

---

# **Developers Manual for .tel Client Applications**

**March 2010**

---



## Disclaimer

Copyright (c) 2008-2010, Telnic Ltd.

All rights reserved.

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

- Redistributions of documentation must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions 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

<b>1. Overview .....</b>	<b>6</b>
<b>1.1. About this document .....</b>	<b>6</b>
<b>1.2. About Telnic and .tel.....</b>	<b>6</b>
<b>1.3. About .tel.....</b>	<b>6</b>
<b>2. Architecture.....</b>	<b>8</b>
<b>2.1. Overview .....</b>	<b>8</b>
<b>2.2. TelHosting Software.....</b>	<b>8</b>
<b>2.3. TelFriends and SO.....</b>	<b>8</b>
<b>2.4. .tel Client Applications.....</b>	<b>9</b>
<b>3. Quick Start .....</b>	<b>10</b>
<b>3.1. How do I read .tel data?.....</b>	<b>10</b>
3.1.1. Dig command.....	10
3.1.2. nslookup.....	10
3.1.3. UNIX host.....	11
3.1.4. Libraries & Scripts.....	11
<b>3.2. How do I write to .tel?.....</b>	<b>11</b>
3.2.1. End-points .....	11
3.2.2. Logging in.....	12
3.2.3. Storing records .....	12
3.2.4. Managing domains/folders.....	12
<b>3.3. Function reference.....</b>	<b>13</b>
<b>4. Anatomy of a .tel domain .....</b>	<b>14</b>
<b>4.1. NAPTR Records .....</b>	<b>14</b>
<b>4.2. LOC Records .....</b>	<b>15</b>
<b>4.3. TXT Records.....</b>	<b>15</b>
<b>5. Looking up .tel domains .....</b>	<b>16</b>

---

<b>5.1. Look up a .tel domain.....</b>	<b>16</b>
<b>5.2. Process NAPTR records .....</b>	<b>18</b>
5.2.1. Handling multiple entries.....	18
5.2.2. Handling contacts with unknown Enumservices .....	19
<b>5.3. Process Protected NAPTR Records .....</b>	<b>19</b>
<b>5.4. Display NAPTR records .....</b>	<b>19</b>
<b>5.5. Process TXT Records.....</b>	<b>21</b>
5.5.1. Version of Structured Data .....	22
5.5.1. Lazy Parsing and Presentation of Structured Keywords.....	22
5.5.2. System Message Types .....	22
5.5.3. Keyword Types .....	23
5.5.4. Structure of Advertisement Records .....	23
<b>5.6. Display TXT Records.....</b>	<b>24</b>
5.6.1. Presentation of Advertisements.....	24
5.6.2. Keyword Label and Value Presentation.....	26
<b>6. Privacy-Related Operations .....</b>	<b>27</b>
<b>6.1. TelFriends Authentication.....</b>	<b>27</b>
<b>6.2. Friending Message Processing .....</b>	<b>29</b>
<b>6.3. Adding a Friend.....</b>	<b>33</b>
<b>7. Managing A .tel domain .....</b>	<b>35</b>
<b>7.1. TelHosting Authentication.....</b>	<b>35</b>
<b>7.2. Updating a .tel domain.....</b>	<b>35</b>
<b>7.3. Adding Private Data.....</b>	<b>35</b>
<b>7.4. Profile Switching .....</b>	<b>35</b>
<b>8. Other Features.....</b>	<b>38</b>
<b>8.1. Address Book Integration.....</b>	<b>38</b>
<b>8.2. Auto-provisioning.....</b>	<b>38</b>
<b>8.3. Search for .tel domains .....</b>	<b>39</b>

---

---

8.4. Exporting / Importing Data.....	40
8.5. Preferences.....	40
9. References.....	41
10. Appendix A: Contacts Display in current .tel client apps .....	41
10.1. Windows Contact Handling .....	41
10.2. Web-based Contact Handling .....	42
11. Appendix B: TXT Display in current .tel clients.....	43
11.1. Patterns of presentation for Keywords .....	43
11.2. Use of keyword presentation patterns .....	43
11.3. Pattern Variants for Specific Keyword sets.....	44
11.4. Keyword Presentation Examples.....	45

## 1. OVERVIEW

### 1.1. About this document

This document is targeted at developers of .tel client applications for PC and mobile devices. The Manual offers a description of the architecture and processes required to create or update .tel client applications.

The document does not include specifics of systems underlying the client applications, and instead focuses on the common aspects. For detailed instructions on setting up the developer environment, getting available tools and samples, please read the Quick Start guide and browse the Developer Forum. The guide does not cover implementation of error handling, installation, or update behaviour of applications.

This Manual requires understanding of DNS basics and key concepts of the .tel top-level domain. For introductory materials, please consult the Developer FAQ. The Manual is based on the API references and other documents available on the developer website; please refer to [References](#) for a full list of relevant resources.

### 1.2. About Telnic and .tel

The .tel TLD (top-level domain) is sponsored by Telnic Ltd. This Organization is responsible to its community of domain owners to ensure they are provided with a reasonable service. As such, the Sponsoring Organization has established an accreditation process for all registrars (organizations accredited by ICANN to register domain names). Organizations that enable provisioning of .tel zones are TelHosting Providers. In addition, the sponsoring organization itself provides a number of services to support the new concept of DNS usage.

### 1.3. About .tel

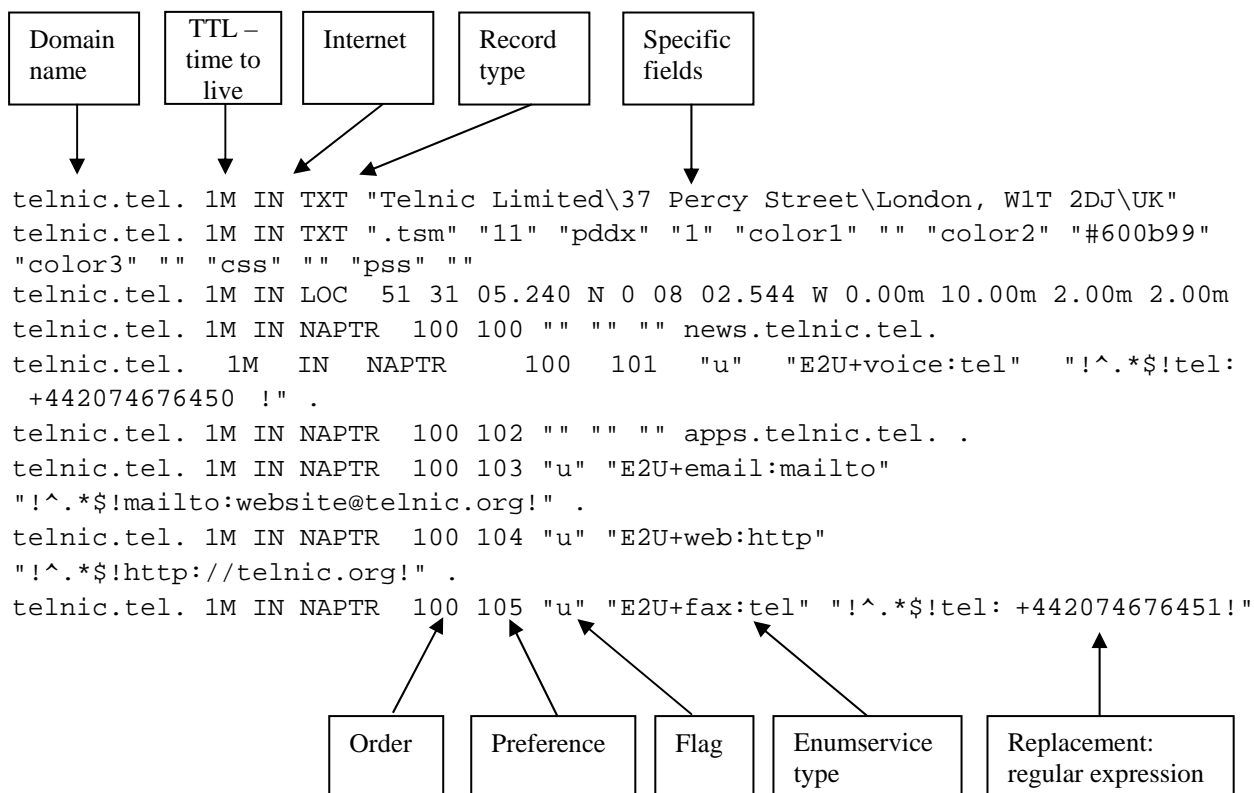
A .tel domain is an innovative top-level domain (TLD) that allows using the DNS as a data store for contact information. The key features of the .tel architecture include:

- Usage of the DNS as a contact data store
- Protection of sensitive contact information
- Proxy front-end and a web interface to manage .tel domains
- Unified SOAP APIs exported for client applications

Unlike traditional domains, .tel does not allow storage of Address or CNAME records; instead, a .tel domain can hold the following records:

- **TXT** - Text string of 255 characters that can contain plain text, a system message, or a set of keyword value pairs like "tc" "London" (tc for Town/City)
- **LOC** - Location record for storing the geographical coordinates as latitude, longitude and altitude with the accuracy of 6 decimal places
- **NAPTR** - Naming Authority Pointer Record with contact information as a URI and an Enumservice that indicates the type of URL information, such as email or phone number. All Enumservice types are supported, and ordering and labelling are available.
- **Other** record types, such as SRV and MX

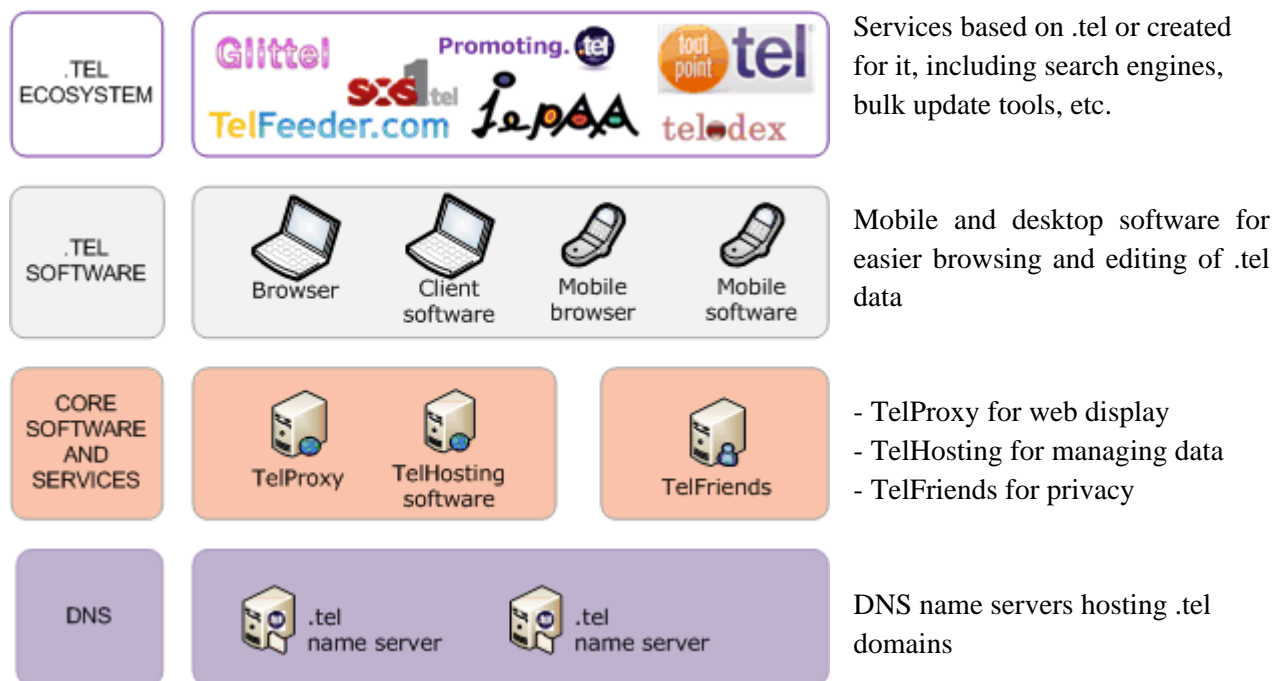
For example:



## 2. ARCHITECTURE

### 2.1. Overview

For adequate storage, retrieval and display of this information, Telnic Ltd. has implemented a multi-level architecture consisting of the following levels:



For examples of software and services, please refer to the developer site, <http://dev.telnic.org>. This section focuses on the core software solutions.

### 2.2. TelHosting Software

TelHosting Software is an open-source Java® application that creates and maintains DNS zone objects, publishing them in .tel DNS name servers. The system is backed by an SQL database; due to an abstraction layer, a variety of popular database products are supported.

TelHosting Software exports a set of SOAP APIs for client applications. The API is divided into the Client part, which allows operations with records, profiles, secondary users, etc., and the Admin part, which is chiefly used by TelHosting Provider personnel for administrator purposes. See the API references at the website. This guide shows how parts of the APIs can be used to implement client applications of varied functionality.

### 2.3. TelFriends and SO

Telnic, the Sponsoring Organization (SO) provides a number of services to the .tel community: the TelProxy that renders .tel data on the web, the TelPages search engine, and TelFriends to manage friending relations and private data sharing. All of these services are sometimes referred to as TelFriends services. In most instances, only TelFriends is meant.

To securely share private information, the .tel owner can activate the TelFriends service and create a list of 'friends' with personalized access to encrypted data stored in secret sub-domains. Anybody with a TelFriends account can be added to a friend list. The 'friending' mechanism is a one-way process, so even

if both participants own .tel domains, they do not become friends unless each one explicitly adds the other to the friend list.

To access private data, log into the .tel page on the web or via a software application. Behind the scenes, the TelFriends service checks whether you are a friend of this domain, and uses your private decryption key to read private data that has been made available for you.

This TelFriends system maintained by Telnic independently of TelHosting software and provides its own open SOAP interface. The SOAP end-points for TelFriends are:

- <https://soap.telfriends.tel/init?wsdl> – initialization and login
- <https://soap.telfriends.tel/member?wsdl> – main operations

## 2.4. .tel Client Applications

The term ".tel client applications" groups all the various plug-ins and standalone applications that operate data published in .tel domains. For a list of currently available applications, please consult the Developer Website, .tel Applications. There are numerous other systems that would benefit from client code that intelligently processes and manages contact information from the .tel DNS.

The function or functions of a .tel application can range from simple lookups to TelFriends authentication, private data sharing, and custom map displays. This document illustrates how the standardized SOAP APIs of the TelHosting Software and the TelFriends system can be used to implement .tel client solutions. Please also check section 3 and the Quick Start document to identify whether your client application needs to implement this or that scenario. More information and live support are available at the Developer Forum.

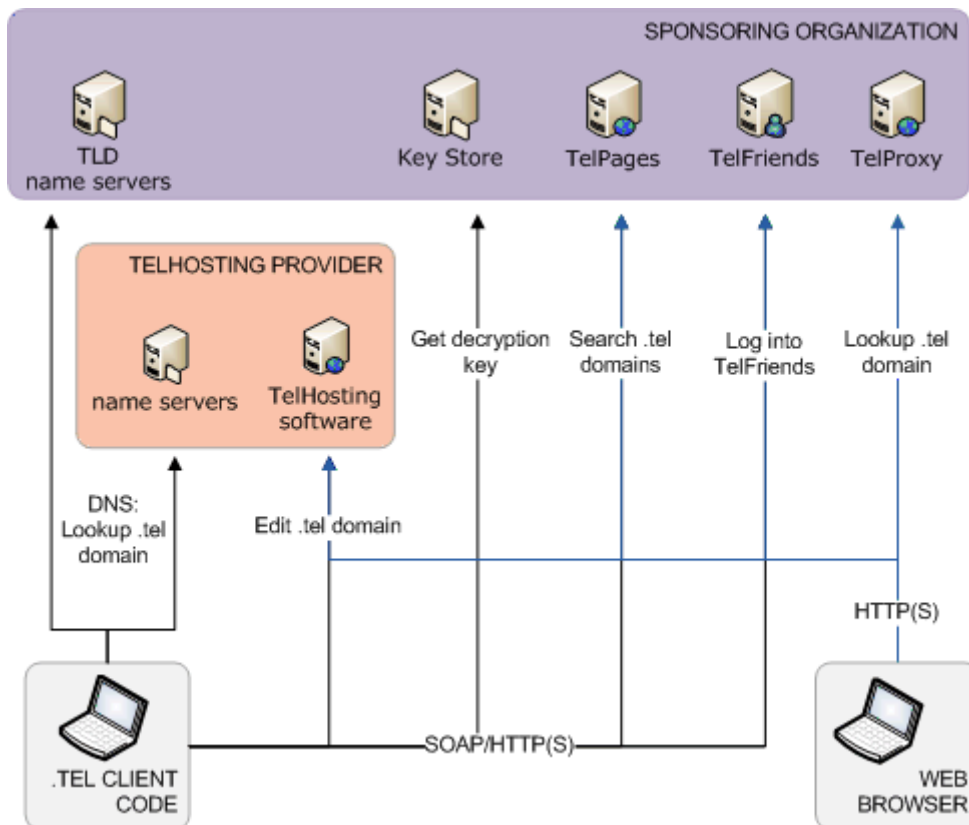


Figure 1: End-user Access Points to .tel components

As shown in Figure 1, end users can view and manage .tel domains in a web browser or a custom client application. Telnic web interface enables users to look up .tel domain names and view available data presented by the TelProxy on a preformatted web page, to search for domains, and to log into the TelFriends and TelHosting Provider systems. With the TelHosting account, .tel domain owners manage their TelHosting Provider profile, publish .tel contacts, etc. With the TelFriends account, users establish friending relations with .tel domain owners to access private encrypted information. All of these actions are also available for client applications via the public SOAP interfaces.

Depending on the client functionality, different operations are available that use corresponding TelHosting or Sponsoring Organization services. The diagram only demonstrates typical operations; the actual SOAP APIs provide many other operations needed to operate .tel domain names. Read on to learn about available functionality.

### 3. QUICK START

Before you begin writing your new .tel application, please read this section to clearly understand the key concepts and functional areas. On any OS and environment, a .tel client application falls into one of the two main categories: reading or writing .tel data – see sections 3.1 and 3.2 respectively.

Section 3.3 offers a finer-grain classification of .tel-related operations, while subsequent parts of the guide provide in-depth documentation on each one. Please remember that method descriptions are in the API references outside this guide.

#### 3.1. How do I read .tel data?

If you are planning to create an application or service that needs to read information in .tel domains, you do not need to use the APIs – just do a direct lookup in the DNS, for example:

##### 3.1.1. Dig command

```
$ dig telnic.tel LOC
telnic.tel. 1M IN LOC 51 31 05.240 N 0 08 02.544 W 0.00m 10.00m 2.00m 2.00m
```

**Note:** Always specify the exact record type you are querying. Do not use the parameter “ANY” because you may get incomplete or confusing results depending on the DNS server configuration.

##### 3.1.2. nslookup

```
$ nslookup -type=naptr _soap._nspapi.mharris.tel
Server:          192.15.1.10
Address:         192.15.1.10#53
```

Non-authoritative answer:

```
_soap._nspapi.mharris.tel naptr = 10 10 "u" "E2U+web:http"
"!^.*$!http://telhosting.domainmonster.com/client!" .
_soap._nspapi.mharris.tel naptr = 10 10 "u" "E2U+web:https"
"!^.*$!https://telhosting.domainmonster.com/client!" .
```

Authoritative answers can be found from:

```

mharris.tel      nameserver = NO.CTH.DNS.NIC.tel.
mharris.tel      nameserver = TO.CTH.DNS.NIC.tel.
mharris.tel      nameserver = SO.CTH.DNS.NIC.tel.
mharris.tel      nameserver = AO.CTH.DNS.NIC.tel.
mharris.tel      nameserver = DO.CTH.DNS.NIC.tel.
D0.CTH.DNS.NIC.tel      internet address = 195.253.47.64
  
```

Please note that the exact nslookup command line options may differ depending on the version/OS. A more modern alternative to nslookup is host.

### 3.1.3. UNIX host

```

$ host -t naptr vlad.tel

vlad.tel has NAPTR record 100 101 "u" "E2U+voice:tel+x-lbl:Current+x-mobile"
"^.*$!tel:+447853326927!" .

vlad.tel has NAPTR record 100 102 "u" "E2U+voice:tel+x-lbl:Direct+x-work"
"^.*$!tel:+442074676464!" .

vlad.tel has NAPTR record 100 103 "u" "E2U+voice:tel+x-lbl:Switchboard+x-
work" "^.*$!tel:+442074676450!" .

vlad.tel has NAPTR record 100 104 "u" "E2U+fax:tel+x-work"
"^.*$!tel:+442074676451!" .

vlad.tel has NAPTR record 100 108 "" "" "" email-me.vlad.tel.
  
```

### 3.1.4. Libraries & Scripts

Alternatively, read .tel data using:

- Library such as [LDNS](#) or Perl [Net::Lookup::DotTel](#) [Perl script](#) for recursive lookup for tracing whole .tel trees.

## 3.2. How do I write to .tel?

To create an application or service that writes to a .tel domain, you need to use one of the standardized APIs (SOAP or AJAX), to login to the TelHosting Software and pass on data.

### 3.2.1. End-points

Because .tel domains can be hosted and maintained by different providers, the end-points for TelHosting interfaces differ. To get the correct API end-point for a .tel domain, query:

**For AJAX:** `_http._nsp._apps.[domain].tel.` or `_https._nsp._apps.[domain].tel.`  
 Append this URL of your TelHosting Provider, with `/g2/json` to get your end-point.

**For SOAP:** `_soap._nspapi.[domain].tel` for NAPTR records. The returned records should point to the addresses of the endpoint, such as `https://MyTelProvider:80/client`

### 3.2.2. Logging in

**For AJAX:** get the end-point and call the login method. Use the returned cookie for all subsequent methods.

**For SOAP:** get the end-point and send the web username and password in the HTTP Headers using BasicAuth. Credentials should be always sent on every SOAP query. SOAP service is normally provided via both HTTP and HTTPS; we suggest using HTTPS for security purposes.

**For TelFriends SOAP:** use the API credentials determined during TelFriends initialization.

### 3.2.3. Storing records

Key points on record management for both interfaces:

- When editing a record, store its ID in the attribute of naptr element.
- When adding a record to all profiles, use profiles=“\_all\_”.
- When entering a URL for a NAPTR record, add a '.' to the end of the replacement field.
- For records, the “owner” attribute determines whether the record is published in the main domain (owner=“@”) or in a sub-domain (owner=sub-domain relative to main domain).
- To create a ‘Go to’ link, add a non-terminal NAPTR record.
- Text and keyword records cannot be sorted, and are returned in a random order.
- Export/import operations are available in SOAP, but not in AJAX.
- When deleting records, AJAX prevents deletion of the last public record.

### 3.2.4. Managing domains/folders

Key points on sub-domain (a.k.a. folders) management:

- There is no fixed technical limit on the number of sub-domain levels, other than that any FQDN must be less than 253 bytes long, including the .tel suffix.
- The total number of sub-domains per domain and the number of records in any domain or sub-domain are limited; check the settings with your .tel provider.
- TelHosting Software treats domains and sub-domains equally, so SOAP does not prohibit deletion of 2<sup>nd</sup> level domain. Make sure you do not delete the apex domain.

### 3.3. Function reference

Please consult the table below to determine which functions your .tel application will need to perform, what needs to be implemented for those functions, and where in the developer's guide you can find relevant information.

Operation	Implementation	Reference
Look up .tels	<ul style="list-style-type: none"> <li>• DNS querying</li> <li>• Processing returned records</li> <li>• Launching the software appropriate for the communication type of each record</li> </ul>	Section 5
Process private data	<ul style="list-style-type: none"> <li>• TelFriends authentication with the user's TelFriends credentials</li> <li>• Retrieval of the list of "friends"</li> <li>• Retrieval of private decryption keys from the TelFriends key store</li> <li>• Decryption of protected NAPTRs</li> <li>• Display of protected NAPTRs to the user (may or may not be the same as displaying public records)</li> </ul> <p>If your program does not need to process encrypted data, it can ignore private records and only process public information.</p>	Section 6
Friend .tel users	<ul style="list-style-type: none"> <li>• TelFriends authentication with the user's TelFriends credentials</li> <li>• Retrieval of pending "friending" messages and a mechanism for sending message responses</li> <li>• Retrieval of private bits from the messages</li> </ul>	Sections 6.1, 6.2
Search for .tels	<p>Choose on the implementation options:</p> <ul style="list-style-type: none"> <li>• Displaying the TelPages web interface for search</li> <li>• Implementing a search via SOAP TelPages interface, simple or more complex with regular expressions and suggestions for better searches</li> </ul>	Section 8.3
Edit own .tel	<ul style="list-style-type: none"> <li>• TelHosting Provider authentication</li> <li>• Retrieval of currently published data set</li> <li>• Editing data and sending it back to the server</li> </ul> <p>Any application must comply with the restrictions on data that can be published within a .tel</p>	Section 7.1
Switch .tel profiles	<ul style="list-style-type: none"> <li>• TelHosting Provider authentication</li> <li>• Retrieval of currently published profiles</li> <li>• Changing the "active" setting and sending the data to server</li> </ul>	Section 7.4

## 4. ANATOMY OF A .TEL DOMAIN

Each domain has a number of required and optional values that it can contain. Specific requirements determining the domain content are:

- Normally contains a fixed A record and a CNAME record with values assigned by the registry. Domain owners or TelHosting Providers cannot change these values, because they are used to generate consistent results to web-based queries on the domain name. No other Address records, CNAME or DNAME records are permitted.
- Normally contains records of the types NAPTR, TXT and LOC.
- Must contain at least one NAPTR record; multiple records of different types are allowed. See section 3.1 for a description of NAPTR record types.
- Must not contain more than one LOC record. Because the LOC record describes the location of the domain owner, and the owner can only be in one place at a time, multiple records of the LOC type are prohibited. See section 3.3 for a description of LOC records.
- May contain one or multiple TXT records representing keywords, by which a domain can be found. See section 3.2 for a description of TXT records.

### 4.1. NAPTR Records

The Naming Authority Pointer (NAPTR) DNS resource record specified in RFC 3403 [6] can hold a contact value as a URI. In addition, NAPTRs include "order" and "preference" fields to indicate the priority of a given record, as well as a reference of an Enumservice type to indicate the type of communication that the given record contains. Both IETF-standardized and custom Enumservices are supported. Among custom Enumservices, a number of descriptive services have been introduced to characterize and label the contact held in the content NAPTR. Descriptive Enumservices can only act in combination with active records, not as separate contacts.

All .tel client applications need to parse and adequately present the following Enumservice types:

Enumservice Group	Description
Standard	Standardized at the IANA Number Registry web site [7]. <b>Types:</b> sip, h323, voice:tel, sms:tel, ems:tel, mms:tel, sms:mailto, ems:mailto, mms:mailto, email:mailto, web:http, web:https, ft:ftp, fax:tel
VoIP and IM	Indicate that the contact can be used to start a communications session including voice/video and Instant messaging. <b>Types:</b> x-voice:skype, x-voice:gtalk, x-im:aim, x-im:icq, x-im:ymsg, x-im:msnim, x-im:xmpp
Protected	Stores encrypted information <b>Type:</b> x-crypto:data:<csid>
Descriptive: location indicator hints	Hold auxiliary textual information that can be presented to the user. Hint to the location of the publishing user. <b>Types:</b> x-mobile, x-work, x-main, x-home, x-transit, x-prs
Descriptive: labels	Contain information for presentation to the user or for interpretation by the client program. <b>Type:</b> x-lbl:text:*[" text]

For instructions on processing and displaying NAPTRs, please refer to section Key Operations in this guide.

## 4.2. LOC Records

A Location record is for expressing geographic location information: latitude, longitude, and altitude, see RFC 1876 [5]. Only one LOC record is allowed per domain. It is assumed that the record indicates the location of the domain owner, and the owner can only be in one place at a time.

Depending on whether you use DNS libraries for the client application, processing LOC records can be more or less tricky. If you use no DNS libraries, you'll need to figure out LOC record processing on your own; see RFC 1876, Appendix A [5] for sample C code. Another example is the iPhone application available on the website, which uses the open source LDNS library to partially resolve processing of this type of record.

## 4.3. TXT Records

The TXT resource record type is defined in the main DNS standard (RFC 1035 [4]). TXT records can be stored in the DNS and can hold a series of strings, each of which is less than or equal to 255 octets in length. Multiple TXT records can exist in a single domain, with all available TXT records being returned in response to a query. The order in which these records are returned is not guaranteed, but strings within a TXT record will be processed in the left to right order.

The formal size limit is the maximum size of a DNS resource record. In practice, of course, creating a massive TXT record or set of TXT records is unwise; the DNS is optimised for relatively short messages, and not all networks react correctly to excessively long DNS messages.

TXT records are used in .tel domains to store non-contact related information about the registrant, and are subdivided into the following groups:

- **Generic descriptions** or messages to be displayed to the user in the normal way.
- **Structured Keywords** for search capabilities. Keywords are identified by the `.tkw` initial string followed by the version number and the keyword itself (type, value pairs). In principle, keywords are independent, but placing several keywords in one TXT record establishes a connection; for instance, a TXT record with an initial keyword type `postalAddress` may contain the subsequent keyword types `townCity`, `postalCode`, and `stateProvince`. The postal address keyword would act as a label to distinguish this address from others entered, and the other keywords would form a chain as specified by the domain owner, so that no client-side collating and formatting of the address is necessary.
- **Structured System Messages** parsed by the client code and not shown to the user. A client application receiving a TXT record starting with `.tsm` should interpret the strings following the version in this record as a system message. Only one system message is allowed per TXT record, and multiple TXT records within a domain must not have contradictory values of this message.
- **Advertisements** to separate sponsored links from contact information in the domain and provide adequate presentation for those links. Advertisements are identified by the `.tad` initial string followed by the version number and the advert fields. These records are stored in a special sub-domain `_ad` of this domain. Any advert records that are found in the domain itself will at present be displayed as plain text in the header; in subsequent versions, the TelProxy and client applications should ignore any malformed or misplaced ads. The client application may choose to lookup and present those sponsored links to the user, or to ignore that sub-folder.

For details on the processing and displaying of TXT records, please see sections 5.5 and 5.6.

## 5. LOOKING UP .TEL DOMAINS

This section provides guidelines regulating the way client applications should query .tel domains, handle retrieved contacts, and manage .tel domain contents. The guidelines are based on the specifications of currently available Telnic-funded client applications, and it is expected that independent client programs will behave in the same way. Code snippets for this guide have been taken from live applications for illustration purposes.

Please note that your application may not need to implement all of the scenarios described below or may perform the operations in a different way. Please check the description of the operation and section 3 or the Quick Start document to identify the functionality that you need to implement in order to perform specific .tel-related tasks with your client program.

### 5.1. Look up a .tel domain

This is a basic operation that in its simplest form involves only direct DNS queries. A client application needs to query the DNS for a specific domain name and display the retrieved results. In this operation, the client queries the TLD name servers for the .tel domain, and the response redirects the client to the TelHosting Provider servers where the queried domain is published.

In addition to basic lookup, the client may be able to retrieve the list of publishers from the TelFriends system, get their private keys, and use them to decrypt any private information that has been made available to the querying user. This case has the following prerequisites:

- The user has an account in the TelFriends member system.
- The client application implements the TelFriends login procedure.
- The client application implements retrieval of private keys and data decryption.

In the extended TelFriends -aware version of the procedure, the client application does the following:

1. Query the TelFriends for the list of publishers. In practice, the client will hold a local cache of this publisher list.
2. Check whether the publisher list includes the queried domain.
  - a. If no, proceed to querying the DNS with the domain name.
  - b. If yes, retrieve the label of the sub-domain, where private data has been published for this user.
3. Query the DNS with the domain (for 2b, sub-domain) to retrieve any TXT, NAPTR, and LOC records, sequentially.
4. Decrypt any protected NAPTRs.
5. Display the results to the user. See sections 5.4 and 5.6 for details on how to display NAPTR and TXT record types.

Figure 2 shows the interactions between the user, client, TelFriends and DNS server when performing a lookup. The diagram assumes that the domain is included in the publishers list and therefore private data will need to be decrypted.

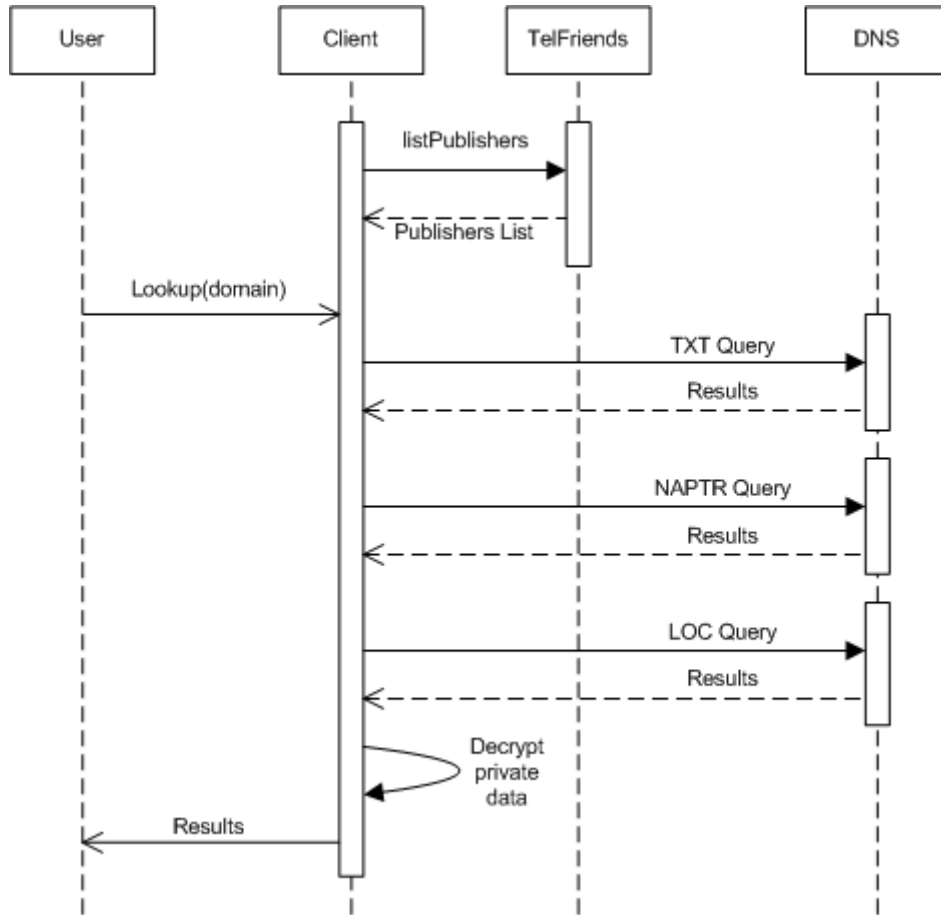


Figure 2: Lookup Procedure

**Sample Code: Blackberry Client Lookup of TXT record**

Taken from the code for the plug-in for Blackberry OS devices,  
 org.telnic.blackberry.lookup.LookupTelname.java.  
**private void** performTxtLookup (**final** String telnameVal)

```

{
    Logger.logMessage( "Performing TXT lookup: " + telnameVal,
Options.KEY_LOG_APPLICATION );
    type = DNS.TYPE_TXT;
    if (!cancelled) {
        lwTxt = new LookupWorker( telnameVal, DNS.TYPE_TXT,
TelnameApp.getDnsConnectionManager() );
    }
    if (!cancelled) {
        lwTxt.setLookupWorkerCompletedListener( this );
    }
    if (!cancelled) {
        lwTxt.start();
    }
}
}

```

**Note:** The lookup procedure may also be affected by the pddx system message keyword, which indicates that private data does not exist within the containing domain. This way, the client does not need to consult the list of publishers to query private sub-domains, see section 5.5.1.

## 5.2. Process NAPTR records

When a client application presents contacts to the user, selecting a contact should initiate an external program designed to support the selected URI and start the appropriate communication session; see Appendix A for lists of applications launched by existing client applications. Sometimes, instead of starting a communication session, the client program can perform a different action; for instance, the client could send commands to an external Bluetooth-connected cell phone to trigger it to dial a telephone number.

When the client program supporting a given service is available but does not respond to a system call using a Launch URI (i.e. the program has not registered itself as a URI handler with the operating system), other methods may be used. For example, to launch the external program on Mac OS X® an Applescript could be run, and on Windows® OS, a special instruction passed on the command line. These cases are shown in the tables of Appendix A as "cmd – x".

In addition, a number of services have web-based solutions. For example, Gmail, Hotmail and Yahoo! Mail can be accessed via web browsers, and so could be used to send email messages to recipients. If so configured, the client program could convert the contact's URL into the launch URLs and pass these to the local web browser; e.g., for the contact <mailto:freddy@foo.com>, the following URLs could be generated:

- Gmail: <http://mail.google.com/mail/?view=cm&fs=1&tf=1&to=freddy@foo.com&fs=1>
- Hotmail: <http://hotmail.msn.com/secure/start?action=compose&to=freddy@foo.com>
- Yahoo! Mail: <http://search.yahoo.com/search?p=%21mail+freddy@foo.com>

### 5.2.1. Handling multiple entries

If a record holds more than one active Enumservice, the client program can choose to present all active Enumservices in one entry or to split the NAPTR into separate entries, one for each active Enumservice. The choice depends on the capabilities and normal presentation style of the client environment.

- **Combined Entry** – All active Enumservices are combined and then presented. Thus `x-voice:skype+x-im:skype` should be presented as “Skype Voice & IM” (using the ampersand character as a combining token). Within a NAPTR, active Enumservices **MUST** refer to the same URI scheme, so the system type (in this case “Skype”) only needs to be presented once.
- **Multiple Entries** - A NAPTR with multiple active Enumservices is split into a set of entries. Each entry has a single active Enumservice, but all entries have all of the auxiliary labels of the original NAPTR. The entries are treated in sequence, so that the “best” entry is that associated with the leftmost active Enumservice, whilst the “worst” entry in the set is that associated with the rightmost active Enumservice. Thus, `x-voice:skype+x-im:skype` should be presented as two entries. The first entry has “Skype Voice”, with the second entry having “Skype IM”.

Note that the client program needs to ignore any duplicate active Enumservices (i.e. multiple copies of the same Enumservice in one NAPTR) and consider only the first (leftmost) copy.

For multiple location indicator hints (LIH, see section 5.4) in one NAPTR, the client program should concatenate the hints with a combining token "and" between each of them.

### 5.2.2. Handling contacts with unknown Enumservices

An Enumservice of an unsupported type might be provisioned via the SOAP API if someone has used a different provisioning system to configure the queried domain.

- If a single NAPTR holds multiple active Enumservices, and one of them is unrecognized, that service can be silently ignored. To select the appropriate icon, use a known Enumservice.
- If the only active Enumservice is unrecognized, the client program should be able to discard the NAPTR or present it by stripping the x- at the front and displaying the auxiliary labels and URI as usual. Use the generic/unknown icon with this contact.

### 5.3. Process Protected NAPTR Records

Protected NAPTR records reference the Enumservice x-crypto type, and contain sensitive encrypted information that the publishing domain owner has made available for the querying user. The publisher and reader of private contacts establish friending relations, which get registered in the TelFriends system, and store encrypted contacts in designated sub-domains. For more information on these algorithms, please refer to the whitepaper "Privacy in .tel".

Processing of protected contacts has the following prerequisites.

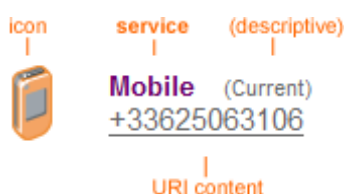
- The user is a member of the TelFriends system.
- The owner of the domain has "friended" the user, has been told the location of that user's public key, and has published encrypted information in a designated sub-domain indicated to that user.
- The application implements TelFriends login and private data decryption.
- The application has retrieved the private key for data decryption.
- The application has found this domain in its publisher list and queried the DNS with the publisher's sub-domain for this user, not the main publicly available domain.

Decryption of NAPTR records can be delegated to an external library of your choice; for instance, the Windows Mobile .tel plug-in uses the modified OpenSSL library for data decryption. You can download that library from the Tools section on the website or use the standard library and adapt to your needs.

### 5.4. Display NAPTR records

This section focuses on how contacts can be presented to the end user. Again, the section gives guidelines based on currently available applications.

A Contact has four Presentational Entities, as shown in the example:



- **Icon Element** is usually tied to the leftmost or only active Enumservice. Some clients may omit the Icon to save space.
- **Service Element** consists of location indicator hints (LIH), if any, followed by the active Enumservice(s). Recommended behaviour

is to highlight this element and clearly distinguish it from the descriptive element. The actual text to be displayed is listed in Table 1 below. See examples of service element display in Table 2.

- **Descriptive** consists of descriptive labels, if any, in parenthesis. If no labels are present, empty parenthesis should not be shown. Hyphens and colons inside a label should be replaced by a space character. Multiple labels should be concatenated with spaces between them.
- **URI content** value of the REGEXP field in the NAPTR record, displayed as a URL. Clicking this value should launch an external program designed to support the selected contact.

The client application may choose to display all NAPTR records in a unified way regardless of their privacy status, or to use different icons to show public records and private information accessible only for this user.

**Table 1: Text presented for Enumservices**

LIH	Presented Text	LIH	Presented Text
x-mobile	Mobile	x-home	Home
x-work	Work	x-transit	In Transit
x-main	Main	x-prs	Premium Rate
Active Enumservice	Presented Text	Active Enumservice	Presented Text
Sip	SIP VoIP call	ft:ftp	FTP File Server Link
h323	H.323 VoIP call	fax:tel	Fax
Voice:tel	Voice Call	x-voice:skype	Skype Voice
sms:tel	SMS Message	x-voice:gtalk	Googletalk Voice
ems:tel	EMS Message	x-im:aim	AIM IM
mms:tel	MMS Message	x-im:icq	ICQ IM
sms:mailto	SMS Message	x-im:ymsg	Yahoo! Messenger IM
ems:mailto	EMS Message	x-im:msnim	MSN IM
mms:mailto	MMS Message	x-im:xmpp	XMPP IM
Email:mailto	Email	ft:ftp	FTP File Server Link
web:http, web:https	Web Link		

**Table 2: Service Element Text Examples**

NAPTR Service Field	Service Element / Presented Text	Comments
"E2U+x-voice:skype+x-im:skype"	<b>Skype Voice &amp; IM</b>	Multiple active Enumservices.
"E2U+x-im:aim+x-lbl:ichat-video"	<b>AIM IM (iChat Video)</b>	An Active Enumservice with a descriptive label
"E2U+web:http+x-lbl:Twitter"	<b>Web Link (Twitter)</b>	
"E2U+voice:tel+x-work+x-mobile+x-lbl:Current"	<b>Work and Mobile Voice Call (Current)</b>	Two Location indicator hints and a descriptive label. Because it's an active voice Enumservice with the x-mobile LIH, the mobile phone icon should be used.
"E2U+voice:tel+x-work+x-prs+x-lbl:Work"	<b>Work and Premium Rate Voice Call (Work)</b>	Two Location indicator hints concatenated together and a

NAPTR Service Field	Service Element / Presented Text	Comments
"E2U+email:mailto+x-main+x-work+x-lbl:Primary"	<b>Main and Work email</b> (Primary)	descriptive label.
"E2U+email:mailto+x-home+x-lbl:Alternate+x-lbl:autoforwards"	<b>Home email</b> (Alternate autoforwards)	A location indicator hint and descriptive labels.
"E2U+email:mailto+x-work+x-lbl:Work"	<b>Work Email</b> (Work)	Note the LIH and label "work". Such cases are permitted.
"E2U+voice:tel+sms:tel+x-home+x-mobile+x-lbl:my-personal:phone"	<b>Home and Mobile Voice Call &amp; SMS Message</b> (my personal phone)	Voice as one of the two active Enumservices and x-mobile as one of the LIHs. The textual service label reflects the order in which these active Enumservices appear in their NAPTRs.
"E2U+sms:tel+voice:tel+x-home+x-mobile+x-lbl:my-personal:phone"	<b>Home and Mobile SMS Message &amp; Voice Call</b> (my personal phone)	
"E2U+sms:tel+voice:tel+x-home+x-lbl:hall-phone"	<b>Home SMS Message &amp; Voice Call</b> (hall phone)	The icon will be a Telephone here, even though it's not the leftmost active Enumservice.

### 5.5. Process TXT Records

Text records are processed in different ways depending on whether the record holds plain text data or structured information. Note that the order of processing records for a domain is not guaranteed, although text strings within one TXT record are processed from left to right.

- Plain-text records – no special processing, displayed as is. See section 5.6.
- Structured data records - require additional processing on a per-TXT record basis:
  - Keywords need to be processed in a special way and displayed to the reader.
  - System messages need to be processed and not displayed to the reader; only message per TXT record currently allowed.
  - Advertisements need to be processed in a special way and may or may not be displayed.

For definitions of keywords and system messages, see section 5.5.3.

The general form for structured data records is:

```
<magic> <version> <primaryType> <primaryValue> [<secondaryType> <secondaryValue>]...
magic = ".tkw" / ".tsm" / ".tad" - TXT record contains keywords/system message/ad
version = 1*2(0..9) - examples: "1", "11", "20"
```

---

IN	TXT	"Fun Contacts" "" "Life outside work"
IN	TXT	".tkw" "1" "s" "Mr" "fn" "James" "fn" "Fenimore" "ln" "Cooper" "jt" "Technical Author" "g" "male" "dob" "15/09/1789"
IN	TXT	".tad" "1" "1" "51" "Free Money" "click here to win" "uri" "http://casino7.info/roomoffer.htm" "uri" "=24342354&z" "desc" "Sierra Resorts offer" "desc" "free gaming chips for members."

---

---

```
IN      TXT      ".tsm" "1"  "pddx" "1"
```

---

### 5.5.1. Version of Structured Data

The client program is expected to be capable of handling a certain major version of this specification. To compare its capability with that indicated as required in the TXT record, it should examine the version string. This version string should be padded with "0" characters to the right if it has less characters than the client's internal capability value. Conversely, the client's internal capability value should be right-padded with "0" characters if it is shorter than the version value in the received structured data record.

Once this is done, the client should treat both its internal capability string and the extended TXT record's version string as if they were integer values, and compare them numerically. If the client program's capability is higher or equal to the version value, then this client can be expected to understand the rest of this structured data. If the first digit in the internal capability string is the same as that in the TXT record's version string, then there is major version compatibility; this means that the client will be able to parse the TXT record content, even if it does not recognise the values of the fields. If the value in the version string is higher than in the client's capability, then the client cannot make this assumption, and must discard the TXT record.

### 5.5.1. Lazy Parsing and Presentation of Structured Keywords

When a client receives a record that holds keywords, the record can be parsed by considering the initial keyword type and value, and then taking each subsequent keyword in turn. The types of these subsequent keywords can be discarded, and their keyword values can be simply concatenated (with a suitable separator between each of the strings). The result can then be presented to the user with no further interpretation; in the majority of cases, the initial keyword type will give the user enough of a semantic "hint" to understand the context and meaning of the record.

### 5.5.2. System Message Types

The system messages are passed as part of the .TSM text system message, and can include the following elements that can be updated using the updateDomain or createDomain command:

- PDDX – indicates that private data does not exist within the domain (1) or that private data is available (0). Clients use this indication as they choose. As an example, the Telnic-developed web proxy client disables the "friend" UI elements when presenting queried .tel domains that include a TXT record with this system message. The rationale for this choice is that with no private data published in this domain, there is little point in sending a friend request message to the domain owner.
- Colour1 to Colour 3 – CSS settings to display the top and title bars and the background with the specified hex colour codes. This only applies to the PC version, and is ignored on mobile browsers. The clients may wish to display or skip this information.
- CSS – number of the template to be applied to the web proxy page. Under construction.
- PSS – behaviour of the Search box on .tel pages. Under construction.

```
".tsm" "11" "pddx" "1" "color1" "" "color2" "#600b99" "color3" "" "css" "" "pss" ""
```

The older TSM version, 1, only contains PDDX. Other values are in version 1.1 onwards.

### 5.5.3. Keyword Types

The current architecture has pre-set keyword types to store typical values, such as addresses, personal details, business characteristics, etc. Sets of keywords are held inside specially formatted TXT records. Each keyword consists of a pair of elements: the keyword type in one string, with the value of that keyword held in the following string. More than one such keyword pair may be held in a single TXT record.

A set of keywords held in a single TXT record is called a group. The limit for the number of keywords held in a TXT record is indeterminate - the absolute limit is tied to the total volume of data carried. Each TXT record can in theory hold up to 64KB of data, but in practice it would be extremely unwise to send DNS messages with this amount of data. Because the keywords inside one TXT record are processed in the order they're specified, grouping keywords allowed coherent data representation with no additional effort on the client side. For instance, separating keyword types for the address components removes the need to parse comma delimited addresses.

The keyword types and values are deliberately loosely defined to allow extension and innovation by the community. The suggested keyword types provide a "core" to which others can be added in the future; for a comprehensive list of currently available keywords, please refer to the API reference, <http://dev.telnic.org/api/client-soap/keywords.html>.

All currently available keyword types are in the US-ASCII range. To limit the potential for misrepresentation, it is recommended that keyword **type** values be provisioned into the TXT record, (regardless of the language preference of the registrant or potential readers), rather than creating new language-specific keyword types. Client programs receiving these TXT records should be able to convert the types into a "local" equivalent for presentation, and provisioning systems should be able to map local keyword type names into these "canonical forms". It is assumed that the contents of the keyword **value** strings are in the Universal Character Set, and are encoded in UTF-8.

### 5.5.4. Structure of Advertisement Records

An advertisement is a TXT record with the .tad prefix, placed in the \_ad sub-domain of the current domain, such as \_ad.hotels.tel or \_ad.florida.us.hotels.tel. The record follows this structure:

```
TXT ".tad" "Version" "DisplayPreference" "Preference" "Title" "Label" "URI" "Description"
```

- Version – Currently “1”
- Display Preference – required numeric value in the range 1-3 where:
  - 1 – Indicates the advert should be displayed at the top of the page
  - 2 – Indicates the advert should be displayed on the right of the page
  - 3 – Indicates the advert should be displayed at the foot of the page
  - 4 – Indicates the advert should be displayed on the right as Related Content, after the Sponsored Links in block positioned as 2
- Preference - numeric value between 0 – 65535 that acts as the sort order for each display set.
- Title – text string up to 255 bytes long representing the title of the ad
- Label – text string up to 255 bytes long (TeProxy shows as the last line of the advert)
- URI – collection of consecutive key / value pairs (where each key is labeled “uri”) that are concatenated to produce the final URI used within the advert; this ensures that URIs in excess of 255 bytes can be created.

- Description - collection of consecutive key / value pairs (where each key is labeled “desc”) that are concatenated to produce the final description to be used within the advert; this ensures descriptions in excess of 255 bytes can be created.

An example of a .tad TXT record and its presentation on TelProxy are shown in section 5.6.

## **5.6. Display TXT Records**

TXT records holding structured keywords are considered separately from generic records and system messages. Typically, these will be spatially offset from other elements being presented – if present, keywords will be displayed after any “header” text and after the contacts.

Client programs must be ready to accept TXT records that do not match the same restrictions as those enforced by the web user interface, and provide a reasonable presentation to the user despite any difficulties they encounter.

An owner may well provision a number of keywords in a domain, and so clients must be ready to handle these without consuming a lot of screen space, whilst presenting a consistent format to users. As the order in which individual TXT records are returned in response to a DNS query is indeterminate, the content of each TXT record must be processed and presented separately. However, within a TXT record, the keywords can be presented in a consistent form by applying a small number of presentational patterns. Unless there are important reasons why this cannot be done for a particular client environment, the following rules should be used to handle groups of keywords.

Some sets of keywords have “natural” presentational forms, and these have dedicated variants on presentational patterns that should be used. For other sets of keywords, or for those keywords that are present on their own in a TXT record, standard patterns should be used.

Within a group (i.e. the keywords that are held in a single TXT record), a client should use indentation to indicate membership of the group, and to separate presentation of different groups. Where members of a group collectively occupy multiple lines, subsequent lines should be indented.

Sequences of similar keywords should be presented in a list rather than presented each on its own line, to minimise the degree of scrolling or other page navigation required of the user. Lists of similar keywords may well occupy more than one line, and where possible lines should be broken on a keyword boundary as needed. Within a list, clients should present subsequent lines indented, to show that this is a continuation of the current list. Where these items are held within a group, this indentation is in addition to any multi-line indentation used for the group itself (i.e. subsequent lines in a list should be indented relative to the line starting that list, regardless of that first line itself being within a group and so indented).

### **5.6.1. Presentation of Advertisements**

Advertisement TXT records can be ignored by the client code or displayed according to the application’s layout. On the web, the .tel advertisements can be displayed at the top or bottom of the page, or on the right (PC proxy only) with the title, label and description parsed appropriately.

Depending on the settings of the proxy page, adverts can be displayed in all 3 positions on a single page impression. The maximum number of adverts per block is currently at:

- Top: maximum 2 adverts
- Right: maximum 5 adverts
- Bottom: maximum 5 adverts

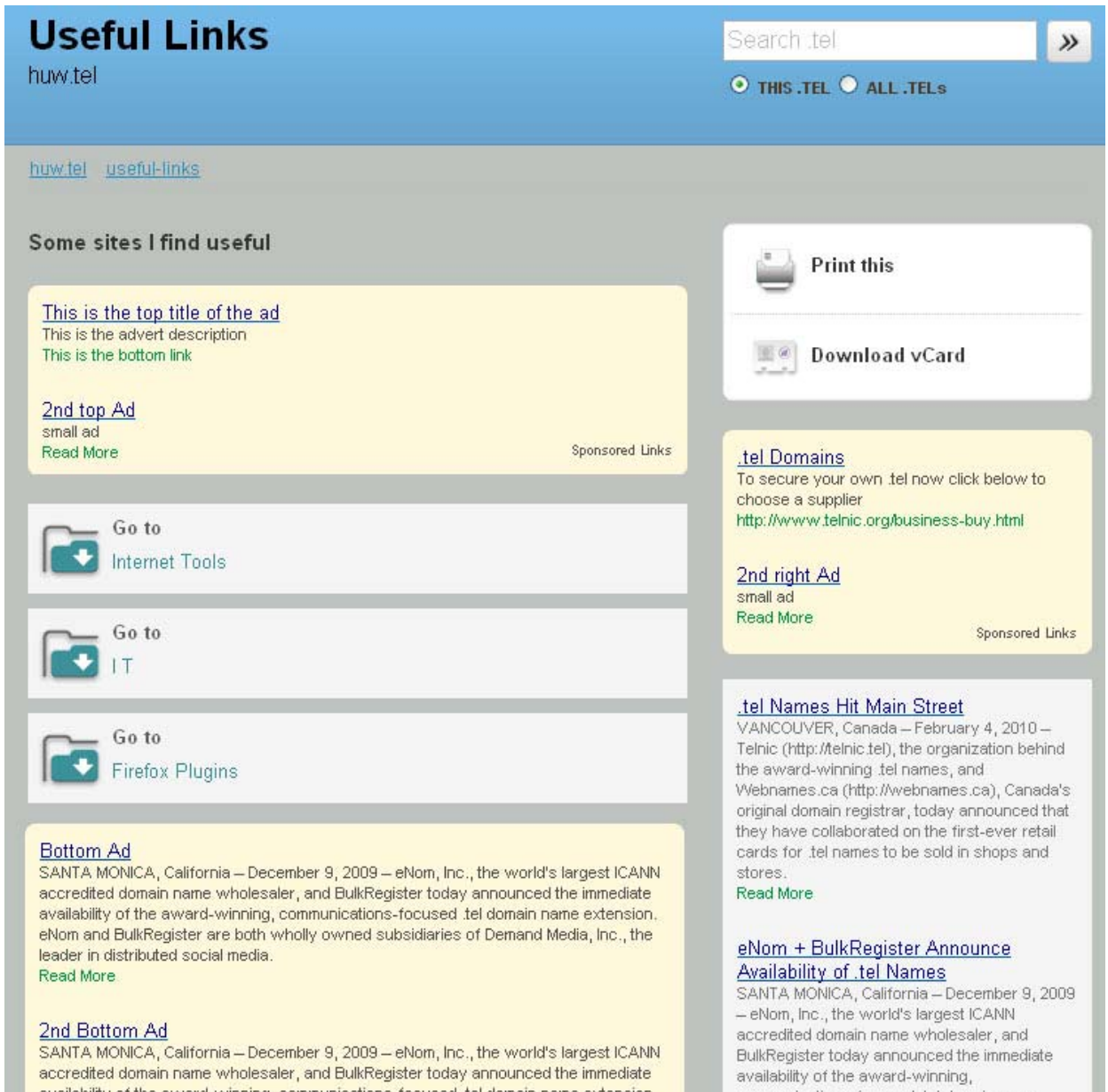
For example:

#### “Free Money

This offer bought to you by Sierra Resorts, in conjunction with Barclays Asset Management PLC. Free gaming chips with each night's stay. Offer void where prohibited by law.  
**click here to win”**

The DNS record for this text can be as follows:

```
IN TXT ".tad" "1" "1" "51" "Free Money" "click here to win" "uri"
"http://casino7.info/roomoffer.htm?x=12312&y" "uri" "=243423543&z=42342343" "desc"
"This offer bought to you by Sierra Resorts" "desc" ", in conjunction with Barclays
Asset Management PLC." "desc" "Free gaming chips with each night's stay." "desc"
"Offer void where prohibited by law."
```



The screenshot shows a web page titled "Useful Links" for the domain huw.tel. The page has a blue header with a search bar and navigation options for ".TEL" and ".TELS". Below the header, there are several sections:

- Some sites I find useful:** A list of links including "Internet Tools", "IT", and "Firefox Plugins".
- Advertisements:** Several yellow boxes containing ad text, such as "This is the top title of the ad" and "Bottom Ad" with details about ICANN accredited domain name wholesalers.
- Utility Buttons:** "Print this" and "Download vCard" buttons.
- News/Articles:** Sections like ".tel Domains" and ".tel Names Hit Main Street" with links to related content.

### 5.6.2. Keyword Label and Value Presentation

Each keyword type has a short and a long form. The long form is specified in “camel case”. A short form (such as `pc`) should be expanded into its long form, prior to further processing. The keyword type is then split into separate words the Camel case boundaries with spaces between the words, and those words are capitalised as appropriate when presented (in this case, `pc` would be converted to its long form equivalent `postalCode`, and then that would be further capitalised and split into words to form “Postal Code”). If the keyword type string is not recognised as a listed short form, it is processed with the same capitalisation and space insertion rules as for the long form.

In many cases, sets of text that are presented to a user need to be distinguishable from one another. Typically, this is done by emphasising one of the texts, whilst leaving the other in its regular form. Where possible, this can be done by presenting text to be emphasised in Bold face, and presenting other texts in plain / Regular face. Where this is not possible, other styles may be used; for instance, if the client device does not support Bold face, Italic text can be used for emphasized text.

Another alternative is to use different density (so that text to be emphasised might be in black, whilst other text would be in grey), or even using different colours. Differentiating between texts using density or colour variations is not recommended unless there is a clear distinction between the text elements, and they are all easy to read. Low contrast (in the case of density variation) or poor colour display (in the case of colour differentiation) may make it much harder for a user to understand (or even read) the data presented to him or her.

Finally, some client implementations have chosen to hide the keyword type or label, presenting it only as a “roll over” or “hover” pop up item. Whilst this may be acceptable for other uses of text within a domain (such as the plain text header elements, if these exist), it has caused significant usability problems when applied to keywords. Users expect to be able to glance at a page and know what each text item represents, and this is not possible for individual keyword types if the labels for those types are not shown. Thus, the user of “hover” pop up items for keyword labels is strongly discouraged. The saving in “screen real estate” comes at an unacceptable cost to user understanding.

For patterns of displaying keyword types and groups of keywords, see Appendix B.

## 6. PRIVACY-RELATED OPERATIONS

### 6.1. TelFriends Authentication

To look up and display private data, "friend" users, and perform all other privacy-related operations, the user needs to be a member of the TelFriends database. The sponsoring organization has a database of users independent of the TelHosting Provider accounts. Each domain owner has an account in the database, and users who don't have .tel domain names but want to view .tel private data can have guest accounts. For more information, please refer to "Privacy in .tel".

To perform privacy-related operations, the client needs to implement the TelFriends login procedure and be able to store TelFriends credentials locally and securely. The log-in procedure can happen when the application starts for the first time, during cold boot, or later, depending on the client implementation.

In the currently available Blackberry client, the cold boot procedure includes TelFriends authentication and goes as follows (also illustrated by Figure 3):

1. The client requests the user's web user name and web password.
2. The client calls the `getChallengeQuestion` operation to obtain the challenge question. It presents the question to the user.
3. The user enters the answer to the question.
4. The client uses the `getAPICredentials` operation to obtain:
  - o the user's TelFriends ID
  - o the user's key pair (encrypted using PKCS#5 PBE with the web password)
  - o the user's API password (encrypted with his public key)
5. The device decrypts the key pair, using the web password the user entered before.
6. The device uses the user's private key to decrypt the API password.
7. The device stores the TelFriends ID and the decrypted API password

All other SO-related operations use the TelFriends ID and API password obtained during the authentication. The client application needs to be able to store those "API credentials" securely. For details on how to use the `getChallengeQuestion` and `getAPICredentials` functions, please refer to the TelFriends API reference.

Guys, I finally cracked the TelFriends cold boot using nothing other than OpenSSL C code. Below is the relevant sample code. I assume that one has already done the SOAP requests to grab the challenge question, and then submit the response.

David, I think you could technically use substantially the same code in Perl to avoid using the filesystem if you really wanted to.

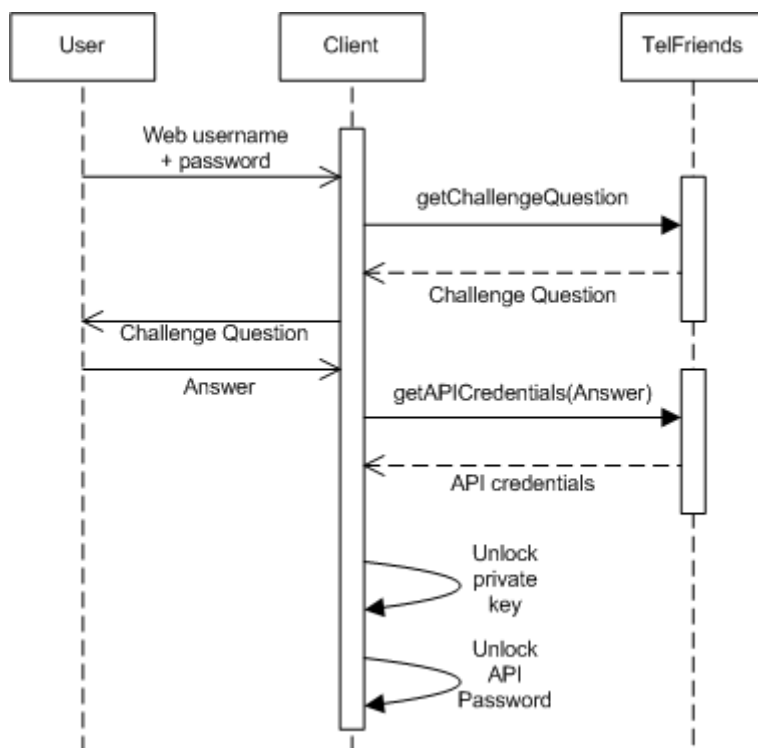


Figure 3: Cold Boot Procedure

**Sample Code: Blackberry Client Unlocking Private Key**

Taken from the code for the plug-in for Blackberry OS devices,  
 org.telnic.blackberry.util.crypto.DecryptionTools.java.

```

byte[] bytes = keyString;

AsymmetricCipherKeyPair keyPair = PKCS5PemReader.decryptPrivateKey( bytes,
    Options.getInstance().getOption( Options.SO_PASSWORD ).toCharArray() );

if (keyPair == null)
{
    // problem extracting key
    throw new CryptoException( TelnameApp.getResourceBundle().getString(
        TelnameAppResource.COLD_BOOT_INTERNAL_ERROR ) );
}

CipherParameters params = keyPair.getPrivate();

if (params instanceof RSAPrivateCrtKeyParameters)
{
    RSAPrivateCrtKeyParameters privKey = (RSAPrivateCrtKeyParameters) params;
    // Store the key in the cache
    RSAKeyCache.setKey( privKey );
}
else
{
    returnMsg = "Unexpected Private Key Type";
}
    
```

**Sample Code: C code to decrypt the private key with the web password & API password with key**

```
// Assumptions: Encrypted private key is inside
// a buffer "encPKBuf" with length "encPKBufLen"
// Encrypted API password is inside
// buffer "encPassBuf" with length "encPassBufLen"

OpenSSL_add_all_algorithms();
BIO *mem = BIO_new_mem_buf(encPKBuf, encPKBufLen);
EVP_PKEY *evpKey = d2i_PKCS8PrivateKey_bio(mem, NULL, NULL, "[WEB
PASSWORD]");
if (evpKey == NULL) {
    // Error
    return NULL;
}
//Extract the rsaKey from the EVP structure
RSA *rsaKey = EVP_PKEY_get1_RSA(evpKey);
uint8_t *bufferPtr = NULL;
size_t bufferPtrSize = 0;
bufferPtrSize = RSA_size(rsaKey);
bufferPtr = malloc( bufferPtrSize * sizeof(uint8_t));
memset((void *)bufferPtr, 0x0, bufferPtrSize);

int movedBytes = RSA_private_decrypt(encPassBufLen, encPassBuf,
    bufferPtr, rsaKey, RSA_PKCS1_PADDING);
if (movedBytes == -1) {
    // ERROR
    return NULL;
}
// Done! API Password is inside "bufferPtr" with length "movedBytes"
```

## 6.2. Friending Message Processing

To establish a connection, two TelFriends members exchange friending messages: a request and a response. When a user receives a friending request, they accept or deny; accepting a request establishes a one-way friending connection between two users, which enables the publishing user to share private data with the reader. Two-way friendship relations are also possible.

In addition to the signature, target user, and free-form text, friending messages hold system information, such as the location of private keys for decryption, and the address of the subdomain with private data. These encryption-related data should be processed but hidden from the user.

Processing friending messages has the following prerequisites:

- The user has an account in the TelFriends member system.
- The client application implements the TelFriends login procedure.
- The client application implements generation and retrieval of friending messages. Preferably, the client application also implements caching of old messages, so that only new ones are retrieved.
- The client application implements public key retrieval and storage.

Figure 4 shows the interaction between the user, plugin, TelFriends and NSP when accepting a friending request. This scenario assumes that among the new retrieved messages, there's a friending request message that the user accepts. Based on this, the application creates a new friend.

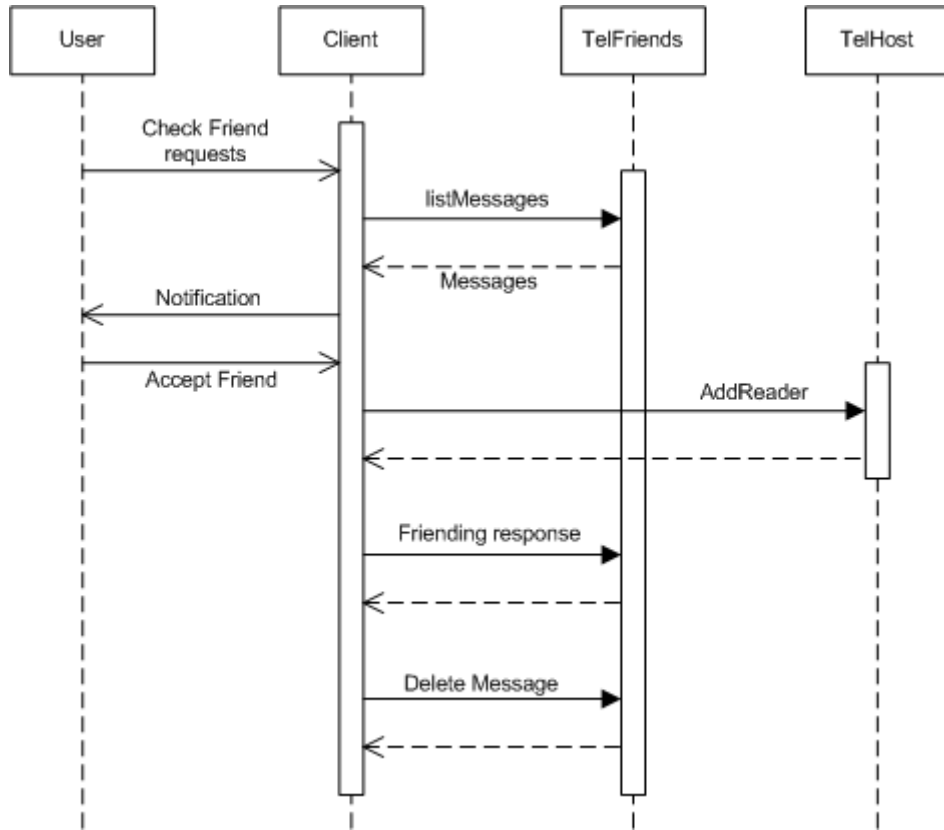


Figure 4: Accepting a Friending Request

**Sample Code: Blackberry Client Accepting a Friending Request**

Taken from the code for the plug-in for Blackberry OS devices,

org.telnic.blackberry.util.xml.TelnicWebServices.java

```

public void acceptRequest (final BlackBerryUICallback ui, final Message
message)
{
    Runnable r = new Runnable()
    {
        public void run ()
        {
            try
            {
                Logger.logMessage( "Accepting friend request",
Options.KEY_LOG_MESSAGE );
                TelnicWebServicesReference.obtainNspCredentials(
message.getTo(), sw, ui );
                // calculate the label here
                String label = TelnicWebServicesReference.generateHash(
(String) message.getTo(), (String) message.getFrom() );
                boolean success = NSPWebServices.addReader( ui,
message.getTo(), message.getFrom(), label, message.getKeydomain(), sw);
                if (!success)
                {
            
```

```
        return;
    }
    String storedLabel = "";
    if (!sw.isCancelled())
    {
        // handle label conflicts in here
        if (sw.getServerString().indexOf( "reader" ) >= 0
            && sw.getServerString().indexOf( "already exists" ) >
0)
        {
            Logger.logMessage( "Reader already exists",
Options.KEY_LOG_MESSAGE );
            // means we have a label conflict, as we have already
added this

            // user - need to fetch the label already assigned
            Hashtable readers = NSPWebServices.listReaders( ui,
(String) message.getTo(), sw );
            if (readers == null)
            {
                return;
            }
            Enumeration e = readers.keys();
            while (e.hasMoreElements())
            {
                String domain = (String) e.nextElement();
                if (domain.equals( message.getFrom() ))
                {
                    //for the specified reader, get the details.
                    storedLabel = NSPWebServices.getReader( ui,
(String) readers.get( domain ), message.getTo(), sw );
                    if (storedLabel == null)
                    {
                        return;
                    }
                    if (storedLabel.equals( "" ))
                    {
                        return;
                    }
                }
            }
        }
    }
    // if we haven't obtained a label from the server, use the
one we made
    if (storedLabel.equals( "" ))
    {
        storedLabel = label;
    }
    // send response message to the reader
    success = SOMemberWebServices.sendFriendResponseMessage( ui,
storedLabel,
(String) message.getFrom(), (String) message.getTo(),
(String) message.getConfirmationcode(), sw );
    if (!success)
    {
        return;
    }
    // finally delete the message
    success = SOMemberWebServices.deleteMessage( ui, (String)
message.getId(), sw );
    if (!success)
    {
```

```
        return;
    }
    // return
    if (!sw.isCancelled() && ui != null)
    {
        final Object[] temp = { new Boolean( false ),
            new
Integer(TelnicWebServicesReference.SOAP_CALL_ACCEPT_REQUEST) };
        ui.callback( temp );
    }
}
catch (Exception e)
{
    Logger.logMessage( "Exception: " + e.toString(),
Options.KEY_LOG_MESSAGE );
    sw.internalError( sw, ui );
}
}
};
final Thread t = new Thread( r );
t.start();
}
```

### Sample code: Accepting a Friend Request

```
my $messages = listMessages($DomainRegistered);
die "No messages" unless ($messages);

# Smush it into an array if it isn't already..
if (ref($messages->{'message'}) ne 'ARRAY') {
    my $temp = $messages->{'message'};
    $messages->{'message'} = [$temp];
}

# In an application, this would be a persistent store
my @message_store = ();

foreach my $message (@{$messages->{'message'}}) {
    my $mesg = getMessage($message);
    push @message_store, $mesg;
}

# Let's accept the first friend in the list
my $friendRequest = $message_store[0];
print Dumper $friendRequest;

if ($friendRequest->{'valid'}) {
    $friendRequest->{'userfrom'} = getUsername($friendRequest->{'from'});
}
else {
    die "Request is not valid..\n";
}

# Attempt to accept a friend request..
print "- A C C E P T I N G   F R I E N D   R E Q U E S T - - - - -\n";
-\n";
```

```
my $readerInfo = createReader(@{$friendRequest}{qw/to userfrom from keydomain
confirmationcode/});

my $cool = sendFriendResponse(
    $readerInfo->{'label'},
    $friendRequest->{'from'},
    $friendRequest->{'to'},
    $friendRequest->{'confirmationcode'}
);

if ($cool) {
    print $friendRequest->{'userfrom'}." is now your friend\n";
    deleteMessage( $friendRequest->{'id'} );
}
else {
    die "Friending was unsuccessful this time\n";
}
```

See the TelFriends and TelHosting API references for detailed descriptions of individual functions. Special cases:

- To reject a friending request, the message is simply deleted from the SO. The user who sent the request does not need to be informed of the rejection.
- To accept an unsolicited friending response, the domain contained in the message is added to the user's publishers list against the sender of the response. An unsolicited response is a message not sent in response to a request and lacking the private code associated with the original request. The client application may or may not distinguish between normal and unsolicited responses.
- To reject an unsolicited friending response, the message is deleted from the SO.

### **6.3. Adding a Friend**

The prerequisites for this task are the same as in section 6.2.

Figure 5 shows the interactions between the user, plugin and TelFriends when adding a friend. A part of the friending procedure is the same as in section 6.2, and consists of friending message exchange. When a positive response to a friending request is received, the sub-domain supplied in the response is added to the user's publisher list.

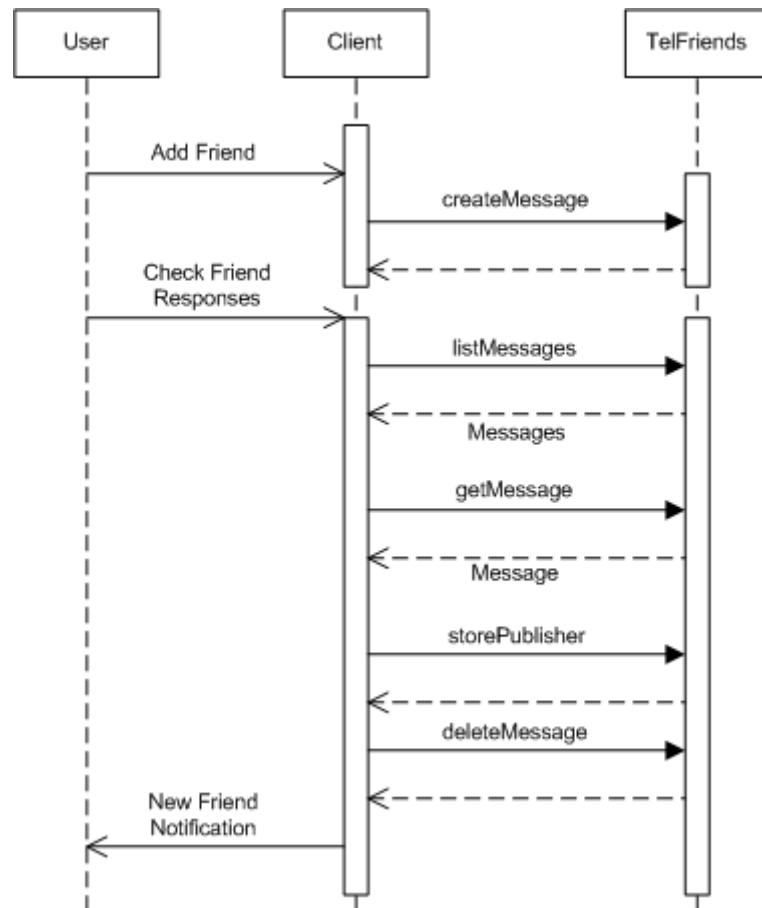


Figure 5: Friending Procedure

**Sample Code: Blackberry Client Adding a Friend**

Taken from the code for the plug-in for Blackberry OS devices,  
 org.telnic.blackberry.util.xml.SOMemberWebServices.java.

```

//the message
final String msg = TelnicWebServicesReference.SOAP_MESSAGE_HEAD +
"<ns8:createMessageRequest>" + "<ns8:from>" + fromMOD
    + "</ns8:from>" + "<ns8:to>" + toMOD + "</ns8:to>"
    + "<ns8:contentType>application/xml+soap</ns8:contentType>"
    + "<ns8:messageType>friendRequest</ns8:messageType>" +
"<ns8:text>"
    + TelnicWebServicesReference.encodeString(textToSend) +
"</ns8:text>" + "</ns8:createMessageRequest>"
    + TelnicWebServicesReference.SOAP_MESSAGE_TAIL;
tw.setRedirect( 0 );
//send the message
final SoapEnvelope sp = tw.sendSoapMessage( msg, "", "createMessageRequest",
    TelnicWebServicesReference.SO_ACCESS );
if (sp == null)
{
    //if soap envelope is null, return error
    tw.noResponse( tw, ui );
    return false;
}
    
```

To remove a friend, the domain can simply be removed from the publishers list. There is no need to send any messages to the friend to inform them.

## 7. MANAGING A .TEL DOMAIN

Client applications can implement manipulation of data published in a .tel domain. If you do not plan to allow users to manage their .tels with your application, you do not need to implement this. You can also implement a simple redirect to Telnic's web interface, so that users selecting "manage your .tel" get redirected to the website and perform updates there. A combination of these approaches is also possible, as with the Blackberry plug-in, which allows profile switching and links to the web interface for other operations.

### 7.1. TelHosting Authentication

To perform any operations on the domain object, the application needs to implement the following:

- Logging in to TelHosting Provider account
- Storage of TelHosting Provider credentials and retrieval of new credentials if they have been changed outside the client application
- Support for switching the Provider and for multiple providers for multiple domains

### 7.2. Updating a .tel domain

This section shows how to publish a new record in a .tel domain and how to edit an existing one.

The current TelHosting Client SOAP API offers the `storeRecord` function, which allows creating a new record or overwriting an existing one, if an existing record ID is specified. Alternatively, you can use the `updateRecords` operation to change one or multiple records grouped by the selected criteria.

A record can be deleted by using the `deleteRecord` operation or by specifying the `<delete>` or `<deleteAll>` element in an `updateRecords` operation.

### 7.3. Adding Private Data

Users often want to limit access to sensitive information published in a .tel domain. To support publication of private data, your application needs to implement SO-related operations, see section 6.

In order to enter private data, your application should create or use existing readers (people who can see the private data) and groups (sets of people and data that they can see). When your application puts records in a group, they automatically become private and can only be seen by the readers who are members of this group.

### 7.4. Profile Switching

Profiles are sets of records that the user can manage as a group. For instance, you can have a "work" profile with your business contacts, and a "home" profile for information on how to reach you off hours. Switching between profiles in real time allows the instantaneous shift of a whole set of contacts, which can be very handy.

Figure 6 shows how the profile switching goes in the current version of the .tel plug-in for Blackberry devices. This client application retrieves the list of currently available profiles, identifies the current one,

and presents the information to the user. The user selects the profile to switch to, and the application sends the corresponding request to the TelHosting Provider.

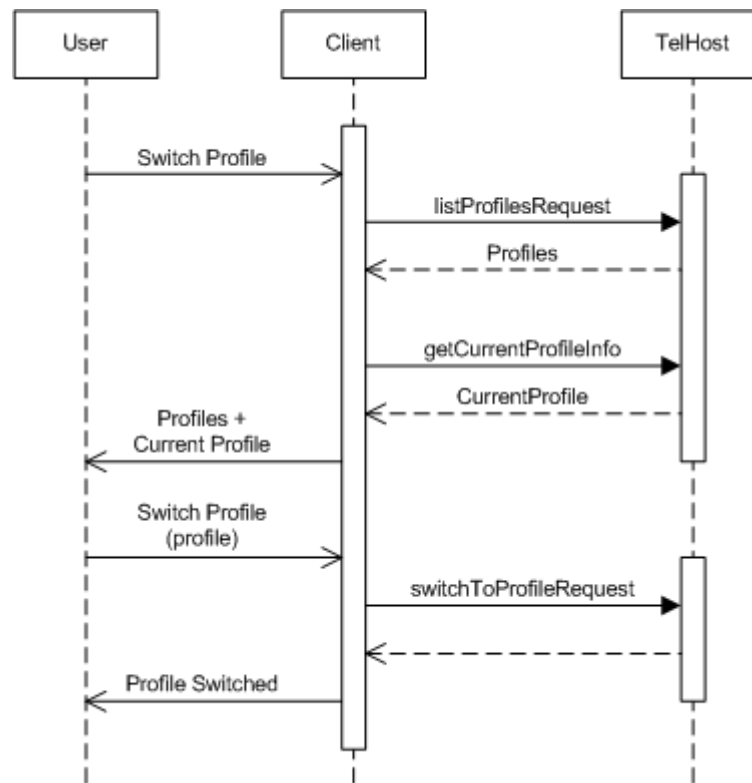


Figure 6: Profile Switching

### Sample Code: Blackberry Client Profile Switching

Taken from the code for the plug-in for Blackberry OS devices,  
org.telnic.blackberry.util.xml.NSPWebServices.java.

```
protected static Vector listProfiles (final BlackBerryUICallback ui, final
String domain,
    final SoapWorker tw)
{
    try
    {
        Logger.logMessage( "Getting profile list",
Options.KEY_LOG_MESSAGE );
        tw.setSoapType(
TelnicWebServicesReference.SOAP_CALL_LIST_PROFILES );

        if (!tw.checkCoverage( tw, ui ))
        {
            return null;
        }

        // list profiles message
String msg = TelnicWebServicesReference.SOAP_MESSAGE_HEAD
    + "<ns7:listProfilesRequest effUserName=\"\" domainName=\"" +
```

```
domain + "\" />"
    + TelnicWebServicesReference.SOAP_MESSAGE_TAIL;

tw.setRedirect( 0 );

// send the message
SoapEnvelope sp = null;
sp = sendMessage(tw, msg, "listProfilesRequest");

if (sp == null)
{
    // if soap envelope is null, return error
    tw.noResponse( tw, ui );
    return null;
}

// process response
final Vector ids = new Vector();
if (sp.getProfileIds() != null)
{
    // extract the ids from the space separated string
    final String profileString = sp.getProfileIds();
    final String Splitter st = new StringSplitter( profileString,
" " );

    while (st.hasMoreCards())
    {
        ids.addElement( st.nextElement() );
    }
    return ids;
}
catch (Exception e)
{
    Logger.logMessage( "Exception: " + e.toString(),
Options.KEY_LOG_MESSAGE );
    tw.internalError( tw, ui );
    return null;
}
}
```

## 8. OTHER FEATURES

### 8.1. Address Book Integration

Some of the existing plug-ins get integrated into the address book of the host device or application. Implementing the integration functionality allows construction of a dynamic address book with contents retrieved from .tel domains.

Your application may simply add a .tel field in the address book or implement full-scale integration with the ability to compare entries from the static address book and freshly retrieved .tel domain data. You may also construct your own .tel address book that stores only .tel domain contacts. To add or update data of the address book, you need to perform a lookup of the specified domain, and implement a comparison algorithm. See the Office Outlook plug-in for an example of native address book integration.

### 8.2. Auto-provisioning

Auto-provisioning is the mechanism by which client software can configure TelHosting settings based on the content of a .tel domain. This way, users won't need to provision the TelHosting system connection information into a freshly installed .tel client program, and won't need to type in SOAP URLs to run TelHosting-related actions. Instead, users type in the name of their domain and supply TelHosting credentials, and the client application gets configuration settings from the domain contents.

This operation requires that the user has a .tel domain, and the client application supports logging into TelHosting Software.

The currently available client applications look at contacts within two reserved sub-domains within a .tel domain name: `_soap._nspapi` and `_http._nspapi` or `_https._nspapi`. The contacts expected in these sub-domains are shown below, and allow the client programs to find the SOAP end point and the human-readable web pages respectively. In this example, the TelHosting system hosts the zone of `example.tel`, and will provision these contacts as shown:

```
$ORIGIN example.tel.  
  
.      IN SOA ...  
  
.      IN NS ...  
  
_soap._nspapi  
IN NAPTR 10 50 "u" "e2u+web:https+x-lbl:1-3"  
"!^.*$!https://exampletelhost.com:8254/action!" .  
IN NAPTR 10 51 "u" "e2u+web:https+x-lbl:1-1"  
"!^.*$!https://exampletelhost.com:8253/action!" .  
  
_https._nspapi  
IN NAPTR 10 50 "u" "e2u+web:https"  
"!^.*$!https://www.exampletelhost.com/login!" .
```

Client programs need to look for contacts in these sub-domains when they are given the .tel domain name. If these are not found or the contacts within them are not usable, then the client program needs to inform the end user that auto-provisioning is not available, and ask for the settings to be entered manually.

Note that the ORDER/PRIORITY field in a NAPTR record reflects the TelHosting Provider's preference. The client may well have its own preference, based on the API version supported. Thus when a client program finds an appropriate contact, the client program looks for an x-1b1 auxiliary Enumservice, and if detected, will match the x-1b1 content (if present) against its own capabilities.

### **8.3. Search for .tel domains**

The .tel domain is a unique offering over and above traditional directory services in that it provides a global presence. Keywords can be placed under each sub-domain to reflect regional differences in how that part of business would like to be indexed. Addresses, keywords and service areas could be localized effectively. This goes some way to alleviating some of the issues with internationalisation.

The Tel-List search scheme uses indices of data gleaned from keyword information published in registered .tel domains. A central system queries each such registered domain regularly, incorporating any keyword data it finds into a search index it maintains. This scheme provides a service by processing search request messages sent from client programs or associated sub-systems, returning an ordered list of .tel domains that best fit the requested criteria. The service uses SOAP over HTTP as its protocol. The querying API allows clients to apply limits to the number of search results, in order to give reasonable message sizes and optimize network performance.

In addition to the keyword data published in a registered domain, a separate “auxiliary text” record can be stored. This will be retrieved at the same time as any keyword data but will not be used in generating the search indices. It can be returned along with the associated domain name in which it was found in response to search requests that match that domain. The client programs and associated sub-systems will also present the keyword (and auxiliary text) data found when executing a direct domain lookup query. This provides “sideways” search functionality.

You may decide to support the search feature in the full, or to use the TelFriends built-in search front-end TelPages, and display that web interface to the user.

With the Tel-List API, the search procedure should go as follows:

1. The user enters the search data.  
Depending on the implementation, your client may support simple queries or more complex ones, which field types and exclude patterns.
2. The client initiates the search by sending a SOAP `searchRequest` message to the SO; see the TelFriends API reference for details on using the search function.
3. The TelFriends returns the search results, if any, sorted by relevance.
4. The application presents the results to the user. Each returned .tel domain should be clickable and trigger the lookup operation.
5. Optionally, the client application may store search history.

The search API allows the application to set the number of search results using the `maxResults` element. This can come in handy for mobile applications, which are limited by the small screen size. Depending on

the implementation, the number of search results can be restricted when performing the search, or in the preferences, as a static value.

#### **8.4. Exporting / Importing Data**

Every accredited TelHosting Provider must support exporting and importing all of a user's data. This is to ensure that .tel owners are independent and can switch TelHosting providers and registrars as needed.

The format of these files is specified, so it would be possible to develop an “offline” client that manipulated these files and re-imported them to the appropriate TelHosting system.

#### **8.5. Preferences**

Some of the current application support user preferences, which store custom application settings. As an example, the Outlook plug-in allows the user to set the TelFriends and TelHosting credentials via the Preferences, and to specify the applications to be launched for various communication sessions.

## 9. REFERENCES

1. TelHosting Client SOAP API Reference, <http://www.dev.telnic.org/api/client-soap/index.html>
2. TelHosting Admin SOAP API Reference, <http://www.dev.telnic.org/api/admin-soap/index.html>
3. Sponsoring Organization SOAP API Reference, <http://www.dev.telnic.org/api/so-soap/index.html>
4. RFC1035, <http://tools.ietf.org/html/rfc1035>
5. RFC1876, <http://tools.ietf.org/html/rfc1876>
6. RFC3403, <http://tools.ietf.org/html/rfc3403>
7. IANA Registry website, <http://www.iana.org/assignments/enum-services>
8. "NAPTR Records in .tel", <http://www.dev.telnic.org/docs/naptr.pdf>

## 10. APPENDIX A: CONTACTS DISPLAY IN CURRENT .TEL CLIENT APPS

In the table of examples shown next, the client is expected to process NAPTRs that include Enumservices as shown, and with the URL generated from the REGEXP field in the form shown in the NAPTR URI column.

### 10.1. Windows Contact Handling

Service	NAPTR URI	Launch URI	Supported Clients
x-im:aim	aim:freddy	aim:goim?screenname=freddy	AOL Messenger
x-im:ymsg	ymsg:freddy	Ymsg:sendim?freddy	Yahoo! Messenger
x-im:msnim	msnim:freddy	Msnim:freddy	MSN Messenger
x-voice:sip, voice:sip, sip	sip:freddy@sip.com	n/a - cmd - dial=freddy@sip.com	X-Lite
voice:tel	tel:+441794833000	n/a - Bluetooth dialer	Bluetooth phone
		n/a - cmd - dial=00441794833000	X-Lite
		Skype:+441794833000? call	Skype
x-im:skype	skype:freddy	Skype:freddy?chat	Skype
x-voice:skype	skype:freddy	Skype:freddy?call	Skype
email:mailto, sms:mailto	mailto:freddy@foo.com	mailto:freddy@foo.com	(System)
ft:ftp	ftp://ftp.foo.com/	ftp://ftp.foo.com/	(System)
web:http,	http://www.foo.com/	http://www.foo.com/	
web:https	https://www.foo.com/	https://www.foo.com/	(System)
sms:tel, ems:tel, mms:tel	tel:+447794833000	n/a - Bluetooth SMS sender	Bluetooth phone

## 10.2. Web-based Contact Handling

Service	NAPTR URI	Launch URI/ Link Generated	Supported Clients
x-im:aim	aim:freddy	aim:goim?screenname=freddy	AOL Messenger
x-voice:aim	aim:freddy	- none -	- none -
x-im:ymsgr	ymsgr:freddy	Ymsgr:sendim?freddy	Yahoo! Messenger
x-voice:ymsgr	ymsgr:freddy	- none -	- none -
x-im:msnim	msnim:freddy	Msnim:chat?contact?freddy	MSN Messenger
x-voice:msnim	msnim:freddy	Msnim:chat?voice?freddy	MSN Messenger
x-voice:sip, voice:sip, sip	sip:freddy@sip.com	sip:freddy@sip.com	
voice:tel	tel:+441794833000	callto:+441794833000	Skype
x-im:skype	skype:freddy	Skype:freddy?chat	Skype
x-voice:skype	skype:freddy	Skype:freddy?call	Skype
email:mailto, sms:mailto	mailto:freddy@foo.com	mailto:freddy@foo.com	(System)
ft:ftp	ftp://ftp.foo.com/	ftp://ftp.foo.com/	(System)
web:http,	http://www.foo.com/	http://www.foo.com/	
web:https	https://www.foo.com/	https://www.foo.com/	(System)
sms:tel, ems:tel, mms:tel	tel:+447794833000	- none -	- none -

## 11. APPENDIX B: TXT DISPLAY IN CURRENT .TEL CLIENTS

### 11.1. Patterns of presentation for Keywords

In following, \*xx\* means emphasise (present in Bold face), whilst <xx> means present in Regular face. In these examples, kt means the content of the keyword type string, whilst kv means the content of the keyword value string.

There are two main patterns for presentation of keywords, each with its own variation for presentation of specific keyword types. These are the indented pattern (with the postal address variant) and the concatenated pattern (with the name and business area variants).

#### ***Indented pattern***

```
*kt1* <kv1>  
  
    *kt2* <kv2>  
  
    ...           ...  
  
    *ktn* <kvn>
```

Note: Where a TXT record contains a single keyword type/value pair or this keyword must be processed separately, it is treated as a degenerate form of this indented pattern (using just the first line of this pattern).

#### ***Concatenated pattern***

```
*kt1* <kv1>, <kv2>, ..., <kvn>
```

Notes:

- In this pattern, only the first keyword type is presented. Subsequent keyword types are suppressed.
- Any sequential elements of the same type in a keyword record should be concatenated with no joining characters. Differing types should be concatenated with comma-space ', ' for example:

```
'ft' 'string 1.' 'ft' 'string 2.' 'ft' 'string 3.' Will be shown as:  
"string1.string2.string3."
```

```
"o" "Telnic Ltd" "d" "Engineering" "jt" "Engineering Director"  
Will be shown as: "Telnic Ltd, Engineering, Engineering Director"
```

### 11.2. Use of keyword presentation patterns

The choice of which presentation pattern to use is deterministic. There are several cases in which dedicated patterns should be used, as the data to be presented is usually shown in a particular way (for example, when shown on business cards or web pages). Those patterns are listed in the following section.

Apart from these cases, it is important for clients to use the standard patterns listed above, so that the querying end user will receive a consistent result. If a particular client platform uses its own user interface

guidelines, then adherence to those guidelines may outweigh the rules specified here. However, such variance from these specifications should be the exception, and where possible these rules should be used.

The rules for processing and presenting keyword data using the patterns shown in this document are:

- If a group holds more than one of the “address-related” keywords types pa (or bpa), a1, a2, a3, tc, sp, or c, the client should present this group using the dedicated Postal Address variant shown below.
- If a group holds “name-related” keyword types s, fn, ln, cn, nn, or bn (in the case of a business name), the client should present this group using the dedicated Name variant (see below).
- Where a group holds “business type” keywords in a usual sequence (bar, followed by one or more keywords each of type bsa) the dedicated Business Area variant shown below should be used.
- Where a sequence of keywords within a group has the same keyword type, the client SHOULD present this sequence using the concatenated pattern (see above).
- Otherwise, the client SHOULD present a group of keywords using the indented pattern (see above).

### **11.3. Pattern Variants for Specific Keyword sets**

#### ***Postal Address variant***

```
*label*      <kv2>
              <kv3>
              ...
              <kvn>
```

Notes:

- Where a TXT record holds a keyword of type postal address, or type business postal address, then the value of that postal address (or business postal address) keyword should be used as the label. If a TXT record does NOT hold such keyword types but DOES have more than one of the other “address-related” keyword types, a default label of “Address:” should be used.
- If keywords other than these “address-related” ones follow on from them in a group, then such keywords should be collated and processed on a new line following on from the postal address related set, using the appropriate pattern for those keywords.

#### ***Business Area Variant***

```
*kv1* ; <kv2>, <kv3>, ..., <kvn>
```

This variant is similar to the standard concatenated pattern, but the label chosen is different (the label is the business area value string), and the client should apply “; ” (Semi-Colon Space) as the concatenation pattern between the business-area value and the following business-sub-area value or values. The client will combine the subsequent (business-sub-area) keywords with the normal (“, “) concatenation pattern.

Notes:

- In this variant, the value of the business-area keyword is emphasised (presented in Bold), whilst the keyword type text itself (“business-area”) is suppressed. Similarly, the subsequent keyword types (all of which must be “business-sub-area”) are suppressed.
- If other keyword types follow on from the business sub-area set in the same TXT record, then these must be processed separately (on another line).

**Name variant**

\*label\* <kv1> <kv2> ... <kvn>

Notes:

- If a group includes the keyword type business name then the label used is the value string of that keyword. If such a keyword type is NOT present in the group containing a set of more than one name-related keywords s, fn, ln, nn, or cn, an explicit label of “Name:” is used instead.
- If cn exists as well as at least one of s, fn, ln or nn, it will be presented on a separate line. If it is not the first “name-related” item in the group, it indented to the same level as the first value.
- If there are other keyword types that follow on in the same group as the “name-related” ones, these are collated and presented starting on a new line after the name-related entries. For example, in the specific case in which a group has name-related types, but also has a sequence of “non-name related” keywords that all share the same keyword type (or have the specific business-area / business-sub-area sequence of types), this sequence will, in turn, be processed using the concatenated pattern or its business-area equivalent.
- The value string for the keyword type nickname should be presented within quotes.

**11.4. Keyword Presentation Examples**

The table below shows groups of keywords, their suggested representation pattern and actual representation.

Keyword group	Pattern	Representation
o:Telnic Limited, jt:Rocket Scientist,g:m, dob:22/11/1963	Indented	<b>Organisation:</b> Telnic Limited <b>Job Title:</b> Rocket Scientist <b>Gender:</b> Male <b>Date of Birth:</b> 22/11/1963
nn:Ratso	Indented with Nickname processing	<b>Nickname:</b> “Ratso”
toyCollections:Thunder birds	Unknown keyword type processing	<b>Toy Collections:</b> Thunderbirds
hi:scouting, hobbiesInterests:bikes	Concatenated with short and long keyword type processing	<b>Hobbies Interests:</b> scouting, bikes
hi:Skydiving, hi:Rock Climbing, hi:BASE Jumping, hi:Extreme	Concatenated with multi-line	<b>Hobbies Interests:</b> Skydiving, Rock Climbing, BASE Jumping,

Skiing, hi:Snowboarding, hi:Sky-Sea Parachuting, hi:Paracending, hi:Microlight flying	folding	Extreme Skiing, Snowboarding, Sea-Sky Parachuting, Paracending, Microlight flying
s:Mrs., fn:jane, ln:SMITH, cn:Jane p. Atwood	Name	<b>Name:</b> Mrs. jane SMITH Jane p. Atwood
cn:jane p. atwood, s:mrs., fn:jane, ln:smith	Name	<b>Name:</b> jane p. Atwood mrs. jane smith
dob: 19490302, cn:jane p. atwood, s:mrs., fn:jane, nn:crusher ln:smith, hi:hiking, hi:diving	Name with subsequent Concatenation	<b>Name:</b> jane p. atwood mrs. jane “crusher” smith <b>Date of Birth:</b> 19490302 <b>Hobbies Interests:</b> hiking, diving
bar:automotive, bsa:retail, businessSubArea:repair s, bsa:sales	Business Area	<b>Automotive;</b> retail, repairs, sales
a1:37 Percy Street, a2:Fitzrovia, tc:London, pc:W1T 2DJ, c:UK	Address Pattern Variant without postal address type	<b>Address:</b> 37 Percy Street Fitzrovia London W1T 2DJ UK
bpa:London HQ, a1:37 Percy Street, tc:London, pc:W1T 2DJ, c:UK	Address Pattern Variant with postal address type	<b>London HQ:</b> 37 Percy Street London W1T 2DJ UK