TelHosting Software API Guide

Version: 1.0
Date: 2025-10-07
Telnic Limited

Contents

1 Overview

The API described in this chapter is exposed via a publicly available web service, using SOAP over HTTP as its protocol. The API works by receiving request messages from the SOAP clients and sending response messages back to them (unless the request message requires no response).

All messages are expressed in XML, their respective syntax is specified using XML schema.

The root WSDL file for the SOAP client is located at
`https://www.managemy.tel/client?wsdl`

XML schema definitions are recursively imported and linked from there, e.g.:

- Common Types: `https://www.managemy.tel/client?xsd=7`
- Domain Types: `https://www.managemy.tel/client?xsd=1`
- Record Types: `https://www.managemy.tel/client?xsd=5`
- Exchange Types: `https://www.managemy.tel/client?xsd=2`

2 Service Interface for Clients

For the convenience of `.tel` users, all the provisioning of the data stored with their TelHosting Provider is possible not only by using the TelHosting Software's interactive web interface, but also from client applications stored locally on the users' PCs and mobile devices. This includes the management of subdomains, public/private data, profiles, and search data. The SOAP APIs defined in this chapter are devised for this purpose.

To allow administrators to act on behalf of a certain user[1], there is an optional attribute `effUserName` (the effective user name) for most operations described in this section. It is excplicitly noted if this attribute is not allowed. If a user who is not an administrator specifies this attribute, its value has to match the name of the user issuing the request.

2.1 Domain and Zone Management

This API provides a means of maintaining zone and domain objects. The user's permissions are used to determine whether he can actually create, update, and delete these objects. Depending on the TelHosting Provider's policy, the user can create second level domains and zones on his own via the API, or is required to use other services of the TelHosting Provider's infrastructure to allocate domains (maybe the online registration system if the TelHosting Provider is also a Telnic registrar). In this case, the TelHosting Provider himself creates the Zone and Domain objects in the system via the internal interface and an administrator account.

The permissions are also used to determine whether the user may create third and higher level domains and separate zones for these, so that they will appear separately in the DNS, maybe even on a different set of name servers.

2.1.1 API operations summary

A summary of the API operations is shown in table 1. The corresponding schema definition can be found in the file `Domain-1.0.xsd`.

2.1.2 `createZone` operation

This operation creates a new zone object with a given domain name. The user has the choice of specifying the name servers to use himself (which must be, of course, among the name servers that are managed by the TelHosting Software) or letting the system decide which name servers to use. The latter has the advantage that the system can distribute the zones equally depending on the capabilities and loads of the name servers. The maximum number of name servers is subject to the user's permissions. If fewer than the specified number of nameservers are defined the system will choose the remaining. In any case, there may not be fewer than two or more than 13 name servers specified for a given zone (either explicitly or using `count`).

Optionally, the `createZone` operation may also check whether the domain for the new zone is belonging to the TelHosting Provider, by comparing the IANA ID of the domain in the Whois with the one stored in the Partition of the user trying to create the zone. Setting the attribute `synchronized` to `true` will check for existing zones and delete all zones that are belonging to other users and may block the zone that is to be created.

The `createZoneRequest` and `createZoneResponse` elements describe the request and response, respectively. The only element present in the request is `<nameservers>`, specifying the name servers to be used for the new zone. It takes an optional attribute `count` that defines the number of name servers to use and has 0 to `count` subelements of the type `<host>` for the domain names of the hosts the user specifies himself. Optional you may give the attribute `<synchronized>` for the `createZoneRequest`.

[1] See section **??** on page **??** for an explanation of this concept.

operation operation	admin admin	user-admin useradmin	user user	description description
`createZone`	yes	yes	permissions	Creates a new zone (Zone object), with automatic and/or manual name server assignment.
`updateZone`	yes	yes	permissions	Updates the name servers within the zone.
`deleteZone`	yes	yes	permissions	Deletes a zone.
`checkZone`	yes	yes	permissions	Returns if a zone is existing.
`getZone`	yes	yes	yes	Retrieves zone information.
`listZones`	yes	yes	yes	List the names of all zones.
`createDomain`	yes	yes	permissions	Creates a new Domain object.
`updateDomain`	yes	yes	permissions	Updates a Domain object.
`deleteDomain`	yes	yes	permissions	Deletes a Domain object.
`getDomain`	yes	yes	yes	Retrieves domain information.
`listDomains`	yes	yes	yes	Lists the names of all domains.
`getLoggingStatistics`	yes	yes	yes	Get logging statistics for a domain.

Table 1: Summary of domain and zone management API; *permissions* means the user's permissions settings determine whether the operation is allowed or not.

Example:

Create a zone for `reggie.tel` with three name servers. Let the system choose the third name server.

Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/domain/types-1.0">
   <soap:Header/>
   <soap:Body>
      <typ:createZoneRequest zoneName="reg.reggie.tel" synchronized="true">
         <typ:nameservers count="3">
            <typ:host>ns1.example.com</typ:host>
            <typ:host>ns2.example.com</typ:host>
         </typ:nameservers>
      </typ:createZoneRequest>
   </soap:Body>
</soap:Envelope>
```

Response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
   <S:Body>
      <createZoneResponse
        xmlns="http://xmlns.telnic.org/ws/nsp/client/record/types-1.0"/>
   </S:Body>
</S:Envelope>
```

2.1.3 `updateZone` operation

This operation updates the information for a given zone object.

The `updateZoneRequest` and `updateZoneResponse` elements describe the request and response, respectively. As in the case of `createZone`, the only element present in the request is `<nameservers>`.

Example:

Update the zone of `reggie.tel` to use only `ns3` and `ns4`.

Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/domain/types-1.0">
   <soap:Header/>
   <soap:Body>
      <typ:updateZoneRequest zoneName="reg.reggie.tel">
         <typ:nameservers>
            <typ:host>ns3.example.com</typ:host>
            <typ:host>ns4.example.com</typ:host>
         </typ:nameservers>
      </typ:updateZoneRequest>
   </soap:Body>
</soap:Envelope>
```

Response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
   <S:Body>
      <updateZoneResponse
        xmlns="http://xmlns.telnic.org/ws/nsp/client/domain/types-1.0"/>
   </S:Body>
</S:Envelope>
```

2.1.4 `deleteZone` operation

This operation deletes a zone object.

The `deleteZoneRequest` and `deleteZoneResponse` elements describe the request and response, respectively. The request does not have any elements, only an attribute `zoneName` specifying the name of the zone object to be deleted.

Example:

Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/domain/types-1.0">
   <soap:Header/>
   <soap:Body>
      <typ:deleteZoneRequest zoneName="reg.reggie.tel"/>
   </soap:Body>
</soap:Envelope>
```

Response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
   <S:Body>
      <deleteZoneResponse
        xmlns="http://xmlns.telnic.org/ws/nsp/client/domain/types-1.0"/>
   </S:Body>
</S:Envelope>
```

2.1.5 `checkZone` operation

This operation returns a boolean value indicating if the given zone is existing.

The `checkZoneRequest` and `checkZoneResponse` elements describe the request and response, respectively.

Example:

Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/domain/types-1.0">
   <soap:Header/>
   <soap:Body>
      <typ:checkZoneRequest zoneName="reg.reggie.tel"/>
   </soap:Body>
</soap:Envelope>
```

Response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
  <S:Body>
    <ns4:checkZoneResponse
      xmlns:ns4="http://xmlns.telnic.org/ws/nsp/client/domain/types-1.0">
        <ns4:exists>true</ns4:exists>
    </ns4:checkZoneResponse>
  </S:Body>
</S:Envelope>
```

2.1.6 `getZone` operation

This operation retrieves the data associated with the zone, i.e. the name servers.

The `getZoneRequest` and `getZoneResponse` elements describe the request and response, respectively. As in the case of `createZone` above, the only element present in the request is `<nameservers>`.

Example:

Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/domain/types-1.0">
   <soap:Header/>
   <soap:Body>
      <typ:getZoneRequest zoneName="reg.reggie.tel"/>
   </soap:Body>
</soap:Envelope>
```

Response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
  <S:Body>
    <ns4:getZoneResponse
      xmlns:ns4="http://xmlns.telnic.org/ws/nsp/client/domain/types-1.0"">
        <ns4:nameservers>
          <ns4:host>ns4.example.com</ns4:host>
          <ns4:host>ns3.example.com</ns4:host>
       </ns4:nameservers>
    </ns4:getZoneResponse>
  </S:Body>
</S:Envelope>
```

2.1.7 `listZones` operation

This operation lists all zones associated with a user.

The `listZonesRequest` and `listZonesResponse` elements describe the request and response, respectively. There are no particular elements or attributes for this kind of request.

Example:

Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/domain/types-1.0">
   <soap:Header/>
   <soap:Body>
      <typ:listZonesRequest/>
   </soap:Body>
</soap:Envelope>
```

Response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
   <S:Body>
      <ns4:listZonesResponse
        xmlns:ns4="http://xmlns.telnic.org/ws/nsp/client/domain/types-1.0">
         <ns4:zoneName>reggie.tel</ns4:zoneName>
         <ns4:zoneName>reg.reggie.tel</ns4:zoneName>
         <ns4:zoneName>greg.reggie.tel</ns4:zoneName>
      </ns4:listZonesResponse>
   </S:Body>
</S:Envelope>
```

2.1.8 `createDomain` operation

This operation creates a domain name in the system. It is required to associate profiles, resource records, and private data to a domain name.

The `createDomainRequest` and `createDomainResponse` elements describe the request and response, respectively. The name of the domain to be created is given by the required `domainName` attribute. The optional element `<displayString>` may be added to the created domain. The domain display string is displayed in several occasions instead of the domain name. The optional element `<googleAnalyticsId>` defines whether the domain should include Google Analytics code when displayed. If omitted it is deactivated, otherwise it has to contain a valid google analytics id. Optionally the following proxy design properties can be set for apex domains (this is not allowed

for sub-domains): the <color1>, the <color2>, the <color3>, the <color4>, the <css> (id for the css template that should be used; note that not all templates make use of all four different colours), the <pss> (the proxy search scope, either NoSearch if no search box should be displayed at the proxy page, SearchThis if search should be restricted to the current domain hierarchy or SearchAll if all .tel domains will be searched), the <hml> (either true if the "Manage .tel" link should be hidden or false (default) in case the "Manage .tel" link should be displayed), the <htl> (either true if the link to Telnic should be hidden or false (default) if the Telnic link should be displayed), the <bkg> (the background image location) and <bip> to define what background image (if any) should be displayed and whether it should be displayed normally (bip=1), stretched (bip=2), or tiled (bip=3), the <hdrbkg> (the header background image location) and <hdrurl> (the header URL) to define what header background image (if any) should be displayed and what it should be linked to. Finally the <showLogging> can be used to decide whether the logging statistics can be viewed in the Control Panel. Compare also the following updateDomain request.

Example:

Creates a domain.

Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/domain/types-1.0">
   <soap:Header/>
   <soap:Body>
      <typ:createDomainRequest domainName="greg.reggie.tel">
        <typ:displayString>The domain of Greg Reggie</typ:displayString>
        <typ:color1>#123456</typ:color1>
        <typ:color3>#000000</typ:color3>
        <typ:css>1</typ:css>
        <typ:pss>SearchThis</typ:pss>
        <typ:htl>true</typ:htl>
        <typ:googleAnalyticsId>1234567</typ:googleAnalyticsId>
        <typ:bkg>http://example.com/myBackground.png</typ:bkg>
        <typ:bip>2</typ:bip>
        <typ:hdrbkg>http://example.com/myImage.jpg</typ:hdrbkg>
        <typ:hdrurl>http://myimage.tel</typ:hdrurl>
        <typ:showLogging>true</typ:showLogging>
      </typ:createDomainRequest>
   </soap:Body>
</soap:Envelope>
```

Response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
   <S:Body>
      <createDomainResponse
        xmlns="http://xmlns.telnic.org/ws/nsp/client/domain/types-1.0"/>
   </S:Body>
</S:Envelope>
```

2.1.9 `updateDomain` operation

This operation updates the name of a domain object, the domain's display string in the system, the Google Analytics setting, and the proxy design properties. The display string is displayed instead of the domain name on many occasions. It can be removed by setting an empty domain display string. The Google Analytics settings can be activated by supplying a valid Google Analytics Id and

deactivated by supplying an empty string. The proxy design properties define how the domain will be displayed at the TelProxy. You can set up to four different colours (defined by their six digit hex code preceded by the # character, e.g., #ff0000 for the colour red), choose a css template (by giving its unique id; note that some css templates do not make us of all four colours), choose a background image and the way it should be displayed (i.e., normal, stretched, tiled), define whether the search on the TelProxy should be disabled (proxy search scope set to `NoSearch`), restricted to your domain hierarchy (proxy search scope set to `SearchThis` or whether all .tel domains will be searched (proxy search scope set to `SearchAll`, which is also the default value). Furthermore you can define whether the link to manage your .tel domain and the link to Telnic should be displayed on your proxy page.

The `updateDomainRequest` and `updateDomainResponse` elements describe the request and response, respectively. All elements present are optional: the `<displayString>`, the `<googleAnalyticsId>`, the `<newDomainName>`, the `<color1>`, the `<color2>`, the `<color3>`, the `<color4>`, the `<css>` (id for the css template that should be used), the `<pss>` (the proxy search scope, either `NoSearch`, `SearchThis` or `SearchAll`), the `<hml>` (hide manage link, either true or false), the `<htl>` (hide Telnic link, either true or false), the `<bkg>` (background image), the `<bip>` (background image display preference), the `<hdrbkg>` (header background image), and the `<hdrurl>` (header URL). Note that all proxy design properties and the Google Analytics Id may only be set for the apex domain and will affect all sub-domains, too. Finally the `<showLogging>` can be used to decide whether the logging statistics can be viewed in the Control Panel.

It is only possible to rename non-apex domains. All sub-domains will be renamed accordingly, i.e., if `b.a.tel` is renamed `z.a.tel` a possible sub-domain `c.b.a.tel` will automatically be renamed `c.z.a.tel`. At the same time all non-terminal NAPTRs in the current user's account which point to one of the domains to be renamed will be changed such that they point to the respective new domain name. All other domain specific items (e.g., all other records, groups, profiles) will remain as they are.

Example:

Updates a domain.

Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/domain/types-1.0">
   <soap:Header/>
   <soap:Body>
      <typ:updateDomainRequest domainName="greg.reggie.tel">
         <!-Optional:->
         <typ:newDomainName>john.reggie.tel</typ:newDomainName>
         <!-Optional:->
         <typ:displayString>The domain of Greg Reggie</typ:displayString>
         <!-Optional:->
         <typ:color1>#000000</typ:color1>
         <!-Optional:->
         <typ:color2>#ffffff</typ:color2>
         <!-Optional:->
         <typ:color3>#89abcd</typ:color3>
         <!-Optional:->
         <typ:css>1</typ:css>
         <!-Optional:->
         <typ:pss>SearchAll</typ:pss>
         <!-Optional:->
         <typ:hml>false</typ:hml>
         <!-Optional:->
         <typ:htl>false</typ:htl>
         <!-Optional:->
         <typ:googleAnalyticsId>1234567</typ:googleAnalyticsId>
         <typ:showLogging>false</typ:showLogging>
      </typ:updateDomainRequest>
   </soap:Body>
</soap:Envelope>
```

Response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
   <S:Body>
      <updateDomainResponse
        xmlns="http://xmlns.telnic.org/ws/nsp/client/domain/types-1.0"/>
   </S:Body>
</S:Envelope>
```

2.1.10 `deleteDomain` operation

This operation removes a domain from the system.

The `deleteDomainRequest` and `deleteDomainResponse` elements describe the request and response, respectively. The name of the domain to be deleted is given by the required `domainName` attribute. The optional attribute `recursive` may be set to true in order to delete the whole domain tree below the given domain.

Example:

Delete the domain `reg-reggie.tel`.

Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/domain/types-1.0">
   <soap:Header/>
   <soap:Body>
      <typ:deleteDomainRequest domainName="reg-reggie.tel" recursive="true"/>
   </soap:Body>
</soap:Envelope>
```

Response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
   <S:Body>
      <deleteDomainResponse
        xmlns="http://xmlns.telnic.org/ws/nsp/client/domain/types-1.0"/>
   </S:Body>
</S:Envelope>
```

2.1.11 `getDomain` operation

This operation returns the display string (if set), the Google Analytics Id, and the proxy design properties (i.e., color1, color2, color3, color4, css template id, proxy search scope, hml, and htl) for a given domain name. For an explanation of these values see the updateDomain request.

The `getDomainRequest` and `getDomainResponse` elements describe the request and response, respectively. The name of the domain to be retrieved is given by the required `domainName` attribute.

Example:

Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/domain/types-1.0">
   <soap:Header/>
   <soap:Body>
      <typ:getDomainRequest domainName="greg.reggie.tel"/>
   </soap:Body>
</soap:Envelope>
```

Response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
   <S:Body>
      <ns4:getDomainResponse
        xmlns:ns4="http://xmlns.telnic.org/ws/nsp/client/domain/types-1.0">
        <ns4:displayString>Cartman's Domain</ns4:displayString>
        <ns4:color1>#db1cdb</ns4:color1>
        <ns4:color2>#f5f5f5</ns4:color2>
        <ns4:color3>#bd42bd</ns4:color3>
        <ns4:color4>#112233</ns4:color4>
        <ns4:css>4</ns4:css>
        <ns4:pss>SearchThis</ns4:pss>
        <ns4:hml>false</ns4:hml>
        <ns4:htl>false</ns4:htl>
        <ns4:googleAnalyticsId>1234567</ns4:googleAnalyticsId>
        <ns4:bkg>http://example.com/myBackground.png</ns4:bkg>
        <ns4:bip>2</ns4:bip>
        <ns4:hdrbkg>http://example.com/myImage.jpg</ns4:hdrbkg>
        <ns4:hdrurl>http://myimage.tel</ns4:hdrurl>
        <ns4:showLogging>true</ns4:showLogging>
      </ns4:getDomainResponse>
   </S:Body>
</S:Envelope>
```

2.1.12 `listDomains` operation

This operation lists all domains.

The `listDomainsRequest` and `listDomainsResponse` elements describe the request and response, respectively. If the optional attribute `apexonly` is set to true only the user's apex domains (i.e., those that do not have any super domain) will be listed. Allowed elements in the request are:

- `<domains>` (optional and only possible if `apexonly` is not used) specifying the list of domains (as a sequence of `<domainName>` elements) whose sub-domains should be listed.

The user can associate data with all the domains listed in the response.

Example 1:

This request will list all apex domains of the current user.

Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/domain/types-1.0">
   <soap:Header/>
   <soap:Body>
      <typ:listDomainsRequest apexonly="true"/>
   </soap:Body>
</soap:Envelope>
```

Response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
   <S:Body>
      <ns4:listDomainsResponse
        xmlns:ns4="http://xmlns.telnic.org/ws/nsp/client/domain/types-1.0">
         <ns4:domainName>reggie.tel</ns4:domainName>
         <ns4:domainName>regina.tel</ns4:domainName>
      </ns4:listDomainsResponse>
   </S:Body>
</S:Envelope>
```

Example 2:

This request will list all sub-domains of reggie.tel and test.regina.tel (including those two domains). Note, that the domain regina.tel will not be listed.

Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/domain/types-1.0">
   <soap:Header/>
   <soap:Body>
      <typ:listDomainsRequest apexonly="false">
         <typ:domains>
            <!-Zero or more repetitions:->
            <typ:domainName>reggie.tel</typ:domainName>
            <typ:domainName>test.regina.tel</typ:domainName>
         </typ:domains>
      </typ:listDomainsRequest>
   </soap:Body>
</soap:Envelope>
```

Response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
   <S:Body>
      <ns4:listDomainsResponse
        xmlns:ns4="http://xmlns.telnic.org/ws/nsp/client/domain/types-1.0">
         <ns4:domainName>reggie.tel</ns4:domainName>
         <ns4:domainName>sub1.reggie.tel</ns4:domainName>
         <ns4:domainName>test.regina.tel</ns4:domainName>
         <ns4:domainName>a.test.regina.tel</ns4:domainName>
         <ns4:domainName>b.test.regina.tel</ns4:domainName>
      </ns4:listDomainsResponse>
   </S:Body>
</S:Envelope>
```

2.1.13 `getLoggingStatistics` operation

This operation retrieves the logging statistics for a specific domain over a specified time period.

The `getLoggingStatistics` and `getLoggingStatistics` elements describe the request and response, respectively. The name of the domain for which statistics should be retrieved is given by the required `domainName` attribute. Two further attributes are mandatory: the `type` which has to be either `mobile` or `desktop`; the `timeFrame` takes the values `lastSevenDays`, `currentCalendarMonth`, `lastCalendarMonth`, `currentCalendarYear`, `lastCalendarYear`, or `allTime`.

Example:

Get the logging statistics for the zone `reggie.tel` for all desktop request within the last seven days.

Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/domain/types-1.0">
   <soap:Header/>
   <soap:Body>
      <typ:getLoggingStatisticsRequest domainName="cartman.tel"
         type="desktop" timeFrame="lastSevenDays"/>
   </soap:Body>
</soap:Envelope>
```

Response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
   <S:Body>
     <ns4:getLoggingStatisticsResponse
       xmlns:ns4="http://xmlns.telnic.org/ws/nsp/client/domain/types-1.0">
         <ns4:pageViewCount>1</ns4:pageViewCount>
         <ns4:phoneClickCount>2</ns4:phoneClickCount>
         <ns4:urlClickCount>3</ns4:urlClickCount>
         <ns4:emailClickCount>4</ns4:emailClickCount>
      </ns4:getLoggingStatisticsResponse>
   </S:Body>
</S:Envelope>
```

2.2 Resource Record Management

This API allows the management of resource records that are published in a domain. For details about resource records, see section **??**.

2.2.1 API operations summary

A summary of the API operations is shown in table 2. The corresponding schema definition can be found in the file `Record-1.0.xsd`.

operation	**description**
`storeRecord`	Adds new or replaces existing record.
`deleteRecord`	Deletes record.
`listRecords`	Lists records of a domain.
`updateRecords`	Bulk update of a domain's content.

Table 2: Summary of the resource record management API.

For all these operations, there is a required attribute, `domainName`, specifying the name of the domain on which to act.

2.2.2 `storeRecord` operation

This operation allows a single record to be stored. If no ID is given, the record is added to the current set of records and assigned a new ID. If an ID is given that already exists, the record associated with it is deleted first, i.e. the new record replaces the old one. It is not required that the previous record is similar to the new one in any aspect. If an ID is given that does not exist, an error message is returned. The response repeats the ID given in the request or returns the newly assigned ID.

The `storeRecordRequest` and `storeRecordResponse` elements describe the request and response, respectively. Allowed elements in the request (exactly one of the top-level elements must be present) with their respective subelements are:

- `<txt>` uses one or multiple subelements `<text>` to specify the free text contact data;
- `<naptr>` for encrypted contact information, subelements:
  - `<order>`,
  - `<preference>`,
  - `<flags>`,
  - `<services>`,
  - exactly one of `<regexp>` or `<replacement>`, and
  - `<longLabel>` (optional);
- `<srv>` has the subelements `<priority>`, `<weight>`, `<port>`, and `<target>`;
- `<mx>` has the subelements `<priority>` and `<exchange>`;
- `<loc>` has the subelements
  - `<latitude>` and `<longitude>` with subelements `<degrees>`, `<minutes>` (optional), and `<seconds>` (optional);
  - `<altitude>`;
  - `<size>` (optional);
  - `<horizPre>` (optional);
  - `<vertPre>` (optional);
- `<ninfo>` uses one or multiple subelements `<text>` to specify the node information;
- `<adsense>` support for the Google AdSense program, subelements:
  - `<displayPreference>`,
  - `<preference>`,
  - `<publisherId>`, and
  - `<adUnitId>`;
- `<video>` support for the Video records, subelements:
  - `<videoUrl>` and
  - `<provider>`;
- `<imagead>` support for the Image Ad records, subelements:
  - `<displayPreference>`,

  - `<preference>`,
  - `<imageUrl>`,
  - `<adLink>`, and
  - `<caption>` (optional);

- `<offer>` support for the Offer records, subelements:
  - `<displayPreference>`,
  - `<preference>`,
  - `<startDate>` (optional),
  - `<endDate>` (optional),
  - `<businessCategory>` (optional),
  - `<online>` (optional),
  - `<promoCode>` (optional),
  - `<discount>` (optional),
  - `<redeem>`,
  - `<title>`,
  - `<imageType>`,
  - `<desc>`,
  - `<offerLink>` (optional),
  - `<imageUri>` (optional),
  - `<terms>` (optional);
- `<file>` support for the File records, subelements:
  - `<fileUrl>`,
  - `<longLabel>` (optional),
  - `<priority>`;
- `<payment>` support for the Payment records, subelements:
  - `<displayPreference>`,
  - `<preference>`,
  - `<provider>`,
  - `<merchantId>`,
  - `<productType>`,
  - `<price>`,
  - `<shipping>` (optional),
  - `<discount>` (optional),
  - `<currency>`,
  - `<productName>`,
  - `<desc>`,
  - `<imageUri>` (optional),
  - `<terms>` (optional);

- `<generic>` has the subelements `<type>` and `<rdata>`.

All these elements can have the following attributes:

- `owner` defines the owner name of the record relative to the domain in which it is contained. Similar to the DNS notation, an at-sign (`@`) denotes the domain's name itself.
- `class` describes the DNS class the record belongs to. It can be either numeric or symbolic (using the well-known constants). The default is `IN`.
- `ttl` sets the *time-to-live* value of the record, which defines the maximum time in seconds a record may be cached by a name server. If omitted, a system default is used.
- `profiles` may contain a list of IDs of profiles with which the record shall be associated. If the attribute is omitted or is empty, the record is associated with the default profile. To prevent the association with any profile, the special value `_none_` may be used. If the record shall appear in all profiles, including those created at a later point in time, the special value `_all_` may be used.
- `groups` may contain a list of IDs of groups with which the record shall be associated. Note that in case of non-concealable records, the attribute must be either omitted or must be empty
- `id` specifies the ID of an existing record that shall be replaced by this record.
- `recursive` flag whether the record should be recursive, i.e., it will also be published in all (existing and future) sub-domains (depending on its profile and group association, of course).

Example 1:

create a `TXT` record with the owner name `sample.reggie.tel`.

Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/record/types-1.0">
   <soap:Header/>
   <soap:Body>
      <typ:storeRecordRequest domainName="reggie.tel">
         <typ:txt ttl="3600" owner="sample" profiles="5" groups="">
            <typ:text>Hello World!</typ:text>
         </typ:txt>
      </typ:storeRecordRequest>
   </soap:Body>
</soap:Envelope>
```

Response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
   <S:Body>
      <storeRecordResponse
        xmlns="http://xmlns.telnic.org/ws/nsp/client/record/types-1.0">
         <id>61</id>
      </storeRecordResponse>
   </S:Body>
</S:Envelope>
```

Example 2:

create a `NAPTR` record with a link to a world wide web page and a long label describing this link in more detail (assumes that a record with ID 58 already exists). Note the terminating dot used in the `<replacement>` element to obtain an absolute domain name.

Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/record/types-1.0">
   <soap:Header/>
   <soap:Body>
      <typ:storeRecordRequest domainName="reggie.tel">
         <typ:naptr groups="12 11" id="58">
            <typ:order>10</typ:order>
            <typ:preference>10</typ:preference>
            <typ:flags></typ:flags>
            <typ:services></typ:services>
            <typ:replacement>www.example.com.</typ:replacement>
            <typ:longLabel>Example link to describe the storeRecord SOAP command</typ:longLabel>
         </typ:naptr>
      </typ:storeRecordRequest>
   </soap:Body>
</soap:Envelope>
```

The response looks like the previous one, except for the ID which is 58 in this case.

2.2.3 `deleteRecord` operation

This operation allows to delete a record with the given ID. The only element in this kind of request is `<id>`, the ID of the record that is to be deleted.

Example:

Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/record/types-1.0">
   <soap:Header/>
   <soap:Body>
      <typ:deleteRecordRequest domainName="reggie.tel">
         <typ:id>58</typ:id>
      </typ:deleteRecordRequest>
   </soap:Body>
</soap:Envelope>
```

Response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
   <S:Body>
      <deleteRecordResponse
        xmlns="http://xmlns.telnic.org/ws/nsp/client/record/types-1.0"/>
   </S:Body>
</S:Envelope>
```

2.2.4 `listRecords` operation

With the help of this operation, the client can retrieve all records associated with a given domain name. There are no elements in this kind of request.

Example:

Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/record/types-1.0">
   <soap:Header/>
   <soap:Body>
      <typ:listRecordsRequest domainName="reggie.tel"/>
   </soap:Body>
</soap:Envelope>
```

Response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
   <S:Body>
      <listRecordsResponse
        xmlns="http://xmlns.telnic.org/ws/nsp/client/record/types-1.0">
         <txt class="1" groups="" id="61" owner="sample" profiles="5" ttl="3600">
            <text>Hello World!</text>
         </txt>
         <naptr class="1" groups="13" id="62" owner="@" profiles="5" recursive="false">
            <order>102</order>
            <preference>10</preference>
            <flags>u</flags>
            <services>E2U+mailto</services>
            <regexp>!^.*$!mailto:myemail@example.com!</regexp>
         </naptr>
         <naptr class="1" groups="12 11" id="68" owner="@" profiles="5" ttl="7200"
           recursive="false">
            <order>99</order>
            <preference>100</preference>
            <flags/>
            <services/>
            <replacement>www.example.com.</replacement>
            <longLabel>Just an example link</longLabel>
         </naptr>
      </listRecordsResponse>
   </S:Body>
</S:Envelope>
```

2.2.5 `updateRecords` operation

This operation can update a set of records at once. Additional flexibility and efficiency is gained by two details. First, the client can choose to delete the whole existing content, simplifying a “replace all” implementation. Second, the client can group records that share the same attributes (`owner`, `class`, `ttl`, `profiles`, `groups`) via group elements, thus saving space in transmission. For records that have a given ID, the same semantics as with the `storeRecord` operation applies: if a record with the given ID already exists, it is replaced. Thus, records do not need to be explictly deleted via the delete element.

The `updateRecordsRequest` and `updateRecordsResponse` elements describe the request and response, respectively. Allowed elements in the request with their respective subelements are:

- `<deleteAll>` indicates that all existing records shall be eliminated before the addition of the records contained in this request;
- `<delete>` identifies a single record which shall be deleted, repetitions are allowed, but `<delete>` must not be used together with `<deleteAll>`;
- `<txt>`, `<naptr>`, `<srv>`, `<mx>`, `<loc>`, `<ninfo>`, `<adsense>`, `<video>`, `<imagead>`, `<offer>`, `<file>`, `<payment>`, and `<generic>` (see `storeRecord` above);
- `<group>` (instead of one of the elements listed in the previous item) defines an element that allows grouping of records that share the same attributes, within a group, elements like `<naptr>` or `<mx>` and further subgroups can be mixed.

Example:

Delete all existing records, then create a text record and a few records belonging to the profile with ID 2, using a subgroup to overwrite the `owner` attribute for the mail exchange records.

Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/record/types-1.0">
   <soap:Header/>
   <soap:Body>
      <typ:updateRecordsRequest domainName="reggie.tel">
         <typ:deleteAll/>
         <typ:txt owner="@" class="in">
            <typ:text>Hallo Welt!</typ:text>
         </typ:txt>
         <typ:group owner="mystuff" profiles="2">
           <typ:naptr class="in" ttl="1000">
             <typ:order>10</typ:order>
             <typ:preference>20</typ:preference>
             <typ:flags>a</typ:flags>
             <typ:services>b</typ:services>
             <typ:regexp>c</typ:regexp>
           </typ:naptr>
           <typ:naptr class="1" groups="13" id="62" owner="@" profiles="5">
             <typ:order>42</typ:order>
             <typ:preference>10</typ:preference>
             <typ:flags>u</typ:flags>
             <typ:services>E2U+mailto</typ:services>
             <typ:regexp>!^.*$!mailto:theemail@example.com!</typ:regexp>
           </typ:naptr>
           <typ:group owner="mail" profiles="2">
              <typ:mx class="in" ttl="3928">
                <typ:priority>20</typ:priority>
                <typ:exchange>mymail1.example.com.</typ:exchange>
              </typ:mx>
              <typ:mx class="in" ttl="3928">
                <typ:priority>20</typ:priority>
                <typ:exchange>mymail2.example.com.</typ:exchange>
              </typ:mx>
           </typ:group>
         </typ:group>
      </typ:updateRecordsRequest>
   </soap:Body>
</soap:Envelope>
```

Response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
   <S:Body>
      <updateRecordsResponse
        xmlns="http://xmlns.telnic.org/ws/nsp/client/record/types-1.0"/>
   </S:Body>
</S:Envelope>
```

2.3 Public/Private Data Interface

This interface provides all the functionality required by clients for private data management, i.e. provisioning of readers and the grouping of readers and associated resource records. Please refer to section **??** for a general description of the concepts behind the organization of private data.

2.3.1 API operations summary

A summary of the API operations is shown in table 3. The corresponding schema definition can be found in the file `Reader-1.0.xsd`.

operation	**description**
`createReader`	Create a new reader for a domain.
`updateReader`	Update an existing reader with new data.
`deleteReader`	Delete an existing reader from a domain.
`getReader`	Get all data of a specific reader.
`listReaders`	List all readers stored for a domain.
`createGroup`	Create a new group for a domain.
`updateGroup`	Update an existing group with new data.
`deleteGroup`	Delete an existing group from a domain.
`getGroup`	Get all data of a specific group.
`listGroups`	List all groups stored for a domain.

Table 3: Summary of the public/private data interface API; *permissions* means the user's permissions settings determine whether the operation is allowed or not.

2.3.2 `createReader` operation

This operation creates a new reader and links it to a domain name for which the reader shall gain access. Optionally, the reader can be associated with an initial set of groups. The reader is identified by his domain name, which is a pseudo domain name assigned by the SO system in the case that the reader is a so-called guest user. The system assigns an ID to this reader and returns it in the response.

The `createReaderRequest` and `createReaderResponse` elements describe the request and response, respectively. Allowed elements in the request are:

- `<name>` is the name of the reader;
- `<readerPseudoDomainName>` specifies the reader's pseudo domain name to be used to calculate the private sub-domain label;

- `<keyLocation>` specifies the domain where the public key of the new reader can be found;
- `<reference>` (optional) specifies the reference value received in the friending request from this reader, if any;
- `<groups>` (optional) specifies the list of groups to which the reader should belong.

The response contains the `<id>` of the new reader as well as a `<label>`, which is part of the subdomain containing private data this reader should be able to read.

Example:

Creates a reader and adds her to the group with id 16 (assuming such a group exists).

Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/reader/types-1.0">
   <soap:Header/>
   <soap:Body>
      <typ:createReaderRequest domainName="reggie.tel">
         <typ:name>Jane Doe</typ:name>
         <typ:readerPseudoDomainName>1234.reader.keys.tel</typ:readerPseudoDomainName>
         <typ:keyLocation>jane.doe.keys.tel</typ:keyLocation>
         <typ:reference>ffb7afb10fc19137ecd42f298ea6acb247c48141</typ:reference>
         <typ:groups>16</typ:groups>
      </typ:createReaderRequest>
   </soap:Body>
</soap:Envelope>
```

Response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
   <S:Body>
      <ns2:createReaderResponse
        xmlns:ns2="http://xmlns.telnic.org/ws/nsp/client/reader/types-1.0">
         <ns2:id>11</ns2:id>
         <ns2:label>61e09be55eb4fd64cc20d802bdfa22722ae23efa</ns2:label>
      </ns2:createReaderResponse>
   </S:Body>
</S:Envelope>
```

2.3.3 `updateReader` operation

This operation allows to update the reader data and/or to update the groups the reader is associated with.

The `updateReaderRequest` and `updateReaderResponse` elements describe the request and response, respectively. Allowed elements in the request are:

- `<id>` specifying which reader to update;
- `<name>` is the name of the reader;
- `<readerPseudoDomainName>` specifies the reader's pseudo domain name to be used to calculate the private sub-domain label;
- `<keyLocation>` specifies the domain where the public key of the new reader can be found;

- `<reference>` (optional) specifies the reference value received in the friending request from this reader, if any;
- `<groups>` (optional) specifies the new list of group ids to which the reader should belong.

In order to remove a reader from all groups, an empty list of groups can be submitted.

The response contains the `<label>` of the new reader, which is part of the subdomain containing private data this reader should be able to read.

Example:

Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/reader/types-1.0">
   <soap:Header/>
   <soap:Body>
      <typ:updateReaderRequest domainName="reggie.tel">
         <typ:id>11</typ:id>
         <typ:name>Jane Doe</typ:name>
         <typ:readerPseudoDomainName>1234.reader.keys.tel</typ:readerPseudoDomainName>
         <typ:keyLocation>jane.doe.keys.tel</typ:keyLocation>
         <typ:reference>ffb7afb10fc19137ecd42f298ea6acb247c48141</typ:reference>
         <typ:groups>16</typ:groups>
      </typ:updateReaderRequest>
   </soap:Body>
</soap:Envelope>
```

Response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
   <S:Body>
      <updateReaderResponse
        xmlns="http://xmlns.telnic.org/ws/nsp/client/reader/types-1.0"/>
         <typ:label>61e09be55eb4fd64cc20d802bdfa22722ae23efa</typ:label>
   </S:Body>
</S:Envelope>
```

2.3.4 `deleteReader` operation

This operation deletes a reader of the specified domain name.

The `deleteReaderRequest` and `deleteReaderResponse` elements describe the request and response, respectively. The only element present in the request is the `<id>` of the reader that is to be deleted.

Example:

Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/reader/types-1.0">
   <soap:Header/>
   <soap:Body>
      <typ:deleteReaderRequest domainName="reggie.tel">
         <typ:id>11</typ:id>
      </typ:deleteReaderRequest>
   </soap:Body>
</soap:Envelope>
```

Response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
   <S:Body>
      <deleteReaderResponse
        xmlns="http://xmlns.telnic.org/ws/nsp/client/reader/types-1.0"/>
   </S:Body>
</S:Envelope>
```

2.3.5 `getReader` operation

Retrieves information about a reader.

The `getReaderRequest` and `getReaderResponse` elements describe the request and response, respectively. The only element present in the request is the `<id>` of the reader.

Example:

Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/reader/types-1.0">
   <soap:Header/>
   <soap:Body>
      <typ:getReaderRequest domainName="reggie.tel">
         <typ:id>11</typ:id>
      </typ:getReaderRequest>
   </soap:Body>
</soap:Envelope>
```

Response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
   <S:Body>
      <ns2:getReaderResponse
        xmlns:ns2="http://xmlns.telnic.org/ws/nsp/client/reader/types-1.0">
         <ns2:name>Jane Doe</ns2:name>
         <ns2:label>61e09be55eb4fd64cc20d802bdfa22722ae23efa</ns2:label>
         <ns2:keyLocation>jane.doe.keys.tel</ns2:keyLocation>
         <ns2:reference>ffb7afb10fc19137ecd42f298ea6acb247c48141</ns2:reference>
         <ns2:readerPseudoDomainName>1234.reader.keys.tel</ns2:readerPseudoDomainName>
         <ns2:groups>16</ns2:groups>
      </ns2:getReaderResponse>
   </S:Body>
</S:Envelope>
```

2.3.6 `listReaders` operation

This operation lists all readers associated with a domain name.

The `listReadersRequest` and `listReadersResponse` elements describe the request and response, respectively. There are no elements in this kind of request.

Example:

Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/reader/types-1.0">
   <soap:Header/>
   <soap:Body>
      <typ:listReadersRequest domainName="reggie.tel"/>
   </soap:Body>
</soap:Envelope>
```

Response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
   <S:Body>
      <ns2:listReadersResponse
        xmlns:ns2="http://xmlns.telnic.org/ws/nsp/client/reader/types-1.0">
         <ns2:reader id="11">Jane Doe</ns2:reader>
         <ns2:reader id="3">John Doe</ns2:reader>
      </ns2:listReadersResponse>
   </S:Body>
</S:Envelope>
```

2.3.7 `createGroup` operation

This creates a new group. The system assigns an ID to this group and returns it in the response.

The `createGroupRequest` and `createGroupResponse` elements describe the request and response, respectively. Allowed elements in the request are:

- `<name>` of the group, will be displayed to the user;
- `<readers>` (optional) specifying a list of the readers that shall initially belong to the created group;
- `<records>` (optional) specifying the IDs of the resource records that shall initially be associated with the created group.

Example:

Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/reader/types-1.0">
   <soap:Header/>
   <soap:Body>
      <typ:createGroupRequest domainName="reggie.tel">
         <typ:name>Friends</typ:name>
         <typ:readers>15 1</typ:readers>
      </typ:createGroupRequest>
   </soap:Body>
</soap:Envelope>
```

Response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
   <S:Body>
      <ns2:createGroupResponse
        xmlns:ns2="http://xmlns.telnic.org/ws/nsp/client/reader/types-1.0">
         <ns2:id>17</ns2:id>
      </ns2:createGroupResponse>
   </S:Body>
</S:Envelope>
```

2.3.8 `updateGroup` operation

Updates a group. The name as well as the associations with the readers and records can be updated. They are left unchanged if not specified in the request.

If associations to resource records are removed which thereafter are not linked to any group, they are removed from the system in order to avoid an accidental exposure of private data.

The `updateGroupRequest` and `updateGroupResponse` elements describe the request and response, respectively. Allowed elements in the request are:

- `<id>` specifying the group to be updated;
- `<name>` (optional) of the group, will be displayed to the user;
- `<readers>` (optional) specifying a new list of readers;
- `<records>` (optional) specifying a new list of resource record IDs.

Example:

Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/reader/types-1.0">
   <soap:Header/>
   <soap:Body>
      <typ:updateGroupRequest domainName="reggie.tel">
         <typ:id>17</typ:id>
         <typ:name>Close Friends</typ:name>
         <typ:readers>15</typ:readers>
         <typ:records>62 71</typ:records>
      </typ:updateGroupRequest>
   </soap:Body>
</soap:Envelope>
```

Response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
   <S:Body>
      <updateGroupResponse
        xmlns="http://xmlns.telnic.org/ws/nsp/client/reader/types-1.0"/>
   </S:Body>
</S:Envelope>
```

2.3.9 `deleteGroup` operation

Deletes a group. Note that all resource records that have exclusively been associated with this group are deleted also, in order to avoid an accidental exposure of private data. If the resource records are still associated with other groups, they are not deleted, however.

Readers are never deleted, even if they have been exclusively associated with the group.

The `deleteGroupRequest` and `deleteGroupResponse` elements describe the request and response, respectively. The only element present in the request is the `<id>` of the group that is to be deleted.

Example:

Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/reader/types-1.0">
   <soap:Header/>
   <soap:Body>
      <typ:deleteGroupRequest domainName="reggie.tel">
         <typ:id>17</typ:id>
      </typ:deleteGroupRequest>
   </soap:Body>
</soap:Envelope>
```

Response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
   <S:Body>
      <deleteGroupResponse
        xmlns="http://xmlns.telnic.org/ws/nsp/client/reader/types-1.0"/>
   </S:Body>
</S:Envelope>
```

2.3.10 `getGroup` operation

Returns the data (i.e. the name, readers, and records) of the group.

The `getGroupRequest` and `getGroupResponse` elements describe the request and response, respectively. The only element present in the request is the `<id>` of the group for which to retrieve data.

Example:

Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/reader/types-1.0">
   <soap:Header/>
   <soap:Body>
      <typ:getGroupRequest domainName="reggie.tel">
         <typ:id>17</typ:id>
      </typ:getGroupRequest>
   </soap:Body>
</soap:Envelope>
```

Response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
   <S:Body>
      <ns2:getGroupResponse
        xmlns:ns2="http://xmlns.telnic.org/ws/nsp/client/reader/types-1.0">
         <ns2:name>Close Friends</ns2:name>
         <ns2:readers>15</ns2:readers>
         <ns2:records>62 71</ns2:records>
      </ns2:getGroupResponse>
   </S:Body>
</S:Envelope>
```

2.3.11 `listGroups` operation

Returns the list of groups that exist for a given domain.

The `listGroupsRequest` and `listGroupsResponse` elements describe the request and response, respectively. There are no elements in this kind of request, but the `domainName` attribute has to be specified, giving the name of the domain whose groups to list. In the response, the name of each group is returned in a separate element, as well as its ID as an attribute.

Example:

Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/reader/types-1.0">
   <soap:Header/>
   <soap:Body>
      <typ:listGroupsRequest domainName="reggie.tel"/>
   </soap:Body>
</soap:Envelope>
```

Response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
   <S:Body>
      <ns2:listGroupsResponse
        xmlns:ns2="http://xmlns.telnic.org/ws/nsp/client/reader/types-1.0">
          <ns2:group id="17">Close Friends</ns2:group>
          <ns2:group id="12">Friends</ns2:group>
          <ns2:group id="5">Family</ns2:group>
      </ns2:listGroupsResponse>
   </S:Body>
</S:Envelope>
```

2.4 Profile Editing and Switching Interface

This interface provides all the functionality required by clients to create, edit, delete, inquire and switch profiles. Please refer to section **??** for a general description of profiles and the involved concepts and procedures.

2.4.1 API operations summary

A summary of the API operations is shown in table 4. The corresponding schema definition can be found in the file `Profile-1.0.xsd`.

operation	description
`createProfile`	Create a new profile for a domain.
`deleteProfile`	Delete an existing profile from a domain.
`updateProfile`	Update an existing profile with new data.
`getProfile`	Get all data of a specific profile.
`listProfiles`	List all profiles stored for a domain.
`listProfilesExt`	List all profiles stored for a domain (extended version).
`switchToProfile`	Switch to the specified profile.
`getActiveProfile`	Determine the currently active profile.

Table 4: Summary of the profile editing and switching API.

2.4.2 `createProfile` operation

This operation creates a new profile. The system assigns an ID to this profile and returns it in the response.

The `createProfileRequest` and `createProfileResponse` elements describe the request and response, respectively. Allowed elements in the request are:

- `<name>` specifies the (unique) name of the profile;
- `<records>` specifies a list of resource record IDs that should be part of the profile.

Additionally, the attribute `default` can be used to make the new profile the default profile.

Example:

The following request creates a “Work” profile. It is associated with the resource records 7, 12, and 23.

Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/profile/types-1.0">
   <soap:Header/>
   <soap:Body>
      <typ:createProfileRequest domainName="reggie.tel">
         <typ:profile default="false">
            <typ:name>Work</typ:name>
            <typ:records>7 12 23</typ:records>
         </typ:profile>
      </typ:createProfileRequest>
   </soap:Body>
</soap:Envelope>
```

Response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
   <S:Body>
      <ns3:createProfileResponse
        xmlns:ns3="http://xmlns.telnic.org/ws/nsp/client/profile/types-1.0">
         <ns3:id>24</ns3:id>
      </ns3:createProfileResponse>
   </S:Body>
</S:Envelope>
```

2.4.3 `deleteProfile` operation

This operation deletes an existing profile. The operation receives a `DeleteProfileRequestMessage` and returns a `DeleteProfileResponseMessage`, expressed by the `deleteProfileRequest` and `deleteProfileResponse` elements from the XSD, respectively.

Note that throughout this API, existing profiles are addressed by their *IDs*, not by their names.

Example:

This request deletes the profile with the ID 24.

Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/profile/types-1.0">
   <soap:Header/>
   <soap:Body>
      <typ:deleteProfileRequest domainName="reggie.tel">
         <typ:profile id="24"/>
      </typ:deleteProfileRequest>
   </soap:Body>
</soap:Envelope>
```

Response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
   <S:Body>
      <deleteProfileResponse
        xmlns="http://xmlns.telnic.org/ws/nsp/client/profile/types-1.0"/>
   </S:Body>
</S:Envelope>
```

Note that you are not allowed to delete the default profile.

2.4.4 `updateProfile` operation

This operation updates an existing profile.

The `updateProfileRequest` and `updateProfileResponse` elements describe the request and response, respectively. The profile to be updated is specified by the attribute `id`. Allowed elements in the request are:

- `<name>` (optional) specifies the (unique) new name of the profile;
- `<records>` (optional) specifies a new list of resource record IDs that should replace the current resource records of this profile; if this element is omitted, the resource records of the profile will not be changed.

Additionally, the attribute `default` can be used to make the new profile the default profile.

Example:

This request updates the profile with the ID 24 to be the default profile.

Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/profile/types-1.0">
   <soap:Header/>
   <soap:Body>
      <typ:updateProfileRequest domainName="reggie.tel">
         <typ:profile id="24" default="true">
            <typ:name>Holidays</typ:name>
            <typ:records>71 73</typ:records>
         </typ:profile>
      </typ:updateProfileRequest>
   </soap:Body>
</soap:Envelope>
```

Response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
   <S:Body>
      <updateProfileResponse
        xmlns="http://xmlns.telnic.org/ws/nsp/client/profile/types-1.0"/>
   </S:Body>
</S:Envelope>
```

Note that there always has to be a default profile. Therefore, an attribute `default="false"` is silently ignored if applied to the default profile, i.e. the default profile will not be changed.

2.4.5 `getProfile` operation

This operation retrieves the data of an existing profile.

The `getProfileRequest` and `getProfileResponse` elements describe the request and response, respectively. The only element present in the request is the `<profile>` for which to retrieve data, specified via the `id` attribute.

Example:

This request retrieves the data of the profile with the ID 24.

Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/profile/types-1.0">
   <soap:Header/>
   <soap:Body>
      <typ:getProfileRequest domainName="reggie.tel">
         <typ:profile id="24"/>
      </typ:getProfileRequest>
   </soap:Body>
</soap:Envelope>
```

Response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
   <S:Body>
      <ns3:getProfileResponse
        xmlns:ns3="http://xmlns.telnic.org/ws/nsp/client/profile/types-1.0">
         <ns3:profile default="true">
            <ns3:name>Holidays</ns3:name>
            <ns3:records>73 71</ns3:records>
         </ns3:profile>
      </ns3:getProfileResponse>
   </S:Body>
</S:Envelope>
```

2.4.6 `listProfiles` operation

This operation retrieves the IDs of all existing profiles of a domain. See `listProfilesExt` below (section 2.4.7) for an extended version returning more information.

The `listProfilesRequest` and `listProfilesResponse` elements describe the request and response, respectively. There are no elements in this kind of request, but the `domainName` attribute has to be specified, giving the name of the domain whose profiles to list.

Example:

Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/profile/types-1.0">
   <soap:Header/>
   <soap:Body>
      <typ:listProfilesRequest domainName="reggie.tel"/>
   </soap:Body>
</soap:Envelope>
```

Response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
   <S:Body>
      <ns3:listProfilesResponse
        xmlns:ns3="http://xmlns.telnic.org/ws/nsp/client/profile/types-1.0">
         <ns3:profiles ids="6 21 24"/>
      </ns3:listProfilesResponse>
   </S:Body>
</S:Envelope>
```

2.4.7 `listProfilesExt` operation

This operation is an extended version of `listProfiles`. It retrieves not only the IDs of all existing profiles of a domain, but also their names, together with information about the active and default profiles.

The `listProfilesExtRequest` and `listProfilesExtResponse` elements describe the request and response, respectively. There are no elements in this kind of request, but the `domainName` attribute has to be specified, giving the name of the domain whose profiles to list.

Example:

Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/profile/types-1.0">
   <soap:Header/>
   <soap:Body>
      <typ:listProfilesExtRequest domainName="reggie.tel"/>
   </soap:Body>
</soap:Envelope>
```

Response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
   <S:Body>
      <ns3:listProfilesExtResponse
        xmlns:ns3="http://xmlns.telnic.org/ws/nsp/client/profile/types-1.0">
         <ns3:profiles active="21" default="6">
            <ns3:profile id="6" name="default"/>
            <ns3:profile id="21" name="Home"/>
            <ns3:profile id="24" name="Holidays"/>
         </ns3:profiles>
      </ns3:listProfilesExtResponse>
   </S:Body>
</S:Envelope>
```

2.4.8 `switchToProfile` operation

This operation switches to a given profile, i.e. makes it the active one.

The `switchToProfileRequest` and `switchToProfileResponse` elements describe the request and response, respectively. The only element present in the request is the `<profile>` to which to switch, specified via the `id` attribute.

Example:

Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/profile/types-1.0">
   <soap:Header/>
   <soap:Body>
      <typ:switchToProfileRequest domainName="reggie.tel">
         <typ:profile id="24"/>
      </typ:switchToProfileRequest>
   </soap:Body>
</soap:Envelope>
```

Response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
   <S:Body>
      <ns3:switchToProfileResponse
        xmlns:ns3="http://xmlns.telnic.org/ws/nsp/client/profile/types-1.0"/>
   </S:Body>
</S:Envelope>
```

2.4.9 `getActiveProfile` operation

This operation determines the currently active profile of a domain.

The `getActiveProfileRequest` and `getActiveProfileResponse` elements describe the request and response, respectively. There are no elements in this kind of request, but the `domainName` attribute has to be specified, giving the name of the domain whose profiles to list.

Example:

Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/profile/types-1.0">
   <soap:Header/>
   <soap:Body>
      <typ:getActiveProfileRequest domainName="reggie.tel"/>
   </soap:Body>
</soap:Envelope>
```

Response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
   <S:Body>
      <ns3:getActiveProfileResponse
        xmlns:ns3="http://xmlns.telnic.org/ws/nsp/client/profile/types-1.0">
         <ns3:profile id="24"/>
      </ns3:getActiveProfileResponse>
   </S:Body>
</S:Envelope>
```

2.5 Search Data Management

This interface provides all the functionality required by clients for search data management, i.e. provisioning of keyword/value pairs for inclusion in the Tel-List search index. Refer to section **??** for a general description of the search data principles.

The API presents an abstraction from the actual representation of the search data in the `.tel` zones; the generation of appropriate resource records, as well as the notification of the SO's zone crawler, is automatically done behind the scenes. The interface is exposed via a publicly available web service, using SOAP over HTTP as its protocol.

2.5.1 API operations summary

A summary of the API operations is shown in table 5. The corresponding schema definition can be found in the file `SearchData-1.0.xsd`.

operation	**description**
`setSearchData`	Set/update search data of a domain, replacing all previous data (also allows the deletion of all search data by providing an empty new set of data).
`getSearchData`	Get the current search data of a domain.

Table 5: Summary of the search data management API.

2.5.2 `setSearchData` operation

This operation sets new search data for a domain. It replaces all search data that may have been present prior to the operation.

The `setSearchDataRequest` and `setSearchDataRequest` elements describe the request and response, respectively. The only element present in the request is `<searchData>`, which contains a sequence of `<keyword>` subelements. These consist of the required attribute `field` (a non-empty string), defining the keyword, and the required attribute `value`, defining the value to be associated with this particular keyword.

These immediate child elements of `<searchData>` define the domain's *primary* keywords. Every primary keyword may optionally have subordinate *secondary* keywords associated with it, which may be specified as `<keyword>` child elements. These child elements carry the same required `field` and `value` attributes, but must not contain any further keywords as child elements themselves.

Example:

This request sets some search data for the domain `reggie.tel`. The "name label" grouping keyword `nl` is used to set various name-related keywords; `bpa` and `pa` are keywords grouping some address data (`tc`, `pc`) specified as secondary keywords.

Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/searchdata/types-1.0">
   <soap:Header/>
   <soap:Body>
      <typ:setSearchDataRequest domainName="reggie.tel">
         <typ:searchData>
            <typ:keyword field="nl" value="">
               <typ:keyword field="s" value="Mr"/>
               <typ:keyword field="fn" value="John"/>
               <typ:keyword field="ln" value="Smith"/>
               <typ:keyword field="nn" value="Johnny"/>
               <typ:keyword field="ms" value="bachelorette"/>
            </typ:keyword>
            <typ:keyword field="bpa" value="My shop">
               <typ:keyword field="tc" value="Bristol"/>
               <typ:keyword field="pc" value="12345"/>
            </typ:keyword>
            <typ:keyword field="pa" value="My home">
               <typ:keyword field="tc" value="London"/>
               <typ:keyword field="pc" value="67890"/>
            </typ:keyword>
         </typ:searchData>
      </typ:setSearchDataRequest>
   </soap:Body>
</soap:Envelope>
```

Response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
   <S:Body>
      <setSearchDataResponse
        xmlns="http://xmlns.telnic.org/ws/nsp/client/searchdata/types-1.0"/>
   </S:Body>
</S:Envelope>
```

2.5.3 `getSearchData` operation

This operation retrieves the current set of search data stored for a given domain.

The `getSearchDataRequest` and `getSearchDataRequest` elements describe the request and response, respectively. There are no elements in this kind of request, but the `domainName` attribute has to be specified, giving the name of the domain whose search data to retrieve.

Example:

Retrieve the search data for the domain `reggie.tel`.

Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/searchdata/types-1.0">
   <soap:Header/>
   <soap:Body>
      <typ:getSearchDataRequest domainName="reggie.tel"/>
   </soap:Body>
</soap:Envelope>
```

Response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
   <S:Body>
      <ns5:getSearchDataResponse
        xmlns:ns5="http://xmlns.telnic.org/ws/nsp/client/searchdata/types-1.0">
         <ns5:searchData>
            <ns5:keyword field="fn" value="John"/>
            <ns5:keyword field="ln" value="Smith"/>
            <ns5:keyword field="bpa" value="My shop">
               <ns5:keyword field="tc" value="Bristol"/>
               <ns5:keyword field="pc" value="12345"/>
            </ns5:keyword>
            <ns5:keyword field="pa" value="My home">
               <ns5:keyword field="tc" value="London"/>
               <ns5:keyword field="pc" value="67890"/>
            </ns5:keyword>
         </ns5:searchData>
      </ns5:getSearchDataResponse>
   </S:Body>
</S:Envelope>
```

2.6 Data Exchange Interface

This interface provides all the functionality required by clients to export and import the whole data that is associated with one or more domains: records, profiles, readers, groups, search data, and zones. The most important applications of this feature are to create backups and to easily migrate one's data to a different TelHosting Provider.

2.6.1 API operations summary

A summary of the API operations is shown in table 6. The corresponding schema definition can be found in the file `Exchange-1.0.xsd`.

operation	admin	useradmin	user	description
`exportData`	yes	yes	permissions	Exports the data of domains and zones.
`importData`	yes	yes	permissions	Imports the data of domains and zones.
`restoreDomain`	yes	yes	permissions	Restores the data of domain hierarchies.

Table 6: Summary of the export/import API; *permissions* means the user's permissions settings determine whether the operation is allowed or not.

2.6.2 `exportData` operation

This operation allows a user to export the data associated with one or more domains and zones in one go.

Note that the `<privateSalt>` – specifying a base-64-encoded string used to create the reference for friending messages – has a fixed length (by default 64, not configurable). For brevity, the examples contain values whose length is reduced by half.

The `exportDataRequest` and `exportDataResponse` elements describe the request and response, respectively.

Allowed elements in the request are:

- `<domains>` (optional) specifying the list of domains (as a sequence of `<domainName>` elements) whose data should be exported. If missing, all domains are exported. If present, only the specified domains are exported; an empty list exports no domains at all. Each `<domainName>` element may have an optional `recursive` attribute. If this is set to true, all sub-domains of the given domain are exported without the need to listing them.

- `<zones>` (optional) specifying the list of zones (as a sequence of `<domainName>` elements) whose data should be exported. If missing, all zones are exported. If present, only the specified domains are exported; an empty list exports no zones at all.

Specifying any domains or zones that do not exist within the user's inventory will make the export fail with an error message.

Example:

Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/exchange/types-1.0">
  <soap:Header/>
  <soap:Body>
    <typ:exportDataRequest>
      <typ:domains>
        <typ:domainName>example.tel</typ:domainName>
      </typ:domains>
      <typ:zones>
        <typ:domainName>example.tel</typ:domainName>
      </typ:zones>
    </typ:exportDataRequest>
  </soap:Body>
</soap:Envelope>
```

Response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
  <S:Body>
    <ns6:exportDataResponse
      xmlns:ns6="http://xmlns.telnic.org/ws/nsp/client/exchange/types-1.0">
    <ns6:data><![CDATA[<?xml version="1.0" encoding="UTF-8"?>
      <exports xmlns="http://xmlns.telnic.org/nsp/exchange-container-1.0">
      <export xmlns="http://xmlns.telnic.org/nsp/exchange-1.0">
      <domains>
        <domain domainName="example.tel">
          <profiles>
            <profile default="true"><name>work</name>
              <records>record17 record16</records>
            </profile>
          </profiles>
          <readers>
            <reader id="reader11"><name>reggie</name>
              <label>42bc4550afb3967205dd25bb3662eff12a72387c</label>
              <keyLocation>d1ql.node1.keys.tel</keyLocation>
              <reference>ae4fup89du02398ruisd9pitrjq3rj4umas3ioufp</reference>
              <readerDomainName>1234.reader.keys.tel</readerDomainName>
            </reader>
          </readers>
          <groups>
            <group><name>friends</name>
              <readers>reader11</readers><records>record16</records><allFriendsFlag>true</allFriendsFlag>
            </group>
            <group><name>family</name>
              <readers></readers><records>record17 record16</records><allFriendsFlag>false</allFriendsFlag>
            </group>
          </groups>
          <records>
            <naptr class="1" id="record17" owner="@">
              <order>100</order><preference>100</preference><flags>u</flags>
              <services>E2U+voice:tel+x-work</services>
              <regexp>!^.*$!tel:+42.4!</regexp>
            </naptr>
            <naptr class="1" id="record16" owner="@">
              <order>100</order><preference>90</preference><flags>u</flags>
              <services>E2U+voice:tel+x-home</services>
              <regexp>!^.*$!tel:+44.1!</regexp>
            </naptr>
          </records>
          <searchData/>
          <privateSalt>D1zzXOl9nAX0v1MF5jgXN3zjy59OU42ftaLjyGCVyG9zbk5hyjugqjku</privateSalt>
        </domain>
      </domains>
      <zones>
        <zone zoneName="example.tel">
          <nameservers>
            <host>ns1.example.tel</host>
            <host>ns2.example.tel</host>
          </nameservers>
        </zone>
      </zones>
      </export>
      </exports>]]>
    </ns6:data>
    </ns6:exportDataResponse>
  </S:Body>
</S:Envelope>
```

2.6.3 `importData` operation

This operation allows a user to import the data associated with one or more domains and zones in one go. The `importDataRequest` and `importDataResponse` elements describe the request and response, respectively.

Allowed elements in the request are:

- `<domains>` (optional) specifying the list of domains whose data should be imported. If missing, all domains found in the data are imported. If present, only the specified domains are imported (if they are included in the data); an empty list imports no domains at all.
- `<zones>` (optional) specifying the list of zones whose data should be imported. If missing, all zones found in the data are imported. If present, only the specified zones are imported (if they are included in the data); an empty list imports no zones at all.
- `<data>` defining the data that is to be imported (see section**??** for details regarding the structure of this data).

For both domains and zones, a specified name that does not exist in the data will be silently ignored.

The Boolean attribute `overwrite` is required and specifies whether existing data will be overwritten or not. More precisely, if set to `true`, after the import a domain for which data is imported will have exactly the data specified in the import data. Thus, other data present in this domain before the import is removed. If `overwrite` is set to `false`, no data is changed for domains that already exist. If an apex domain is going to be imported and the same apex domain already exists, the import is only possible if the existing domain has the exact same structure (i.e., sub-domain hierarchy) as the domain to be imported. If this is not the case the command `restoreDomain` should be used instead.

Note that the `<privateSalt>` – specifying a base-64-encoded string used to create the reference for friending messages – has a fixed length (by default 64, not configurable). For brevity, the examples contain values whose length is reduced by half.

In the response, the domains and zones that were actually imported are reported.

Example:

Request:

```
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/exchange/types-1.0">
  <soap:Header/>
  <soap:Body>
    <typ:importDataRequest>
      <typ:domains>
        <typ:domainName>example.tel</typ:domainName>
      </typ:domains>
      <typ:zones/>
      <typ:data><![CDATA[<?xml version="1.0" encoding="UTF-8"?>
        <exports xmlns="http://xmlns.telnic.org/nsp/exchange-container-1.0">
        <export xmlns="http://xmlns.telnic.org/nsp/exchange-1.0">
        <domains>
          <domain domainName="example.tel">
            <profiles>
              <profile default="true"><name>always</name>
                <records>record42 record43</records>
              </profile>
            </profiles>
            <readers/>
            <groups/>
            <records>
              <naptr class="1" id="record42" owner="@">
                <order>100</order><preference>100</preference><flags>u</flags>
                <services>E2U+voice:tel+x-work</services>
                <regexp>!^.*$!tel:+42.4!</regexp>
              </naptr>
              <naptr class="1" id="record43" owner="@">
                <order>100</order><preference>100</preference><flags>u</flags>
                <services>E2U+voice:tel+x-mobile</services>
                <regexp>!^.*$!tel:+42.543342!</regexp>
              </naptr>
            </records>
            <searchData/>
            <privateSalt>D1zzXOl9nAX0v1MF5jgXN3zjy59OU42ftaLjyGCVyG9zbk5hyjugqjku</privateSalt>
          </domain>
        </domains>
        <zones>
          <zone zoneName="example.tel">
            <nameservers>
              <host>ns1.example.tel</host>
              <host>ns2.example.tel</host>
            </nameservers>
          </zone>
        </zones>
        </export>
        </exports>]]>
      </typ:data>
    </typ:importDataRequest>
  </soap:Body>
</soap:Envelope>
```

Response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
   <S:Body>
      <ns3:importDataResponse
        xmlns:ns3="http://xmlns.telnic.org/ws/nsp/client/exchange/types-1.0">
         <ns3:domains>
            <ns3:domainName>example.tel</ns3:domainName>
         </ns3:domains>
         <ns3:zones/>
      </ns3:importDataResponse>
   </S:Body>
</S:Envelope>
```

2.6.4 `restoreDomain` operation

This operation allows a user to import the data associated with one or more domains and replaces any existing domains of the same name. The `restoreDomainRequest` and `restoreDomainResponse` elements describe the request and response, respectively.

Allowed elements in the request are:

- `<domains>` specifying the list of apex domains whose data should be imported. All sub-domains present in the data will also be imported.
- `<data>` defining the data that is to be imported (see section**??** for details regarding the structure of this data).

If a specified domain name does not exist in the data an error will be reported.

If the apex domain already exists within in the target account then it will be deleted (along with the entire sub-domain hierarchy).

Note that the `<privateSalt>` – specifying a base-64-encoded string used to create the reference for friending messages – has a fixed length (by default 64, not configurable). For brevity, the examples contain values whose length is reduced by half.

In the response, the domains and zones that were actually imported are reported.

Example:

Request:

```
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:typ="http://xmlns.telnic.org/ws/nsp/client/exchange/types-1.0">
  <soap:Header/>
  <soap:Body>
    <typ:restoreDomainRequest>
      <typ:domains>
        <typ:domainName>example.tel</typ:domainName>
      </typ:domains>
      <typ:data><![CDATA[<?xml version="1.0" encoding="UTF-8"?>
        <exports xmlns="http://xmlns.telnic.org/nsp/exchange-container-1.0">
        <export xmlns="http://xmlns.telnic.org/nsp/exchange-1.0">
        <domains>
          <domain domainName="example.tel">
            <profiles>
              <profile default="true"><name>always</name>
                <records>record42 record43</records>
              </profile>
            </profiles>
            <readers/>
            <groups/>
            <records>
              <naptr class="1" id="record42" owner="@">
                <order>100</order><preference>100</preference><flags>u</flags>
                <services>E2U+voice:tel+x-work</services>
                <regexp>!^.*$!tel:+42.4!</regexp>
              </naptr>
              <naptr class="1" id="record43" owner="@">
                <order>100</order><preference>100</preference><flags>u</flags>
                <services>E2U+voice:tel+x-mobile</services>
                <regexp>!^.*$!tel:+42.543342!</regexp>
              </naptr>
            </records>
            <searchData/>
            <privateSalt>D1zzXOl9nAX0v1MF5jgXN3zjy59OU42ftaLjyGCVyG9zbk5hyjugqjku</privateSalt>
          </domain>
        </domains>
        <zones>
          <zone zoneName="example.tel">
            <nameservers>
              <host>ns1.example.tel</host>
              <host>ns2.example.tel</host>
            </nameservers>
          </zone>
        </zones>
        </export>
        </exports>]]>
      </typ:data>
    </typ:restoreDomainRequest>
  </soap:Body>
</soap:Envelope>
```

Response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
   <S:Body>
      <ns3:restoreDomainResponse
        xmlns:ns3="http://xmlns.telnic.org/ws/nsp/client/exchange/types-1.0">
         <ns3:domains>
            <ns3:domainName>example.tel</ns3:domainName>
         </ns3:domains>
      </ns3:restoreDomainResponse>
   </S:Body>
</S:Envelope>
```

List of Tables