



# Beispielprodukt

**-Systemspezifikationen: SW-Spezifikation-**

**TelData**

**Version: A-Muster**

Projektbezeichnung	Artio	
Projektleiter	Herr Karlapp	
Verantwortlich	Herr Deynet	SW-Architekt
Erstellt am		
Zuletzt geändert	19.06.2007 16:34	
Bearbeitungszustand	<input type="checkbox"/>	in Bearbeitung
	<input type="checkbox"/>	vorgelegt
	<input checked="" type="checkbox"/>	fertig gestellt
Dokumentablage	SW-Spezifikation TelData.doc	
V-Modell-XT Version	Version 1.2.1	

Das V-Modell® XT ist urheberrechtlich geschützt. Copyright © 2006 V-Modell® XT Autoren und andere. Alle Rechte vorbehalten.

Das V-Modell® XT ist unter der Apache License Version 2.0 freigegeben.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

## Weitere Produktinformationen

Mitwirkend	[nicht beteiligt] [nicht beteiligt] [nicht beteiligt] [nicht beteiligt] [nicht beteiligt]	Prüfer Logistikentwickler Systemsicherheitsbeauftragter SW-Entwickler Ergonomieverantwortlicher
Erzeugung	Produktumfang eines SW-Moduls <ul style="list-style-type: none"> <li>• Implementierungs-, Integrations- und Prüfkonzept SW [Dateiname]</li> <li>• SW-Architektur [Dateiname]</li> </ul>	

## Änderungsverzeichnis

Änderung			Geänderte Kapitel	Beschreibung der Änderung	Autor	Zustand
Nr.	Datum	Version				
1	02.06.05	1.1	Alle	Initiale Produkterstellung	Deynet	
2	08.06.05	1.2	3	Sequenzdiagramme hinzugefügt	Deynet	
3	23.07.05	1.3	Alle	Gesamtes Dokument überarbeitet	Deynet	
4	30.08.05	1.4	3	Sequenzdiagramme überarbeitet, Beschreibung der Schnittstellen überarbeitet	Deynet	
5	13.10.05	1.5	1	Einleitung korrigiert	Deynet	
6	30.10.05	1.6	3	Sequenzdiagramme korrigiert	Deynet	

## Prüfverzeichnis

Die folgende Tabelle zeigt einen Überblick über alle Prüfungen – sowohl Eigenprüfungen wie auch Prüfungen durch eigenständige Qualitätssicherung – des vorliegenden Dokumentes.

Datum	Geprüfte Version	Anmerkungen	Prüfer	Neuer Produktzustand
5.11.05	1.6	Review	Deynet	

## Inhalt

1	Einleitung.....	6
2	SW-Element-Überblick.....	7
3	Schnittstellenbeschreibung.....	8
4	Nicht-funktionale Anforderungen.....	17
5	Anforderungsverfolgung.....	17
6	Abbildungsverzeichnis.....	17



## 1 Einleitung

Die SW-Spezifikation beschreibt alle funktionalen und nicht-funktionalen Anforderungen an ein SW-Element (SW-Einheit, SW-Komponente oder SW-Modul). Zur Erstellung der Spezifikation werden die Anforderungen aus den Spezifikationen übergeordneter Systemelemente beziehungsweise SW-Elemente abgeleitet. Die Spezifikation dient als Vorgabe und Hilfsmittel für Entwurf und Dekomposition der SW-Architektur. Sollten im Laufe der weiteren Entwicklung des SW-Elements Änderungen nötig sein, ist zunächst immer die SW-Spezifikation anzupassen. Die Prüfspezifikation Systemelement definiert die Prüffälle zum Nachweis der Schnittstellen und Anforderungen der Spezifikation.

Wesentliche Inhalte der SW-Spezifikation sind die Beschreibung der Anforderungen an das SW-Element sowie die Festlegung der Schnittstellen, die es zu bedienen hat. Zusätzlich wird die Verfeinerung und Zuordnung von Anforderungen und Schnittstellen zu untergeordneten SW-Elementen beschrieben.

Im Rahmen der Anforderungsverfolgung wird sichergestellt, dass alle Anforderungen an das Element bei der Verfeinerung auf die nächste Hierarchieebene berücksichtigt werden. Die Erstellung der SW-Spezifikationen erfolgt Hand in Hand mit dem Architektorentwurf der SW-Einheiten. Zur Sicherstellung der Konsistenz zwischen Spezifikationen und Architektur ist der SW-Architekt verantwortlich für die Erstellung beider Produkte.

Anforderungen aus der SW-Spezifikation können sich auf die Spezifikation Logistische Unterstützung auswirken. Ebenso können Anforderungen der Logistik die SW-Spezifikation beeinflussen.

## 2 SW-Element-Überblick

Das Telefonmodul TelData.c / TelData.h dient der Speicherung eines Telefonbuchs. Es enthält Funktionen zur Speicherung und Verwaltung von Telefonbucheinträgen. Zu diesen Funktionen zählen u.a. auch die Sortierung aller Telefonbucheinträge und das Auslesen einzelner oder mehrerer Telefonbucheinträge.

Auf das Telefonmodul greift nur das Modul TelApi zu. Dieses steht dann als Schnittstelle zum Telefonbuch für andere Systemteile zur Verfügung.

Nachdem ein oder mehrere Telefonbucheinträge dem Telefonbuch hinzugefügt wurden, wird das Telefonbuch durch Aufruf einer Funktion sortiert.

Daraufhin stehen verschiedene Abfragefunktionen zur Verfügung. Mit Hilfe dieser Funktionen können ein oder mehrere Telefonbucheinträge ausgelesen werden. Es besteht die Möglichkeit, über folgende Suchkriterien ein oder mehrere Telefonbucheinträge auszulesen:

- Auslesen mehrerer Einträge beginnend ab einer Indexnummer
- Auslesen mehrerer Einträge beginnend ab einem Anfangsbuchstaben
- Auslesen eines Eintrags anhand eines Index
- Auslesen eines Eintrags anhand einer Telefonnummer

Des Weiteren gibt es eine Funktion, mit der die aktuelle Anzahl der gespeicherten Telefonbucheinträge abgefragt werden kann.

Generell wird ein Zeiger ein oder mehrerer Telefonbucheinträge vom Aufrufer im Funktionskopf übergeben. Das Telefonbuchmodul schreibt nach Aufruf der jeweiligen Funktion die zurück zu liefernden Daten an die Stelle des übergebenen Zeigers.

### 3 Schnittstellenbeschreibung

Das SW-Modul TelData besitzt folgende Schnittstellen:

«Modul»TelData
<pre> vInitTpbPointer(void) enSetTelEntry(byte bTpbType, tstTelTpbEntry *pstEntry) : tenError vSortLists(void) enGetTelEntries(word wStartEntry, byte* pbNumberOfEntries, tstTelTpbEntryCan* pastData, byte bTpbType) : tenError enGetTelEntriesByCapital(byte bTpbType, tstTelTpbEntryCan* pastData, byte* pbNumberOfEntries byte bLetter, byte bOffset) : tenError enGetTelEntryByEntryNum(tstTelTpbEntry* pstEntry, byte bTpbType) : tenError enGetTelEntryByTelNum(tstTelTpbEntry* pstTpbEntry, byte bTpbType) : tenError enGetNumberOfTpbEntries(byte bTpbType, word* pwNumberOfEntries) : tenError </pre>

Abbildung 1: Schnittstellen des SW-Moduls TelData

Es existieren zwei generelle Strukturen zur Übertragung bzw. zum Auslesen eines oder mehrerer Telefonbucheinträge. Als Rückgabe der Funktionen dient eine Enumeration. Diese werden im Folgenden beschrieben.

#### 3.1 tstTelTpbEntryCan

Mit dieser Struktur kann ein Telefonbucheintrag an das CAN-Modul übermittelt werden. Falls bei dem zu übermittelnden Eintrag ein Name im Telefonbuch existiert, wird dieser übermittelt. Ansonsten wird die Telefonnummer des Eintrags übergeben.

Die Struktur enthält folgende Daten:

«struct» tstTelTpbEntryCan
<pre> +abData[nMaxNumberLengthCan] : byte +bLength : byte +wEntryNum : word </pre>

Abbildung 2: Struktur tstTelTpbEntryCan

Dabei bezeichnet

- `abData[nMaxNumberLengthCan]` den Namen des Telefonbucheintrags, bzw., falls dieser nicht existiert, die Telefonnummer des Eintrags,
- `bLength` die Länge des Namens bzw. der Telefonnummer des Eintrags und
- `wEntryNum` die Indexnummer des Telefonbucheintrags.

#### 3.2 tstTelTpbEntry

Diese Struktur stellt einen vollständigen Telefonbucheintrag dar. Er besteht aus folgenden Komponenten:

«struct» tstTelTpbEntry
<pre> +wEntryNum : word +bLengthOfNumber : byte +bLengthOfName : byte +abNumber[nMaxNumberLength] : byte +abName[nMaxNameLength] : byte </pre>

Abbildung 3: Struktur tstTelTpbEntry

Dabei bezeichnet

- `wEntryNum` die Indexnummer im Telefonbuch
- `bLengthOfNumber` die Länge der Telefonnummer
- `bLengthOfName` die Länge des Namen
- `abNumber[nMaxNumberLength]` die Telefonnummer des Anrufers und
- `abName[nMaxNameLength]` den Namen des Anrufers.

### 3.3 tenError

Als Rückgabewert der aufgerufenen Funktionen dient die Enumeration `tenError`. Sie kann, abhängig von der jeweiligen Funktion, folgende Werte annehmen:

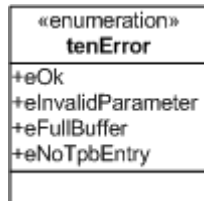


Abbildung 4: Struktur `tenError`

Dabei bezeichnet

- `eOk` „kein Fehler“,
- `eInvalidParameter` „falscher Übergabeparameter“,
- `eFullBuffer` „Puffer ist voll“ und
- `eNoTpbEntry` „kein passender Telefonbucheintrag gefunden“

Die eigentlichen Rückgabewerte (z.B. die Telefonbucheinträge) werden als Zeiger beim Aufruf der Funktion übergeben. Diese können dann nach Aufruf der jeweiligen Funktion ausgelesen bzw. weiterverarbeitet werden.

Im Folgenden werden die einzelnen Funktionen zur Verwendung des Telefonmoduls beschrieben.

### 3.4 vInitTpbPointer(void)

Die Funktion initialisiert die Pointer des Telefonbuchs.

### 3.5 enSetTelEntry(byte bTpbType, const tstTelTpbEntry\* pstEntry)

#### Eingabe:

`byte bTpbType`

Telefonbuchbuchtyp. Wird bisher nicht verwendet.

`tstTelTpbEntry* pstEntry`

Pointer auf die Struktur des zu schreibenden Telefonbucheintrags.

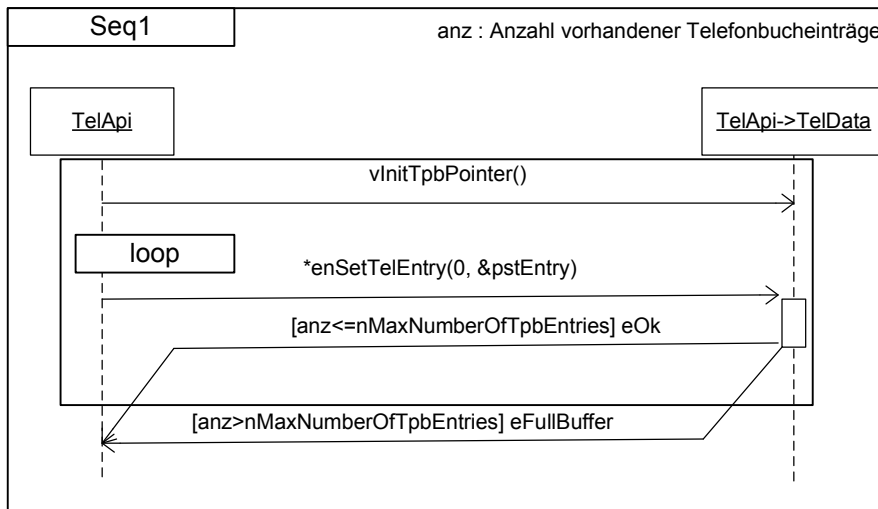
#### Ausgabe:

`tenError` (Rückgabe)

- `eOK` (Eintrag wurde hinzugefügt)
- `eFullBuffer` (Eintrag konnte nicht hinzugefügt werden, da max. Anzahl erreicht ist)

Mit Hilfe der Funktion kann ein Telefonbucheintrag gespeichert werden. Dabei wird die Variable `bTpbType` bisher nicht verwendet.

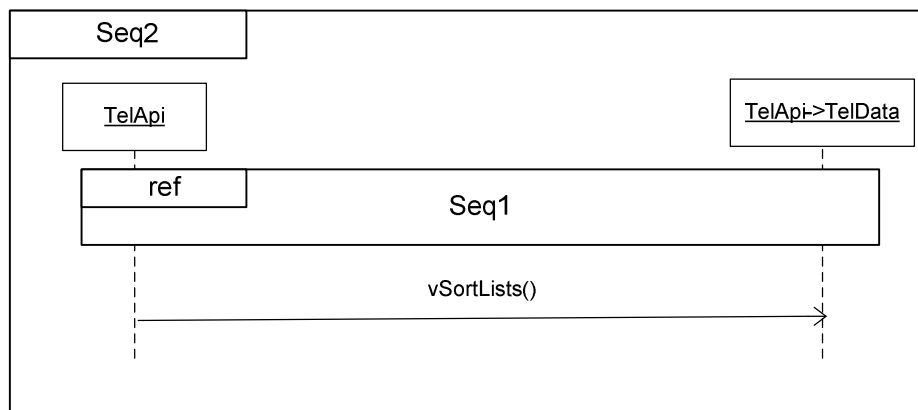
`pstEntry` ist ein Pointer auf die Struktur des zu schreibenden Telefonbucheintrags. Der Rückgabewert ist vom Typ `tenError`.



Seq 1: Das Telefonbuch wird initialisiert und Einträge werden hinzugefügt.

### 3.6 vSortLists(void)

Mit dieser Funktion werden alle im Telefonbuch befindlichen Einträge sortiert. Nachdem Telefonbucheinträge in das Telefonbuch geschrieben wurden, muss diese Funktion aufgerufen werden, um Einträge korrekt lesen zu können.



Seq 2: Nachdem Einträge dem Telefonbuch hinzugefügt wurden, sortiert `vSortLists()` dieses.

### 3.7 `enGetTelEntries(word wStartEntry, byte* pbNumberOfEntries, tstTelTpbEntryCan* pastData, byte bTpbType)`

#### Eingabe:

`byte bTpbType`

Telefonbuchtyp. Wird bisher nicht verwendet.

`word wStartEntry`

Index des als ersten zu holenden Eintrags.

`byte* pbNumberOfEntries`

Anzahl der zu holenden Einträge.

tstTelTpbEntryCan\*  
pastData

Pointer auf ein Array von Telefonbucheinträgen.  
In diesen werden die zurückgegebenen Telefonbucheinträge geschrieben.

### Ausgabe:

tenError (Rückgabe)

- eOK (Daten konnten geliefert werden)
- eInvalidParameter (die übergebenen Parameter sind nicht gültig)

byte\* pbNumberOfEntries

Hier wird die eigentliche Anzahl der zurückgelieferten Einträge gespeichert.

tstTelTpbEntryCan\*  
pastData

Pointer auf ein Array von Telefonbucheinträgen.  
In diesen werden die zurückgegebenen Telefonbucheinträge geschrieben.

Mit dieser Funktion können ein oder mehrere Telefonbucheinträge gelesen werden. Es muss der Starteintrag und die Anzahl angegeben werden. Falls der letzte Eintrag erreicht wird, wird am Anfang des Telefonbuchs wieder begonnen.

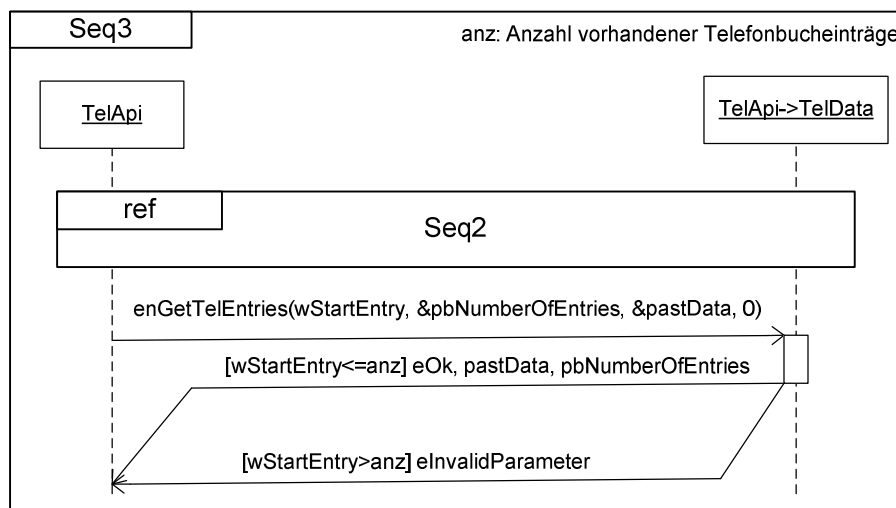
wStartEntry gibt dabei den Index des als ersten zu holenden Eintrags an.

pbNumberOfEntries beschreibt die Anzahl der zu holenden Einträge. Man übergibt hier einen Pointer der Variable, damit die eigentliche Anzahl der übergebenen Einträge hier gespeichert werden kann. Die Anzahl der übergebenen Einträge unterscheidet sich jedoch nur von der Anzahl der angeforderten Einträge, wenn die Anzahl der zu holenden Einträge die der insgesamt im Telefonbuch gespeicherten Einträge übersteigt. Ansonsten wird, falls das Ende des Telefonbuchs erreicht ist, am Anfang des Telefonbuchs begonnen.

In dem übergebenen Pointer pastData werden die zurückgegebenen Telefonbucheinträge gespeichert.

Die Variable bTpbType wird nicht verwendet.

Der Rückgabewert ist vom Typ tenError.



Seq 3: Einfaches laden mehrerer Telefonbucheinträge.

### 3.8 enGetTelEntriesByCapital(byte bTpbType, tstTelTpbEntryCan\* pastData, byte\* pbNumberOfEntries, byte bLetter, byte bOffset)

#### Eingabe:

byte bTpbType	Telefonbuchbuchtyp. Wird bisher nicht verwendet.
tstTelTpbEntryCan* pastData	Pointer auf ein Array von Telefonbucheinträgen. In diesen werden die zurückgegebenen Telefonbucheinträge geschrieben.
byte* pbNumberOfEntries	Anzahl der zu holenden Einträge.
byte bLetter	Die Variable beschreibt den Anfangsbuchstaben der zu holenden Einträge.
byte bOffset	Wird bisher nicht verwendet.

#### Ausgabe:

tenError (Rückgabe)	<ul style="list-style-type: none"> <li>• eOK (Daten konnten geliefert werden)</li> <li>• eInvalidParameter (die übergebenen Parameter sind nicht gültig)</li> </ul>
tstTelTpbEntryCan* pastData	Pointer auf ein Array von Telefonbucheinträgen. In diesen werden die zurückgegebenen Telefonbucheinträge geschrieben.
byte* pbNumberOfEntries	Hier wird die eigentliche Anzahl der zurückgelieferten Einträge gespeichert.

Mit dieser Funktion können ein oder mehrere Telefonbucheinträge, beginnend ab einem Anfangsbuchstaben ausgelesen werden. Falls der letzte Eintrag eines Anfangsbuchstabens erreicht, wird an Anfang des nächsten Buchstabens weiter gelesen.

Die Variable `bTpbType` wird nicht verwendet.

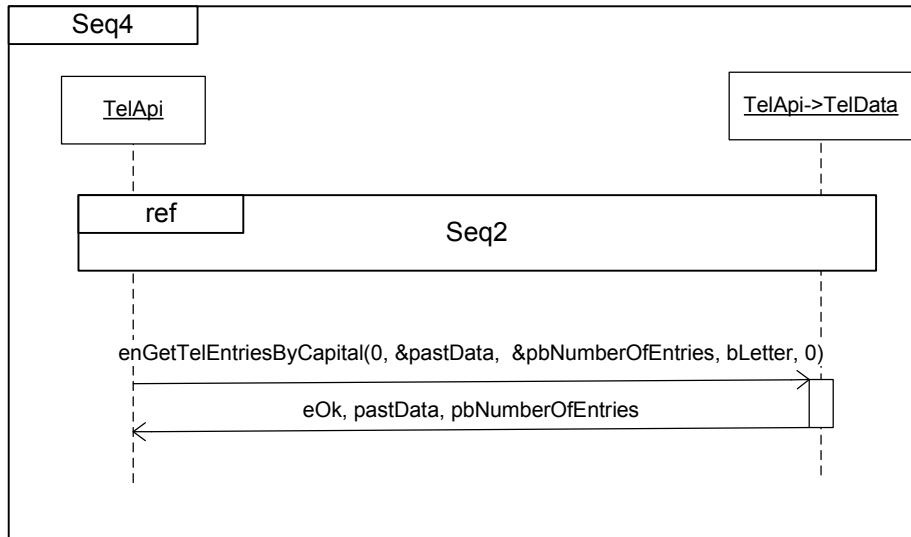
In dem übergebenen Pointer `pastData` werden die zurückgegebenen Telefonbucheinträge gespeichert.

`pbNumberOfEntries` beschreibt die Anzahl der zu holenden Einträge. Man übergibt hier einen Pointer der Variable, damit die eigentliche Anzahl der übergebenen Einträge hier gespeichert werden kann. Die Anzahl der übergebenen Einträge unterscheidet sich jedoch nur von der Anzahl der angeforderten Einträge, wenn die Anzahl der zu holenden Einträge die der insgesamt im Telefonbuch gespeicherten Einträge übersteigt. Ansonsten wird, falls das Ende des Telefonbuchs erreicht ist, am Anfang des Telefonbuchs begonnen.

`bLetter` beschreibt den Anfangsbuchstaben der zu holenden Einträge.

`bOffset` wird nicht berücksichtigt.

Der Rückgabewert ist vom Typ `tenError`.



**Seq 4: Mehrere Telefonbucheinträge werden nach dem übergebenen Anfangsbuchstaben geordnet geladen.**

### 3.9 enGetTelEntryByEntryNum(*tstTelTpbEntry\** pstEntry, byte bTpbType)

#### Eingabe:

byte bTpbType

Telefonbuchbuchtyp. Wird bisher nicht verwendet.

*tstTelTpbEntry\** pstEntry

In diesem Telefonbucheintrag wird der Index des zu holenden Eintrags übergeben.

#### Ausgabe:

tenError (Rückgabe)

- eOK (Daten konnten geliefert werden)
- eNoTpbEntry (der übergebene Index existiert nicht)

*tstTelTpbEntry\** pstEntry

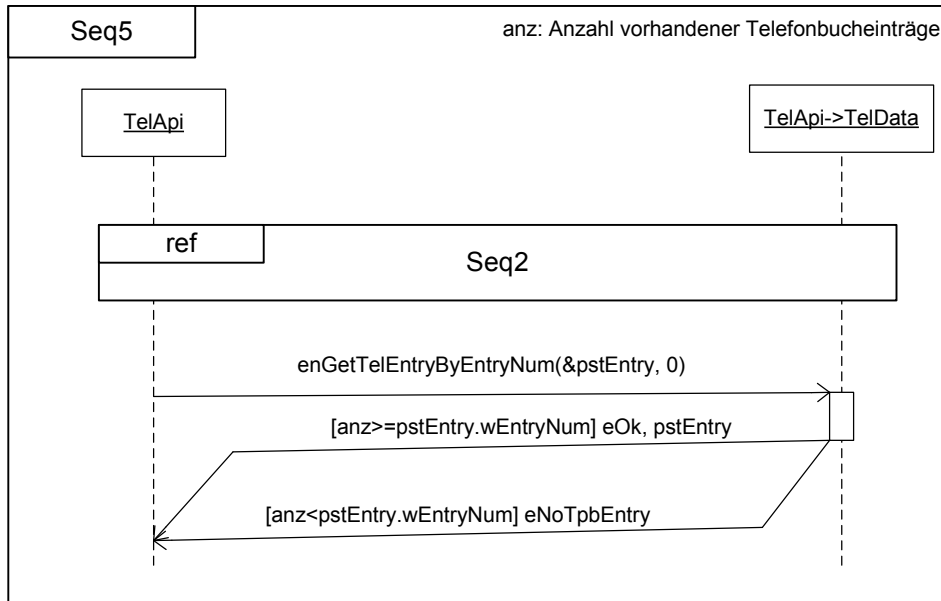
In diese Struktur wird der Eintrag mit dem übergebenen Index gespeichert.

Mit Hilfe dieser Funktion können einzelne Telefonbucheinträge aus dem Telefonbuch gelesen werden.

*pstEntry* ist ein Pointer auf einen Telefonbucheintrag vom Typ *tstTelTpbEntry*. In dieser Struktur wird der Index des zu holenden Telefonbucheintrags übergeben. In diese Variable wird dann auch der Telefonbucheintrag geschrieben.

Die Variable *bTpbType* wird nicht verwendet.

Der Rückgabewert ist vom Typ *tenError*.



**Seq 5: Ein Telefonbucheintrag wird anhand seiner Indexnummer ausgelesen.**

### 3.10 enGetTelEntryByTelNum(tstTelTpbEntry\* pstTpbEntry, byte bTpbType)

#### Eingabe:

byte bTpbType

Telefonbuchbuchtyp. Wird bisher nicht verwendet.

tstTelTpbEntry\*  
pstTpbEntry

In diesem Telefonbucheintrag wird die Telefonnummer des zu holenden Eintrags übergeben.

#### Ausgabe:

tenError (Rückgabe)

- eOK (Daten konnten geliefert werden)
- eNoTpbEntry (die übergebene Telefonnummer existiert nicht)

tstTelTpbEntry\*  
pstTpbEntry

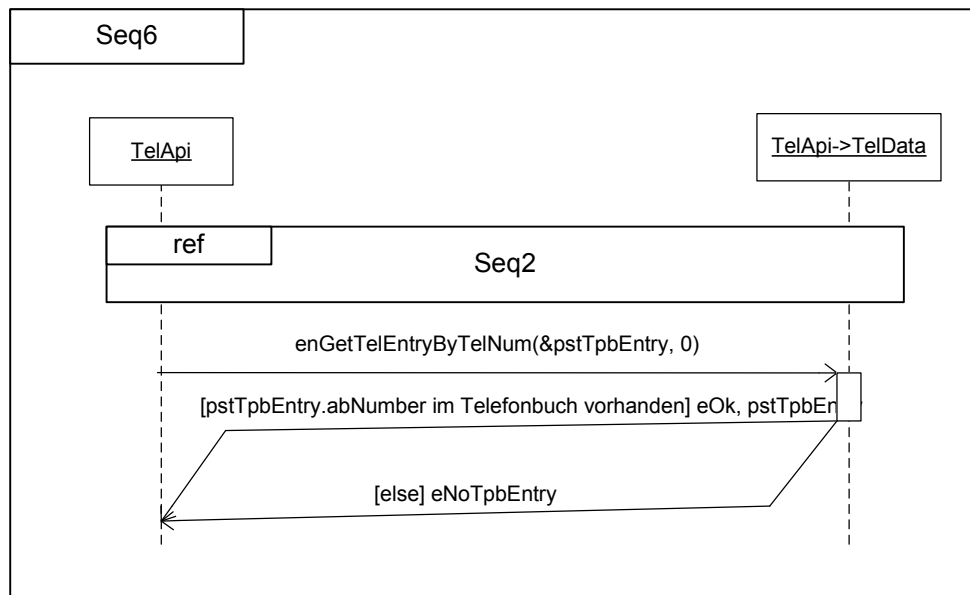
In diese Struktur wird der Eintrag mit der übergebenen Telefonnummer gespeichert.

Mit dieser Funktion kann ein Telefonbucheintrag anhand seiner Telefonnummer ausgelesen werden. Dabei wird eine evtl. vorhandene Länderkennzahl identifiziert und berücksichtigt.

pstTpbEntry ist ein Pointer vom Typ tstTelTpbEntry. In dieser Variablen wird die Telefonnummer übergeben und der ausgelesene Telefonbucheintrag gespeichert.

Die Variable bTpbType wird nicht verwendet.

Der Rückgabewert ist vom Typ tenError.



Seq 6: Ein Telefonbucheintrag wird anhand seiner Telefonnummer ausgelesen.

### 3.11 enGetNumberOfTpbEntries(byte bTpbType, word\* pwNumberOfEntries)

#### Eingabe:

byte bTpbType

Telefonbuchbuchtyp. Wird bisher nicht verwendet.

word\* pwNumberOfEntries

Pointer auf die Variable, in der die Anzahl der vorhandenen Telefonbucheinträge gespeichert wird.

#### Ausgabe:

tenError (Rückgabe)

- eOK (Daten konnten geliefert werden)

word\* pwNumberOfEntries

Anzahl vorhandener Telefonbucheinträge.

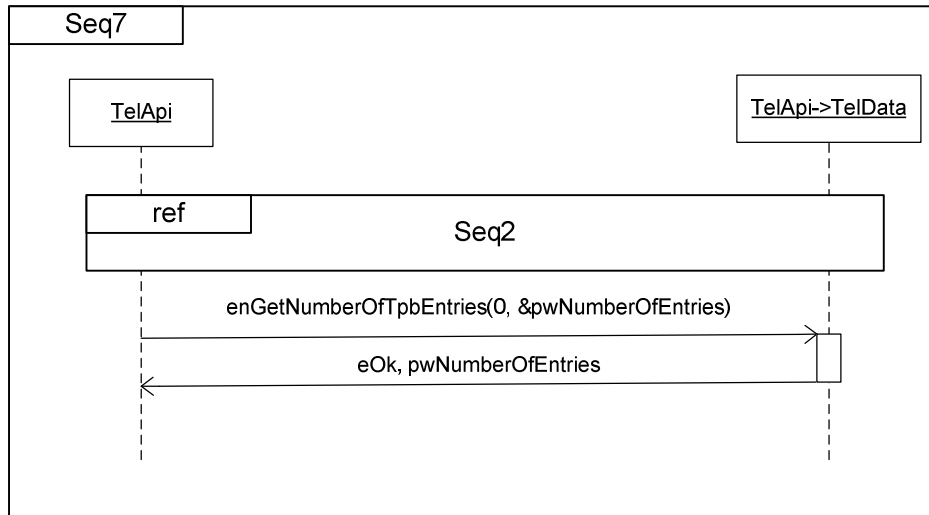
Diese Funktion dient dem Auslesen der Anzahl der gespeicherten Telefonbucheinträge im Telefonbuch.

Die Variable bTpbType wird nicht verwendet.

Es wird der Zeiger pwNumberOfEntries übergeben; in diesem wird die Anzahl der vorhandenen Telefonbucheinträge gespeichert.

Der Rückgabewert ist vom Typ tenError und kann folgende Werte zurückgeben:

- eOk (das Datum konnte korrekt geliefert werden)



**Seq 7: Die Anzahl der vorhandenen Telefonbucheinträge wird gelesen.**

## 4 Nicht-funktionale Anforderungen

1. Die Sortierung der Telefonbucheinträge muss innerhalb von 450ms abgeschlossen sein.

## 5 Anforderungsverfolgung

TelData	TelApi
3.1	3.1
3.2	3.2
3.3	3.3
3.4	3.4
3.5	3.5
3.6	-
3.7	3.8
3.8	3.9
3.9	3.11
3.10	3.10
3.11	3.12

## 6 Abbildungsverzeichnis

Abbildung 1: Schnittstellen des SW-Moduls TelData.....	8
Abbildung 2: Struktur tstTelTpbEntryCan .....	8
Abbildung 3: Struktur tstTelTpbEntry .....	8
Abbildung 4: Struktur tenError .....	9

Seq 1: Das Telefonbuch wird initialisiert und Einträge werden hinzugefügt. ....	10
Seq 2: Nachdem Einträge dem Telefonbuch hinzugefügt wurden, sortiert vSortLists() dieses. ....	10
Seq 3: Einfaches laden mehrerer Telefonbucheinträge.....	11
Seq 4: Mehrere Telefonbucheinträge werden nach dem übergebenen Anfangsbuchstaben geordnet geladen.....	13
Seq 5: Ein Telefonbucheintrag wird anhand seiner Indexnummer ausgelesen. ....	14
Seq 6: Ein Telefonbucheintrag wird anhand seiner Telefonnummer ausgelesen. ....	15
Seq 7: Die Anzahl der vorhandenen Telefonbucheinträge wird gelesen. ....	16