# Dokumentation
# Eine Bibliothek zum Versand von SMS Nachrichten
## Version 1.0

Kontakt:

BEYOND THE NET Internet Service GmbH
Bonnerstr. 31
50389 Wesseling
Phone: +49 2236 88 11 8-0
Fax: +49 2236 88 11 8-88
E-Mail: info@btn.de
Web: www.btn.de

# Inhalt

Kapitel 2 ..... 5
Installation ..... 5
2.1 Windows ..... 5
2.1.1 Binärpaket ..... 5
2.1.2 Compilieren des Quelltextes ..... 5
2.2 Linux ..... 5
2.2.1 Binärpakete ..... 5
2.2.2 Compilieren des Quelltextes ..... 5
2.3 UNIX ..... 5
Kapitel 3 ..... 6
Programmierung in C ..... 6
3.1 Einbindung der Bibliothek ..... 6
3.1.1 Windows ..... 6
3.1.2 Linux ..... 6
3.2 Versenden einer einfachen SMS ..... 7
3.3 Benutzung der erweiterten Funktionen ..... 7
3.3.1 Erstellen der btnCreds- und btnSms-Strukturen ..... 7
3.3.2 Setzen der Versandoptionen ..... 7
3.3.3 Versand durchführen ..... 8
3.3.4 Ergebnisauswertung ..... 8
3.3.5 Speicher freigeben ..... 9
3.4 Beispiel ..... 9
Kapitel 4 ..... 11
Programmierung in anderen Programmiersprachen ..... 11
4.1 PHP ..... 11
4.2 VisualBasic 6.0 ..... 11
4.2.1 Allgemeines ..... 11
4.2.2 Beispiel ..... 12
4.3 Delphi 5 ..... 13
4.3.1 Allgemeines ..... 13
4.3.2 Beispiel ..... 13
4.4.1 Allgemeines ..... 15
4.4.2 Beispiel ..... 15

Kapitel 5 ..... 16
Referenz ..... 16
5.1 Konstanten ..... 16
5.1.1 Strukturtypen ..... 16
5.1.2 Zugangsdaten ..... 16
5.1.3 Proxy-Server ..... 16
5.1.4 Tarife ..... 17
5.1.5 Nachrichtentypen ..... 17
5.1.6 Ergebnis ..... 18
5.1.7 Benutzerinformationen ..... 19
5.2 Strukturen ..... 20
5.3 Funktionen ..... 28
5.3.1 Zugangsdaten ..... 28
5.3.3 Funktionen nur für SMS Nachrichten ..... 32
5.3.4 Funktionen nur für MMS Nachrichten ..... 35
5.3.5 Ergebnisauswertung ..... 36
5.3.6 Benutzerinformationen ..... 38
5.3.7 Empfang von SMS Nachrichten ..... 40
Anhang A ..... 42
Lizenz ..... 42

# Kapitel 1
## Einleitung

Die btnSMS Bibliothek stellt Funktionen zur Verfügung um SMS Nachrichten über den Dienst von BEYOND THE NET zu versenden. Dabei wurde besonderen Wert darauf gelegt, dass eine möglichst reibungslose Integration in bestehende Software möglich ist. So können z.B. mit nur einem einzigen Funktionsaufruf eine einfache 160 Zeichen SMS Nachricht schnell und einfach versendet werden. Mehr Optionen stellt die Bibliothek über Funktionen, die auf verschiedenen Strukturen arbeiten zur Verfügung.

Als Lizenz wird die GNU Lesser Public License - LGPL (siehe Seite 45) verwendet. Einer Einbindung in kommerzielle Projekte und damit die Erweiterung bestehender Applikationen um die Funktionalität des SMS Nachrichtenversandes steht also nichts im Wege. Außerdem haben Sie jederzeit die Möglichkeit den Quellcode einzusehen. Dadurch ergeben sich im Wesentlichen zwei Vorteile. Zum einen kann genau überprüft werden, welche internen Vorgänge stattfinden und wie diese implementiert sind. Zum anderen wird die Fehlersuche extrem vereinfacht. Wir geben uns größte Mühe unsere Software fehlerfrei zu halten. Trotzdem gibt es wohl keine Software die man als vollkommen fehlerfrei. Durch den vorhandenen Quellcode haben Sie auch die Möglichkeit eventuelle Fehler schneller zuzuordnen, ob ein Fehler in der Bibliothek oder im aufrufenden Programm vorliegt. Hinweise auf Fehler nehmen wir sehr ernst und wir sind dankbar um jeden Erfahrungsbericht über unsere Arbeit.

Die SMS Nachrichten werden über eine Schnittstelle versendet die denen der Webservices ähnlich ist. Der Client sendet eine XML-Datei über das http-Protokoll an den Server von BEYOND THE NET. Dieser Antwortet ebenfalls mit einer XML-Datei, die der Client dann auswerten kann.
Dieses Verfahren basiert auf offenen Standards und ist somit sehr Zukunftssicher. Um die korrekte Implementation der Protokolle und Standards zu gewährleisten, benutzt btnSMS zwei andere freie Bibliotheken. Für das http-Protokoll wird libCURL eingesetzt.
Diese, unter einer MIT/X-Basierenden Lizenz zur Verfügung stehende Bibliothek, ist auf UNIX-Systemen weit verbreitet, aber auch ohne Probleme unter Windows einsetzbar. Für die Erstellung und Auswertung der XML-Dateien wird die ebenfalls unter einer MIT-Lizenz verfügbare libXML2 eingesetzt. Die Lizenzen dieser Bibliotheken erlauben ebenfalls den Einsatz in kommerziellen Projekten.
Diese Dokumentation wurde mit LaTEXerstellt.

# Kapitel 2

## Installation

### 2.1 Windows

#### 2.1.1 Binärpaket

Die Binärpakete für Windows zeichnen sich dadurch aus, dass die darin enthaltene DLL-Datei eine \All-in-One" Lösung ist. Alle zusätzlich benötigten Bibliotheken (libCURL und libXML2) wurden statisch in die DLL-Datei gelinkt, so dass bei der Installation eines Programms, das btnSMS benutzt, nur die eine DLL-Datei (btnsms.dll) verfügbar gemacht werden muss.

#### 2.1.2 Compilieren des Quelltextes

Das compilieren des Quellcodes unter Windows ist nicht trivial. Hier sind verschiedene Optionen je nach Anwendungsfall zu setzen. Insbesondere bestehen mehrere Möglichkeiten die Bibliothek zu linken. Um eine DLL-Datei zu erstellen, wie sie im Binärpaket enthalten ist, müssen zunächst spezielle Versionen von libCURL und libXML2 erstellt werden, die statisch gelinkt werden können, aber trotzdem die Laufzeitbibliothek für dynamisch gelinkte Programme verwenden. Einfacher ist es die Bibliothek komplett dynamisch zu linken. Dann sind aber neben der daraus entstehenden btnsms.dll noch eine curl.dll und eine libxml.dll erforderlich.
Das Quellcode Paket für Windows wird demnächst eine Projekt-Datei für Microsoft Visual Studio .NET 2003 enthalten, in der alle diese Möglichkeiten berücksichtigt werden.

### 2.2 Linux

#### 2.2.1 Binärpakete

Wir werden uns bemühen in Zukunft für verschiedene Linux Distributionen Binärpakete im jeweiligen Paketformat anzubieten. Da wir aber nicht die ganze Bandbreite an Distributionen ständig vorhalten können, sind wir dabei auf die Mithilfe andere angewiesen. Sollten Sie für eine bestimmte Distribution ein Binärpaket von btnSMS erstellt haben, senden Sie es uns bitte zu. Nach einer Prüfung werden wir es dann auf den entsprechenden Webseiten zum Download stellen.

#### 2.2.2 Compilieren des Quelltextes

Auf Linux-Systemen ist es üblicher den Quelltext selbst zu compilieren. Durch das GNU Build System ist dies auch sehr einfach möglich. In dem Archiv ist eine Datei mit dem Namen INSTALL vorhanden. In dieser Datei befindet sich die genaue Anleitung wie der Vorgang des compilierens durchzuführen ist und welche Voraussetzungen dafür zu erfüllen sind.

### 2.3 UNIX

Uns liegen bisher keine Erfahrungsberichte vor, ob btnSMS auch auf anderen UNIX-ähnlichen Betriebssystemen lauffähig ist. Durch den Einsatz der GNU-build-tools sollte eine Portierung allerdings ohne Probleme möglich sein. Wenn Sie Erfahrungen mit btnSMS auf anderen UNIX Systemen gemacht haben, würden wir uns freuen von Ihnen zu hören.

# Kapitel 3

## Programmierung in C

### 3.1 Einbindung der Bibliothek

#### 3.1.1 Windows

Damit sie die Funktionen der btnSMS-Bibliothek in C nutzen können werden die Header-Dateien (*.h), eine Import-Bibliothek (btnsms.lib) und die DLL-Datei (btnsms.dll) benötigt. Die Header-Dateien enthalten Prototypen für die in der Bibliothek vorhandenen Funktionen. Damit die Einbindung besonders einfach ist, muss nur eine Header-Datei (die dann weitere Dateien einbindet) von Ihnen über die #include Direktive ihres Compilers eingebunden werden. Dies ist die Datei btnsms.h. Außerdem muss ihr Programm gegen die Bibliothek gelinkt werden. Es gibt zwei Methoden, wie sie in ihrem C Programm die Funktionen aus der btnsms DLL-Datei verwenden können. Dabei handelt es sich um die \Explizite Verknüpfung" und die \Implizite Verknüpfung". Letztere ist die am häufigsten verwendete und einfachere Methode, die daher auch an dieser Stelle kurz beschrieben wird. Um die btnSMS DLL-Datei \explizit" zu verknüpfen, konsultieren Sie bitte das Handbuch ihres Compilers.

Für die \Implizite Verknüpfung" wird in den Binärpaketen für Windows eine sogenannte \Import-Bibliothek" mitgeliefert. In den Linker-Optionen ihres Compilers müssen sie angeben, dass sie das Programm gegen die Datei btnsms.lib linken möchten. Dann genügt es, wenn sich bei Ausführung ihres Programms die DLL-Datei btnsms.dll im gleichen Verzeichnis beendet, wie die ausführbare (.exe) Datei ihres Programms. Alternativ kann btnsms.dll auch in das Windows-Verzeichnis oder das Windows-System-Verzeichnis kopiert werden.
Eine einfache Möglichkeit diese Aufgaben durchzuführen ist die Verwendung eines Installationsprogramms.
Um ein solches zu erstellen gibt es auch sehr gute kostenlose Tools.
Eines ist z.B. der NSIS-Compiler. Um damit ein Installationsprogramm zu erstellen, muss ein kleines Script geschrieben werden, welches die Aufgaben der Installation definiert. Als Ausgabe bekommt man eine ausführbare Datei, die das Programm enthält und bei Ausführung installiert. Zu finden ist NSIS unter http://nsis.sourceforge.net/.

#### 3.1.2 Linux

Damit die Bibliothek unter Linux nutzbar ist, müssen die Pakete libxml2 und curl installiert sein. Soll die Bibliothek aus dem Quellcode kompiliert werden, müssen auch die dazugehörigen \Development-Pakete" installiert werden. Diese enthalten die benötigten Header-Dateien um c die Bibliothek aus dem Quellcode zu • übersetzen (siehe INSTALL-Datei in der Quellcode-Distribution für Linux). Ist die Bibliothek einmal auf dem Linux-System installiert, muss nur noch dem Linker angeben werden, dass gegen die btnSMS-Bibliothek gelinkt werden soll. Dies geschieht beim GCC (der unter Linux am häufigsten verwendete Compiler) mit der Option\-lbtnsms".
Sollte die Bibliothek nicht im Standard-Pfad installiert worden sein, kann bei der Compilierung evtl. die Angabe eines zusätzlichen Include-Pfades mit der Option \- I/path/to/include/_les" nötig sein. Beim Linkvorgang ist dann auch der Pfad zur Bibliotheksdatei
(.so) als Angabe notwendig. Dies passiert mit der Option \-L/path/to/lib/_le".

## 3.2 Versenden einer einfachen SMS

Um eine einfache SMS Nachricht zu versenden, bedarf es nur eines einzigen Funktionsaufrufes. verwendet wird dazu die btnSmsSendSimple(. . . )-Funktion (siehe Referenz Seite 32). Als Parameter muss der Funktion die BTN-BenutzerID, das zugehörige Passwort, die Empfänger-Telefonnummer und der Text der Nachricht angegeben werden. Der Rückgabewert der Funktion ist vom Typ int und kann mit Hilfe der Fehlercode-Konstanten (siehe Seite 18) ausgewertet werden.

## 3.3 Benutzung der erweiterten Funktionen

Um beim Versand von SMS Nachrichten diverse Optionen einzustellen, können die erweiterten Funktionen benutzt werden. Der Grundsätzliche Ablauf verläuft dann im Normalfall in folgenden Schritten:

1. Erstellen einer Struktur mit den Zugangsdaten (btnCreds) und einer Struktur für den SMS Versand (btnSms).

2. Anwendung verschiedener Funktionen auf die Strukturen um Empfänger, Text und bestimmte Optionen zu setzen oder zu verändern.

3. Den durch die Struktur beschriebenen Versand durchführen.

4. Die beim Versand erhaltene Ergebnis-Struktur (btnResult) auswerten.

5. Speicher für die erstellten Strukturen freigeben.

### 3.3.1 Erstellen der btnCreds- und btnSms-Strukturen

Die Funktion btnCredsNew(. . . ) reserviert Speicher für eine neue btnCreds-Struktur und initialisiert diese mit den als Parameter übergebenen Werten. Als Parameter der Funktion müssen die BTN-BenutzerID und das dazugehörige Passwort übergeben werden (siehe Referenz Seite 28). Der zurückgegebene Zeiger ist vom Typ ptrBtnCreds und sollte in einer entsprechenden Zeigervariablen gespeichert werden.

Die Funktion btnSmsNew(. . . ) reserviert Speicher für eine neue btnSms-Struktur und initialisiert diese mit sinnvollen Standardwerten. Vom Typ her ist die Nachricht dann zunächst eine 160 Zeichen Text SMS die im Standard-Tarif versendet wird. Alle zusätzlichen Versandoptionen sind deaktiviert. Der zurückgegebene Zeiger ist vom Typ ptrBtnSms und sollte in einer entsprechenden Zeigervariablen gespeichert werden.

### 3.3.2 Setzen der Versandoptionen

Damit eine Nachricht versendet werden kann, müssen in der btnSms-Struktur zumindest die Zugangsdaten, der Nachrichtentext und mindestens eine Empfänger-Telefonnummer gesetzt werden.
Um die Zugangsdaten zu setzen wird die Funktion btnMessageSetCreds(. . . ) verwendet. Dieser übergibt man als ersten Parameter einen Zeiger auf die zu manipulierende btnSms- Struktur und als zweiten Parameter den Zeiger auf die zuvor erstellte btnCreds-Struktur (siehe Referenz Seite 31).
Mit der Funktion btnMessageSetText(. . . ) kann der Text der SMS Nachricht angegeben werden (siehe Referenz Seite 31). Erster Parameter ist, wie üblich, die btnSms-Struktur, die manipuliert werden soll. Als zweiter Parameter wird der Text als Zeichenkette übergeben. Damit sichergestellt ist, dass kein Speicher überlauf stattfindet muss als dritter Parameter die Länge des Textes angegeben werden.

Letztlich muss noch mindestens eine Empfänger-Rufnummer angegeben werden, an die die SMS Nachricht gesendet wird. Dies erledigt die Funktion btnMessageAddDestination(. . . ) (siehe Referenz Seite 30). Auch dieser Funktion wird als erster Parameter ein Zeiger auf die zu manipulierende btnSms-Struktur übergeben. Parameter zwei ist eine Zeichenkette die eine hinzuzufügender Telefonnummer enthält. Parameter drei muss wiederum die Länge der übergebenen Zeichenkette enthalten. Diese Funktion kann nahezu beliebig oft mit verschiedenen Telefonnummern aufgerufen werden. Die SMS Nachricht wird dann an alle jemals übergebenen Telefonnummern versendet. Es ist allerdings zu beachten, dass bei mehrmaligem Aufruf mit derselben Telefonnummer die Nachricht auch genauso oft an diese Nummer gesendet wird, wie die Funktion aufgerufen wurde.
An dieser Stelle können noch weitere Versandoptionen mit Hilfe von anderen Funktionen gesetzt werden. Diese entnehmen Sie bitte zunächst der Referenz ab Seite 17.

### 3.3.3 Versand durchführen

Schließlich wird der Versand der SMS Nachricht(en) dann mit der Funktion btnSmsSend(. . . ) durchgeführt. Diese Funktion benötigt als einzigen Parameter einen Zeiger auf die Zuvor erstellte btnSms-Struktur. Zurückgegeben wird ein Zeiger auf eine btnResult-Struktur, die wie im Folgenden beschrieben, zur Ergebnisauswertung verwendet werden kann. Diese sollte mit Hilfe einer entsprechenden Zeigervariablen vom Typ ptrBtnResult gespeichert werden.

### 3.3.4 Ergebnisauswertung

Um festzustellen ob der Versand der SMS Nachrichten geglückt ist, sollte zunächst mit der Funktion btnResultGetErrorcode(. . . ) der Fehlercode aus der btnResult-Struktur ausgelesen werden. Als einziger Parameter muss hier der beim Versand erhaltene Zeiger übergeben werden. Der Rückgabewert ist vom Typ int und kann folgendermaßen ausgewertet werden: Wert Bedeutung

1 - 1024 Fehler der durch den Server bei Beyond The Net gemeldet wurde. Eine genauere Auswertung kann mit Hilfe der Konstanden auf Seite 18 durchgeführt werden.

1025 – 2048: Fehler auf Netzwerk Ebene. Eine genauere Auswertung kann mit Hilfe der Fehlercodes der libCURL-Bibliothek durchgeführt werden, wobei hier vom zurückgegebenen Wert 1024 subtrahiert werden muss (siehe http://curl.haxx.se/libcurl/c/libcurl-errors.html).

2049 Fehler auf Ebene des http-Protokolls. Vom zurückgegebenen Wert muss 2048 subtrahiert werden. Dann entspricht der Fehlercode dem entsprechenden http-Statuscode (siehe http://de.wikipedia.org/wiki/HTTP-Statuscodes).

Ist alles korrekt verlaufen sollte der zurückgegebene Wert der Konstante
BTN ERROR SUCCESS entsprechen, also 0 sein. Dann wurden die SMS Nachrichten an alle Empfänger erfolgreich dem BTN Server übergeben. Neben den anderen BTN ERROR XXX Fehlercodes, bei denen definitiv keine Nachrichten versendet wurden, nimmt der Wert BTN ERROR PARTIAL SUCCESS eine Sonderrolle ein. Wird er zurückgegeben konnten die SMS Nachrichten nicht an alle Empfänger versendet werden, aber zumindest an einen Teil davon. An welche dies der Fall ist, kann mit den Funktionen btnResultGetFirstResultItem(. . . ) und btnResultItem. . . (. . . ) herausgefunden werden.
Im Fehlerfall kann zusätzlich mit der Funktion btnResultGetErrorMsg(. . . ) eine genauere Fehlerbeschreibung ausgelesen werden. Diese dient nur der weiteren Beschreibung des Fehlers und kann sich von Version zu Version unangekündigt auch ändern. Es existieren weitere Funktionen um eine detailliertere Auswertung vorzunehmen. Diese und die hier beschriebenen können in der Referenz ab Seite 17 nachgeschlagen werden.

### 3.3.5 Speicher freigeben

Zu jeder von der btnSMS-Library erstellten Struktur gibt es eine entsprechende Funktion den Speicher, der dabei reserviert wurde auch wieder freizugeben. Für eine btnCreds Struktur ist es die Funktion btnCredsFree(. . . ), für eine btnSms-Struktur ist es die Funktion btnSmsFree(. . . ) und für eine btnResult-Struktur ist es die Funktion btnResultFree(. . . ).

## 3.4 Beispiel

```
#include <stdio.h>
#include <btnsms.h>

#define USERNAME "TST00000"
#define PASSWORD "test"

int main(int argc, char *argv[]) {
        ptrBtnCreds creds;

ptrBtnSms sms;
ptrBtnResult result;

if (argc != 3) {
        fprintf(stderr,"Wrong Number of Arguments\n");
        fprintf(stderr,"Usage: sendsms <destination> <text>\n");
        exit(1);

}

creds = btnCredsNew(USERNAME,PASSWORD);
sms = btnSmsNew();
btnMessageSetCreds((ptrBtnMessage)sms,creds);
btnMessageAddDestination((ptrBtnMessage)sms,argv[1],strlen(argv[1]));
btnMessageSetText((ptrBtnMessage)sms,argv[2],strlen(argv[2]));

if (sms != 0) {

        printf("Sending SMS Message to %s\n",argv[1]);

} else {

        fprintf(stderr,"Could not create btnSms structure\n");
        exit(1);
}

result = btnMessageSend(sms);

if (result != 0 &&
        btnResultGetErrorcode(result) == BTN_ERROR_SUCCESS) {
                printf("Sucess\n");
                exit(0);
} else {

if (result == 0) {
        fprintf(stderr,"Client error\n");
```

```
} else {
        fprintf(stderr,
                "Failed\nErrorcode: %d\n",
                btnResultGetErrorcode(result));
        fprintf(stderr,
                "Errormessage: %s\n",
                btnResultGetErrorMsg(result));
                exit(btnResultGetErrorcode(result));
                }
        }
}
```

# Kapitel 4

## Programmierung in anderen Programmiersprachen

Grundsätzlich ist die Bibliothek von jeder Programmiersprache aus nutzbar, die unter Windows die exportierten Funktionen einer DLL-Datei und unter Linux die Funktionen eines \shared-object-File" aufrufen kann. Der direkte Zugriff auf Strukturen ist nicht erforderlich.
Es empfiehlt sich allerdings die in der Bibliothek verwendeten Konstanten (siehe Seite 17) in der jeweiligen Programmiersprache nachzubilden. Sollten Sie einen sogenannten Wrapper in einer anderen Programmiersprache erstellt haben und waren Sie dazu bereit diesen unter einer freien Lizenz zu Verfügung zu stellen, wurden wir uns freuen von Ihnen zu hören. Dieser konnte dann an dieser Stelle aufgenommen werden.

### 4.1 PHP

Es existiert eine sogenannte \Extension" für PHP. Dafür muss zunächst die Basis-Bibliothek auf dem entsprechenden Rechner installiert sein. Die Extension verbindet dann die Bibliothek mit PHP. Diese muss in der entsprechenden PHP-Datei nur mit der Funktion dl() eingebunden werden. Für die PHP Extension gibt es eine eigene Dokumentation von BEYOND THE NET. Diese enthält auch eine ausführliche Funktionsreferenz, da die Funktionen auf die Bedürfnisse der Programmiersprache angepasst wurden.

### 4.2 VisualBasic 6.0

#### 4.2.1 Allgemeines

Die btnSMS-Bibliothek kann auch Problemlos mit Visual Basic 6.0 benutzt werden. Dazu enthält die Bibliothek bereits Funktionen die mit der Aufrufkonvention \ stdcall". Auf der Homepage unter http://w3.btn.de/entwickler.php kann für Visual Basic 6.0 ein Archiv mit einem Beispiel und einer Datei, die bereits die entsprechenden Prototypen enthält, heruntergeladen werden.
Zu beachten ist in jedem Fall, dass bei allen Funktionen aus der btnSMS-Bibiliothek in Visual Basic 6.0 immer das Pre_x \std " vor dem in dieser Dokumentation beschriebenen Funktionsnamen stehen muss.

In Visual Basic 6.0 gibt es keine Zeiger. Daher werden alle Zeigerwerte in Variablen vom Typ Long gespeichert. Die Funktionen sind auch entsprechende deklariert, dass sie diese Werte als Parameter übernehmen und als Rückgabwerte zurückgeben.

### 4.2.2 Beispiel

```
Private Declare Function std_btnCredsNew Lib "btnsms.dll" (ByVal username As String, ByVal
Private Declare Function std_btnCredsSetCustomer Lib "btnsms.dll" (ByVal creds As Long, ByVal
Private Declare Sub std_btnCredsFree Lib "btnsms.dll" (ByVal creds As Long)
Private Declare Function std_btnSmsSendSimple Lib "btnsms.dll" (ByVal username As String,
Private Declare Function std_btnSmsSend Lib "btnsms.dll" (ByVal msg As Long) As Long
Private Declare Function std_btnSmsNew Lib "btnsms.dll" () As Long
Private Declare Function std_btnMessageAddDestination Lib "btnsms.dll" (ByVal msg As Long,
Private Declare Function std_btnMessageAddDestinationGreet Lib "btnsms.dll" (ByVal msg As
Private Declare Function std_btnSmsSetText Lib "btnsms.dll" (ByVal msg As Long, ByVal text
Private Declare Sub std_btnSmsSetCreds Lib "btnsms.dll" (ByVal msg As Long, ByVal creds As
Private Declare Function std_btnSmsSetOriginator Lib "btnsms.dll" (ByVal msg As Long, ByVal
Private Declare Function std_btnSmsSetGreetSubst Lib "btnsms.dll" (ByVal msg As Long, ByVal
Private Declare Sub std_btnSmsSetType Lib "btnsms.dll" (ByVal msg As Long, ptype As Long)
Private Declare Sub std_btnSmsSetTarif Lib "btnsms.dll" (ByVal msg As Long, tarif As Long)
Private Declare Sub std_btnSmsSetDelivery Lib "btnsms.dll" (ByVal msg As Long, day As Long,
Private Declare Function std_btnSmsSetStatusReport Lib "btnsms.dll" (ByVal msg As Long, ByVal
Private Declare Sub std_btnSmsFree Lib "btnsms.dll" (ByVal msg As Long)
Private Declare Sub std_btnResultFree Lib "btnsms.dll" (ByVal result As Long)
Private Declare Function std_btnResultGetErrorcode Lib "btnsms.dll" (ByVal result As Long)
Private Declare Function std_btnResultGetErrorMsg Lib "btnsms.dll" (ByVal result As Long) Private
Declare Function std_btnResultGetAccountBalance Lib "btnsms.dll" (ByVal result As Private
Declare Function std_btnResultGetStatusReportId Lib "btnsms.dll" (ByVal result As Private
Declare Function std_btnResultGetFirstResultItem Lib "btnsms.dll" (ByVal result As Private
Declare Function std_btnResultItemGetNext Lib "btnsms.dll" (ByVal resultitem As Long) Private
Declare Function std_btnResultItemGetDestination Lib "btnsms.dll" (ByVal resultitem Private
Declare Function std_btnResultItemGetErrorcode Lib "btnsms.dll" (ByVal resultitem Private
Declare Function std_btnResultItemGetResult Lib "btnsms.dll" (ByVal resultitem As Private
Declare Function std_btnResultItemGetErrorMsg Lib "btnsms.dll" (ByVal resultitem As Private
Declare Function std_btnUserstatusRequest Lib "btnsms.dll" (ByVal creds As Long) As Private
Declare Function std_btnUserstatusGetErrorcode Lib "btnsms.dll" (ByVal status As Long) Private
Declare Function std_btnUserstatusGetErrorMsg Lib "btnsms.dll" (ByVal status As Long) Private
Declare Function std_btnUserstatusGetUserid Lib "btnsms.dll" (ByVal status As Long) Private
Declare Function std_btnUserstatusGetActive Lib "btnsms.dll" (ByVal status As Long) Private
Declare Function std_btnUserstatusGetType Lib "btnsms.dll" (ByVal status As Long) Private
Declare Function std_btnUserstatusGetAccountBalance Lib "btnsms.dll" (ByVal status Private
Declare Function std_btnUserstatusGetToday Lib "btnsms.dll" (ByVal status As Long) Private
Declare Function std_btnUserstatusGetLimit Lib "btnsms.dll" (ByVal status As Long) Private
Declare Sub std_btnUserstatusFree Lib "btnsms.dll" (ByVal status As Long)
Private Sub Command1_Click()
Dim msg As Long
Dim creds As Long
Dim result As Long
Dim dummy_s As String
Dim dummy_l As Long
Dim wurst As Variant
msg = std_btnSmsNew()
creds = std_btnCredsNew("TST00000", "test")
std_btnSmsSetCreds msg, creds
dummy_l = std_btnMessageAddDestination(msg, "+491712345678", Len("+491712345678"))
dummy_s = std_btnMessageSetText(msg, "Testnachricht", Len("Testnachricht"))
result = std_btnMessageSend(msg)
Label1.Caption = msg
Label2.Caption = creds
Label3.Caption = std_btnResultGetErrorcode(result)
End Sub
```

## 4.3 Delphi 5

### 4.3.1 Allgemeines

Um die btnSMS-Bibliothek mit Delphi 5 zu benutzen wird eine Unit benötigt, die alle Funktionen aus der Bibliothek als externe Funktionen einbindet. Diese Unit können sie auf der Internetseite zu unserer Bibliothek unter http://w3.btn.de/entwickler.php herunterladen.
Zusätzlich wird natürlich auch die Bibliothek als solche benötigt. Hier greifen sie am besten auf eine der vorkompilierten Versionen für Windows zurück. Die in dem Archiv enthaltene DLL-Datei muss sich entweder im Windows-Verzeichnis (meist C:\WINDOWS oder C:\WINNT) oder in demselben Verzeichnis wie die ausführbare Datei (.exe) ihres Programms beenden.

Die Beschreibung wie die Funktionen zu benutzen sind, kann dieser Dokumentation für die Sprache C entnommen werden mit folgenden zusätzlichen Hinweisen:

- Alle Zeiger sind vom Typ Pointer und sollten in entsprechenden Zeigervariablen gespeichert werden.

- Alle Zeichenketten (Strings) die an die Funktionen übergeben werden, müssen Nullterminierte Strings sein. Delphi arbeitet normalerweise mit sogenannten \Pascal-Strings". Um einen Null-terminierten String zu erhalten schreiben sie einfach PChar('inhalt der Zeichenkette').

- Alle Strings die von den Funktionen zurückgegeben werden sind ebenfalls Nullterminierte Strings. Diese können Sie mit string(. . . ) wieder in einen normalen Pascal String umwandeln.

### 4.3.2 Beispiel

Es folgt ein Beispiel wie die btnSMS-Bibliothek in Delphi 5 zu benutzen ist. Dabei handelt es sich nur um die Quellcode-Datei eines Formulars. Das gesamte Beispiel-Projekt ist in dem Archiv mit der Delphi Unit enthalten und kann von unserer Internetseite http://w3.btn.de/entwickler.php runtergeladen werden.

```
unit btnSMSExampleForm;

interface

uses
        Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
        StdCtrls, btnsms;
type
        TBtnSMSExample = class(TForm)
        buttonSendMessage: TButton;
        labelErrorCode: TLabel;
        labelErrorMessage: TLabel;
        editDestination: TEdit;
        editText: TEdit;
        labelDestination: TLabel;
        labelText: TLabel;
        editUsername: TEdit;
        editPassword: TEdit;
        labelUsername: TLabel;
        labelPassword: TLabel;
        procedure buttonSendMessageClick(Sender: TObject);
```

```
private
        { Private-Deklarationen }
public
        { Public-Deklarationen }
end;


var
        BtnSMSExample: TBtnSMSExample;

Implementation

{\$R *.DFM}

procedure TBtnSMSExample.buttonSendMessageClick(Sender: TObject);

        var
        creds, msg, result :Pointer;
        username, password, destination, text :PChar;
        errorcode :Integer;
begin
        { convert Pascal-strings into PChar's }
        username := PChar(editUsername.Text);
        password := PChar(editPassword.Text);
        destination := PChar(editDestination.Text);
        text := PChar(editText.Text);
        { build the SMS message }
        creds := btnCredsNew(username,password);
        msg := btnSmsNew();
        btnMessageSetCreds(msg,creds);
        btnMessageAddDestination(msg,destination,StrLen(destination));
        btnMessageSetText(msg,text,StrLen(text));
        { send the SMS }
        result := btnMessageSend(msg);
        { do some error reporting }
        errorcode := btnResultGetErrorcode(result);
        labelErrorcode.Caption := 'Error Code: '+IntToStr(errorcode);
        if errorcode <> BTN_ERROR_SUCCESS then
        begin
        labelErrorMessage.Caption := 'Error Message: '+string(btnResultGetErrorMsg(result));
        end;
        { free all structures }
        btnSmsFree(msg);
        btnResultFree(result);
        btnCredsFree(creds);

end;

end.
```

## 4.4 C

### 4.4.1 Allgemeines

Natürlich kann die btnSMS-Bibliothek auch in einer Umgebung mit verwaltetem Code, wie z.B. bei C# eingesetzt werden. Allerdings müssen die Funktionen die benutzt werden sollen einzeln mit einer DllImport-Anweisung importiert werden. Damit diese auch funktioniert muss über eine using-Anweisung der Namesraum System.Runtime.InteropServices eingebunden werden. Die DLL muss sich nachher bei der Ausführung des Programms in einem, für das Programm auffindbaren Ort beenden (siehe Seite 6).

### 4.4.2 Beispiel

```
using System;
using System.Runtime.InteropServices;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace WindowsApplication1
{
public partial class Form1 : Form
{
        [DllImport("btnsms.dll", EntryPoint = "btnSmsSendSimple")]
        public static extern int btnSmsSendSimple(string userid,
                                                  string password,
                                                  string dest,
                                                  string text);
public Form1()
{
        InitializeComponent();

}
private void button1_Click(object sender, EventArgs e)
{
        btnSmsSendSimple("TST00000", "test", "+491712345678", "Testnachricht");
                }
        }
}
```

# Kapitel 5

## Referenz

Die Referenz ist für Programmierer gedacht, die diese Bibliothek benutzen mochten um SMS Nachrichten zu versenden. Es werden keine Konstanten, Strukturen und Funktionen erläutert, die nur intern verwendet werden. Die dazugehörige Dokumentation findet sich im Quelltext der Bibliothek als Kommentare.

### 5.1 Konstanten

#### 5.1.1 Strukturtypen

**BTN MESSAGE TYPE SMS 1**

Gibt an, dass diese Struktur den Versand von SMS Nachrichten beschreibt.

**BTN MESSAGE TYPE MMS 2**

Gibt an, dass diese Struktur den Versand von MMS Nachrichten beschreibt.

#### 5.1.2 Zugangsdaten

**BTN LENGTH USERNAME 10**

Ein Benutzername kann maximal 10 Zeichen lang sein.

**BTN LENGTH PASSWORD 15**

Das Passwort kann maximal 15 Zeichen lang sein.

**BTN LENGTH CUSTOMER 10**

Die Kundennummer kann maximal 10 Zeichen lang sein.

#### 5.1.3 Proxy-Server

**BTN PROXY TYPE SOCKS4 4**

Ein SOCKS4-Proxy-Server wird durch die Zahl 4 gekennzeichnet.

**BTN PROXY TYPE SOCKS5 0**

Ein SOCKS5-Proxy-Server wird durch die Zahl 5 gekennzeichnet.

### 5.1.4 Tarife

**BTN TARIF ECO 15**
Die Kennung für den Eco-Tarif ist 15. Diese Kontante kann der Funktion btnSmsSetTarif(. . . ) übergeben werden um diesen Tarif zum Versand zu benutzen.

**BTN TARIF STANDARD 20**

Die Kennung für den Standard-Tarif ist 20. Diese Kontante kann der Funktion btnSmsSetTarif(. . . ) übergeben werden um diesen Tarif zum Versand zu benutzen.

**BTN TARIF PREMIUM 30**

Die Kennung für den Premium-Tarif ist 30. Diese Kontante kann der Funktion btnSmsSetTarif(. . . ) übergeben werden um diesen Tarif zum Versand zu benutzen.

### 5.1.5 Nachrichtentypen

**BTN TYPE NORMAL 65**

Die Kennung für eine normale 160 Zeichen SMS Nachricht ist 65. Diese Konstante kann der Funktion btnSmsSetType(. . . ) übergeben werden um den Typ der zu versendenden SMS Nachricht einzustellen.

**BTN TYPE LONG 66**

Die Kennung für eine lange SMS Nachricht, die auch mehr als 160 Zeichen enthalten kann und als mehrere verkettete SMS gesendet wird ist 66. Diese Konstante kann der Funktion btnSmsSetType(. . . ) übergeben werden um den Typ der zu versendenden SMS Nachricht einzustellen.

**BTN TYPE FLASH 67**

Die Kennung für eine 160 Zeichen SMS Nachricht, die direkt auf dem Display des Empfängers angezeigt wird ist 67. Diese Konstante kann der Funktion btnSmsSetType(. . . ) übergeben werden um den Typ der zu versendenden SMS Nachricht einzustellen.

### 5.1.6 Ergebnis

**BTN ERROR SUCCESS 0**

Die Kennung für einen gänzlich erfolgreichen Versand ist 0. Mit dieser Konstante kann zusammen mit der Funktion btnResultGetErrorcode(. . . ) überprüft werden, ob der Versand an alle Empfänger funktioniert hat.

**BTN ERROR NOCREDIT 1**

Wenn kein Guthaben mehr auf Ihrem Prepaid-Konto mehr vorhanden ist, wird ein Fataler Fehler mit dem Fehlercode 1 zurückgegeben. Dies kann mit dieser Konstante und der Funktion btnResultGetErrorcode(. . . ) überprüft werden.

**BTN ERROR WRONGCREDS 2**

Sind BenutzerID und/oder Passwort falsch, wird ein Fehlercode 2 zurückgegeben. Dies kann mit dieser Konstante und der Funktion btnResultGetErrorcode(. . . ) überprüft werden.

**BTN ERROR INTERNAL 3**

Ist auf der Serverseite ein interner Fehler passiert, wird ein Fehlercode 3 zurückgegeben. Dies kann mit dieser Konstante und der Funktion btnResultGetErrorcode(. . . ) überprüft werden.

**BTN ERROR NOACCESS 4**

Falls der angegebene Benutzeraccount für die angeforderten Versandoptionen nicht berechtig ist, wird ein Fehlercode 4 zurückgegeben. Dies kann mit dieser Konstante und der Funktion btnResultGetErrorcode(. . . ) überprüft werden.

**BTN ERROR CLOSEDACCOUNT 5**

Ist der angegebene Benutzeraccount gesperrt worden, wird ein Fehlercode 5 zurückgegeben. Dies kann mit dieser Konstante und der Funktion btnResultGetErrorcode(. . . ) überprüft werden.

**BTN ERROR WRONGIP 6**

Ist für den angegebenen Benutzeraccount die Überprüfung der IP-Adresse aktiviert und entspricht die IP-Adresse des requests nicht einer der zuvor bei BTN angegebenen, wird ein Fehlercode 6 zurückgegeben. Dies kann mit dieser Konstante und der Funktion btnResultGetErrorcode(. . . ) überprüft werden.

**BTN ERROR WRONGSMS 7**

Werden für den Versand mehrere miteinander unvereinbare Optionen aktiviert, oder fehlen bestimmte Angaben für den Versand, so wird ein Fehlercode 7 zurückgegeben. Dies kann mit dieser Konstante und der Funktion btnResultGetErrorcode(. . . ) überprüft werden.

**BTN ERROR WRONGXML 9**

Ist die XML-Datei nicht wohlgeformt oder entspricht sie nicht den Spezifikationen der DTD, wird ein Fehlercode 9 zurückgegeben. Dies kann mit dieser Konstante und der Funktion btnResultGetErrorcode(. . . ) überprüft werden

**BTN ERROR PARTIAL SUCCESS 200**

Der Versand der SMS Nachrichten hat nur an bestimmte Empfänger funktioniert. Welche Empfängernummern fehlgeschlagen sind, kann mit den btnResultItem. . . (. . . )-Funktionen herausgefunden werden.

### 5.1.7 Benutzerinformationen

**BTN USERTYPE FREE 10**

Die Kennung für einen Benutzer der nur kostenlose SMS Nachrichten senden darf, ist 10. Dieser Wert kann mit dem Rückgabewert der Funktion btnUserstatusGetType(. . . ) abgeglichen werden.

**BTN USERTYPE PREPAID 20**

Die Kennung für einen Benutzer der SMS Nachrichten auf Guthabenbasis versendet ist 20. Dieser Wert kann mit dem Rückgabewert der Funktion btnUserstatusGetType(. . . ) abgeglichen werden.

**BTN USERTYPE BILLED 30**

Die Kennung für einen Benutzer der SMS Nachrichten auf Rechnung versendet, ist 30. Dieser Wert kann mit dem Rückgabewert der Funktion btnUserstatusGetType(. . . ) abgeglichen werden.

## 5.2 Strukturen

Im Normalfall ist es nicht notwendig auf Elemente von Strukturen direkt zuzugreifen. Es gibt für alle Operationen entsprechende Funktionen. Dadurch wird eine größtmögliche Kompatibilität mit anderen Programmiersprachen als C/C++ erreicht, die entweder keine Strukturen in dieser Form kennen oder die eine zu C/C++ inkompatible Struktur-Implementation besitzen.
Trotzdem werden hier alle Strukturen im Detail beschrieben, damit deren Verwendung eindeutig definiert ist.

```
struct btnCreds f
        char username[BTN LENGTH USERNAME+1];
        char password[BTN LENGTH PASSWORD+1];
        char customer[BTN LENGTH CUSTOMER+1];
        char *proxy url;
        size t proxy url size;
        long proxy port;
        int proxy type;
}
```

Diese Struktur enthält die zur Autorisierung benötigten Informationen. Grundsätzlich ist es so gedacht, dass in einem Programm eine solche Struktur angelegt wird, die dann über Zeiger immer wieder referenziert werden kann.

***Beschreibung der Elemente:***

***username***
Enthält den Benutzernamen mit der maximalen Länge
BTN LENGTH USERNAME.

***password***
Enthält das Passwort des Benutzers mit der maximalen Länge
BTN LENGTH PASSWORD.

***customer***
Enthält eine Kundennummer mit der maximalen Länge
BTN LENGTH CUSTOMER.
Diese ist frei definierbar. Alle in einem Monat verwendeten Kundennummern werden auf der Rechnung gesondert aufgeschlüsselt
.

***proxy url***
Enthält die URL eines Proxy-Servers, falls ein solcher benutzt werden soll. Handelt es sich um einen Null-Zeiger, wird kein Proxy benutzt.

***proxy url size***
Falls in proxy url die Adresse eines Proxy-Server angegeben ist, enthält dieses Feld die Länge dieser Adresse.

***proxy port***
Falls ein anderer Port als der Standard-HTTP-Port 80 für den Proxy-Server verwendet werden soll, muss er in diesem Feld eingestellt sein. Eine 0 bedeutet hier die Standardeinstellung.

***proxy type***
Mit diesem Feld kann der Typ des Proxy-Server eingestellt werden. Eine 0 steht hier für einen Standard-HTTP-Proxy (siehe auch Seite 17).

```
struct btnDestination f
        char *destination;
        char *greeting;
        size t destSize;
        size t greetSize;
        ptrBtnDestination next;
}
```

Diese Struktur enthält alle Informationen die für den Versand an eine spezielle Empfängernummer vonnöten sind. Da es sich um eine verkettete liste handelt, ist auch ein Verweis auf das nächste Element in der Liste enthalten.

***Beschreibung der Elemente:***

***destination***
Enthält die Empfänger-Telefonnumer eines Empfängers der SMS Nachricht (eine btnSms Struktur) zu der diese btnDestination Struktur gehört. Beim tatsächlichen Versand darf dieses Element nicht leer und nicht null sein.

***greeting***
Enthält den Text, der als persönliche Begrüßung in der Nachricht an diesen Empfänger eingef• ugt wird. Dies kann z.B. der Name des Empfängers sein. Dieses Element muss nur dann gefüllt sein, wenn auch eine SMS Nachricht versendet wird, bei der persönliche Begrüßungen eingefügt werden (siehe btnSmsSetGreetSubst(. . . ) auf Seite
33). Ansonsten darf es leer oder null sein.

***destSize***
Die Länge des Puffers auf den der Zeiger *destination* zeigt.

***greetSize***
Die Länge des Puffers auf den der Zeiger greeting zeigt.

***next***
Zeigt auf die nächste btnDestination-Struktur in der Liste der Empfänger.

```
struct btnMessage f
        int mtype;
        ptrBtnCreds credentials;
        ptrBtnDestination _rstDestination;
        ptrBtnDestination lastDestination;
        size t countDestination;
        char *originator;
        size t originatorSize;
        char *message;
        size t messageSize;

}
```

In dieser Struktur sind die gemeinsamen Elemente von btnSms und btnMms zusammengefasst. Diese können dadurch von einheitlichen Funktionen benutzt werden, egal ob es sich um eine Struktur handelt die den Versand von SMS oder MMS Nachrichten handelt. Durch das Element mtype kann unterschieden werden, welche Struktur tatsächlich vorhanden ist.

***Beschreibung der Elemente dieser Struktur:***

***mtype***
Enthält den Typ der btnMessage-Struktur.
***credentials***
Zeiger auf die, zum Versand dieser Nachricht benutzte, btnCreds-Struktur.
***_rstDestination***
Eine verkettete Liste die alle Empfängernummern enthält, an die diese Nachricht versendet wird.
***lastDestination***
Ende der der Verketteten List der Empfänger.
***countDestination***
Die momentane Anzahl an Empfängernummern in der Liste.
***originator***
Der Absender der Nachrichten.
***originatorSize***
Die Länge des Absenders der Nachrichten.
***message***
Der Nachrichtentext.
***messageSize***
Die Länge des Nachrichtentextes.

```
struct btnSms f
        int mtype;
        ptrBtnCreds credentials;
        ptrBtnDestination _rstDestination;
        ptrBtnDestination lastDestination;
        size t countDestination;
        char *originator;
        size t originatorSize;
        char *message;
        size t messageSize;
        char* greetSubst;
        size t greetSubstSize;
        char *binaryData;
        size t binaryDataSize;
        char *binaryDataType;
        size t binaryDataTypeSize;
        int tarif;
        int type;
        time t deliveryDate;
        int ags;
        char *statusEmail;
        size t statusEmailSize;
        int statusHours;

}
```

Eine solche Struktur bildet genau einen Versandvorgang einer SMS Nachricht an (nahezu) beliebig viele Empfänger ab. Die wichtigsten Elemente sind dabei der Zeiger auf die Zugangsdaten (credentials), die Liste der Empfängernummern (_rstDestination, lastDestination) und der Text der Nachricht (message).

***Beschreibung der Elemente dieser Struktur:***

***mtype***
Enthält den Typ der btnMessage-Struktur, bei einer SMS Nachricht ist dieser immer 1 (BTN MESSAGE TYPE SMS).
***credentials***
Zeiger auf die, zum Versand dieser SMS Nachricht benutzte, btnCreds-Struktur.
***_rstDestination***
Eine verkettete Liste die alle Empfängernummern enthält, an die diese SMS Nachricht versendet wird.
***lastDestination***
Ende der der Verketteten List der Empfänger.
***countDestination***
Die momentane Anzahl an Empfängernummern in der Liste.
***Originator***
Der Absender der SMS Nachrichten.
***originatorSize***
Die Lange des Absenders der SMS Nachrichten.
***message***
Der Nachrichtentext.
***messageSize***
Die Länge des Nachrichtentextes.

***greetSubst***
Der Ersetzungstext für eine persönliche Begrüßung.
***greetSubstSize***
Die Länge des Ersetzungstextes für eine persönliche Begrüßung.
***binaryData***
Die zu versendenden Binärdaten.
***binaryDataSize***
Die Länge der zu versendenden Binärdaten.
***binaryDataType***
Datentyp der zu versendenden Binärdaten.
***binaryDataTypeSize***
Die Länge der Datentypangabe.
***Tarif***
Der Tarif in dem die SMS Nachrichten versendet werden.
***Type***
Der Typ der SMS Nachrichten.
***deliveryDate***
Das Datum, an dem die SMS Nachrichten ausgeliefert werden.
***Flag***
Verschiedene Flags für zusätzliche Optionen beim Versand. Enthält im Moment nur das \Statusreport" als Bit 0 Flag.
***statusEmail***
Die E-Mail-Adresse an die der Statusreport gesendet wird.
***statusEmailSize***
Die Länge der Email-Adresse.
***statusHours***
Die Anzahl an Stunden die nach dem Versand gewartet wird, bis der Statusreport erstellt wird.

```
struct btnMms f
        int mtype;
        ptrBtnCreds credentials;
        ptrBtnDestination _rstDestination;
        ptrBtnDestination lastDestination;
        size t countDestination;
        char *originator;
        size t originatorSize;
        char *message;
        size t messageSize;
        char *headline;
        size t headlineSize;
        char *picture;
        size t pictureSize;
        char *sound;
        size t soundSize;

}
```

Mit einer btnMms-Struktur wird genau ein Versandvorgang einer MMS Nachricht an nahezu beliebig viele Empfänger abgebildet. Dabei ist es egal ob nur ein Bild, ein Ton, ein Text oder jegliche Kombination aus diesen Teilen versendet wird.

***Beschreibung der Elemente dieser Struktur:***

***mtype***
Enthält den Typ der btnMessage-Struktur, bei einer MMS Nachricht ist dieser immer 2 (BTN MESSAGE TYPE MMS).
***credentials***
Zeiger auf die, zum Versand dieser MMS Nachricht benutzte, btnCreds-Struktur.
***_rstDestination***
Eine verkettete Liste die alle Empfängernummern enthält, an die diese MMS Nachricht versendet wird.
***lastDestination***
Ende der der Verketteten List der Empfänger.
***countDestination***
Die momentane Anzahl an Empfängernummern in der Liste.
***originator***
Der Absender der MMS Nachrichten.
***originatorSize***
Die L• ange des Absenders der MMS Nachrichten.
***message***
Der Nachrichtentext.
***messageSize***
Die L• ange des Nachrichtentextes.
***headline***
Der Betre_ der Nachricht.
***headlineSize***
Die Länge des Betreffs.
***picture***
Zeiger auf einen Puffer für die Binärdaten eines Bildes in der MMS.
***pictureSize***
Größe des Puffers für das Bild.
***sound***
Zeiger auf einen Puffer für die Binärdaten einer Audiodatei in der MMS.

***soundSize***
Größe des Puffers für die Audiodaten.

```
struct btnResultItem f
char *destination;
int errorcode;
char *result;
char *errorMsg;
ptrBtnResultItem next;

]
```

Die btnResult-Struktur, die man nach dem Versand von SMS Nachrichten auslesen kann, enthält eine Liste von btnResultItem-Strukturen. Jede dieser Strukturen repräsentiert dabei eine Empfängernummer. Sollte der Versand nicht vollständig sondern nur teilweise erfolgreich gewesen sein, kann die Liste dieser Strukturen dazu benutzt werden, herauszufinden, an welche Empfängernummern nicht erfolgreich versendet werden konnte.

***Beschreibung der Elemente dieser Struktur:***

***destination***
Die Empfängernummer.
***errorcode***
Der Fehlercode.
***errorMsg***
Die ausführliche Fehlernachricht.
***next***
Das nächste Element in der Liste der btnResultItem-Strukturen.

```
struct btnResult f
int errorcode;
char *errorMsg;
char *accountBalance;
char *statusReportId;
ptrBtnResultItem _rst, last;

}
```

Werden SMS Nachrichten, die durch eine btnSms-Struktur beschrieben wurden versendet, gibt die btnSmsSend(. . . )-Funktion einen Zeiger auf eine btnResult-Struktur zurück. Diese enthält Informationen darüber, wie der Versand verlaufen ist.

***Beschreibung der Elemente dieser Struktur:***

***errorcode***
Der Fehlercode für die gesamte Versendung.
***errorMsg***
Die ausführliche Fehlernachricht für die gesamte Versendung.
***accountBalance***
Bei einem Vorausbezahlten Konto, das Restguthaben nach dem Versand.
***statusReportId***
Die eindeutig Identifikationsnummer dieses Versands.
***first***
Das erste Element in der Liste der btnResultItem-Strukturen.
***last***
Das letzte Element in der Liste der btnResultItem-Strukturen.

```
struct btnUserstatus f
int errorcode;
char *errorMsg;
char *userid;
int active;
int type;
char *accountBalance;
int today;
int limit;

}
```

Ein Zeiger auf eine solche Struktur wird von der Funktion btnUserstatusRequest(. . . ) zurückgegeben. Sie enthält diverse Statusinformationen über den verwendeten Benutzer.

***Beschreibung der Elemente dieser Struktur:***

***errorcode***
Der Fehlercode dieser Abfrage.
***errorMsg*** I
m Fehlerfall die ausführliche Fehlernachricht.
***userid***
Die BTN BenutzerID des Kunden.
***active***
Ob der Kunde Aktiviert ist oder nicht.
***Type***
Den Kundentyp (Kostenlos, Vorausbezahlt, auf Rechnung).
***accountBalance***
Bei einem Vorausbezahlten Konto das akutelle Guthaben.
***today***
Die Anzahl der heute bereits versendeten SMS Nachrichten.
***limit***
Das Tageslimit an SMS Nachrichten.

```
struct btnIncomingResult f
int errorcode;
char *errorMsg;
ptrBtnIncomingResultItem _rst, last;

}
```

Werden SMS Nachrichten über die Funktion btnSmsReceive(. . . ) empfangen, so gibt diese Funktion bei einer empfangenen Nachricht oder einem Fehler einen Zeiger auf eine btnIncomingResult-Struktur zurück, wenn eine Nachricht empfangen wurde oder ein Fehler aufgetreten ist. Falls keine Nachricht empfangen wurde, geben die Funktionen einen Null-Zeiger zurück.

***Beschreibung der Elemente dieser Struktur:***

***errorcode***
Der Fehlercode für den Empfangsversuch.
***errorMsg***
Die ausführliche Fehlernachricht für den Empfangsversuch.
***first***
Das erste Element in der Liste der btnIncomingResultItem-Strukturen.
***Last***
Das letzte Element in der Liste der btnIncomingResultItem-Strukturen.

```
struct btnIncomingResultItem f
char *originator;
char *text;
ptrIncomingBtnResultItem next;

}
```

Die btnIncomingResult-Struktur, die man nach dem Empfang von SMS Nachrichten auslesen kann, enthält eine Liste von btnIncomingResultItem-Strukturen. Jede dieser Strukturen repräsentiert dabei eine empfangene Nachricht. Es ist wichtig alle Nachrichten in der Liste auszulesen, da diese nur ein einziges Mal übermittelt werden.

***Beschreibung der Elemente dieser Struktur:***

***originator***
Die Absendernummer.
***text***
Der Text der Nachricht.
***next***
Das nächste Element in der Liste der btnIncomingResultItem-Strukturen.

## 5.3 Funktionen

### 5.3.1 Zugangsdaten

**ptrBtnCreds btnCredsNew(char *username, char *password)**
Mit dieser Funktion wird Speicher für eine neue btnCreds-Struktur reserviert. Dabei werden username und password in die neue Struktur hineinkopiert. Nach dem Aufruf dieser Funktion wird also kein weiterer Zugriff_ auf die übergebenen Zeiger erfolgen.

**Rückgabewert:**
Zurückgegeben wird ein Zeiger auf die neue Struktur. Der Speicher für diese Struktur kann mit Hilfe der Funktion btnCredsFree(. . . ) wieder freigegeben werden.

**Beschreibung der Parameter:**
***username*** Muss auf einen Null-terminierten String zeigen, der die BTN BenutzerID enthält.
***password*** Muss auf einen Null-terminierten String zeigen, der das zu der BenutzerID gehörende Passwort enthält.

**int btnCredsSetCustomer(ptrBtnCreds creds, char *customer)**
Diese Funktion setzt in einer btnCreds-Struktur zusätzlich noch das Feld der \Kundennummer". Diese ist frei wählbar. Jede Kundennummer, die in einem Monat verwendet wird, taucht gesondert aufgeschlüsselt auf der Rechnung auf. Dadurch ist beispielsweise der Versand von SMS Nachrichten einfach verschiedenen Kostenstellen zuzuordnen.

**Beschreibung der Parameter:**
***creds*** Ein Zeiger auf die btnCreds-Struktur die manipuliert werden soll.
***customer*** Muss auf einen Null-terminierten String zeigen, der eine frei wählbare Kundennummer enthält.

**Rückgabewert**:
Diese Funktion gibt immer 1 zurück.

**Char * btnCredsSetProxyUrl(ptrBtnCreds creds, char *proxy url, size t size)**

Hier kann die URL eines Proxyserver gesetzt werden, falls dieser für die Verbindung zum Internet notwendig ist. Das Format lautet bei einem HTTP-Proxy z.B. http://proxy.mein-unternehmen.de[:Port]. Damit werden alle Verbindungen über diesen Proxy-Server hergestellt.

**Beschreibung der Parameter:**
***creds*** Ein Zeiger auf die btnCreds-Struktur die manipuliert werden soll.
***Proxy_url*** Muss die URL des Proxyservers enthalten. Ein Null-Zeiger deaktiviert den Proxy wieder.
***size*** Die Länge des Puffers auf den proxy url zeigt.

**Rückgabewert**:
Gibt einen Zeiger auf den neu reservierten Speicherbereich für die URL der Proxy-Server zurück.

**Void btnCredsSetProxyPort(ptrBtnCreds creds, long port)**

Die Angabe der Portnummer kann auch in die URL des Proxy-Servers integriert werden. Sollte dies nicht gewünscht sein, so kann sie hier auch über den separaten Funktionsaufruf angegeben werden.

**Beschreibung der Parameter:**
***creds*** Ein Zeiger auf die btnCreds-Struktur die manipuliert werden soll.
***port*** Die Portnummer, auf der der Proxy-Server angesprochen werden soll.
**void btnCredsSetType(ptrBtnCreds creds, int type)**
Hier kann auch ein anderer Typ von Proxy-Server eingestellt werden. Im Normalfall wird ein HTTP-Proxy angenommen. Durch die Angabe der Konstanten BTN PROXY TYPE SOCKS4 und BTN PROXY TYPE SOCKS5 werden auch diese Arten unterstützt.

**Beschreibung der Parameter:**
***creds*** Ein Zeiger auf die btnCreds-Struktur die manipuliert werden soll.
***Type*** Übergabe einer Konstante, die den Typ des Proxy-Servers angibt.

**int btnCredsSetLogin(ptrBtnCreds creds, char *proxy username, char *proxy password)**
Falls der Proxy-Server einen Login erwartet, kann dieser mit Hilfe dieser Funktion gesetzt werden.

**Rückgabewert:**
Gibt bei Erfolg 1 zurück, ansonsten 0.

**Beschreibung der Parameter:**
***creds*** Ein Zeiger auf die btnCreds-Struktur die manipuliert werden soll.
***proxy username*** Der Name des Benutzerkontos, der zum Proxy-Server gesendet werden soll.
***proxy password*** Das zu dem Benutzerkonto gehörende, gültige Passwort.

**Void btnCredsFree(ptrBtnCreds creds)**
Die zuvor mit btnCredsNew(. . . ) erstellte Struktur, auf die der übergebene Zeiger weisen muss, kann mit dieser Funktion wieder freigegeben werden.

### 5.3.2 Gemeinsame Funktionen für SMS und MMS Nachrichten

**ptrBtnDestination btnMessageAddDestination(ptrBtnMessage msg, char *dest, size t size)**

Diese Funktion dient dazu einer btnMessage-Struktur eine weitere Empfängertelefonnummer hinzuzufügen. Dabei wird für den neuen Empfänger Speicher reserviert, so dass ein späterer Zugriff auf den Puffer des übergebenen Zeigers nicht mehr notwendig ist.

**Beschreibung der Parameter:**

***msg*** Ein Zeiger auf die btnMessage-Struktur, die manipuliert werden soll.
***dest*** Ein Zeiger auf einen Puffer, der die hinzuzufügende Empfängernummer enthält.
***size*** Die Länge des Puffers auf den dest zeigt.

**Rückgabewert**:
Zurückgegeben wird ein Zeiger auf die neu erstellte btnDestination-Struktur oder Null falls ein Fehler aufgetreten ist.

```
ptrBtnDestination btnMessageAddDestinationGreet(
ptrBtnMessage msg,
char *dest, size t destSize,
char *greeting, size t greetSize
)
```

Diese Funktion dient dazu einer btnMessage-Struktur eine weitere Empfängertelefonnummer mit persönlicher Begrüßung hinzuzufügen. Dabei wird für den neuen Empfänger und die Begrüßung Speicher reserviert, so dass ein späterer Zugriff auf den Puffer der übergebenen Zeiger nicht mehr notwendig ist.

Damit die persönliche Begrüßung tatsächlich auch in die Nachricht eingefügt wird, muss mit der Funktion btnSmsSetGreetSubst(. . . ) ein Zeichenfolge definiert werden, die im Text der Nachricht vorkommt und die durch den hier (pro Empfänger) definierten Text ersetzt werden soll.

**Beschreibung der Parameter:**
***msg*** Ein Zeiger auf die btnMessage-Struktur, die manipuliert werden soll.
***dest*** Ein Zeiger auf einen Puffer, der die hinzuzufügende Empfängernummer enthält.
***destSize*** Die L• ange des Pu_ers auf den dest zeigt.
***greeting*** Ein Zeiger auf einen Puffer der den Text, der als persönliche Begrüßung eingefügt werden soll, enthält.
***greetSize*** Die Länge des Puffers auf den greeting zeigt.

**Rückgabewert:**
Zurückgegeben wird ein Zeiger auf die neu erstellte btnDestination-Struktur oder Null falls ein Fehler aufgetreten ist.

**char * btnMessageSetText(ptrBtnMessage msg, char *text, size t size)**
Mit dieser Funktion kann der Text der zu versendenden Nachricht gesetzt werden. Für den Text wird neuer Speicher allokiert, so dass nach dem Aufruf dieser Funktion kein Zugriff mehr auf den übergeben Zeiger stattfindet.

**Beschreibung der Parameter:**
***msg*** Ein Zeiger auf die btnMessage-Struktur, die manipuliert werden soll.
***text*** Ein Zeiger auf einen Puffer, der den Text der Nachricht enthält.
***size*** Die Länge des Puffers auf den text zeigt.

**Rückgabewert:**
Gibt einen Zeiger auf den neu allokierten Speicher für den Text der Nachricht zurück. Falls ein Fehler auftrat, wird ein Null-Zeiger zurückgegeben.

**ptrBtnResult btnMessageSend(ptrBtnMessage msg)**
Diese Funktion versendet die durch die übergebene btnMessage-Struktur beschriebenen SMS oder MMS Nachrichten.

**Beschreibung der Parameter:**
***msg*** Ein Zeiger auf die btnMessage-Struktur, deren SMS oder MMS Nachrichten versendet werden sollen.

**Rückgabewert:**
Zurückgegeben wird eine btnResult-Struktur, die mit den entsprechenden Funktionen ausgewertet werden kann. Diese kann muss der Auswertung mit btnResultFree(. . . ) wieder freigegeben werden. Tritt ein Fehler auf, der das Zurückgeben einer solchen Struktur verhindert (z.B. zu wenig freier Speicher) wird ein Null-Zeiger zurückgegeben.

**Void btnMessageSetCreds(ptrBtnMessage msg, ptrBtnCreds creds)**
Diese Funktion setzt die Zugangsdaten, die beim Versand benutzt werden. Da diese sich im Allgemeinen nicht • andern, werden diese in einer seperaten btnCreds-Struktur gespeichert, die über mehrere Versand-Operationen hinweg benutzt werden kann. Die Daten werden nicht aus der übergebenen btnCreds-Struktur herauskopiert. Diese muss also beim Aufruf von btnMessageSend(. . . ) noch im Speicher vorhanden sein und darf nicht freigegeben werden.

**Beschreibung der Parameter:**
***msg*** Ein Zeiger auf die btnMessage-Struktur, die manipuliert werden soll.
***creds*** Ein Zeiger auf eine btnCreds-Struktur, deren Zugangsdaten zum Versand verwendet werden.

**char * btnMessageSetOriginator(ptrBtnMessage msg, char *orig, size t size)**

Mit dieser Funktion wird der Absender der SMS oder MMS Nachricht gesetzt. Dieser darf bei SMS entweder aus 16 Ziffern zzgl. Pluszeichen (+) oder 11 alphanumerischen Zeichen bestehen, bei MMS aus einer gültigen Telefonnummer oder einer Email-Adresse. Für den Empfänger wird zusätzlich Speicher reserviert, so dass ein späterer Zugriff auf den übergebenen Zeiger nicht mehr notwendig ist.

**Rückgabewert:**
Diese Funktion gibt einen Zeiger auf den neu reservierten Speicherbereich für den Absender zurück. Trat ein Fehler auf, wird Null zurückgegeben.

**Beschreibung der Parameter:**
***msg*** Ein Zeiger auf die btnMessage-Struktur, die manipuliert werden soll.
***orig*** Ein Zeiger auf einen Puffer, der den Absender der Nachricht enthält.
***size*** Die Länge des Puffers auf den orig zeigt.

**Rückgabewert:**
Ein Zeiger auf den für den Absender reservierten Speicherbereich. Falls ein Fehler auftrat, ein Null-Zeiger.

### 5.3.3 Funktionen nur für SMS Nachrichten

**ptrBtnSms btnSmsNew()**
Mit dieser Funktion kann eine neue btnSms-Struktur erzeugt werden. Dabei wird für die Struktur entsprechend Speicher allokiert und die Elemente der Struktur auf sinnvolle Standardwerte gesetzt, so dass eine einfache 160 Zeichen SMS Nachricht versendet werden kann, wenn die Zugangsdaten (btnMessageSetCreds(. . . ), mindestens ein Empfänger (btnMessageAddDestination(. . . )) und der Text (btnMessageSetText(. . . )) gesetzt werden.

**Rückgabewert:**
Die Funktion gibt einen Zeiger auf die neue BtnSms-Struktur zurück. Diese kann dann mit Hilfe der anderen Funktionen manipuliert werden. Sollte ein Fehler auftreten (z.B. weil nicht mehr genügend Speicher vorhanden ist) gibt die Funktion einen Null-Zeiger zurück. int btnSmsSendSimple(char *username, char *password, char *dest, char *text)

Mit dieser Funktion kann schnell und einfach eine 160 Zeichen SMS Nachricht ohne frei definierbaren Absender an einen einzelnen Empfänger versendet werden. Bei dieser Funktion ist sicherzustellen, dass alle übergebenen Zeiger auf Strings auch mit einer Null terminiert sind.

**Beschreibung der Parameter:**
***username*** Muss auf einen Null-terminierten String zeigen, der die BTN BenutzerID enth• alt.
***password*** Muss auf einen Null-terminierten String zeigen, der das zu der BenutzerID gehörende Passwort enthält.
***dest*** Muss auf einen Null-terminierten String zeigen, der die Empfängernummer für die SMS Nachricht enthält.
***text*** Muss auf einen Null-terminierten String zeigen, der den Text der zu versendenden SMS Nachricht enthält.

**Rückgabewert:**
Der Rückgabewert zeigt an, ob der Versand erfolgreich war oder nicht. Er kann mit Hilfe der entsprechenden Konstanten (siehe Seite 18) ausgewertet werden.

**Chat * btnSmsSetGreetSubst(ptrBtnSms msg, char *greetSubst, size t size)**
Mit dieser Funktion kann eine Zeichenfolge definiert werden, die durch eine persönliche Begrüßung, die für jeden Empfänger unterschiedlich sein kann, ersetzt wird. Diese Zeichenfolge muss natürlich im Text der Nachricht, also in dem mit btnSmsSetText(. . . ) gesetzten Text, auch tatsächlich vorkommen.
Dabei wird für die Zeichenfolge Speicher reserviert, so dass ein späterer Zugriff auf den Puffer des übergebenen Zeigers nicht mehr notwendig ist.

**Beschreibung der Parameter:**
***msg*** Ein Zeiger auf die btnSms-Struktur, die manipuliert werden soll.
***greetSubst*** Ein Zeiger auf einen Puffer, der die Zeichenfolge enthält, die ersetzt werden soll.
***size*** Die Länge des Puffers auf den greetSubst zeigt.

**Rückgabewert:**
Ein Zeiger auf den für den Ersetzungstext reservierten Speicherbereich. Falls ein Fehler auftrat, ein Null-Zeiger.

**Char * btnSmsSetBinaryData(ptrBtnSms msg, char *buf, size t size)**
Bei SMS Nachrichten, die Logos und Klingeltöne enthalten, müssen die Binärdaten mit dieser Funktion gesetzt werden. Was es dabei für Möglichkeiten gibt ist unter XX zu erfahren. Die Binärdaten, auf die buf zeigt, werden in einen neu reservierten Speicherbereich kopiert, so dass nach dem Aufruf dieser Funktion kein Zugriff mehr auf den buf erfolgen muss.

**Beschreibung *der* Parameter:**
***msg*** Ein Zeiger auf die btnSms-Struktur, die manipuliert werden soll.
***buf*** Zeiger auf einen Puffer mit den Binärdaten.
***size*** Länge des Puffers buf.

**Rückgabewert**:
Ein Zeiger auf den für die Binärdaten reservierten Speicherbereich. Falls ein Fehler auftrat, ein Null-Zeiger.

**Void btnSmsSetType(ptrBtnSms msg, int type)**
Diese Funktion • andert den Typ einer SMS Nachricht. Dabei sollten die Konstanten für die verschiedenen Typen auf Seite 18 verwendet werden.

**Beschreibung der Parameter:**
***msg*** Ein Zeiger auf die btnSms-Struktur, die manipuliert werden soll.
***type*** Der neue Typ der zu versendenden SMS Nachricht.

**Void btnSmsSetTarif(ptrBtnSms msg, int tarif)**
Diese Funktion ändert den Tarif einer SMS Nachricht. Dabei sollten die Konstanten für die verschiedenen Tarife auf Seite 18 verwendet werden.

**Beschreibung der Parameter:**
***msg*** Ein Zeiger auf die btnSms-Struktur, die manipuliert werden soll.
***tarif*** Der neue Tarif der zu versendenden SMS Nachricht.

```
oid btnSmsSetDelivery(
ptrBtnSms msg,
int day,
int month,
int year,
int hour,
int minute,

)
```

SMS Nachrichten können auch termingesteuert versendet werden. Dazu muss mit dieser Funktion der genaue Termin angegeben werden.

**Beschreibung der Parameter:**
***msg*** Ein Zeiger auf die btnSms-Struktur, die manipuliert werden soll.
***day*** Der Tag im Monat, an dem die SMS Nachricht versendet werden soll. (1-31)
***month*** Der Monat im Jahr, an dem die SMS Nachricht versendet werden soll. (1-12)
***year*** Das Jahr, in dem die SMS Nachricht versendet werden soll.
***hour*** Die Stunde am Tag, in der die SMS Nachricht versendet werden soll.
***minute*** Die Minute in der Stunde, zu der die SMS Nachricht versendet werden soll.

**Char * btnSmsSetStatusReport(ptrBtnSms msg, char *email, size t size, int hours)**
Mit dieser Funktion wird f• ur die zu versendenden SMS Nachrichten ein Statusreport angefordert. Dieser wird nach der angegebenen Anzahl an Stunden an die angegebene Email-Adresse geschickt. Nach dem Versand kann mit der btnResultGetStatusReportId(. . . )- Funktion eine eindeutige Nummer ermittelt werden, mit der die E-Mail einem Versand eindeutig zuzuordnen ist. Für die E-Mail-Adresse wird ausreichend Speicher reserviert, so dass nach dem Aufruf dieser Funktion kein Zugriff mehr auf den übergebenen Zeiger mehr stattfindet.

**Beschreibung der Parameter:**
***msg*** Ein Zeiger auf die btnSms-Struktur, die manipuliert werden soll. Email Ein Zeiger auf einen Puffer, der die E-Mail-Adresse enthält, an die der Statusreport gesendet werden soll.
***size*** Die Länge des Puffers der Email-Adresse.
***hours*** Die Anzahl an Stunden, die nach dem Versand gewartet werden soll, bis der Statusreport erstellt wird.

**Rückgabewert:**
Ein Zeiger auf den für die E-Mail-Adresse reservierten Speicherbereich. Falls ein Fehler auftrat, ein Null-Zeiger.

**void btnSmsFree(ptrBtnSms msg)**
Gibt allen zuvor allozierten Speicher für eine btnSms-Struktur wieder frei. Diese Funktion sollte nach jedem Versand aufgerufen werden, wenn die btnSms-Struktur nicht wiederverwendet werden soll.

**Beschreibung der Parameter:**
***msg*** Ein Zeiger auf eine btnSms-Struktur, deren Speicher wieder freigegeben werden soll.

### 5.3.4 Funktionen nur für MMS Nachrichten

**ptrBtnMms btnMmsNew()**
Mit dieser Funktion kann eine neue btnMms-Struktur erzeugt werden. Dabei wird für die Struktur entsprechend Speicher allokiert und die Elemente der Struktur auf sinnvolle Standardwerte gesetzt. Eine MMS Nachricht kann versendet werden, wenn die Zugangsdaten (btnMessageSetCreds(. . . )), eine Headline (btnMmsSetHeadline(. . . ), mindestens in Empfänger (btnMessageAddDestination(. . . )) und entweder ein Text (btnMessageS-etText(. . . )), ein Bild (btnMmsSetPicture(. . . )) oder ein Sound (btnMmsSetSound(. . . )) gesetzt werden.

**Rückgabewert:**
Die Funktion gibt einen Zeiger auf die neue btnMms-Struktur zurück. Diese kann dann mit Hilfe der anderen Funktionen manipuliert werden. Sollte ein Fehler auftreten (z.B. weil nicht mehr genügend Speicher vorhanden ist) gibt die Funktion einen Null-Zeiger zurück.

**char * btnMmsSetHeadline(ptrBtnMms msg, char *headline, size t size)**
Diese Funktion setzt die Headline, also quasi den Betreff, der zu versendenden MMS Nachricht. Dabei wird für den Text entsprechend Speicher reserviert, so dass nach dem Aufruf dieser Funktion nicht mehr auf den übergebenen Zeiger Bezug genommen wird.

**Beschreibung der Parameter:**
***msg*** Ein Zeiger auf die btnMms-Struktur, die manipuliert werden soll.
***headline*** Ein Zeiger auf einen Puffer, der die zu sendende Headline der MMS Nachricht enthält.
***size*** Die Länge des Puffers auf den headline zeigt.

**Rückgabewert:**
Die Funktion gibt einen Zeiger auf den Puffer zurück, der für die Headline alloziert wurde.

**Char * btnMmsSetPicture(ptrBtnMms msg, char *buf, size t size)**
Mit Hilfe dieser Funktion kann der MMS Nachricht ein Bild hinzugefügt werden. Dabei sollte es sich im Moment ausschließlich um JPEG-Bilder handeln. In einer späteren Version können auch andere Bildtypen übergeben werden.

**Beschreibung der Parameter:**
***msg*** Ein Zeiger auf die btnMms-Struktur, die manipuliert werden soll.
***buf*** Ein Zeiger auf die Binärdaten des Bildes, das hinzugefügt werden soll.
***size*** Die Länge des Puffers auf den buf zeigt.

**Rückgabewert:**
Die Funktion gibt einen Zeiger auf den Puffer zurück, der für das Bild alloziert wurde.

**char * btnMmsSetSound(ptrBtnMms msg, char *buf, size t size)**
Mit Hilfe dieser Funktion kann der MMS Nachricht eine Audiodatei hinzugefügt werden. Dabei sollte es sich im Moment ausschließlich um MIDI-Dateien handeln. In einer späteren Version können auch andere Audiotypen übergeben werden.

**Beschreibung der Parameter:**
***msg*** Ein Zeiger auf die btnMms-Struktur, die manipuliert werden soll.
***buf*** Ein Zeiger auf die Binärdaten des Sounds, der hinzugefügt werden soll.
***size*** Die Länge des Puffers auf den buf zeigt.

**Rückgabewert:**
Die Funktion gibt einen Zeiger auf den Puffer zurück, der für den Sound alloziert wurde.

**void btnMmsFree(ptrBtnMms msg)**
Gibt allen zuvor allozierten Speicher für eine btnMms-Struktur wieder frei. Diese Funktion sollte nach jedem Versand aufgerufen werden, wenn die btnMms-Struktur nicht wiederverwendet werden soll.

**Beschreibung der Parameter:**
***msg*** Ein Zeiger auf eine btnMms-Struktur, deren Speicher wieder freigegeben werden soll.

### 5.3.5 Ergebnisauswertung

**void btnResultFree(ptrBtnResult result)**
Gibt allen zuvor allozierten Speicher für eine btnResult-Struktur wieder frei. Diese Funktion sollte nach der Auswertung der Rückantwort immer aufgerufen werden. Dabei ist zu beachten, dass die Zeiger auf Zeichenketten, die von den verschiedenen Auswertungsfunktionen (ab Seite 37 - btnResultGetFatalErrorMsg(. . . ), btnResultGetAccountBa- lance(. . . ), usw.) zurückgegeben werden nicht mehr gültig sind. Der Inhalt muss vorher an eine andere Stelle im Speicher kopiert werden.

**Beschreibung der Parameter:**
***result*** Ein Zeiger auf eine btnResult-Struktur, deren Speicher wieder freigegeben werden soll. int btnResultGetErrorcode(ptrBtnResult result)
Mit dieser Funktion kann der Fehlercode einer btnResult-Struktur ausgelesen werden. Mit Hilfe der entsprechenden Konstanten (siehe Seite 18) kann überprüft werden, ob und wenn ja, woran der Versand gescheitert ist.

**Beschreibung der Parameter:**
***result*** Ein Zeiger auf eine btnResult-Struktur, deren Fataler Error Code zurückgegeben werden soll.

**Rückgabewert:**
Falls der Versand komplett erfolgreich war wird die Konstante BTN ERRORCODE SUCCESS (also 0) zurückgegeben. Ansonsten ein anderer Wert, der entweder einer der BTN ERRORCODE . . . Konstanten entspricht oder ein mit Bit 10 (1024) maskierter Wert ist, der einen Fehler auf Netzwerkebene darstellt. Fehlercodes > 1024 sind also immer Fehler der zugrundeliegenden libCURL, die auf diese Art und Weise \durchgeschleift" werden. Dabei entspricht der zurückgegebene Code minus 1024 dem entsprechenden libCURL
Fehlercode. Diese sind unter folgender Adresse zu finden:
*http://curl.haxx.se/libcurl/c/libcurl-errors.html*

**char * btnResultGetErrorMsg(ptrBtnResult result)**
Falls die der Versand, aus dem die übergangene btnResult-Struktur hervorgegangen ist, fehlgeschlagen ist, kann mit dieser Funktion eine genaue Fehlerbeschreibung ermittelt werden Diese gibt über den Fehlercode hinaus Auskunft darüber, was falsch gelaufen ist. Nach dem Aufruf von btnResultFree(. . . ) ist der von dieser Funktion zurückgegebene Zeiger nicht mehr gültig.

**Beschreibung der Parameter:**
***result*** Ein Zeiger auf eine btnResult-Struktur, deren Fatale Fehlerbeschreibung zurückgegeben werden soll.

**Rückgabewert:**
Im Fehlerfall die ausführliche Fehlerbeschreibung, ansonsten einen Null-Zeiger.

**char * btnResultGetAccountBalance(ptrBtnResult result)**
Falls die zum Versand benutzte BenutzerID über ein Prepaid-Konto verfügt, kann mitdieser Funktion das aktuelle Guthaben nach dem gerade erfolgten Versand abgefragt werden.

**Beschreibung der Parameter:**
***result*** Ein Zeiger auf eine btnResult-Struktur, deren Restguthaben zurückgegeben werden soll.

**Rückgabewert:**
Dabei handelt es sich um eine Zeichenkette, die das Guthaben in Euro, mit einem Punkt (.) als Dezimaltrennzeichen, enthält. War der benutzte Account kein Prepaid-Konto, wird von dieser Funktion eine Null zurückgegeben.

**char * btnResultGetStatusReportId(ptrBtnResult result)**
Wurde beim Versand ein Statusreport angefordert, so kann mit dieser Funktion die eindeutige Identifikationsnummer des Statusreports abgefragt werden. Diese taucht in der Email, die nach der angegebenen Zeit erstellt wird, wieder auf. Dies ermöglicht eindeutig die Verbindung eines Versands und der Status-Email herzustellen.

**Beschreibung der Parameter:**
***result*** Ein Zeiger auf eine btnResult-Struktur, deren Identifikationsnummer des Statusreports abgefragt werden soll.

**Rückgabewert:**
Falls eine Identifikationsnummer vorliegt wird diese zurückgegeben, ansonsten ein Null-Zeiger.

**ptrBtnResultItem btnResultGetFirstResultItem(ptrBtnResult result)**
Auch wenn beim Versand der SMS Nachrichten kein fataler Fehler aufgetreten ist, so kann es doch sein, dass die Nachricht an einzelne Empfängernummern nicht zugestellt werden konnte. Für jede Empfängernummer existiert eine btnResultItem-Struktur. Das erste Element aus dieser Liste wird von dieser Funktion zurückgegeben.

**Beschreibung der Parameter:**
***result*** Ein Zeiger auf eine btnResult-Struktur, deren erstes Element der Liste der btnResul- tItem-Elemente zurückgegeben werden soll.

**Rückgabewert:**
Ein Zeiger auf die erste btnResultItem-Struktur. Das erste Folgeelement erhält man, wenn man die Funktion btnResultItemGetNext(. . . ) auf das von dieser Funktion zurückgegebene anwendet.

**ptrBtnResultItem btnResultItemGetNext(ptrBtnResultItem resultitem)**
Gibt das nächste Folgeelement der übergebenen btnResultItem-Struktur zurüuck. Ist das übergebene Element das letzte Element in der Liste gibt diese Funktion Null zurück.

**Beschreibung der Parameter:**
***result*** Ein Zeiger auf eine btnResultItem-Struktur, deren Folgeelement der Liste der btnResultItem-Elemente zurückgegeben werden soll.

**Rückgabewert:**
Existiert ein weiteres Element in der Liste der btnResultItem-Strukturen, so wird es zurückgegeben, ansonsten ein Null-Zeiger.

**char * btnResultItemGetDestination(ptrBtnResultItem resultitem)**
Mit dieser Funktion kann ermittelt werden, für welche Empfängernummer diese btnResultItem-Struktur den Status des Versands enthält.

**Beschreibung der Parameter:**
***result*** Ein Zeiger auf eine btnResultItem-Struktur, deren Empfängernummer zurückgegeben werden soll.

**Rückgabewert:**
Die Empfängernummer, für die diese BtnResultItem-Struktur die Statusinformationen enthält.

**int btnResultItemGetErrorcode(ptrBtnResultItem resultitem)**
Gibt den Fehlercode für die in der btnResultItem-Struktur enthaltene Empfängernummer zurück. Eine Null (0) ist dabei ein erfolgreicher Versand und eine Eins (1) stellt einen Fehler dar. Der Grund für einen Fehler kann dann mit der Funktion btnResultItemGetErrorMsg(. . . ) ermittelt werden.

**Beschreibung der Parameter:**
***result*** Ein Zeiger auf eine btnResultItem-Struktur, deren Fehlercode zurückgegeben werden soll.

**Rückgabewert:**
Diese Funktion gibt 0 zurück, wenn der Versand an diesen Empfänger erfolgreich war und 1, falls der Versand nicht erfolgreich war.

**char * btnResultItemGetErrorMsg(ptrBtnResultItem resultitem)**
Falls der Versand an den von dieser btnResultItem-Struktur beschriebenen Empfänger fehlschlug, kann mit dieser Funktion eine Fehlerbeschreibung ermittelt werden, die den Grund für das Fehlschlagen angibt. Ist kein Fehler aufgetreten, gibt diese Funktion einen Null-Zeiger zurück, der nicht benutzt werden darf.

**Beschreibung der Parameter:**
***result*** Ein Zeiger auf eine btnResultItem-Struktur, deren Fehlerbeschreibung zurückgegeben werden soll.

**Rückgabewert:**
Falls der Versand an diesen Empfänger nicht erfolgreich war, wird eine ausführliche Beschreibung des Fehlers zurückgegeben. War der Versand erfolgreich, gibt diese Funktion einen Null-Zeiger zurück.

### 5.3.6 Benutzerinformationen

**btnUserstatus btnUserstatusRequest(ptrBtnCreds creds)**
Mit dieser Funktion können verschiedene Informationen über das eigene Benutzerkonto abgefragt werden. So kann z.B. vor einem Versand überprüft werden, ob genug Guthaben auf dem Kundenkonto vorhanden ist.

**Beschreibung der Parameter:**
***creds*** Ein Zeiger auf eine btnCreds-Struktur, die die Zugangsdaten für das abzufragende Benutzerkonto enthält.

**Rückgabewert:**
Zurückgegeben wird ein Zeiger auf eine btnUserstatus-Struktur. Diese kann mit den entsprechenden
Funktionen ausgewertet werden. Trat ein Fehler auf, der das Erzeugen einer solchen Struktur verhindert, wird ein Null-Zeiger zurückgegeben.

**int btnUserstatusGetErrorcode(ptrBtnUserstatus status)**
Mit dieser Funktion kann der Fehlercode einer btnUserstatus-Struktur ausgelesen werden. Mit Hilfe der entsprechenden Konstanten (siehe Seite 18) kann überprüft werden, ob und wenn ja, woran die Abfrage gescheitert ist.

**Beschreibung der Parameter:**
***status*** Ein Zeiger auf eine btnUserstatus-Struktur, die ausgelesen werden soll.

**Rückgabewert:**
Falls die Abfrage erfolgreich war wird die Konstante BTN ERRORCODE SUCCESS (also 0) zurückgegeben. Ansonsten ein anderer Wert, der einer der BTN ERRORCODE . . . Konstanten entspricht.

**char * btnUserstatusGetErrorMsg(ptrBtnUserstatus status)**
Falls die Abfrage der Benutzerinformationen nicht erfolgreich war, kann mit dieser Funktion zusätzlich zu dem Fehlercode noch eine genaue Fehlerbeschreibung ermittelt werden.

**Beschreibung der Parameter:**
***status*** Ein Zeiger auf eine btnUserstatus-Struktur, die ausgelesen werden soll.

**Rückgabewert:**
Die ausführliche Fehlerbeschreibung, falls ein Fehler aufgetreten ist, ansonsten ein Null-Zeiger.

**char * btnUserstatusGetUserid(ptrBtnUserstatus status)**
Mit dieser Funktion kann zur Kontrolle nochmal die abgefragte BenutzerID ausgelesen werden.

**Beschreibung der Parameter:**
***status*** Ein Zeiger auf eine btnUserstatus-Struktur, die ausgelesen werden soll.

**Rückgabewert:**
Die abgefragte BTN BenutzerID. Verlief die Abfrage nicht erfolgreich, gibt diese Funktion einen Null-Zeiger zurück.

**int btnUserstatusGetActive(ptrBtnUserstatus status)**
Ermittelt ob der das abgefragte Benutzerkonto aktiv und in der Lage ist SMS Nachrichten zu versenden. Sollte diese Funktion angeben, dass das Benutzerkonto deaktiviert wurde, setzen Sie sich bitte umgehend mit BEYOND THE NET in Verbindung um den Sachverhalt zu klären.
**Beschreibung der Parameter:**
***status*** Ein Zeiger auf eine btnUserstatus-Struktur, die ausgelesen werden soll.

**Rückgabewert:**
Bei einem aktiven Benutzerkonto wird eine 1 zurückgegeben, bei einem inaktiven eine 0. Ist die Abfrage fehlgeschlagen, so wird immer eine 0 zurückgegeben.

**int btnUserstatusGetType(ptrBtnUserstatus status)**
Mit dieser Funktion kann herausgefunden werden, von welchem Typ das abgefragte Benutzerkonto ist. Dabei gibt es die Möglichkeiten, dass der Benutzer nur kostenlose (in der Funktionalität eingeschränkte) SMS Nachrichten senden darf, dass er Nachrichten nur auf Guthabenbasis versenden darf, oder dass er eine monatliche Rechnung über die versendeten Nachrichten erhalt.

**Beschreibung der Parameter:**
***status*** Ein Zeiger auf eine btnUserstatus-Struktur, die ausgelesen werden soll.

**Rückgabewert:**
Der Rückgabewert entspricht einer der BTN USERTYPE . . . Konstanten (siehe Seite 20).
Ist die Abfrage fehlgeschlagen wird immer eine Null zurückgegeben.

**char * btnUserstatusGetAccountBalance(ptrBtnUserstatus status)**
Falls die abgefragte BenutzerID über ein Prepaid-Konto verfügt, kann mit dieser Funktion das aktuelle Guthaben auf dem Kundenkonto ausgelesen werden.

**Beschreibung der Parameter:**
***status*** Ein Zeiger auf eine btnUserstatus-Struktur, die ausgelesen werden soll.

**Rückgabewert:**
Dabei handelt es sich um eine Zeichenkette, die das Guthaben in Euro, mit einem Punkt(.) als Dezimaltrennzeichen, enthält. War der benutzte Account kein Prepaid-Konto, wird von dieser Funktion eine Null zurückgegeben. Ist die Abfrage fehlgeschlagen wird immer ein Null-Zeiger zurückgegeben.

**int btnUserstatusGetToday(ptrBtnUserstatus status)**
Abfrage der Anzahl der heute bereits versendeten SMS Nachrichten.

**Beschreibung der Parameter:**
***status*** Ein Zeiger auf eine btnUserstatus-Struktur, die ausgelesen werden soll.

**Rückgabewert:**
Die Anzahl der heute bereits versendeten SMS Nachrichten. Ist die Abfrage fehlgeschlagen wird immer eine Null zurückgegeben.

**int btnUserstatusGetLimit(ptrBtnUserstatus status)**
Abfrage des Tageslimits des Benutzers.

**Beschreibung der Parameter:**
***status*** Ein Zeiger auf eine btnUserstatus-Struktur, die ausgelesen werden soll.

**Rückgabewert:**
Die Anzahl an SMS Nachrichten, die dieser Benutzer pro Tag maximal versenden darf. Ist die Abfrage fehlgeschlagen wird immer eine Null zurückgegeben.

**void btnUserstatusFree(ptrBtnUserstatus status)**
Diese Funktion gibt allen Speicher, der bei der Erstellung einer btnUserstatus-Struktur reserviert wurde. Zu beachten ist dabei, dass damit die von den Auswertungsfunktionen zurückgegebenen Zeiger ihre Gültigkeit verlieren!

**Beschreibung der Parameter:**
***status*** Ein Zeiger auf eine btnUserstatus-Struktur, die freigegeben werden soll.

### 5.3.7 Empfang von SMS Nachrichten

**ptrBtnIncomingResult btnSmsReceive(ptrBtnCreds creds, int timeout)**
Diese Funktion empfängt für den durch die • übergebene BtnCreds-Struktur bestimmten Benutzer eine oder mehrere SMS Nachrichten. Diese Funktion blockiert so lange bis derübergebene Timeout erreicht wurde oder wenn mindestens eine Nachricht empfangen wurde.Wird der Timeout auf 0 gesetzt, ist der Timeout deaktiviert und die Funktion blockiert solange bis eine Nachricht empfangen wurde.

**Beschreibung der Parameter:**
***cerds*** Ein Zeiger auf die btnCreds-Struktur, mit den Zugangsdaten des Benutzers, für die Nachrichten emfpangen werden sollen.
***timeout*** Ein Timeout in Sekunden, nach dessen Ablauf die Funktion ohne Ergebnis zurückkehren soll, falls keine Nachricht empfangen wurde.

**Rückgabewert:**
Bei einer empfangenen Nachricht oder bei Auftritt eines Fehlers wird eine btnIncomingResult-Struktur zurückgegeben, die entsprechend ausgewertet werden kann. Diese muss nach der Auswertung mit btnIncomingResultFree(. . . ) wieder freigegeben werden. Sollten keine Fehler aufgetreten sein und innerhalb des Timeouts keine Nachrichten vorliegen, wird ein Null-Zeiger zurückgegeben.

# Anhang A

## Lizenz

GNU LESSER GENERAL PUBLIC LICENSE
Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed. [This is the _rst released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software{to make sure the software is free for all its users. This license, the Lesser General Public License, applies to some specially designated software packages{typically libraries{of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you _rst think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object _les to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.
To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated

libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.
When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the \Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library. The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a \work based on the library" and a \work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

GNU LESSER GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION#

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called \this License"). Each licensee is addressed as \you".

A \library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables. The \Library", below, refers to any such software library or work which has been distributed under these terms. A \work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term \modification".) \Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition _les, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library. You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.
2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.
b) You must cause the _les modified to carry prominent notices stating that you changed the _les and the date of any change.
c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.
d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful. (For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. . You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices. Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy. This option is useful when you wish to copy part of the code of the Library into a program that is not a library.
4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange. If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place

satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a \work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a \work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than

a \work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables. When a \work that uses the Library" uses material from a header _le that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object _le uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object _le is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a \work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications. You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable \work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions _les in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the \work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception,

the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable. It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.
9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.
10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.
11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

    If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances. It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

    This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and \any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.
14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY \AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
END OF TERMS AND CONDITIONS

Ende des Dokuments