```

Key	Description
command	The type of command the API response applies to. This is usually the command you requested, but can in some cases also be a sub-task the API needed to perform in order to finish the main request.
handle	The resource handle provided for the request.
code	The response code. For all possible response codes and their meanings see chapter 5.
msg	The corresponding response message. Each response code has it's own textual message to give a better description of the response code.
error	An array containing all the error messages (if any) belonging to the request.
params	An array containing all the parameters provided for the request. This is useful for debugging perposes as can be seen how certain parameters are received by the API.
svTRID	The YourSRS server's transaction ID. This ID is unique and identifies your specific request. Use this in all support requests.
response	Object containing response variables. This object can be empty when the request did not result in any response variables. This can be because of the type of the request (eg. create, update, delete) or because of an error that occurred during the request.

The above keys will always be present in each response. In addition a key "children" can be present as well containing responses from sub-requests made during the process of your request. Those sub-requests all resulted in their own response object with the same layout of the main response.



In case any of the children resulted in an error the main response will use that child's response code and message as its own.

In addition an error message with the following format will be set as part of the main response.

```
Command '<command>' with handle '<handle>' resulted in the following response message: '<message>'
```

3. Domain Commands

Contains functionality to manage domains. Calls made to the resource type domains require a domain_handle and one of the domain actions listed below.

- Check domain
- Info domain
- Create domain
- Update domain
- Renew domain
- Transfer domain
- Transfercancel domain
- Transferrespond domain
- Trade domain
- Delete domain

Check domain

Checks the availability of a domain.

Parameters

No additional parameters available for check domain.

Request

```
Method: POST
URL:    https://httpapi.yoursrs.com/v1/domains/example.com/check
BODY:
{
  "login_handle": "example_user",
  "login_pass": "*****"
}
```

Response

```
{
  "command": "domain:check",
  "handle": "example.com",
  "code": 1000,
  "msg": "Command completed successfully",
  "error": [
  ],
  "params": {
    "handle": "example.com"
  },
  "svTRID": "158759-2-yoursrs",
  "response": {
    "avail": "0"
  }
}
```

Info domain

Retrieve information about a certain domain.

Parameters

No additional parameters available for info domain.

Request

```
Method: POST
URL:    https://httpapi.yoursrs.com/v1/domains/example.com/info
BODY:
{
  "login_handle": "example_handle",
  "login_pass": "*****"
}
```










Response

```
{
  "command": "domain:info",
  "handle": "example.com",
  "code": 1000,
  "msg": "Command completed successfully",
  "error": [
  ],
  "params": {
    "handle": "example.com"
  },
  "svTRID": "158833-2-yoursrs",
  "response": {
    "name": "example.com",
    "roid": "8444-DOMAIN",
    "status": "requested",
    "registrant": "example_registrant_handle",
    "billing": "example_billing_handle",
    "tech": "example_tech_handle",
    "admin": "example_admin_handle",
    "ns": [
      {
        "host": "ns1.example.com"
      },
      {
        "host": "ns2.example.com"
      },
      {
        "host": "ns3.example.com"
      },
      {
        "host": "ns4.example.com",
        "addr": "1.2.3.4"
      }
    ]
  },
  "renew": true,
  "clID": "example_user",
  "crDate": 1257526246,
  "upDate": 1257526246,
  "exDate": 1289062246
}
```

Create domain

Register a new domain.

Parameters

Key	Required	Description
ns		Array of nameservers
template		DNS template ID
template_link		Boolean
autorenew		Boolean
registrant		Contact handle to be used for registrant
admin		Contact handle to be used for admin
tech		Contact handle to be used for tech
billing		Contact handle to be used for billing
contact_data		Array of new contact handles to be created



ns

Either an array of nameservers or a valid template ID are required for each domain create command.



registrant, admin, tech, billing

The 'registrant', 'admin', 'billing', 'tech' contact handles are required unless a new contact has to be created for that handle.

In that case the corresponding contact data have to be provided under 'contact_data'.



contact_data

Contact data is an optional associative array containing data for new contact handles to be created. These can be supplied for any of the registrant, admin, tech or billing handles.

If contact data is supplied for any of the handles in the contact_data array the corresponding handle should be left empty.

Request

```
Method: POST
URL: https://httpapi.yoursrs.com/v1/domains/example.com/create
BODY:
{
  "autorenew":true,
  "template":null,
  "template_link":null,
  "ns":[
    {
      "host":"ns1.example.com"
    },
    {
      "host":"ns2.example.com"
    },
    {
      "host":"ns3.example.com"
    },
    {
      "host":"ns4.example.com",
      "addr":"1.2.3.4"
    }
  ],
  "registrant":"example_registrant_handle",
  "admin":"example_admin_handle",
  "tech":"example_tech_handle",
  "billing":"example_billing_handle",
  "contact_data":[]
},
"login_handle":"example_handle",
"login_pass":"*****"
}
```

Response

```
{
  "command": "domain:create",
  "handle": "example.com",
  "code": 1001,
  "msg": "Command completed successfully; action pending",
  "error": [
  ],
  "params": {
    "autorenew": true,
    "template": null,
    "template_link": null,
    "ns": {
      {
        "host": "ns1.example.com"
      },
      {
        "host": "ns2.example.com"
      },
      {
        "host": "ns3.example.com"
      },
      {
        "host": "ns4.example.com",
        "addr": "1.2.3.4"
      }
    },
    "registrant": "example_registrant_handle",
    "admin": "example_admin_handle",
    "tech": "example_tech_handle",
    "billing": "example_billing_handle",
    "contact_data": [
    ],
    "handle": "example.com"
  },
  "svTRID": "158765-6-yoursrs",
  "response": {
    "domain": "example.com",
    "crDate": 1257526246,
    "exDate": 1289062246
  },
}
```

```

"children":[
  {
    "command":"contact:check",
    "handle":"example_registrant_handle",
    "code":1000,
    "msg":"Command completed successfully",
    "error":[

    ],
    "params":[

    ],
    "svTRID":"158765-2-yoursrs",
    "response":{"
      "example_registrant_handle":"0",
      "avail":"0"
    }
  },
  {
    "command":"contact:check",
    "handle":"example_admin_handle",
    "code":1000,
    "msg":"Command completed successfully",
    "error":[

    ],
    "params":[

    ],
    "svTRID":"158765-3-yoursrs",
    "response":{"
      "example_admin_handle":"0",
      "avail":"0"
    }
  },
  {
    "command":"contact:check",
    "handle":"example_tech_handle",
    "code":1000,
    "msg":"Command completed successfully",
    "error":[

    ],
    "params":[

    ],
    "svTRID":"158765-4-yoursrs",
    "response":{"
      "example_tech_handle":"0",
      "avail":"0"
    }
  },
  {
    "command":"contact:check",
    "handle":"example_billing_handle",
    "code":1000,
    "msg":"Command completed successfully",
    "error":[

    ],
    "params":[










    ],
    "svTRID":"158765-5-yoursrs",
    "response":{"
      "example_billing_handle":"0",
      "avail":"0"
    }
  }
]
}

```

Update domain

Update an existing domain.

Parameters

Key	Required	Description
ns		Array of nameservers
autorenew		Boolean
template		DNS template ID
template_link		Boolean
admin		Contact handle to be used for admin
billing		Contact handle to be used for billing
tech		Contact handle to be used for tech
contact_data		Array containing information about new contact handles.
lock		Boolean



Parameters requirement

For a domain update non of the parameters is required and only the parameters that are delivered will be used to update the domain.



Remove data

In order to remove data you will have to send the parameter with an empty string. The only exception to this is the ns parameter, which will require an empty array.

 **ns**

Either an array of nameservers or a valid template ID are required for each domain create command. To remove nameservers from a domain you can nest another array inside the ns array called remove. The same goes for adding domains.

PHP Example:

```
$params['ns'] = array(
    'add' = array(
        0 => array(
            'host' => 'example_ns_host',
            'addr' => '123.123.123.123'
        )
    ),
    'remove' = array(
        0 => array(
            'host' => 'example_ns_host2'
        )
    )
);
```

In the above example the nameservers under 'add' will be added to the list of nameservers and the nameservers under 'remove' will be removed from the list of nameservers. Alternatively you can provide an array of nameservers like the example under create domain without the 'add' and 'remove' tags. In that case the server will automatically decide which nameservers to add and which are to be removed. **Note that in that case the list provided should always contain all the nameservers you want to use for the domain as any nameserver currently in use for the domain that's not in the list of provided nameservers will be removed.**

Request

```
Method: POST
URL: https://httpapi.yoursrs.com/v1/domains/example.com/update
BODY:
{
  "auto_renew":"false",
  "lock":"true",
  "template":"",
  "template_link":"",
  "ns":[
    {
      "host":"ns1.example.com",
      "addr":"123.123.123.123"
    },
    {
      "host":"ns2.example.com",
      "addr":"123.123.123.124"
    },
    {
      "host":"ns3.example.com",
      "addr":"123.123.123.125"
    }
  ],
  "login_handle":"example_handle",
  "login_pass":"*****"
}
```

Response

```
{
  "command":"domain:update",
  "handle":"example.com",
  "code":1001,
  "msg":"Command completed successfully; action pending",
  "error":[
  ],
  "params":{
    "auto_renew":"false",
    "lock":"true",
    "template":"",
    "template_link":"",
    "ns":[
      {
        "host":"ns1.example.com",
        "addr":"123.123.123.123"
      },
      {
        "host":"ns2.example.com",
        "addr":"123.123.123.124"
      },
      {
        "host":"ns3.example.com",
        "addr":"123.123.123.125"
      }
    ]
  },
  "handle":"example.com"
},
"svTRID":"158872-3-yoursrs",
"response":[
],
```

```

"children":[
  {
    "command":"domain:info",
    "handle":"example.com",
    "code":1000,
    "msg":"Command completed successfully",
    "error":[

    ],
    "params":[


    ],
    "svTRID":"158872-2-yoursrs",
    "response":{
      "name":"example.com",
      "roid":"8444-DOMAIN",
      "status":"ok",
      "registrant":"example_registrant_handle",
      "billing":"example_billing_handle",
      "tech":"example_tech_handle",
      "admin":"example_admin_handle",
      "ns":[
        {
          "host":"ns1.example.com"
        },
        {
          "host":"ns2.example.com"
        },
        {
          "host":"ns3.example.com"
        },
        {
          "host":"ns4.example.com",
          "addr":"1.2.3.4"
        }
      ],
      "renew":true,
      "clID":"example_user",
      "crDate":1257526246,
      "upDate":1257759087,
      "exDate":1289062246
    }
  }
]
}

```

Renew domain

Renew a domain registration period.

Parameters

Key	Required	Description
curExpDate		Timestamp of the current expiration date. This value ensures that repeated <renew> commands do not result in multiple unanticipated successful renewals.



The parameter 'curExpDate' has to be the same as the value exDate you'd get from a domain:info request on the domain. If the date does not match a 2306 response code will be returned.

Request

```
Method: POST
URL: https://httpapi.yoursrs.com/v1/domains/example.com/renew
BODY:
{
  "curExpDate":1289062246,
  "login_handle":"example_handle",
  "login_pass":"*****"
}
```











Response

```
{
  "command":"domain:renew",
  "handle":"example.com",
  "code":1000,
  "msg":"Command completed successfully",
  "error":[
  ],
  "params":{
    "curExpDate":1289062246,
    "handle":"example.com"
  },
  "svTRID":"159688-2-yoursrs",
  "response":{
    "domain":"example.com",
    "exDate":1320598246
  }
}
```

Transfer domain

Transfer a domain.

Parameters

Key	Required	Description
ns		Array of nameservers
template		DNS template ID
template_link		Boolean
auth		The auth code of the domain
registrant		The registrant handle
admin		The admin handle
tech		The tech handle
billing		The billing handle
request_type		The request type: "transfer" or "trade"
contact_data		Array of new contact handles to be created

ns
Either an array of nameservers or a valid template ID are required for each domain create command.

registrant, admin, tech, billing
The 'registrant', 'admin', 'billing', 'tech' contact handles are required unless a new contact has to be created for that handle. In that case the corresponding contact data have to be provided under 'contact_data'.

contact_data
Contact data is an optional associative array containing data for new contact handles to be created. These can be supplied for any of the registrant, admin, tech or billing handles. If contact data is supplied for any of the handles in the contact_data array the corresponding handle should be left empty.

Request

```
Method: POST
URL: https://httpapi.yoursrs.com/v1/domains/example.com/transfer
BODY:
{
  "auth": "byqb2h5sif3x",
  "template": null,
  "template_link": null,
  "request_type": "transfer",
  "ns": [
    {
      "host": "ns1.example.com"
    },
    {
      "host": "ns2.example.com"
    },
    {
      "host": "ns3.example.com"
    },
    {
      "host": "ns4.example.com",
      "addr": "1.2.3.4"
    }
  ],
  "registrant": "example_registrant_handle",
  "admin": "example_admin_handle",
  "tech": "example_tech_handle",
  "billing": "example_billing_handle",
  "contact_data": null
}
```

Response

```
{
  "command": "domain:transfer",
  "handle": "example.com",
  "code": 1000,
  "msg": "Command completed successfully",
  "error": [],
  "params": {
    {
      "auth": "byqb2h5sif3x",
      "template": null,
      "template_link": null,
      "request_type": "transfer",
      "ns": [
        {
          {
            "host": "ns1.example.com"
          },
          {
            "host": "ns2.example.com"
          },
          {
            "host": "ns3.example.com"
          },
          {
            "host": "ns4.example.com",
            "addr": "1.2.3.4"
          }
        ],
        "registrant": "example_registrant_handle",
        "admin": "example_admin_handle",
        "tech": "example_tech_handle",
        "billing": "example_billing_handle",
        "contact_data": [],
        "handle": "example.com"
      },
      "svTRID": "457655-6-yoursrs",
      "response": {
        {
          "domain": "example.com",
          "trStatus": "requested",
          "reID": "example_user",
          "reDate": 1268045605,
          "acID": "unknown",
          "acDate": 1268564008
        },
        "children": [
          {
            {
              "command": "contact:check",
              "handle": "example_registrant_handle",
              "code": 1000,
              "msg": "Command completed successfully",
              "error": [],
              "params": [],
              "svTRID": "457655-2-yoursrs",
              "response": {
                {
                  "example_registrant_handle": "0",
                  "avail": "0"
                }
              }
            },
          ],
        },
      },
    }
  }
}
```

```

{
  {
    "command": "contact:check",
    "handle": "example_admin_handle",
    "code": 1000,
    "msg": "Command completed successfully",
    "error": [],
    "params": [],
    "svTRID": "457655-3-yoursrs",
    "response":
    {
      "example_admin_handle": "0",
      "avail": "0"
    }
  },
  {
    "command": "contact:check",
    "handle": "example_tech_handle",
    "code": 1000,
    "msg": "Command completed successfully",
    "error": [],
    "params": [],
    "svTRID": "457655-4-yoursrs",
    "response":
    {
      "example_tech_handle": "0",
      "avail": "0"
    }
  },
  {
    "command": "contact:check",
    "handle": "example_billing_handle",
    "code": 1000,
    "msg": "Command completed successfully",
    "error": [],
    "params": [],
    "svTRID": "457655-5-yoursrs",
    "response":
    {
      "example_billing_handle": "0",
      "avail": "0"
    }
  }
}
]

```

Transfercancel domain

Cancel a transfer request.

Parameters

No additional parameters available for domain transfercancel.

Request

```
Method: POST
URL:    https://httpapi.yoursrs.com/v1/domains/example.com/transfercancel
BODY:
{
  "login_handle": "example_handle",
  "login_pass": "*****"
}
```

Response

```
{
  "command": "domain:transfercancel",
  "handle": "example.com",
  "code": 2301,
  "msg": "Object not pending transfer",
  "error": [
  ],
  "params": {
    "handle": "example.com"
  },
  "svTRID": "158887-2-yoursrs",
  "response": [
  ]
}
```

Transferrespond domain

Approve or decline a transfer request.

Parameters

Key	Required	Description
auth	<input checked="" type="checkbox"/>	The auth code of the domain
approve	<input checked="" type="checkbox"/>	Boolean

Request

```
Method: POST
URL: https://httpapi.yoursrs.com/v1/domains/example.com/transferrespond
BODY:
{
  "auth": "byqb2h5sif3x",
  "approve": "true",
  "login_handle": "test_user",
  "login_pass": "12345"
}
```

Response


```
{
  "command": "domain:transferrespond",
  "handle": "example.com",
  "code": 1000,
  "msg": "Command completed successfully",
  "error": [],
  "params": {
    "auth": "byqb2h5sif3x",
    "approve": "true",
    "handle": "example.com"
  },
  "svTRID": "158885-1-yoursrs",
  "response": [
  ]
}
```


Trade domain

Trade a domain to a different registrant.

Parameters

Key	Required	Description
registrant		The registrant handle
contact_data		Contact data for a new contact

 **registrant**
Use the registrant handle to trade a domain to an already existing contact.

 **contact_data**
The contact_data parameter is meant to create a new contact handle.

Request

```
Method: POST
URL: https://httpapi.yoursrs.com/v1/domains/example.com/trade
BODY:
{
  "contact_data":{
    "registrant":{
      "handle":null,
      "email":"info@example.com",
      "name":"John Doe",
      "org":"Example Organization",
      "street":[
        "Example Lane 128",
        "Example Additional Address Line 1",
        "Example Additional Address Line 2"
      ],
      "city":"Example City",
      "sp":"Example State or Province",
      "pc":"1234PC",
      "cc":"NL",
      "voice":"+31.123456789",
      "fax":"+31.123456789",
      "gender":"male",
      "lang":"EN"
    }
  },
  "login_handle":"example_handle",
  "login_pass":"*****"
}
```

Response

```
{
  "command": "domain:trade",
  "handle": "example.com",
  "code": 1001,
  "msg": "Command completed successfully; action pending",
  "error": [
  ],
  "params": {
    "handle": "example.com",
    "contact_data": {
      "registrant": {
        "handle": "new_example_handle",
        "email": "info@example.com",
        "name": "John Doe",
        "org": "Example Organization",
        "street": [
          "Example Lane 128",
          "Example Additional Address Line 1",
          "Example Additional Address Line 2"
        ],
        "city": "Example City",
        "sp": "Example State or Province",
        "pc": "1234PC",
        "cc": "NL",
        "voice": "+31.123456789",
        "fax": "+31.123456789",
        "gender": "male",
        "lang": "EN"
      }
    }
  },
  "svTRID": "158893-4-yoursrs",
  "response": [
  ],
}
```

```

"children":[
  {
    "command":"contact:check",
    "handle":"new_example_handle",
    "code":1000,
    "msg":"Command completed successfully",
    "error":[
    ],
    "params":[
    ],
    "svTRID":"158893-2-yoursrs",
    "response":{
      "new_example_handle":"1",
      "avail":"1"
    }
  },
  {
    "command":"contact:create",
    "handle":"new_example_handle",
    "code":1000,
    "msg":"Command completed successfully",
    "error":[
    ],
    "params":{
      "handle":"new_example_handle",
      "email":"info@example.com",
      "name":"John Doe",
      "org":"Example Organization",
      "street":[
        "Example Lane 128",
        "Example Additional Address Line 1",
        "Example Additional Address Line 2"
      ],
      "city":"Example City",
      "sp":"Example State or Province",
      "pc":"1234PC",
      "cc":"NL",
      "voice":"+31.123456789",
      "fax":"+31.123456789",
      "gender":"male",
      "lang":"EN"
    },
    "svTRID":"158893-3-yoursrs",
    "response":{
      "id":"new_example_handle",
      "crDate":1257767891
    }
  }
]
}

```

Delete domain

Used to delete a domain registration.

Parameters

No additional parameters available for domain delete.

Request

```
Method: POST
URL:    https://httpapi.yoursrs.com/v1/domains/example.com/delete
BODY:
{
  "login_handle": "example_handle",
  "login_pass": "*****"
}
```

Response

```
{
  "command": "domain:delete",
  "handle": "example.com",
  "code": 1001,
  "msg": "Command completed successfully; action pending",
  "error": [
  ],
  "params": {
    "handle": "example.com"
  },
  "svTRID": "159697-2-yoursrs",
  "response": [
  ]
}
```

4. Contact Commands

Contains functionality to manage contacts.

Calls made to the resource type contacts require a contact_handle and one of the resource actions listed below.

- Check contact
- Info contact
- Create contact
- Update contact
- Delete contact

Check contact

Check the availability of a contact handle.

Parameters

No additional parameters available for contact check.

Request

```
Method: POST
URL: https://httpapi.yoursrs.com/v1/contacts/example_contact/check
BODY:
{
  "login_handle": "example_handle",
  "login_pass": "*****"
}
```

Response

```
{
  "command": "contact:check",
  "handle": "example_contact",
  "code": 1000,
  "msg": "Command completed successfully",
  "error": [
  ],
  "params": [
  ],
  "svTRID": "158893-2-yoursrs",
  "response": {
    "example_contact": "1",
    "avail": "1"
  }
}
```

Info contact

Retrieve information about a contact.

Parameters

No additional parameters available for contact info.

Request

```
Method: POST
URL:    https://httpapi.yoursrs.com/v1/contacts/example_contact/info
BODY:
{
  "login_handle": "example_handle",
  "login_pass": "*****"
}
```






Response

```
{
  "command": "contact:info",
  "handle": "example_contact",
  "code": 1000,
  "msg": "Command completed successfully",
  "error": [
  ],
  "params": {
    "handle": "example_contact"
  },
  "svTRID": "159673-2-yoursrs",
  "response": {
    "id": "example_contact",
    "roid": "7374-CONTACT",
    "status": "ok",
    "name": "John Doe",
    "org": "Example Organization",
    "street": [
      "Example address line 1",
      "Example address line 2",
      "Example address line 3"
    ],
    "city": "Example city",
    "sp": "Example state or province",
    "pc": "1234PC",
    "cc": "NL",
    "voice": "+31.123456789",
    "fax": "+31.123456789",
    "email": "info@example.com",
    "gender": "male",
    "clID": "example_handle",
    "crDate": 1257335759
  }
}
```

Create contact

Create a new contact.

Parameters

Key	Required	Description
email		Email address of the contact
name		Name of the contact
org		Organization of the contact
street		Array of address rows (max 3)
city		City of the contact
sp		State or province of the contact
pc		Zipcode of the contact
cc		Country code of the contact (ISO3166 format)
voice		Phone number of the contact (E164a format)
fax		Fax number of the contact
gender		Gender of the contact (male or female)
lang		Perefered language, ignored if null

Request

```

Method: POST
URL: https://httpapi.yoursrs.com/v1/contacts/example_contact/create
BODY:
{
  "email":"info@example.com",
  "name":"John Doe",
  "org":"Example Organization",
  "street":[
    "Example address line 1",
    "Example address line 2",
    "Example address line 3"
  ],
  "city":"Example City",
  "sp":"Example state or province",
  "pc":"1234PC",
  "cc":"NL",
  "voice":"+31.123456789",
  "fax":"+31.123456789",
  "gender":"male",
  "language":"EN",
  "login_handle":"example_handle",
  "login_pass":"*****"
}

```













Response

```
{
  "command": "contact:create",
  "handle": "example_contact",
  "code": 1000,
  "msg": "Command completed successfully",
  "error": [
  ],
  "params": {
    "email": "info@example.com",
    "name": "John Doe",
    "org": "Example Organization",
    "street": [
      "Example address line 1",
      "Example address line 2",
      "Example address line 3"
    ],
    "city": "Example city",
    "sp": "Example state or province",
    "pc": "1234PC",
    "cc": "NL",
    "voice": "+31.123456789",
    "fax": "+31.123456789",
    "gender": "male",
    "language": "EN"
  },
  "svTRID": "158875-2-yoursrs",
  "response": {
    "id": "example_contact",
    "crDate": 1257759786
  }
}
```

Update contact

Update a contact handle. Used to alter the contact data on a handle.

Parameters

Key	Required	Description
email		Email address of the contact
name		Name of the contact
org		Organization of the contact
street		Array of address rows (max 3)
city		City of the contact
sp		State or province of the contact
pc		Zipcode of the contact
cc		Country code of the contact (ISO3166 format)
voice		Phone number of the contact (E164a format)
fax		Fax number of the contact
gender		Gender of the contact (male or female)
lang		Perefered language, ignored if null



Parameters requirement

For a contact update non of the parameters is required and only the parameters that are delivered will be used to update the contact.



Remove data

In order to remove data you will have to send the parameter with an empty string. The only exception to this is the street parameter, which will require an empty array.

Request

```
Method: POST
URL: https://httpapi.yoursrs.com/v1/contacts/example_contact/update
BODY:
{
  "email":"info@example.com",
  "name":"John Doe",
  "org":"Example organization",
  "street":[
    "Example address line 1",
    "Example address line 2",
    "Example address line 3"
  ]
  "city":"Example city",
  "sp":"Example state or province",
  "pc":"1234PC",
  "cc":"NL",
  "voice":"+31.123456789",
  "fax":"+31.123456789",
  "gender":"male",
  "login_handle":"example_handle",
  "login_pass":"*****"
}
```

Response

```
{
  "command":"contact:update",
  "handle":"example_contact"
  "code":1000,
  "msg":"Command completed successfully",
  "error":[]

},
"params":{
  "email":"info@example.com",
  "name":"John Doe",
  "org":"Example organization",
  "street":[
    "Example address line 1",
    "Example address line 2",
    "Example address line 3"
  ]
  "city":"Example city",
  "sp":"Example state or province",
  "pc":"1234PC",
  "cc":"NL",
  "voice":"+31.123456789",
  "fax":"+31.123456789",
  "gender":"male",
  "handle":"example_contact"
},
"svTRID":"158875-2-yoursrs",
"response":[]

}
```

Delete contact

Delete a contact.

Parameters

No additional parameters available for contact delete.

Request

```
Method: POST
URL:    https://httpapi.yoursrs.com/v1/contacts/example_contact/delete
BODY:
{
  "login_handle": "example_handle",
  "login_pass": "*****"
}
```

Response

```
{
  "command": "contact:delete",
  "handle": "example_contact",
  "code": 1000,
  "msg": "Command completed succesfully",
  "error": [
  ],
  "params": {
    "handle": "example_contact"
  }
  "svTRID": "159504-2-yoursrs",
  "response": [
  ]
}
```

5. API Response Codes

All <response> messages have a result code, a message with the description of the result code and optionally additional error messages.

As the HTTP API is closely related to our EPP API the response codes are identical. Those of you who are already familiar with our EPP API will probably recognise the response codes.

Although the response codes listed below are all valid for the EPP API implementation, several of those will not likely be returned by the HTTP API as a pre-filter on commands available has already been done by the HTTP API.

The response codes that can be returned are:

1000 - Command completed successfully

This is the usual response code for a successfully completed command that is not addressed by any other 1xxx-series response code.

1001 - Command completed successfully; action pending

This response code will be returned when responding to a command that requires offline activity before the requested action can be completed.

2000 - Unknown command

This response code will be returned when the server receives a command element that is not defined by EPP.

2001 - Command syntax error

This response code will be returned when the server receives an improperly formed command element.

2002 - Command use error

This response code will be returned when the server receives a properly formed command element but the command cannot be executed due to a sequencing or context error.

For example, a <logout> command cannot be executed without having first completed a <login> command.

2003 - Required parameter missing

This response code will be returned when the server receives a command for which a required parameter value has not been provided.

2004 - Parameter value range error

This response code will be returned when the server receives a command parameter whose value is outside the range of values specified by the protocol.

2005 - Parameter value syntax error

This response code will be returned when the server receives a command containing a parameter whose value is improperly formed.

2100 - Unimplemented protocol version

This response code will be returned when the server receives a command element specifying a protocol version that is not implemented by the server.

2101 - Unimplemented command

This response code will be returned when the server receives a valid command that is not implemented by the server. For example, a <transfer> command can be unimplemented for certain object types.

2102 - Unimplemented option

This response code will be returned when the server receives a valid command that contains a protocol option that is not implemented by the server.

2103 - Unimplemented extension

This response code will be returned when the server receives a valid command that contains a protocol command extension that is not implemented by the server.

2104 - Billing failure

This response code will be returned when the server attempts to execute a billable operation and the command cannot be completed due to a client-billing failure.

2105 - Object is not eligible for renewal

This response code will be returned when the client attempts to <renew> an object that is not eligible for renewal in accordance with server policy.

2106 - Object is not eligible for transfer

This response code will be returned when a client attempts to <transfer> an object that is not eligible for transfer in accordance with server policy.

2200 - Authentication error

This response code will be returned when the server notes an error when validating client credentials.

2201 - Authorization error

This response code will be returned when the server notes a client-authorization error when executing a command. This error is used to note that a client lacks privileges to execute the requested command.

2202 - Invalid authorization information

This response code will be returned when the server receives invalid command authorization information when attempting to confirm authorization to execute a command. This error is used to note that a client has the privileges required to execute the requested command, but the authorization information provided by the client does not match the authorization information archived by the server.

2300 - Object pending transfer

This response code will be returned when the server receives a command to transfer of an object that is pending transfer due to an earlier transfer request.

2301 - Object not pending transfer

This response code will be returned when the server receives a command to confirm, reject, or cancel the transfer of an object when no command has been made to transfer the object.

2302 - Object exists

This response code will be returned when the server receives a command to create an object that already exists in the repository.

2303 - Object does not exist

This response code will be returned when the server receives a command to query or transform an object that does not exist in the repository.

2304 - Object status prohibits operation

This response code will be returned when the server receives a command to transform an object that cannot be completed due to server policy or business practices.

For example the server might have received a <delete> command for an object whose status prohibits deletion.

2305 - Object association prohibits operation

This response code will be returned when the server receives a command to transform an object that cannot be completed due to dependencies on other objects that are associated with the target object.

For example, the server can disallow <delete> commands while an object has active associations with other objects.

2306 - Parameter value policy error

This response code will be returned when the server receives a command containing a parameter value that is syntactically valid but semantically invalid due to local policy.

For example, the server can support a subset of a range of valid protocol parameter values.

2307 - Unimplemented object service

This response code will be returned when the server receives a command to operate on an object service that is not supported by the server.

2308 - Data management policy violation

This response code will be returned when the server receives a command whose execution results in a violation of server data management policies.

For example, removing all attribute values or object associations from an object might be a violation of the server's data management policies.

2400 - Command failed

This response code will be returned when the server is unable to execute a command due to an internal server error that is not related to the protocol. The failure can be transient.

2500 - Command failed; server closing connection

This response code will be returned when the server receives a command that cannot be completed due to an internal server error that is not related to the protocol. The failure is not transient and will cause other commands to fail as well.

2501 - Authentication error; server closing connection

This response code will be returned when the server notes an error when validating client credentials and a server-defined limit on the number of allowable failures has been exceeded.

2502 - Session limit exceeded; server closing connection

This response code will be returned when the server receives a <login> command and the command cannot be completed because the client has exceeded a system-defined limit on the number of sessions that the client can establish. It might be possible to establish a session by ending existing unused sessions and closing inactive connections.

6. Appendix - PHP Examples

Below are some snippets in PHP giving examples of some basic calls to the HTTP API.

Generic sendRequest

For each request made to the HTTP API a common snippet of code can be used.

The purpose of this snippet is to receive the request parameters. After that format the request and send it to either the production or test environment of the HTTP API.

The result of that request is then captured in a variable and returned as the result of this function.

```
function sendRequest($url, $params, $handle, $password) {
    $params['login_handle'] = $handle;
    $params['login_pass'] = $password;

    $curl = curl_init();

    // Set the URL for the request in the cURL options.
    curl_setopt($curl, CURLOPT_URL, $url);

    curl_setopt($curl, CURLOPT_FAILONERROR, TRUE);

    // This setting will return the response of the HTTP API rather than print it on screen.
    curl_setopt($curl, CURLOPT_RETURNTRANSFER, TRUE);

    // Make sure you enable the POST setting in cURL as the API only accepts POST commands.
    curl_setopt($curl, CURLOPT_POST, TRUE);

    // Set the parameters to submit with the POST request and make sure they're JSON Encoded.
    curl_setopt($curl, CURLOPT_POSTFIELDS, json_encode($params));

    // The production environment uses a SSL connection, for the test environment however you'd be
    looking to disable
    // the settings.
    curl_setopt($curl, CURLOPT_SSL_VERIFYPEER, true);
    curl_setopt($curl, CURLOPT_SSL_VERIFYHOST, 2);

    // Disable the SSL Verification for the test environment.
    if ($this->settings->get("plugin_realtimeregister_TestMode") == 1) {
        curl_setopt($curl, CURLOPT_SSL_VERIFYPEER, false);
        curl_setopt($curl, CURLOPT_SSL_VERIFYHOST, 1);
    }

    /* Capture the response of the request. */
    $result = curl_exec($curl);

    /* Could not connect to API, curl returned false */
    if ($result === false) {
        $msg = "Curl errno " . curl_errno($curl) . " : " . curl_error($curl);
        curl_close($curl);
        return $msg;
    }

    /* Try to decode the response */
    $response = json_decode($result);

    /* Close cURL */
    curl_close($curl);

    /* Response could not be decoded */
    if (!$response) {
        return "Received invalid response. Please try again.";
    }

    /* An error occurred */
    if ($response->code >= 2000) {
        $error = $response->error;
        array_unshift($error, $response->msg);
        return $response->msg;
    }

    return $response;
}
```

Check Domain

```
$domain = "test.nl";
$params = array();
$request_url = "https://httpapi.yoursrs.com/v1/domains/" . urlencode($domain) . "/check";

$response = sendRequest($request_url, $params, "tester", "pass1234");

print_r($response);
```

Create Domain

```
$domain = "test.nl";
$params = array();

$request_url = "https://httpapi.yoursrs.com/v1/domains/" . urlencode($domain) . "/create";

$params['autorenew'] = true;
$params['template'] = null;
$params['template_link'] = null;
$params['ns'][] = array("host" => $hostname1, "addr" => $address1);
$params['ns'][] = array("host" => $hostname2);

$registrant = array(
    'name' => "demo registrant",
    'email' => "demo@realtimeregister.com",
    'org' => "Demo Organisation",
    'street' => array( 'street1', 'street2', 'street3' ),
    'city' => "Demo City",
    'sp' => "Demo State or Province",
    'pc' => "1111AB",
    'cc' => "NL",
    'voice' => "+31.123456789"
);

$params['contact_data']['registrant'] = $registrant;
$params['admin'] = "admin handle";
$params['billing'] = "billing handle";
$params['tech'] = "tech handle";

$response = sendRequest($request_url, $params, "tester", "pass1234");

print_r($response);
```

Update Domain

Below is a snippet that shows a domain update call in PHP. Here we change the autorenew setting of the domain.

```
$domain = "test.nl";

$params = array();
$params['autorenew'] = true;

$request_url = "https://httpapi.yoursrs.com/v1/domains/" . urlencode($domain) . "/update";

$response = sendRequest($request_url, $params, "tester", "pass1234");

print_r($response);
```

Delete Domain

One of the most simple requests is the delete domain. All it requires is a domain name which is part of the request URL.

```

$domain = "test.nl";
$params = array();

$request_url = "https://httpapi.yoursrs.com/v1/domains/" . urlencode($domain) . "/delete";

$response = sendRequest($request_url, $params, "tester", "pass1234");

print_r($response);

```

Check Contact

```

$contact = "tester";
$request_url = "https://httpapi.yoursrs.com/v1/contacts/" . urlencode($contact) . "/check";
$params = array();

$response = sendRequest($request_url, $params, "tester", "pass1234");

print_r($response);

```

Create Contact

```

$contact = "tester";
$request_url = "https://httpapi.yoursrs.com/v1/domains/" . urlencode($domain) . "/create";

$params = array(
  'name' => "tester", // string Name
  'email' => "tester@test.nl", // string Email
  'org' => "", // string Organisation
  'street' => array("Schrevenweg 5"), // array Address rows (max 3)
  'city' => 'Zwolle', // string City
  'sp' => 'Overijssel', // string State / Province
  'pc' => '8024HB', // string Zipcode / Postal code
  'cc' => 'NL', // string Country Code (ISO3166 format)
  'phone' => '+31.384530752', // string Phone number (E164a format)
  'fax' => '+31.384524734', // string Fax number (E164a format)
  'gender' => 'male', // string Gender (male or female)
  'language' => 'NL' // string Language (NL or EN)
);

$response = sendRequest($request_url, $params, "tester", "pass1234");

print_r($response);

```

Update Contact

```

$contact = "tester";
$request_url = "https://httpapi.yoursrs.com/v1/domains/" . urlencode($domain) . "/update";

$params = array();
$params['email'] = "t.ester@test.nl";

$response = sendRequest($request_url, $params, "tester", "pass1234");

print_r($response);

```

Delete Contact

```

$contact = "tester";
$request_url = "https://httpapi.yoursrs.com/v1/domains/" . urlencode($domain) . "/delete";

$params = array();

$response = sendRequest($request_url, $params, "tester", "pass1234");

print_r($response);

```