

TelFriends SOAP API Reference

April 2010



DISCLAIMER

Copyright (c) 2008-2010, Telnic Ltd.

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the Telnic Ltd nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS DOCUMENTATION IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Table of Contents

About this Document	5
Revision History	5
Overview	6
Overview.....	6
Key Concepts	6
Message Format.....	8
List of Namespaces	9
List of Schema and WSDL Files	9
Related Links	9
Key Store.....	11
getKey11	
Message Store	13
Message Store	13
createMessage	15
listMessages	16
getMessage	18
deleteMessage.....	19
getBlacklist	19
addToBlacklist	20
removeFromBlacklist	20
subscribe.....	21
unsubscribe	21
Publisher Store	23
Publisher Store	23
storePublisher	23
listPublishers	24
deletePublisher.....	25
Tel User Management.....	26
Tel User Management	26

getUserInfo	26
listDomainNames	27
changePassword	28
changeChallenge	29
Client Initialization	30
Client Initialization	30
getChallengeQuestion	30
getAPICredentials	31
getPKCS12	33

1. ABOUT THIS DOCUMENT

Sponsoring Organization (SO) systems provide a number of services for enabling privacy in .tel domains, which get represented via a web front-end [TelFriends](#). This document describes the SOAP API of the SO services, a.k.a TelFriends interface.

The reference is targeted at developers interested in creating new [.tel client applications](#) and enhancing existing ones. The document assumes that you are familiar with application programming standard SOAP (Simple Object Access Protocol) and that you understand the key concepts of the .tel innovative technology. See [Related Links](#) for references to relevant materials.

With this reference, you will learn about:

- The objects used in the interface
- The functions available to manipulate all those objects: encryption and decryption keys, messages, users, etc.
- Specific usage requirements for each function
- The "cold boot" initialization procedure and operations that client applications can use to get authenticated with the SO system

1.1. Revision History

Version Description

- | | |
|-----|--|
| 1.0 | SO SOAP API reference; initial version. |
| 1.1 | Message Store API updated with getApexDomain operation |
| 1.2 | TelPages API moved to a separate document |

2. OVERVIEW

2.1. Overview

The .tel top-level domain (TLD) uses the Domain Name System (DNS) as a data store for contact information in text records, location records, and Naming Authority Pointer records (NAPTR). To enable .tel domain owners to manage access to their sensitive information published in NAPTR records, the Sponsoring Organization implements a privacy model. With this model, domain owners can encrypt private information before publication in the .tel DNS and grant access to this protected data to select users, which may or may not own a .tel domain.

The privacy model is implemented in the TelFriends system, which supports data encryption, friending relation management, and other operations grouped into the following APIs:

- [Key Store](#) that holds public and private keys of SO members that are used to encrypt and decrypt protected content
- [Message Store](#) that contains friending request and response messages that SO members send to establish friending relations and share protected data, as well as manages the black list blocking users from sending friend requests to this account
- [Publisher Store](#) that holds lists of users (publishers) that have granted access to their .tel protected content to other users (readers)
- [Tel User Management Module](#) which allows storage and retrieval of information on SO members (users)

All APIs work by receiving request messages from the clients and sending response messages back to them unless the request message requires no response. All messages are expressed in XML, and their respective syntax is specified by an XML schema; see [List of Schema Files](#). The messages must follow the specific format, see [Message Format](#).

In addition to the SO services, client applications need to interact with TelHosting Software, which exposes a number of SOAP APIs as well, sub-divided into [Client APIs](#) and [Admin APIs](#). The TelHosting Software interfaces are not covered in this document.

2.2. Key Concepts

This reference is based on a number of basic concepts, around which the whole API is organized.

Table 1: Key Concepts

Object Name	Description
.tel client application	An application that manipulates contact data published in .tel domain records. The application uses the SOAP APIs to interact with the TelHosting Provider, the <i>Sponsoring Organization</i> , and can query .tel DNS directly to perform its functions.

	See currently available applications at Telnic Developer Area .
.tel Sponsoring Organization	<p>Telnic Ltd. is the sponsoring organization (SO) for the .tel domain. SO provides a number of services, including infrastructure for TelHosting Providers, .tel member database, and storage of keys for record encryption.</p> <p>In relation to .tel client applications, the SO provides a SOAP API to support the "friending" mechanism for private data exchange. That API covers initialization in the SO system, key/credential management, and processing of friending messages.</p>
User	<p>Can mean the user of a .tel client application or the user of the SO system, as in Tel User Management.</p> <p>Do not mix with members of the SO system with "full" or "guest" user profiles. Do not mix with members of TelHosting Provider systems, who own .tel domains registered by that provider and have user accounts in the Provider's systems.</p>
Message	<p>A friending request or response message stored in the message store for the SO member. These messages are used to establish connections between different .tel domains to exchange private data. See Friending Messages for a description of the message format and purpose. The concept of "black lists" is also a part of the interface, cf. getBlacklist.</p>
Publisher	<p>A .tel domain owner with a user account in the SO system, who has agreed to publish private data for a Reader. The SO system keeps track of all connections between SO members, so that each user has a Publisher Store with links to all .tel domains, where private data is available for this member.</p>
Reader	<p>A user who has been granted access to a Publisher .tel domain owner. Reader may or may not own .tel domains and in turn act as Publishers for other Readers. If a Reader has no .tel domain associated with them in the SO system, that Reader has a "guest" account in the SO member database.</p>
Key pair	<p>A pair of private and public keys for encryption and decryption of private data. Each Reader has a key pair, and a Publisher uses the Reader's public key to encrypt data, whereas the Reader uses the private key to decrypt that data. The Key store is used to collect public keys, so that Publishers can retrieve those keys to encrypt their data for each Reader individually. Predefined and uploaded keys are supported.</p>
Pseudo domain name	<p>An unique identifier assigned to each member of the SO database and stored in the Reader facet.</p>

2.3. Message Format

Request

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:typ="_xmlns.telnic.org_namespace_here_">
<soap:Header/>
<soap:Body>
...
</soap:Body>
</soap:Envelope>
```

Instead of `_xmlns.telnic.org_namespace_here_`, insert the corresponding namespace from the [List of Namespaces](#).

For example:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
  <S:Body>
    <getMessageResponse
xmlns="http://xmlns.telnic.org/ws/so/member/messagestore/types-1.0"
xmlns:ns2="http://xmlns.telnic.org/ws/so/member/usermanagement/types-1.0"
xmlns:ns3="http://xmlns.telnic.org/ws/so/common-1.0"
xmlns:ns4="http://xmlns.telnic.org/ws/so/member/publisherstore/types-1.0"
xmlns:ns5="http://xmlns.telnic.org/ws/so/member/keystore/types-1.0">
      <id>3316</id>
      <to>testdomain.tel</to>
      <from>565.readers.keys.nic.tel</from>
      <received>2009-04-15T17:45:45.534+02:00</received>
      <contentType>text/plain</contentType>
      <messageType>friendRequest</messageType>
      <text>date: 20090415:154534
          key-domain: d565ql.node65.keys.tel
          reference: 09dc3892ae0895e048b0651cfabc471fc37bd146
          cover-note: and this is to someone else...</text>
    </getMessageResponse>
  </S:Body>
</S:Envelope>
```

Response

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
  <S:Body>
  ...
</S:Body>
</S:Envelope>
```

For example:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:typ="http://xmlns.telnic.org/ws/so/member/messagestore/types-1.0">
  <soap:Header/>
  <soap:Body>
    <typ:createMessageRequest>
      <typ:from>622.readers.keys.nic.tel</typ:from>
      <typ:to>622.readers.keys.nic.tel</typ:to>
      <typ:contentType>text/plain</typ:contentType>
      <typ:messageType>friendResponse</typ:messageType>
      <typ:text>date: 20090317:154913
          sub-domain-id:
40a497ebbb09e92b34a87f19a4742af3047fcb50.u19tangeal.tel
          reference: d7a5249c53234c969495635e6e47e194878d764f
          cover-note: this is unsolicited so should be discarded</typ:text>
    </typ:createMessageRequest>
  </soap:Body>
</soap:Envelope>
```

2.4. List of Namespaces

Name	API Group
http://xmlns.telnic.org/ws/so/member/keystore/types-1.0	Key Store
http://xmlns.telnic.org/ws/so/member/messagestore/types-1.0	Message Store
http://xmlns.telnic.org/ws/so/member/publisherstore/types-1.0	Publisher Store
http://xmlns.telnic.org/ws/so/member/usermanagement/types-1.0	Tel User
http://xmlns.telnic.org/ws/so/init/types-1.0	Client Initialization

2.5. List of Schema and WSDL Files

The various schema files can be found inside the distribution of TelHosting Software or with this reference. One schema matches one API function group.

WSDL file	Schema	API Group
Member-Service-1.0.wsdl	KeyStore-1.0.xsd	Key Store
	MessageStore-1.0.xsd	Message Store
	PublisherStore-1.0.xsd	Publisher Store
	UserManagement-1.0.xsd	Tel User
Init-Service-1.0.wsdl	Init-1.0.xsd	Client Initialization

2.6. Related Links

- Telnic Developer Area, <http://dev.telnic.org/index.html>
- Telnic Developers FAQ, <http://dev.telnic.org/pages/faq.html>
- Telnic .tel Community Area, <http://telnic.org/community-landing.html>
- TelFriends End-User Guide, http://telnic.org/downloads/telfriends_guide.pdf
- Telnic Developers Manual, <http://dev.telnic.org/docs/devguide.pdf>
- TelHosting Client SOAP API, <http://dev.telnic.org/api/client-soap/index.html>
- TelHosting Admin SOAP API, <http://dev.telnic.org/api/admin-soap/index.html>

- SOAP, Simple Object Access Protocol, <http://www.w3.org/TR/soap/>
- Telnic Whitepaper, Privacy in tel, <http://dev.telnic.org/docs/privacy.pdf>
- RFC2535, <http://tools.ietf.org/html/rfc2535>
- RFC3066, <http://tools.ietf.org/html/rfc3066>
- ISO 8601 summary, http://www.iso.org/iso/support/faqs/faqs_widely_used_standards/widely_used_standards_other/date_and_time_format.htm, XSD definition <http://books.xmlschemata.org/relaxng/ch19-77049.html>
- IANA defining DNS Parameters, <http://www.iana.org/assignments/dns-parameters>
- RKEY DNS Resource Record, <http://tools.ietf.org/html/draft-reid-dnsextr-rkey-00>

3. KEY STORE

3.1. getKey

The operation returns information about a key and/or the key data itself. The caller can choose what shall be returned, depending on the need (e.g. download of the key or verification of the synchronization state). If no key is stored in the system, the response is empty.

getKeyRequest

By default, both attributes have the value "1" to request both information and data.

Attribute Name	Description
info	Boolean value requesting information about the key.
data	Boolean value requesting the key data.

getKeyResponse

The response includes the `key` element, which includes the requested information or data about the key, specifically:

Element Name	Description
info	The information about the key in the subelements: <ul style="list-style-type: none">• <code>publicKeyHash</code> - the hash of the public key• <code>keyLocation</code> - the .tel domain where the key is located• <code>lastChange</code> - the date and time of the last key change
data	The key data itself, consisting of the following subelements: <ul style="list-style-type: none">• <code>publicKey</code> - the encrypted public key• <code>privateKey</code> - the encrypted private key

Example

Request

```
<getKeyRequest
  xmlns="http://xmlns.telnic.org/ws/so/member/keystore/types-1.0"
  info="1" data="1">
</getKeyRequest>
```

Response

```
<getKeyResponse
  xmlns="http://xmlns.telnic.org/ws/so/member/keystore/types-1.0">
  <key>
```

```
<info xmlns="http://xmlns.telnic.org/ws/so/common-1.0">
  <publicKeyHash alg="SHA-1">
    UjBsR09EbGhjZ0dTQUxNQUFBUNBRU1tQ1p0dU1GUXhEUzhi
  </publicKeyHash>
  <lastChange>2001-12-17T09:30:47-05:00</lastChange>
  <keyLocation>d937djql.node123.keys.tel</keyLocation>
</info>
<data>R0lGODlhcgGSALMAAAQCAEMmCZtuMFQxDS8b</data>
</key>
</getKeyResponse>
```

4. MESSAGE STORE

4.1. Message Store

The private data publishing infrastructure requires the "friending" process in order to initiate the exchange of private data. To conduct this process, the clients of the publisher and the reader need to communicate. Because it's unrealistic to expect both clients to be online at the same time and because some client programs may not be addressable (i.e. use dynamically assigned IP addresses and DNS names), a store-and-forward mechanism has been implemented by the SO for communication purposes.

The SO services include a *message store*, which enables the clients of the publisher and the reader to establish a friending connection. The message store enables sending friend requests, receiving confirmations (friend responses), and processes both message types. When a friend response is received, the messaging system checks whether the response is solicited, and does the following:

- **Solicited response with a valid reference:** the sender is added to users publisher list and message is stored to be presented to the user. When the user acknowledges receipt of the message, it is deleted.
- **Solicited response with an invalid reference:** the message is deleted without presentation to the user.
- **Unsolicited response** (commonly called invitation, reference = 0): the message is stored to be presented to the user. The user can accept and add the sender to the friend list, or reject and not do so. After the receiver has acted on the message, it is deleted from the store. Note that the current web interfaces do not allow sending unsolicited responses. This functionality is not fully tested.

The message store has been designed to put as few constraints as possible to the messages exchanged, in order to be flexible enough for other uses. This includes the transmission of binary data, which is, for example, required to transmit encrypted messages. This section defines the format and restrictions applied to friend messages.

Note that each domain has a separate mailbox addressed by the domain name. For forward compatibility with idN, both cleartext and Punycode names are accepted and treated as being equivalent. In addition, a separate message box exists for the "reader" facet of the user. To gain a homogeneous addressing scheme, a special domain name is associated with each user. This domain name does not materialize in the DNS, however. The name can be queried via the [user management API](#).

Message Format

For each message, the following metadata is stored:

- the user who created the message
- the date the message was stored
- the message size

- the message format (MIME)
- the message type (`friendRequest` or `friendResponse`), if supplied by the submitting client
- the transport format
- a unique id of the message

The message store reproduces the message bodies in the same format, i.e. no conversion between the formats is performed.

Because the message store supports binary messages, it also supports for different data transport mechanisms:

- base64 binary data. It is guaranteed that the binary data is transmitted unchanged.
- text data for plain text messages and non-embedded XML messages. Due to the embedding within XML, the text is subject to character set conversion, normalization of the line terminators and handling of control characters. Whitespace is preserved, however.
- XML embedded messages. It is only guaranteed that the XML infoset is preserved. Namespace prefixes, the order of attributes and so on might change. Restrictions to special XML constructs like processing instructions, entity references may apply.

See the format of the SOAP message format for requests and responses in the [Message Format](#) page.

Limitations

To prohibit misuse of the message store as a general exchange of data, the following limitations apply:

- The total number of messages in the message box is limited (currently: 100). If a mailbox overflows, the system throws an error to the sender that the message could not be delivered. The mailbox owner does not get warned if they mailbox is full, but they do not get new messages.
- The size of a message is limited (currently 10KB).
- The number of messages per submitting user is limited; a mechanism prevents a denial of service in the case that the submitted messages are not read by the receiver over a longer period of time (currently: 4).

The recommended values are dynamic parameters and can be changed if necessary.

Using this API

To operate with friend messages, a client application needs to:

1. Complete the authentication procedure, see [Client Initialization](#).

2. Log into the SO system using the SO ID and API password retrieved at the previous step.
3. Call the methods of this API group as needed; see the table below for a full list of available operations.

Table 6: Message Store API

Operation	Description
createMessage	creates a message to another user
listMessages	lists all messages in a message box
getMessage	retrieves a single message
deleteMessage	deletes a single message
getBlacklist	retrieves the blacklist
addToBlacklist	adds another user to the blacklist
removeFromBlacklist	removes another user from blacklist
subscribe	subscribes to notifications about new messages
unsubscribe	unsubscribes from notifications about new messages
getApexDomain	returns the apex (second-level) domain for given sub-domain name

4.2. createMessage

This operation allows a user to create a message to another user. The message may be rejected, because the addressee does not exist or message box limits have been reached.

createMessageRequest

Element Name	Description
from	Sender .tel domain name
to	Receiver .tel domain name
contentType	Type of the message content - encrypted or not.
messageType	<i>optional</i> Friending request or response
actual message	The message data in one of the following formats: <ul style="list-style-type: none"> • binary - base64 encrypted • text - plain text with some XML restrictions applied; see

Message Store for details

- XML - XML message format

createMessageResponse

No elements.

Example

In the example, reggie.tel sends an encrypted friending request message to george.tel.

Request

```
<createMessageRequest
  xmlns="http://xmlns.telnic.org/ws/so/member/messagestore/types-
  1.0">
  <from>reggie.tel</from>
  <to>george.tel</to>
  <contentType>application/x-encrypted</contentType>
  <messageType>friendingRequest</messageType>
  <binary>R0lGODlhcgGSALMAAAQCAEMmCZtuMFQxDS8b</binary>
</createMessageRequest>
```

Response

```
<createMessageResponse
  xmlns="http://xmlns.telnic.org/ws/so/member/messagestore/types-
  1.0"/>
```

4.3. listMessages

With this operation, a client can retrieve a list of the messages that are stored by the system for a specified identity. The client can decide whether it wants to retrieve the ids only or whether it wants to have details about the messages included. The former can be used to limit the traffic size if the client simply wants to check for new messages.

listMessagesRequest

Attribute Name	Description
includeInfo	Boolean value requesting that message metadata be included into the response.

Element Name	Description
mBox	Name of the .tel domain, for which the list of messages needs to be retrieved.

listMessagesResponse

A sequence of "message" elements with the following elements. If metadata has been requested, the size and format of the message are included in the response as well.

Element Name	Description
id	Message id.
from	Sender's .tel domain name.
received	Date and time when the message was received.
contentType	Type of the message content - encrypted or not.
messageType	Friending request or response.
size	<i>optional</i> Size of the message (in bytes).
format	<i>optional</i> The format in which the message is received.

Example

The example requests new message for the domain reggie.tel. A friending request and a response are returned.

Request

```
<listMessagesRequest
  xmlns="http://xmlns.telnic.org/ws/so/member/messagestore/types-
1.0"
  includeInfo="1">
  <mbox>reggie.tel</mbox>
</listMessagesRequest>
```

Response

```
<listMessagesResponse
  xmlns="http://xmlns.telnic.org/ws/so/member/messagestore/types-
1.0">
  <message>
    <id>M39281</id>
    <from>george.tel</from>
    <received>2001-12-17T09:30:47-05:00</received>
    <contentType>application/x-encrypted</contentType>
    <messageType>friendingRequest</messageType>
    <size>3453</size>
    <format>binary</format>
  </message>
  <message>
    <id>Z45939</id>
    <from>albert.tel</from>
    <received>2001-12-17T09:30:47-05:00</received>
    <contentType>application/x-encrypted</contentType>
    <messageType>friendingResponse</messageType>
    <size>4322</size>
    <format>binary</format>
  </message>
</listMessagesResponse>
```

4.4. getMessage

This operation retrieves a single message from the message store. The message is identified by its unique id.

getMessageRequest

Element Name	Description
id	The id of the message to be retrieved.

getMessageResponse

Element Name	Description
id	id of the retrieved message.
to	Receiver's .tel domain name
from	Sender's .tel domain name
received	Date and time when the message was received
contentType	Type of the message content - encrypted or not.
messageType	Friending request or response
actual message	The message data in one of the following formats: <ul style="list-style-type: none">• binary - base64 encrypted• text - plain text with some XML restrictions applied; see Message Store for details• XML - XML message format

Example

Request

```
<getMessageRequest
  xmlns="http://xmlns.telnic.org/ws/so/member/messagestore/types-
1.0">
  <id>Z45939</id>
</getMessageRequest>
```

Response

```
<getMessageResponse
  xmlns="http://xmlns.telnic.org/ws/so/member/messagestore/types-1.0">
  <id>Z45939</id>
  <to>reggie.tel</to>
  <from>albert.tel</from>
  <received>2001-12-17T09:30:47-05:00</received>
```

```
<contentType>application/x-encrypted</contentType>
<messageType>friendingResponse</messageType>
<binary>R0lGODlhcgGSALMAAAQCAEMmCZtuMFQxDS8b</binary>
</getMessageResponse>
```

4.5. deleteMessage

A client can use this operation to delete a message from the message box.

deleteMessageRequest

Element Name	Description
id	The id of the message to be retrieved.

deleteMessageResponse

No elements.

Example

Request

```
<deleteMessageRequest
  xmlns="http://xmlns.telnic.org/ws/so/member/messagestore/types-1.0">
  <id>Z45939</id>
</deleteMessageRequest>
```

Response

```
<deleteMessageResponse
  xmlns="http://xmlns.telnic.org/ws/so/member/messagestore/types-1.0"/>
```

4.6. getBlacklist

This operation retrieves the blacklist of the user, i.e. all users whose messages to the calling user are automatically blocked.

getBlacklistRequest

No elements.

getBlacklistResponse

Element Name	Description
userName	The user name of the SO member.

Example

Request

```
<getBlacklistRequest
  xmlns="http://xmlns.telnic.org/ws/so/member/messagestore/types-
```

1.0"/>

Response

```
<getBlacklistResponse
  xmlns="http://xmlns.telnic.org/ws/so/member/messagestore/types-
1.0">
  <userName>kenny</userName>
  <userName>stan</userName>
</getBlacklistResponse>
```

4.7. addToBlacklist

Using this operation a client can add one or more users to the blacklist of the calling user.

addToBlacklistRequest

Element Name	Description
--------------	-------------

userName	The user name of the SO member to be added to the list.
----------	---

addToBlacklistResponse

No elements.

Example

Request

```
<addToBlacklistRequest
  xmlns="http://xmlns.telnic.org/ws/so/member/messagestore/types-
1.0">
  <userName>butters</userName>
  <userName>chef</userName>
</addToBlacklistRequest>
```

Response

```
<addToBlacklistResponse
  xmlns="http://xmlns.telnic.org/ws/so/member/messagestore/types-
1.0"/>
```

4.8. removeFromBlacklist

This operation allows to remove one or more users from the blacklist.

removeFromBlacklistRequest

Element Name	Description
--------------	-------------

userName	The user name of the SO member to be deleted from the list.
----------	---

removeFromBlacklistResponse

No elements.

Example

Request

```
<removeFromBlacklistRequest
  xmlns="http://xmlns.telnic.org/ws/so/member/messagestore/types-
1.0">
  <userName>kenny</userName>
  <userName>chef</userName>
</removeFromBlacklistRequest>
```

Response

```
<removeFromBlacklistResponse
  xmlns="http://xmlns.telnic.org/ws/so/member/messagestore/types-
1.0"/>
```

4.9. subscribe

This operation allows the client to add a SOAP API URL that should be notified by the SO about new messages for the user.

subscribeRequest

Element Name	Description
notificationUrl	The URL where to send notification of new messages in the SO system.

subscribeResponse

No elements.

Example

Request

```
<subscribeRequest
  xmlns="http://xmlns.telnic.org/ws/so/member/messagestore/types-
1.0">
  <notificationUrl>http://my.nsp.com/notify</notificationUrl>
</subscribeRequest>
```

Response

```
<subscribeResponse
  xmlns="http://xmlns.telnic.org/ws/so/member/messagestore/types-
1.0"/>
```

4.10. unsubscribe

The client can use this operation to no longer receive notifications about new messages for the user to the given SOAP API URL.

unsubscribeRequest

Element Name	Description
--------------	-------------

notificationUrl The URL to be removed from notification list.

unsubscribeResponse

No elements.

Example

Request

```
<unsubscribeRequest
  xmlns="http://xmlns.telnic.org/ws/so/member/messagestore/types-
1.0">
  <notificationUrl>http://my.nsp.com/notify</notificationUrl>
</unsubscribeRequest>
```

Response

```
<unsubscribeResponse
  xmlns="http://xmlns.telnic.org/ws/so/member/messagestore/types-
1.0"/>
```

5. PUBLISHER STORE

5.1. Publisher Store

Because SO members can use different client programs, the private data infrastructure needs a place where the relationships between the publishers and the readers are stored. In the .tel architecture, the publisher store includes information on the list of .tel domain owners, that have made private data available for a given reader.

For each publisher, the system stores the following set of data:

- the domain of the publisher (which represents more or less the identity of the publisher)
- the secure label, which the publisher uses for the reader; the label resembles the DNS label

Using this API

To operate with the list of publishers, a client application needs to:

1. Complete the authentication procedure, see [Client Initialization](#).
2. Log into the SO system using the SO ID and API password retrieved at the previous step.
3. Call the methods of this API group as needed; see the table below for a full list of available operations.

Table 7: Publisher Store API

Operation	Description
storePublisher	stores information related to a publisher
listPublishers	list all publishers with the related information
deletePublisher	removes the data for the specified publisher

5.2. storePublisher

This operation allows the client to store information related to a publisher. If data already exists for the specified publisher, it is overwritten.

storePublisherRequest

Element Name	Description
entry	The entry containing information on the publisher user to be added to the publisher store. The subelements are: <ul style="list-style-type: none">• publisher - .tel domain name• label - the secure label of this domain; resembles a label in the DNS

storePublisherResponse

No elements.

Example

Request

```
<storePublisherRequest
  xmlns="http://xmlns.telnic.org/ws/so/member/publisherstore/types-
1.0">
  <entry>
    <publisher>george.tel</publisher>
    <label>x483292</label>
  </entry>
</storePublisherRequest>
```

Response

```
<storePublisherResponse
  xmlns="http://xmlns.telnic.org/ws/so/member/publisherstore/types-
1.0"/>
```

5.3. listPublishers

This operation lists all publishers for the calling user.

listPublishersRequest

No elements.

listPublishersResponse

Element Name	Description
entry	The entry containing information on each publisher in the store. The subelements are: <ul style="list-style-type: none"> publisher - .tel domain name label - the secure label of this domain

Example

Request

```
<listPublishersRequest
  xmlns="http://xmlns.telnic.org/ws/so/member/publisherstore/types-
1.0"/>
```

Response

```
<listPublishersResponse
  xmlns="http://xmlns.telnic.org/ws/so/member/publisherstore/types-
1.0">
  <entry>
    <publisher>george.tel</publisher>
```

```

    <label>x483292</label>
  </entry>
<entry>
  <publisher>john.tel</publisher>
  <label>a4939272</label>
</entry>
</listPublishersResponse>

```

5.4. deletePublisher

This operation deletes the specified publisher.

deletePublisherRequest

Element Name	Description
publisher	The .tel domain name of the publisher to be deleted from the store.

deletePublisherResponse

No elements.

Example

Request

```

<deletePublisherRequest
  xmlns="http://xmlns.telnic.org/ws/so/member/publisherstore/types-1.0">
  <publisher>john.tel</publisher>
</deletePublisherRequest>

```

Response

```

<deletePublisherResponse
  xmlns="http://xmlns.telnic.org/ws/so/member/publisherstore/types-1.0"/>

```

6. TEL USER MANAGEMENT

6.1. Tel User Management

This API provides operations related to the user management. It provides means to discover information about the user and associated domains, which is not available elsewhere. It also allows to change the web password and the challenge question and response.

Using this API

To operate with user details, a client application needs to:

1. Complete the authentication procedure, see [Client Initialization](#).
2. Log into the SO system using the SO ID and API password retrieved at the previous step.
3. Call the methods of this API group as needed; see the table below for a full list of available operations.

Table 8: Tel User Management API

Operation	Description
getUserInfo	returns various information about the user account
listDomainNames	returns a list of domains that are associated with the user
changePassword	changes the Web password
changeChallenge	changes the API password challenge

6.2. getUserInfo

This operation returns the user type (guest or member), the current user name and the assigned pseudo domain name (currently used by the message store) and location of the public key.

getUserInfoRequest

By default, no elements because information is retrieved for the calling user. The optional elements allow querying for a different user. Only one of the optional elements can be present in a request. If the user found is not the one who issued the request, no information about the user type will be provided.

Element Name	Description
userName	<i>optional</i> SO name of the user to get information for.
userPseudoDomainName	<i>optional</i> Pseudo domain name of the user to get information for.
domainName	<i>optional</i> Domain name associated with the user to get information for.

getUserInfoResponse

By default, it returns information about the user issuing the SOAP request.

Element Name	Description
type	<i>optional</i> The type of the user (member or guest); only specified if the user specified in the query is not the calling user.
userName	Current user name.
userPseudoDomainName	Pseudo domain name assigned to the queried user.
keyLocation	<i>optional</i> The domain name under which the public key will be published. This element will not be present if the queried user has no key stored in the SO.
privateUserSalt	Private user signature used to create the reference for friending messages.
soId	The user's SO serial number.

Example

Request

```
<getUserInfoRequest
  xmlns="http://xmlns.telnic.org/ws/so/member/usermanagement/types-1.0"/>
```

Response

```
<getUserInfoResponse
  xmlns="http://xmlns.telnic.org/ws/so/member/usermanagement/types-1.0">
  <type>member</type>
  <userName>johndoe</userName>
  <userPseudoDomainName>x38294.soid.tel</userPseudoDomainName>
  <keyLocation>d0ql.node0.keys.tel</keyLocation>
  <privateUserSalt>
    cQDjwkZ6NQP4HLkXcdnFdmj9JuLEhmfuwXh4OZdD0X6wI17imo39tcQSP+GvQmYPNSNqx4n
    ayBx1jkaZZfU0QQ==
  </privateUserSalt>
  <soId>g12345</soId>
</getUserInfoResponse>
```

6.3. listDomainNames

This operation lists all domain names that are assigned to the user. For guest users, an empty list is returned.

listDomainNamesRequest

No elements.

listDomainNamesResponse

Element Name	Description
--------------	-------------

domainName	The name of the domain(s) associated with the calling user.
------------	---

Example

Request

```
<listDomainNamesRequest
xmlns="http://xmlns.telnic.org/ws/so/member/usermanagement/types-1.0"/>
```

Response

```
<listDomainNamesResponse
xmlns="http://xmlns.telnic.org/ws/so/member/usermanagement/types-1.0">
  <domainName>reggie.tel</domainName>
  <domainName>george.tel</domainName>
</listDomainNamesResponse>
```

6.4. changePassword

This operation allows a user to change the web password that is used to access the SO systems. The password that a primary .tel user uses to access the community member database is not altered.

changePasswordRequest

Element Name	Description
--------------	-------------

oldPassword	The old SO password for this user.
newPassword	The new SO password for this user.

changePasswordResponse

No elements.

Example

Request

```
<changePasswordRequest
xmlns="http://xmlns.telnic.org/ws/so/member/usermanagement/types-1.0">
  <oldPassword>abcde</oldPassword>
  <newPassword>12345</newPassword>
</changePasswordRequest>
```

Response

```
<changePasswordResponse
xmlns="http://xmlns.telnic.org/ws/so/member/usermanagement/types-
```

1.0"/>

6.5. changeChallenge

This operation allows to change the challenge question and response that is used in conjunction with the client initialization API.

changeChallengeRequest

Element Name	Description
question	The new SO challenge question.
response	The response to the new question.

changeChallengeResponse

No elements.

Example

Request

```
<changeChallengeRequest
xmlns="http://xmlns.telnic.org/ws/so/member/usermanagement/types-
1.0">
  <question>What's my favourite colour?</question>
  <response>pale stone grey</response>
</changeChallengeRequest>
```

Response

```
<changeCallengeResponse
xmlns="http://xmlns.telnic.org/ws/so/member/usermanagement/types-
1.0"/>
```

7. CLIENT INITIALIZATION

7.1. Client Initialization

The client initialization interface provides the functionality to "cold-boot" a new .tel client application from scratch. In contrast to all the other SO APIs, which require the specification of a user's SO id and the API password (the so-called "API credentials") for authentication purposes, the client initialization API requires the specification of the following data:

- the user's web user name
- his web password
- the answer to his challenge question

In turn, the SO system provides the client with the user's API credentials.

Please refer to the [Developer's Manual](#) for a detailed description on how the initialization procedure goes.

Using this API

Below is the list of available APIs in this group.

Table 9: Client Initialization API

Operation	Description
getChallengeQuestion	returns the user's challenge question
getAPICredentials	returns the user's API credentials
getPKCS12	returns a PKCS#12 key store from a PKCS#8 key pair

7.2. getChallengeQuestion

This operation allows a client to request the challenge question that's on record for the user.

getChallengeQuestionRequest

No elements.

getChallengeQuestionResponse

Element Name	Description
challengeQuestion	The current challenge question.

Example

Request

```
<getChallengeQuestionRequest  
  xmlns="http://xmlns.telnic.org/ws/so/init/types-1.0"/>
```

Response

```
<getChallengeQuestionResponse
  xmlns="http://xmlns.telnic.org/ws/so/init/types-1.0">
  <challengeQuestion>Mother's maiden name?</challengeQuestion>
</getChallengeQuestionResponse>
```

7.3. getAPICredentials

This operation allows a client to obtain the API credentials.

getAPICredentialsRequest

In addition to the user's web user name and password (provided over HTTP authentication), the answer to the user's challenge question must be provided to obtain the API credentials.

Element Name	Description
--------------	-------------

challengeAnswer The answer to the current challenge question.

getAPICredentialsResponse

Element Name	Description
--------------	-------------

soid The user's id in the SO database.

apiPassword The user's API password encrypted with the user's public key.

key *optional* The user's key, taken from the SO [Key Store](#). The key data is protected by the user's web password. The private key may be used to decrypt the API password.

Subelements:

- info - metadata for the public key: the hash, the key location, and the date and time of the last change
- data - representation of the key pair: privateKey and publicKey

The key and publicKeyHash elements are mutually exclusive, only one of them can be present in a response.

publicKeyHash *optional* The hash of the public key. Is returned only when the key has been removed from the SO Key store (by the user or by the system when deleting the user). In this case, the SO returns only the hash of the key, so that the client program can at least check whether the returned API password was encrypted with the expected version of the key. See Response 2 for an example of the hash key response.

The key and publicKeyHash elements are mutually exclusive,

only one of them can be present in a response.

`privateUserSalt` Private user signature used to create the reference for friending messages. This element is one per user.

Example

Request

```
<getAPICredentialsRequest
  xmlns="http://xmlns.telnic.org/ws/so/init/types-1.0">
  <challengeAnswer>Cockwood</challengeAnswer>
</getAPICredentialsRequest>
```

Response 1: Full API Credentials

```
<getAPICredentialsResponse
  xmlns="http://xmlns.telnic.org/ws/so/init/types-1.0"
  xmlns:common="http://xmlns.telnic.org/ws/so/common-1.0">
  <soid>g12345</soid>
  <apiPassword>R0lGODlhcgGSALMAAAQCAEMmCZtuMFQxDS8b</apiPassword>
  <key>
    <info>
      <common:publicKeyHash alg="SHA-1">
        UjBsR09EbGhjZ0dTQUxNQUFBUUNBRU1tQ1p0dU1GUXhEUzhi
      </common:publicKeyHash>
      <common:lastChange>2001-12-17T09:30:47.0Z</common:lastChange>
      <common:keyLocation>g12345.node123.keys.tel</common:keyLocation>
    </info>
    <data>
      <common:publicKey>ADCBCiQKBgQDEEkSugaW922FmpTv</common:publicKey>
      <common:privateKey>
        MIGfMA0GCSqGSIsb3DQEBAQUAA4GNADCBiQKBgQDEEkSugaW92
      </common:privateKey>
    </data>
  </key>
  <privateUserSalt>
    cQDjwkZ6NQP4HLkXcdnFdmj9JuLEhmfuwXh4OZdD0X6wI17imo39tcQSP+GvQmYPNSNqx4na
    yBx1jkaZZfU0QQ==
  </privateUserSalt>
</getAPICredentialsResponse>
```

Response 2: Partial API Credentials

```
<getAPICredentialsResponse
  xmlns="http://xmlns.telnic.org/ws/so/init/types-1.0">
  <soid>g12345</soid>
  <apiPassword>R0lGODlhcgGSALMAAAQCAEMmCZtuMFQxDS8b</apiPassword>
  <publicKeyHash alg="SHA-1">
    UjBsR09EbGhjZ0dTQUxNQUFBUUNBRU1tQ1p0dU1GUXhEUzhi
  </publicKeyHash>
  <privateUserSalt>
    cQDjwkZ6NQP4HLkXcdnFdmj9JuLEhmfuwXh4OZdD0X6wI17imo39tcQSP+GvQmYPNSNqx4na
    yBx1jkaZZfU0QQ==
  </privateUserSalt>
</getAPICredentialsResponse>
```

7.4. getPKCS12

Using this operation, the client can request to obtain a PKCS#12 key store from a PKCS#8 key pair. This is intended for Windows Mobile® platform APIs only handling PKCS#12 format keys (whereas many SO functions use PKCS#8 keys).

getPKCS12Request

Element Name	Description
publicKey	The ASN.1 DER encoding of the SubjectPublicKeyInfo type defined by the X.509 standard
privateKey	The ASN.1 DER encoding of the EncryptedPrivateKeyInfo type defined by the X.509 standard. The element has an optional attribute "password", containing the password with which the key has been decrypted. If omitted, the key is decrypted using the user's SO web password.
keyStorePassword	<i>optional</i> The password with which the returned PKCS#12 key store will be encrypted. If omitted, the key store will be encrypted using the user's SO web password.

getPKCS12Response

Element Name	Description
keyStore	The passed key pair as a PKCS#12 key store.

Example

Request

```
<getPKCS12Request xmlns:typ="http://xmlns.telnic.org/ws/so/init/types-1.0">
  <publicKey>MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDEEkSugaW922FmpTvU8KQSn
  I10X0IX6RTNT6w4fpcqDTujiCIvclmzOUvPV5IJkfmFm23EUTfCeDUny4ps2gAp8kFiWVzf8nKG
  QOGhI1tzL62jLkku674JYpi/MEiQGqVsX2XYBJtsCrcl7BDr7+4T57/zzykBMFEFNep0Y14ipwi
  dAQAB</publicKey>
  <privateKey>MIICoTAbBgkqhkiG9w0BBQMwDgQIb5Wx5OFQsRACAggABIICGft4Cyy9yCG6
  X1OZ4KB7f88S2Utg3S4ZptVFjfUz2EGsbaRmqBtNGIXNtK9dTxIrWfDY7i0qNlAMshi+J0fcEQ
  rSjZwkRvqK5KGC4a1VITtHNEBcoC3Et7Kc5VloSfipPuJwbWz1Z02/dpmXF1hGLV2/1ARkvZpTm
  Y2suOMT5mtZEhXJ4OC1psJlShi+wQj5RYcCqqjUllgAx11DeuRX/nPzilldF6KnFAtYnzL9WzOu
  wIIZiCpODG1Xqk3kU5FMSqgar5pD5WY1Bqu0Ego3yfPnSAA0SxtqppxwplNalgdIh4+FFSIy/wl
  itQkkbxz+50J4uAnIc8wv3sgQX3KppBxjImQohKg7fQ2IHPFoKYSQJDWfBshjuQAbNCCaePcU15
  ZY+WMB7F1nqMQJ34NUJxhgDmvXdaLSOsTOnUdSOQYYuMfLahOY/4yUHhU5VlmxGcZshg7XfjaIh
  N44JMoUs+vIoJ8cyfHwyuCuyWwgPeYOboimjgPrFfHUOqtpJk582v4GJki+ogbismw4J98vHu6k
  dkKaf4BYzmKmSmc7eqT3vsgSu2DaalSoLvclVyB7WXi4lyCmFHxeNoUx/jLBdJUB4mhaFmj5ic3
  HxzmHAA4Q7phneAMZNS31sC5kHamnDSTW5Doj2jmkp95I9Ws8wk4nVPHfXkMn76n7MceFxxKxiQ9
  dIwGarZ2UhrS4L9v2vWc156MBAPNYgvlVhfytUs00TztptyPTni8S17GcvKp2XZS8EYVwEfLP8q
  7pgjDN6TxD3KC9hoq9ulvtf2VrMs91I3TiAne1m8g9h02ce1YYkpPJ1PB1ii3Bb2HSI6BOEdT0p
  NkYrnM4FT+S/tkMOWs=</privateKey>
  <keyStorePassword>test</keyStorePassword>
</getPKCS12Request>
```

Response

```
<getPKCS12Response xmlns="http://xmlns.telnic.org/ws/so/init/types-1.0">
  <keyStore>MIAcAQMwgAYJKoZIhvcNAQcBoIAKgASCA+gwgDCABgkqhkiG9w0BBwGggCSABI
idDTCCAwkwwgMFBgsgqhkig9w0BDAoBAqCCArIwggKuMCgGCiqGSIB3DQEMAQMwGgQUkDw7X0mYO
j+tDDpsBfhC4x8c/IICAgQABIICgLub0imgillk9cOr6RJ4v0f9ueHF+5Y48vB9EnZmFW1EyLCT
g70oHYiNCN+Y/byCA05akJ0JtiCzkm9EZ6+F508InG11tOLhelM2MB2AOFj6YHCvxq8HVKcv5Yu
As+B1F81GrUlyuEfPXhuDqLwnazoQ3XRP8+G4JNsX+7WTetGjuZ4mq94wUqrOLCbr8fg+aGfCQN
jJ6HjmfPFaS/uRWB88a9Xf2h2pocc0VLPu57At8qydHAe4nxkA3JuXFqIkaONy+whGCdPUGfzKdi
3gyIFKnZYjhhicFE9wf41P00Um0042hvKNq6GLqjCqb3Qxvs+7h63oxhX1VbENHKSXJ2NV/HzRd
nviJrlIprIs0ZhLF95QnTFEGJsdSOJUotm6BGErp21M0+sms10hEeaqlyxndefq2uQoTfmR+3Eg
+7iR3iDWx0zIYCehRe0CoqgsNqXaCwshV4jMXJ6yBTb9SNTeam1EdgA1aRQRV+q5MkHSPBD4mlC
Le9djE/8NSkE9/A4jz2e6j+5z4cQe8r+z5RddRcap08UTpFcGKIs2I9A/fqosRJ+bc/7EJKvvg0
lBI6XazIsWDBzFQF8yTlCCEOE1KFS2Rc4x1UuIEO6hecgyo5KKAolpUGOvy3qDRcIM3yQ4HX+8
cGKjCJkrfkmx3sscTTbS+8Xk8xWB5JqKd0j1I9u4nR7gUWFITrBD+5eSu7RGYblGK3gAxlpXTYp
DacvEkVo/1r37NljbRWP+zIglS6LL2FS7GG8P19Uwyc1gMckltFsUW8ZI3qUEG1iI6Eu/Jyi/p
shzlpSoleudUA04WHUIJWYyD6gsfUlKvPRYClWFYMSyo+48mh80WOS/BYxQDAZBgkqhkiG9w0BC
RQxDB4KAG0AeQBrAGUAeTAjBgkqhkiG9w0BCRUxFgQU6i83w85/kgM1N0/Jp2BaNBuS8oMAAAAA
AAAwgAYJKoZIhvcNAQcGoIAwGAIBADCABgkqhkiG9w0BBwEwKAYKKoZIhvcNAQwBBjAaBBSRwcj
qKS0glOdr056CU5b1hBhNgICBACggASCAlgIi5Qjzgz+Tz06n5ZWVl63pINfDyUTJ/fsKBmZ4ep
vFO/PWi+yJDoxV1S5fdxi2nBdnUnW5qPDbYhZXrrKmJfntyVWJURtoehOxymrFhXdWKuhUgIYJ4
Q3zzEkLqEY6KYILfvqrZA2X4xJ9Q1L4BIIB948mz1/r4TzqqMJAwoVJCxuKwtDEvchD9JE0RuCd
MiClpvbm8Y9x4+gI29x3zx9lrKRoJm5WOF3XkwWKK9poH2px6MkA7VusjnecTVLLfrz/Rs4dMoB
uPYEh/ixJGAbEvfahcMTMThdtLG253sqnbNv7bpborFI2tVA6J/AZ+uWkq5G0MwNIjHmuSFh0L
FnjofqLtBDK67PQdzqgyWcqYrzUE6IvQc/8xOAwAudNpPN4vpgDExVyD/mKIZWhtkgLuQ6AXVsi
yH5DZENHCGMas8su8w+I0hStHo90oIjwf3bw7XUwkqX3hYsMDnKtTgXierOEnsvUbzalv/sAtKo
YtVQ5ZSJLwp6Au9UR4fV5aB+xcRDy9xoj+bmdd5SEhR6mRL6HGI8xzazvu0m6PAfR0Wk/qwhs13
lg/6OhwHrdYMKELsoPVvjMXVorRFyn5hBx7SiVTthK5kSqBHaVFZik7/f8pkwchu9SkAsSeywmZ
2H2eeH+9Rnk1CePu7oR77y/c3kKV5MseUsksGIqm5hlCU2TD6qOfQYKeMBx51WaVAi+tdOFEnso
UVLef5BeGrvYZMB8wH72gc1GnGOZVHrReeQzR40ZGmc3B0bUNNLskrhmb6Sm5pAVmphSQwcepGz
FgrheEp4xRjunOeLAAAAAAAAAAAAAAAAAAAAAAAAAMd0wITAJBgUrDgMCGGUABBS/B180gemVRmJ
BW74ZXiT+U6BWugQUPM7LxNYPALpQ75DjSTUSPHk05R0CAgQAAAA=</keyStore>
</getPKCS12Response>
```