



**emerion Reseller XML-RPC  
Application Programming Interface (API)  
Specification**

## Table of Contents:

<b>A. State of this document .....</b>	<b>4</b>
<b>B. Syntax used in this document .....</b>	<b>4</b>
<b>C. General .....</b>	<b>4</b>
<b>1. Object: Transaction .....</b>	<b>5</b>
Transaction.begin() .....	5
Transaction.authenticate( %trxParams, \$user <string>, \$md5_digest <string>) .....	6
Transaction.close( %trxParams ) .....	6
<b>2. Object: Domain.....</b>	<b>7</b>
Domain.get( %trxParams, \$domainname <string>) .....	7
Table.Object.Domain.1 : .....	9
Table.Object.Domain.2: .....	9
Table.Object.Domain.3 : .....	9
Domain.lookup( %trxParams, \$domainname <string>, \$batch_lookup <boolean>) .....	10
Domain.placeOrder( %trxParams, %domain) .....	11
<b>3. Object: EmailPOP3.....</b>	<b>12</b>
EmailPOP3.get( %trxParams, \$domainname <string>, \$email_address <string>) .....	12
Table.Object.EmailPOP3.1: .....	13
Table.Object.EmailPOP3.2: .....	13
EmailPOP3.create( %trxParams, %email_address) .....	14
Table.Object.EmailPOP3.3: .....	15
Table.Object.EmailPOP3.4: .....	15
Table.Object.EmailPOP3.5: .....	15
EmailPOP3.modify( %trxParams, %email_address) .....	16
EmailPOP3.delete( %trxParams, \$email_address <string>) .....	16
<b>4. Object: Forward.....</b>	<b>17</b>
Forward.get( %trxParams, \$email_address <string>) .....	17
Table.Object.Forward.1: .....	17
Forward.create( %trxParams, \$email_address <string>, \$forward_address <string>) .....	18
Table.Object.Forward.2: .....	18
Forward.modify( %trxParams, %new_forwarder) .....	19
Table.Object.Forward.3: .....	19
Table.Object.Forward.4: .....	19
Forward.setKeepCopy( %trxParams, \$email_address <string>, \$keep_copy <boolean>) .....	20
Table.Object.Forward.5: .....	20
Forward.delete( %trxParams, %forwarder) .....	21
Table.Object.Forward.6: .....	21
Table.Object.Forward.7: .....	21

<b>5. Object: Autoresponder .....</b>	<b>22</b>
Autoresponder.get( %trxParams, \$email_address <string>).....	22
Table.Object.Autoresponder.1:.....	22
Autoresponder.create( %trxParams, %autoresponder ).....	23
Autoresponder.modify( %trxParams, %autoresponder).....	24
Autoresponder.delete( %trxParams, \$email_address <string>).....	24
<b>6. Object: Alias .....</b>	<b>25</b>
Alias.get( %trxParams, \$domainname <string>, \$email_address <string>).....	25
Table.Object.Alias.1: .....	26
Table.Object.Alias.2: .....	26
Alias.create( %trxParams, \$email_address <string>, \$alias_address <string>, \$blacklist <boolean>).27	
Alias.modify( %trxParams, \$alias_address <string>, \$blacklist <boolean>).....	27
Alias.delete( %trxParams, \$alias_address <string>) .....	28
<b>7. Object: Zone .....</b>	<b>29</b>
Zone.get( %trxParams, \$domainname <string>).....	29
Table.Object.Zone.1: .....	30
Zone.modify(%trxParams %zone) .....	31
Table.Object.Zone.2: .....	31
<b>8. Object: Activate.....</b>	<b>32</b>
Activate for all method calls .....	32
<b>9. Object: Domainpointer .....</b>	<b>33</b>
Domainpointer.get( %trxParams, \$domainname <string>, \$fqdn <string>).....	33
Table.Object.Domainpointer.1: .....	33
Table.Object.Domainpointer.2: .....	34
Domainpointer.modify( %trxParams, %pointer <struct>).....	35
<b>Appendix A .....</b>	<b>36</b>
1. Input Syntax Correction of Contact Set fields.....	36
2. Check for Titles in the fields firstname, lastname .....	36
3. Check for Common Names in the company field .....	37
4. Autoresponder Text Check.....	38

# emerion Reseller XML-RPC Application Programming Interface (API) Specification

Version: 1.0.0, 2005-01-23

Last-changed-by: Bernd Hilmar

Authors: Bernd Hilmar, Michael Paesold

References: <http://www.xmlrpc.com/spec>

## A. State of this document

This document describes the functions of the Reseller API. There also exists a reference implementation of the client part for Perl. It demonstrates the plausibility and usefulness of the specification.

## B. Syntax used in this document

Prefix:

%	Describes a hash (<struct> in XML-RPC)
@	Describes an array
\$	Describes a variable

## C. General

Each transaction starts with the object "Transaction". Once the transaction is started, methods of any objects can be called. After all work is done, the transaction should closed.

Available objects are described below.

The description of each object contains the following:

- Available methods
- Table of method calls  
The first column describes the method call and the second column describes the result (return)
- Notes, detailed information regarding the method calls
- The description of the return hashes

There is a generic return hash, named %result in each object.

### Generic Return Hash (<struct>):

```
%result (
    status_code <int>           // numeric result code
    status_string <string>     // result/error description
    status_detail <string>     // only set if information is available
    session_id <string>       // the session id changes with each call(!)
)
```

The parameters of the Generic Return Hash are called {statusParams}.

## 1. Object: Transaction

Methods: begin, authenticate, close

### Method calls:

Call	Return
Transaction.begin()	%begin_result
Transaction.authenticate( %trxParams, \$user <string>, \$scr_md5 <string>)	%result
Transaction.close( %trxParams )	%result

### Notes:

```
%trxParams = (
    transaction_id <int>
    session_id <string>
)
```

On account activation for the Reseller API you will get the following information:

```
$user          = $ControlPanelLogin (already known)
$password      = [will be submitted at API account activation]
$secret_key    = [will be submitted at API account activation]
```

### Description :

Each time your application wants to communicate with the emerion ControlPanel via the Reseller API, a transaction must be started. The transaction must then be authenticated. Once the authentication was successful, methods of any objects can be sent. At the end the transaction, it should be closed.

### Call:

#### Transaction.begin()

To start a transaction "Transaction.begin" is called without any parameter.

### Result:

#### %begin\_result( {statusParams}, \$challenge <string>)

The hash (<struct>) %begin\_result contains the status parameters and the variable \$challenge. \$challenge is a unique ID for each transaction.

```
%begin_result (
    status_code <int>
    status_string <string>
    session_id <string>
    transaction_id <int>
    challenge <string>
)
```

To proceed with the authentication you must call the next method.

**Call:****Transaction.authenticate( %trxParams <struct>, \$user <string>, \$md5\_digest <string>)**

You must now create an MD5 Digest.

\$md5\_digest is an MD5 Digest of your secret key, the challenge you got from the server with the method "Transaction.begin", and your password.

To authenticate your transaction \$user (your ControlPanel Login = your account id), and the generated \$md5\_digest must be sent as parameter.

**Result:**

As result the generic hash (<struct>) %result is returned:

```
%result (
  status_code <int>
  status_string <string>
  status_detail <string> // only set if information is available
  session_id <string>
)
```

If the transaction is successfully authenticated, you can call methods of any other object.

**Note: In each %result of a method call you get a new session\_id.**

You must use this retrieved \$session\_id in the next method call you send.

**Call:****Transaction.close( %trxParams <struct>)**

After all method calls are sent, you must close the transaction with "Transaction.close".

If the transaction will not be closed, a timeout will occur and the transaction will be closed automatically.

**Result:**

As result the generic hash (<struct>) %result is returned:

```
%result (
  status_code <int>
  status_string <string>
  status_detail <string> // only set if information is available
  session_id <string>
)
```

## 2. Object: Domain

Methods: get, lookup, placeOrder

### Method calls:

Call	Return
Domain.get( %trxParams, \$domainname*)	%result(\$count, @domains)
Domain.lookup( %trxParams, \$domainname, \$batch_lookup)	%result(@status)
Domain.placeOrder( %trxParams, %domain)	%result

### Notes:

- \* = parameter optional
- All other parameters are mandatory
- IDN domain names: \$domainname must be the ACE String of the domain name.

### Description :

#### Call:

**Domain.get( %trxParams <struct>, \$domainname <string>)**

Using this call you can retrieve a particular domain or all domains in your reseller account. If \$domainname is given, only this domain is retrieved, if \$domainname is not set, all domains will be retrieved.

#### Result:

**%result  
( {statusParams}, \$count <int>, @domains[%domain(%contact\_set,%services)])**

The result contains the status parameters, the variable \$count with the number of found domains and an array containing a hash (<struct>) for each found domain. If the parameter \$domainname is not given, all domains of the account are retrieved.



**Field descriptions:****Table.Object.Domain.1 :**

%domain: O=Optional, M=Mandatory (required) = always set, R=Read only (cannot be set)

domainname	M	full qualified domainname
status	R	possible values: active/expired/canceled/gone(=transfer away)
create_date	R	creation date at emerion
expire_date	R	expire date in the registry
cancel_date	R/O	emerion cancel date (=00000000T00:00:00 if domain is not canceled)
order_type	M	possible values: new/transfer
order_date	M	date/time when order has been placed
order_closed	R	date/time when order has been finished = date/time of service activation
order_ip	R	ip number of the client who placed the order

**Table.Object.Domain.2:**

%contact\_set: O=Optional, M=Mandatory (required) = always set

company	O	company name if domain owner is a company
firstname	M	firstname of the domain owner
lastname	M	lastname of the domain owner
address	M	address of the domain owner
zip	M	zip code of the domain owner
city	M	city of the domain owner
country	M	two letter ISO country code
phone	M	phone number of the domain owner
fax	O	fax number of the domain owner
email	M	valid email address of the domain owner (very important)
nichandle	R/O	nic handle if required by the registry
category	O	possible values: private/org if supported by the registry
whois_show_phone	M	0 = if not supported by the registry or if set to No
whois_show_fax	M	0 = if not supported by the registry or if set to No
whois_show_email	M	0 = if not supported by the registry or if set to No

**Table.Object.Domain.3 :**

%services: O=Optional, M=Mandatory (required) = always set, R=Read only

webpace	M/R	amount of assigned webpace in MB
email	M/R	number of assigned email addresses
email_aliases	M/R	number of external aliases in use
subhost	M/R	number of assigned subhosts
ftp	M/R	number of assigned ftp accounts
realm	M/R	realm allowed or not
logging	M/R	logging enabled or not
mysql	M/R	amount of assigned databases
htdig	M/R	htdig enabled or not
mailinglist	M/R	number of assigned mailinglists
cronjobs	M/R	number of assigned cron jobs
sslhost	M/R	shared ssl enabled or not
traffic	M/R	amount of traffic in GB included

**Call:**

**Domain.lookup( %trxParams <struct>, \$domainname <string>, \$batch\_lookup <boolean>)**

This call returns a hash with the status parameters and an array, where each element is a hash (<struct>) with the domainname and the availability of the requested domain name.

If \$batch\_lookup is set to true (1), the availability for all possible TLDs will be looked up.

**Warning:** Please use the feature batch\_lookup with care.

Availability of domain names is not cached, it will always be retrieved from the appropriate registry. The process can take a while, but it is ensured that the results are correct at the time of the request.

**Result:**

Parameter: \$batch\_lookup = 0

```
%result (
  status_code <int>
  status_string <string>
  status_detail <string> // only set if information is available
  session_id <string>
  @status (
    %domain (
      domainname <string>
      available <Boolean> // available=1, taken=0
    )
  )
)
```

Parameter: \$batch\_lookup = 1

```
@status (
  %domain (
    domainname // domainname tld-1
    available <Boolean> // available=1, taken=0
  )
  %domain (
    domainname // domainname tld-2
    available <Boolean> // available=1, taken=0
  )
  %domain (
    domainname // domainname tld-3
    available <Boolean> // available=1, taken=0
  )
  ...
)
```

**Call:****Domain.placeOrder( %trxParams <struct>, %domain <struct>)**

This call places an order directly in the emerion domain name registration system. For this call the hash (<struct>) %domain must be sent. The result is the generic result hash. If the returned status code is successful, the order has been placed, otherwise the status code tells, why the order could not be placed.

**Important notice on input data checks:** Titles are checked and removed from the fields "firstname" and "lastname". If only a title is entered in this fields, the order cannot be placed because a mandatory field is empty. The field "company" is checked against common names. If a common name stands alone in the company field it is removed automatically. No error occurs in this case, because the field "company" is not mandatory.

For title checks see Appendix A.

```
%domain (
  domainname <string>
  order_status <string> // test, live
  www_ip <string> // optional
  article_id <int> // optional, not set if account is reseller
  order_service <string> // optional

  %contact_set = ( // hash reference (<struct>)
    company <string> // optional
    firstname <string> // mandatory
    lastname <string> // mandatory
    address <string> // mandatory
    zip <string> // mandatory
    city <string> // mandatory
    country <string> // two letter ISO country code, mandatory
    phone <string> // mandatory
    fax <string> // optional
    email <string> // mandatory
  )
)
```

**Result:**

The result returns all parameters of the generic hash (<struct>) %result and the additional parameter order\_status. order\_status defines the status of the order, if the domain name is a new registration or a transfer of an existing domain name.

```
%result (
  status_code <int>
  status_string <string>
  status_detail <string> // only set if information is available
  session_id <string>
  order_status <string> // new or transfer
)
```

### 3. Object: EmailPOP3

Methods: get, create, modify, delete

#### Method calls:

Call	Return
EmailPOP3.get( %trxParams, \$domainname*, \$email_address*)	%result(\$count, @emails)
EmailPOP3.create( %trxParams, %email_address)	%result
EmailPOP3.modify( %trxParams, %email_address)	%result
EmailPOP3.delete( %trxParams, \$email_address)	%result

#### Notes:

- \* = parameter optional
- All other parameters are mandatory

#### Description:

##### Call:

**EmailPOP3.get( %trxParams <struct>, \$domainname <string>, \$email\_address <string>)**

This method retrieves one or more email addresses. The parameters \$domainname and \$email\_address are optional. If both parameters are empty all email addresses of the whole account are retrieved. If \$domainname is set and \$email\_address is empty, all email addresses of the specified domain name are retrieved. If \$email\_address is set, only the requested email address is retrieved. To get a single email address, \$domainname must be set and the domain name must match the domain part of the email address:

*Example:*

```
$domainname      = "myname.com" ;
$email_address   = "name@myname.com" ;
```

##### Result:

**%result({statusParams}, \$count <int>, @emails[%email(%spam\_virus\_protection)])**

The result contains the statusParams, the variable \$count with the number of retrieved email addresses and an array containing the hash (<struct>) %email for each found email address.

```

%result (
  status_code <int>
  status_string <string>
  status_detail <string> // only set if information is available
  session_id <string>
  count <int> // number of email addresses in the array
  @emails (
    %email ( // field description see Table.Object.EmailPOP3.1
      email_address <string>
      username <string>
      password <string>
      pop_server <string>
      smtp_server <string>
      blacklist <boolean>
      password_modified_by_owner <boolean>
      alias_assigned <int>
      %spam_virus_protection ( // Table.Object.EmailPOP3.2
        status <boolean>
        quarantine_url <string>
        quarantine_username <string>
        quarantine_password <string>
        accessable_by_webmail <boolean>
      )
    )
  )
)

```

### Field descriptions:

#### Table.Object.EmailPOP3.1:

%email: O=Optional, M=Mandatory (required) = always set, R=Read only (cannot be set)

email_address	M	full email address including the domain part
username	M/R	username is set by the system
password	M	email password
pop_server	M/R	set by the system
smtp_server	M/R	set by the system
blacklist	M	possible values: 0=blacklist off/1=blacklist on
password_modified_by_owner	O	possible values: 0=no/1=yes
alias_assigned	R	number of aliases assigned to the pop3 account

#### Table.Object.EmailPOP3.2:

%spam\_virus\_protection: M=Mandatory (required) = always set, R=Read only (cannot be set)

status	M	possible values: 0=off/1=on
quarantine_url	R	web-URL of the quarantine folder
quarantine_username	R	username for the login of the quarantine folder
quarantine_password	R	password for the login of the quarantine folder
accessable_by_webmail	R	is the quarantine folder access able by web mail? 0=no/1=yes

**Call:****EmailPOP3.create( %trxParams <struct>, %email\_address <struct>)**

This method creates an email address by sending the hash %email\_address as parameter. Please note, the activation of the spam- and virus protection must be done separately with the method "modify".

**Important note:** When you call the "create" method, the email address is created in our database, but not activated on the mail server. A "waiting request" for service activation is initiated. You can now create more email addresses (within a loop). At the end you must activate the created addresses. The activation of a service on the mail server is done with the object "Activate".

```
%email_address ( // Description Table.Object.EmailPOP3.3
    email_address <string>
    auto_password <boolean>
    password <string>
    blacklist <boolean>
)
```

**Result:**

The result includes the details for the newly created email addresses:

```
%result (
    status_code <int>
    status_string <string>
    status_detail <string> // only set if information is available
    session_id <string>
    waiting_request <boolean> // Description Table.Object.EmailPOP3.4
    %email_address = ( // Description Table.Object.EmailPOP3.5
        email_address <string>
        username <string>
        password <string>
        pop_server <string>
        smtp_server <string>
        blacklist <boolean>
    )
)
```

**Field descriptions:****Table.Object.EmailPOP3.3:**

%email\_address: O=Optional, M=Mandatory (required)

email_address	M	full email address including the domain part
auto_password	M	1=creates password automatically, 0=no auto password
password	M	field is empty if auto_password=1
blacklist	M	0=no blacklist, 1=enable blacklist (=blacklist on)

**Table.Object.EmailPOP3.4:**

%result: R=Read only

waiting_request	R	1=request to activate the service has been set, 0=no request
-----------------	---	--

**Table.Object.EmailPOP3.5:**

%email: R=Read only

username	R	Username, created by the system
password	R	Plain text password
pop_server	R	pop3 server set by the system
smtp_server	R	smtp server set by the system
blacklist	R	0=blacklist disable, 1= blacklist enabled (=blacklist on)

**Call:****EmailPOP3.modify( %trxParams <struct>, %email\_address <struct>)**

To modify any parameter of an existing email address, the same hash like at EmailPOP3.create is sent. It is not possible to modify the email address itself. To change an email address the current one must be deleted and a new one created.

The element email\_address must match an existing address. Two values can be modified with one request. If the password is blank and auto\_password is set to false (0), the password is left untouched. The boolean value for the blacklist must be the current value, if the current value is different to the set value, the blacklist flag will be changed.

```
%email_address (
    email_address <string>
    auto_password <boolean>
    password <string>
    blacklist <boolean>
)
```

**Please note:** Modification of the Spam and Virus Protection is not supported. Spam and Virus Protection can only be enabled or disabled in the ControlPanel.

**Result:**

The result is the generic hash (<struct>) %result hash

```
%result (
    status_code <int>
    status_string <string>
    status_detail <string> // only set if information is available
    session_id <string>
)
```

**Call:****EmailPOP3.delete( %trxParams <struct>, \$email\_address <string>)**

To delete an existing address, only the email address is sent as parameter. An email address can only be deleted if the spam and virus protection is first disabled in the ControlPanel.

**Please note:** The email address is deleted, but the action is not activated on the mail server. A waiting request for service activation is initiated. You can delete more email addresses (within a loop) and at the end apply the changes to the mail server with the object "Activate".

**Result:**

The result is the generic hash (<struct>) %result hash

```
%result (
    status_code <int>
    status_string <string>
    status_detail <string> // only set if information is available
    session_id <string>
    waiting_request <boolean>
)
```

## 4. Object: Forward

Methods: get, create, modify, setKeepCopy, delete

### Method calls:

Call	Return
Forward.get( %trxParams, \$email_address)	%result(\$count, @forwarders)
Forward.create( %trxParams, \$email_address, \$forward_address)	%result
Forward.modify( %trxParams, %new_forwarder)	%result
Forward.setKeepCopy( %trxParams, \$email_address, \$keep_copy)	%result
Forward.delete( %trxParams, \$email_address, \$forward_address)	%result

### Description:

#### Call:

**Forward.get( %trxParams <struct>, \$email\_address <string>)**

To get forwarders of an email address, the email address is sent as parameter.

#### Result:

**%result({statusParams}, \$count <int>, @forwarders[%forwarder])**

The result contains the status parameters, the variable \$count and the array @forwarders, containing the hash (<struct>) %forwarder for each found forward.

```
%result (
    status_code <int>
    status_string <string>
    status_detail <string> // only set if information is available
    session_id <string>
    count <int>
    keep_copy <boolean>
    email_address <string>
    @forwarders (
        forward_address <string>
    )
)
```

### Field descriptions:

#### Table.Object.Forward.1:

count	R	number of found forwarders in the array
keep_copy	R	1=forwarded mails will be left in the mailbox, 0=forward only

%forwarder: R=Read only

email_address	R	the POP3 email address
forward_address	R	the destination email address where emails should be forwarded to

**Call:**

**Forward.create( %trxParams <struct>, \$email\_address <string>, \$forward\_address <string>)**

To create a new forwarder the POP3 email address \$email\_address is the first parameter, \$forward\_address, the destination email address, where emails should be forwarded to, is the second parameter.

**Result:**

The result hash (<struct>) %result contains the status parameters {statusParams}, the email address, the created forward address and the keep\_copy flag.

```
%result (
  status_code <int>
  status_string <string>
  status_detail <string> // only set if information is available
  session_id <string>
  email_address <string> // see Table.Object.Forward.2
  forward_address <string>
  keep_copy <boolean>
)
```

**Field descriptions:****Table.Object.Forward.2:**

%result: R=read only, additional parameters

email_address	R	the POP3 email address
forward_address	R	the existing destination email address where emails are forwarded
keep_copy	R	The current status of keep_copy

**Call:****Forward.modify( %trxParams <struct>, %new\_forwarder <struct>)**

To modify an existing forwarder, the hash (<struct>) %new\_forwarder is sent as parameter.

```
%new_forwarder ( // field description Table.Object.Forward.3
    email_address <string>
    forward_address <string>
    new_forward_address <string>
)
```

**Field descriptions:****Table.Object.Forward.3:**

%new\_forwarder: M=Mandatory (Field required)

email_address	M	the POP3 email address
forward_address	M	the existing destination email address where emails should be forwarded
new_forward_address	M	the new destination email address where emails should be forwarded

**Result:**

The result hash (<struct>) %result contains the status parameters {statusParams}, the email address, the new forward address and the current keep\_copy flag.

```
%result (
    status_code <int>
    status_string <string>
    status_detail <string> // only set if information is available
    session_id <string>
    email_address <string> // additional params Table.Object.Forward.4
    forward_address <string>
    keep_copy <boolean>
)
```

**Field descriptions:****Table.Object.Forward.4:**

%result: R=Read Only

email_address	R	the POP3 email address
forward_address	R	the existing destination email address where emails are forwarded
keep_copy	R	The current status of keep_copy

**Call:**

**Forward.setKeepCopy( %trxParams <struct>, \$email\_address <string>, \$keep\_copy <boolean>)**

This method sets the option if forwarded emails for an existing POP3 account should be kept in the local mailbox or not. The first parameter in this call is the existing POP3 email address, the second parameter is the boolean value, if a copy of the forwarded mail should be kept or not.

This method is only used to modify the option to keep a copy or not. The current status of this flag is retrieved in the result.

**Result:**

As result the generic hash (<struct>) %result with additional parameters is returned.

```
%result (
  status_code <int>
  status_string <string>
  status_detail <string> // only set if information is available
  session_id <string>
  email_address <string>
  keep_copy <boolean> // the modified (new) status
)
```

**Field descriptions:****Table.Object.Forward.5:**

Forward.setKeepCopy Parameters: M=Mandatory (Field required)

email_address	M	the POP3 email address
keep_copy	M	1=keep a copy of the forwarded email in the local mailbox 0=no copy kept, email only forwarded

**Call:****Forward.delete( %trxParams <struct>, %forwarder <struct>)**

For this calls the hash %forwarder is used. A single forwarder can be deleted, or all forwarders can be deleted.

**Important notice:** If you like to delete all forwarders, the field "forward\_address" must be blank. If the field "forward\_address" contains an address, the flag "delete\_all" is ignored.

```
%forwarder (
    email_address <string>
    forward_address <string>
    delete_all <boolean>
)
```

**Table.Object.Forward.6:**

Forward.delete Parameters: M=Mandatory (Field required)

email_address	M	the POP3 email address
forward_address	M/O	the forward address that should be deleted
delete_all	M	1=delete all forwarders 0=delete only the given forward address

**Result:**

As result the generic hash (<struct>) %result is returned:

```
%result (
    status_code <int>
    status_string <string>
    status_detail <string> // only set if information is available
    session_id <string>
    email_address <string>
    delete_count <int>
)
```

**Field descriptions:****Table.Object.Forward.7:**

Forward.delete Additional parameters: R=Read only

email_address	R	the email address where the forwarder has been deleted
delete_count	R	the number of deleted forwarders

## 5. Object: Autoresponder

Methods: get, create, modify, delete

### Method calls:

Call	Return
Autoresponder.get( %trxParams, \$email_address)	%result(%autoresponder)
Autoresponder.create( %trxParams, %autoresponder)	%result
Autoresponder.modify( %trxParams, %autoresponder)	%result
Autoresponder.delete( %trxParams, \$email_address)	%result

### Description:

#### Call:

**Autoresponder.get( %trxParams <struct>, \$email\_address <string>)**

To get the autoresponder of an email address, the email address is sent as parameter.

#### Result:

**%result({statusParams}, %autoresponder <struct>)**

The result contains the status parameters and the hash (<struct>) %autoresponder with all values of the autoresponder.

\$count is not retrieved, because an email address can contain only one autoresponder.

```
%result (
  status_code <int>
  status_string <string>
  status_detail <string> // only set if information is available
  session_id <string>
  %autoresponder ( // see Table.Object.Autoresponder.1
    email_address <string>
    status <boolean>
    text <string>
  )
)
```

### Field descriptions:

#### Table.Object.Autoresponder.1:

%autoresponder: M=Mandatory (Field required), O=Optional, R=Read Only

email_address	M	the POP3 email address
status	R	0=Autoresponder is off 1=Autoresponder is on
text	M	The customized text of the automatic answer.

**Call:****Autoresponder.create( %trxParams <struct>, %autoresponder <struct> )**

To create an autoresponder the hash %autoresponder is sent as parameter.  
At creation of the autoresponder all fields are required.

```
%autoresponder (
    email_address <string>
    text <string>
)
```

**Result:**

As result the generic hash (<struct>) %result and as additional parameter the hash (<struct>) %autoresponder is returned. %autoresponder is returned to verify the current status and the format of the text of the autoresponder as saved on the mail server.

```
%result (
    status_code <int>
    status_string <string>
    status_detail <string> // only set if information is available
    session_id <string>
    %autoresponder (
        email_address <string>
        status <boolean>
        text <string>
    )
)
```

**Call:****Autoresponder.modify( %trxParams <struct>, %autoresponder <struct>)**

Using the modify method the text of the autoresponder can be modified, the hash %autoresponder is sent as parameter, "email\_address" and "text" are required.

```
%autoresponder (
    email_address <string>
    text <string>
)
```

**Result:**

As result the generic hash (<struct>) %result and as additional parameter the hash (<struct>) %autoresponder is returned:

```
%result (
    status_code <int>
    status_string <string>
    status_detail <string> // only set if information is available
    session_id <string>
    %autoresponder (
        email_address <string>
        status <boolean>
        text <string>
    )
)
```

**Call:****Autoresponder.delete( %trxParams <struct>, \$email\_address <string>)**

To delete an autoresponder the email address is sent as parameter.

**Result:**

As result the generic hash (<struct>) %result and as additional parameter the hash (<struct>) %autoresponder is returned:

```
%result (
    status_code <int>
    status_string <string>
    status_detail <string> // only set if information is available
    session_id <string>
    %autoresponder (
        email_address <string>
        status <boolean>
    )
)
```

## 6. Object: Alias

Methods: get, create, modify, delete

### Method calls:

Call	Return
Alias.get( %trxParams, \$domainname*, \$email_address*)	%result(\$count, @aliases)
Alias.create( %trxParams, \$email_address, \$alias_address, \$blacklist)	%result
Alias.modify( %trxParams, \$alias_address, \$blacklist)	%result
Alias.delete( %trxParams, \$alias_address)	%result

### Notes:

An alias cannot be modified. To change an alias address the alias must be deleted and then created.

- \* = parameter optional
- All other parameters are mandatory

**Please note:** Modification of the Spam and Virus Protection is not supported. Spam and Virus Protection can only be enabled or disabled in the ControlPanel.

### Description:

#### Call:

**Alias.get( %trxParams <struct>, \$domainname <string>, \$email\_address <string>)**

This method retrieves one or more email aliases. The parameters \$domainname and \$email\_address are optional. If both parameters are empty, all email aliases of the whole account are retrieved. If \$domainname is set and \$email\_address is empty, all email aliases of the specified domain name are retrieved. If \$email\_address is set, only the aliases for the requested email address are retrieved. To get a single email alias, \$domainname must be set and the domain name must match the domain part of the email address:

*Example:*

```
$domainname      = "myname.com" ;
$email_address   = "name@myname.com" ;
```

#### Result:

**%result({statusParams}, \$count, @aliases[%alias])**

The result contains the status parameters and the array @aliases, \$count is the number of found aliases of the requested email address. Each element in @aliases is a hash (<struct>) %alias.

```

%result (
  status_code <int>
  status_string <string>
  status_detail <string> // only set if information is available
  session_id <string>
  count <int> // number of aliases found
  @aliases (
    %alias ( // Description see Table.Object.Alias.1
      alias_address <string>
      email_address <string>
      is_external <boolean>
      blacklist <boolean>
      %spam_virus_protection ( // Table.Object.Alias.2
        status <boolean>
        quarantine_url <string>
        quarantine_username <string>
        quarantine_password <string>
        accessable_by_webmail <boolean>
      )
    )
  )
)

```

**Field descriptions:****Table.Object.Alias.1:**

%alias: M=Mandatory (Field required), O=Optional, R=Read Only (cannot be set)

alias_address	M	the Alias email address
email_address	M	corresponding POP3 email address
blacklist	M	0=blacklist disabled, 1=blacklist enabled
is_external	R	0=local alias 1=external alias (forward function)

**Table.Object.Alias.2:**

%spam\_virus\_protection: M=Mandatory (required) = always set, R=Read only (cannot be set)

status	M	possible values: 0=off/1=on
quarantine_url	R	web-URL of the quarantine folder
quarantine_username	R	username for the login of the quarantine folder
quarantine_password	R	password for the login of the quarantine folder
accessable_by_webmail	R	is the quarantine folder access able by web mail? 0=no/1=yes

**Call:**

**Alias.create( %trxParams <struct>, \$email\_address <string>, \$alias\_address <string>, \$blacklist <boolean>)**

To create an alias, the first parameter must be the POP3 email address for which the alias should be created. The second parameter is the alias address. Both parameters are mandatory. Please note that the system automatically recognizes if the created alias is an external alias or not. An alias is always external if it does not belong the same domain name as the associated POP3 email address.

Modification of the Spam and Virus Protection is not supported.

Spam and Virus Protection can only be enabled or disabled in the ControlPanel.

**Important note:** On "create" the alias address is created, but not activated on the mail server. A waiting request for service activation is initiated. You can create more alias addresses (within a loop). At the end you must activate the created addresses. The activation of a service on the mail server is done with the object "Activate".

**Result:**

The following %result hash is returned:

```
%result (
  status_code <int>
  status_string <string>
  status_detail <string> // only set if information is available
  session_id <string>
  waiting_request <boolean>
  %alias = (
    alias_address <string>
    email_address <string>
    is_external <boolean>
    blacklist <boolean>
  )
)
```

**Call:**

**Alias.modify( %trxParams <struct>, \$alias\_address <string>, \$blacklist <boolean>)**

This call is used to enable or disable the blacklist function for an alias address.

Aliases cannot be modified. To change an alias, it must first be deleted and then created.

**Result:**

As result the generic hash (<struct>) %result is returned:

```
%result (
  status_code <int>
  status_string <string>
  status_detail <string> // only set if information is available
  session_id <string>
)
```

**Call:****Alias.delete( %trxParams <struct>, \$alias\_address <string>)**

To delete an existing alias address only the alias address is sent as parameter.

**Please note:** The alias address is deleted, but the action is not activated on the mail server. A waiting request for service activation is initiated. You can delete more alias addresses (within a loop) and at the end you must apply the changes to the mail server with the object "Activate".

**Result:**

As result the generic hash (<struct>) %result is returned:

```
%result (  
    status_code <int>  
    status_string <string>  
    status_detail <string> // only set if information is available  
    session_id <string>  
    waiting_request <boolean>  
)
```

## 7. Object: Zone

Methods: get, modify

### Method calls:

Call	Return
Zone.get( %trxParams, \$domainname)	%result, %zone
Zone.modify( %trxParams, \$domainname)	%result

This object is used to get and modify the Zone of a domain name.

**Please note:** It is only possible to modify existing A-records. To add or remove an A-record please use the ControlPanel. Modifications of the MX records can be done only in the ControlPanel. This is a security reason, because due to wrong MX entries mails can be lost and local mail addresses can be out of function.

### Description:

#### Call:

#### **Zone.get( %trxParams <struct>, \$domainname <string>)**

The Zone.get call retrieves all MX entries and all A records of the requested domain name. To get the zone entries the domain name is sent as parameter.

#### Result:

As result the generic hash %result is returned with the additional parameter %zone. The hash (<struct>) zone contains arrays of the retrieved zone entries. Each array contains a hash (<struct>) for each entry.

```
%result (
  status_code <int>
  status_string <string>
  status_detail <string> // only set if information is available
  session_id <string>
  domainname <string>
  %zone = (
    @MX = (
      %records = (
        fqdn <string>
        priority <int>
        mailserver <string>
      )
    )
    @A = (
      %records = (
        fqdn <string>
        ip <string>
      )
    )
  )
)
```

**Field descriptions:****Table.Object.Zone.1:**

%zone: R=Read Only (cannot be set)

@MX	R	MX Array contains a hash (struct) for each found MX Record
fqdn	R	<b>full qualified domain name</b> , the domain name of the MX record of the local domain, usually the domain name itself in case of an MX record
priority	R	The priority of the MX record
mailserver	R	The full qualified domain name of the mail server
@A	R	The "A" Array contains a hash (struct) for each found A Record
fqdn	R	<b>full qualified domain name</b> of the A-Record
ip	R	The IP number assigned to the full qualified domain name

**Call:****Zone.modify(%trxParams <struct>, %zone <struct>)**

The method call requires the %zone as parameter.

**Attention:** If the A-record is assigned to an existing MX-record it cannot be modified. If the A-record has the IP of an emerion web server and a web server is assigned to the domain name, it cannot be modified. In this case you must downgrade the current package (Resellers must set the web space to "0" in the Reseller area in the ControlPanel).

```
%zone = (
  domainname <string>
  @A = (
    %records = (
      fqdn <string>
      ip <string>
    )
  )
)
```

**Field descriptions:****Table.Object.Zone.2:**

%zone: M=Mandatory (Field required)

domainname	M	The domain name where to modify the zone
@A	R	The "A" Array must contain a hash (struct) for each existing A-record you like to modify.
fqdn	R	<b>full qualified domain name</b> of the A-Record
ip	R	The IP number you like to assign to the full qualified domain name

**Result:****%result({statusParams})**

The result contains the status parameters. The status is only successful if a waiting request exists, otherwise an error will be returned.

**Please note:** After the zone is modified successfully, it must be activated on the name server with the method "Activate.Zone".

```
%result (
  status_code <int>
  status_string <string>
  status_detail <string> // only set if information is available
  session_id <string>
  waiting_request <boolean>
)
```

## 8. Object: Activate

Methods: POP3, Alias

### Method calls:

Call	Return
Activate.POP3( %trxParams )	%result
Activate.Alias( %trxParams )	%result
Activate.Zone( %trxParams )	%result

This object is very important, because it is used to apply changes on the real servers.

**Please note:** This object activates any changes on the servers of the live environment. A waiting request is initiated upon modification of a service that needs special action on the server to apply the changes. The Object "Activate" should be used only once per transaction. If you want to create or delete more email addresses or aliases, create or delete them all, and at the end activate the changes on the using the appropriate method of the "Activate" object.

**Test mode:** Note that you must also implement the call in test mode, but as long as you are in test mode, no waiting requests are created. If you send the method call while no waiting request is active, you will get a result code of "490". This means there is no waiting request because of the test mode. You can safely ignore that error.

The appropriate method of "Activate" must be called after:

- EmailPOP3.create
- EmailPOP3.delete
- Alias.create
- Alias.delete
- Zone.modify

### Description:

#### Call:

#### Activate for all method calls

The method call only requires the transaction parameters.

#### Result:

**%result({statusParams})**

The result contains the status parameters. The status is only successful if a waiting request exists, otherwise an error will be returned.

## 9. Object: Domainpointer

Methods: get, modify

### Method calls:

Call	Return
Domainpointer.get( %trxParams, \$domainname, \$fqdn)	%result, %pointer
Domainpointer.modify( %trxParams, %pointer)	%result

This object is used to get and modify the domain pointer. It only works if a domain pointer already exists for a domain name.

### Description:

#### Call:

**Domainpointer.get( %trxParams <struct>, \$domainname <string>, \$fqdn <string>)**

To get all values of the domain pointer, the base domain name and the full qualified domain name are sent as parameter.

#### Field descriptions:

#### Table.Object.Domainpointer.1:

Parameters: M=Mandatory (required)

domainname	M	The domainname itself (domainname.com)
fqdn	M	<b>full qualified domain name</b> , the domain name of the pointer including www (www.domainname.com) or any other sub domain

**Result:**

As result the generic hash %result is returned with the additional parameter %pointer. The hash (<struct>) pointer contains all values of the domainpointer.

```
%result (
  status_code <int>
  status_string <string>
  status_detail <string> // only set if information is available
  session_id <string>
  domainname <string>
  %pointer = ( // Description see Table.Object.Domainpointer.2
    fqdn <string>
    target_url <string>
    frame_redirect <boolean>
    use_meta <boolean>
    title <string>
    keywords <string>
    description <string>
    shortcut_icon_url <string>
    is_ssl_host <boolean>
  )
)
```

**Field descriptions:****Table.Object.Domainpointer.2:**

%pointer: M=Mandatory, O=Optional, R=Read Only (cannot be set)

domainname	M	The domain name
fqdn	M	<b>full qualified domain name</b> , the full qualified domain name of the domain pointer
target_url	O	The target URL where the domain name is forwarded to
frame_redirect	O	0=direct redirect 1=frame redirect
use_meta	O	0=Meta tags not used 1=Meta tags used
title	O	Title used in HTML
keywords	O	Keywords used for meta tags
description	O	Description used for meta tags
shortcut_icon_url	O	URL to the shortcut icon shown in the location bar of the browser
is_ssl_host	O	0=Target URL is a non-secure server (http://) 1=Target URL is a secure server (https://)

**Call:****Domainpointer.modify( %trxParams <struct>, %pointer <struct>)**

To modify the domain pointer the hash (<struct>) %pointer is sent as parameter.

```
%pointer = ( // Description see Table.Object.Domainpointer.2
    domainname <string>
    fqdn <string>
    target_url <string>
    frame_redirect <boolean>
    use_meta <boolean>
    title <string>
    keywords <string>
    description <string>
    shortcut_icon_url <string>
    is_ssl_host <boolean>
)
```

**Result:**

As result the generic hash (<struct>) %result is returned:

```
%result (
    status_code <int>
    status_string <string>
    status_detail <string> // only set if information is available
    session_id <string>
)
```

## Appendix A

### 1. Input Syntax Correction of Contact Set fields

White space at the beginning and the end of fields is removed.

The first letter in the fields firstname, lastname, address, city are converted to upper case.

All letters in the field zip are converted to upper case.

### 2. Check for Titles in the fields firstname, lastname

Title	match
.	contains
Dr	contains, covered with white space
Ing	contains, covered with white space
Prof	contains, covered with white space
Mag	contains, covered with white space
Dipl	contains, covered with white space
Fam	contains, covered with white space
Fa	contains, covered with white space
Gh	contains, covered with white space
Hr	contains, covered with white space
Fr	contains, covered with white space
Privat	contains, covered with white space
Dir	contains, covered with white space
DI	contains, covered with white space
Dipl-Ing	contains, covered with white space
-Ing	contains, covered with white space
Kommerzialrat	contains, covered with white space
Ordination	contains, covered with white space
Dipl Ing	contains, covered with white space
Familie	contains, covered with white space
Firma	contains, covered with white space
Gasthof	contains, covered with white space
Herr	contains, covered with white space
Frau	contains, covered with white space
Direktor	contains, covered with white space
Magister	contains, covered with white space
Professor	contains, covered with white space
Ord	contains, covered with white space

“Covered with white space” means, that the title is removed, if it matches at the beginning of a string followed by a white space, or the string contains the title and white space is before and after the title, or the string contains white space followed by the title.

### 3. Check for Common Names in the company field

Dots are not removed in the field company.

Common Name	match
Familie	exact
Firma	exact
Gasthof	exact
Herr	exact
Frau	exact
Privat	exact
Direktor	exact
Kommerzialrat	exact
Ordination	exact
Dr	exact
Dr.	exact
Ing	exact
Prof	exact
Prof.	exact
Mag	exact
Mag.	exact
Ing.	exact
Fam.	exact
Fa.	exact
Magister	exact
Professor	exact
Ord.	exact
Ord	exact
Dipl-Ing	exact
Dipl.-Ing.	exact
Dipl.-Ing	exact
Dipl-Ing.	exact
Dipl Ing	exact
Dipl. Ing.	exact
Dipl. Ing	exact
Dipl Ing.	exact

“Exact” means, that the common name is removed if it matches exactly. If the common name stands in context with any other word or name, it is not removed from the company field.

Dots are not removed in the field company.

#### 4. Autoresponder Text Check

<b>field</b>	<b>match</b>
text	minimum 5 Characters [a-zA-Z], with space excluded