

# **BEYOND THE NET**

Internet Service GmbH

## **Dokumentation: XML/HTTP-Post Interface**

Stand: 10. Januar 2011

# Inhaltsverzeichnis

<b>Einleitung.....</b>	<b>1</b>
<b>Versenden von SMS Nachrichten.....</b>	<b>2</b>
Versand einer einfachen SMS Nachricht.....	2
Versand einer SMS Nachricht mit variablem Absender.....	5
Versand einer SMS Nachricht mit Termin.....	6
Versand von WAP-Push-/Browser-Nachrichten.....	7
Versand eines Nokia Operator Logos.....	8
Versand eines Nokia Gruppensymbols.....	10
Versand einer Nokia Bildmitteilung.....	11
Versand von Nokia Klingeltönen.....	13
Versand von Siemens Daten.....	14
Versand von reinen Binärdaten.....	15
<b>Zusätzliche Optionen.....</b>	<b>18</b>
Tarif festlegen.....	18
Priorität festlegen.....	18
Personalisierte Nachrichten versenden.....	19
Statusreport anfordern.....	19
<b>Auswertung der Rückantwort.....</b>	<b>21</b>
Bestätigungen.....	21
Angabe des Kontostandes.....	22
Fatale Fehler.....	23
Fehlercodes.....	24
<b>Nachrichten mit Termin stornieren.....</b>	<b>25</b>
URL.....	25
Aufbau der zu sendenden XML-Datei.....	25
Aufbau der zu empfangenden XML-Datei.....	26
<b>Beispiele.....</b>	<b>28</b>
Java.....	28
Perl.....	30
PHP.....	31
<b>Anhang.....</b>	<b>33</b>
A. Übersicht über die Schlüsselemente.....	33
B. DTD: btn-sms-send.....	41
C. DTD: btn-sms-response.....	42
<b>Index.....</b>	<b>43</b>

# Einleitung

Neben dem Email-to-SMS Gateway bietet BEYOND THE NET nun auch die Möglichkeit des Versands von SMS Nachrichten über ein XML/HTTP POST Interface. Diese Art des Versands von SMS Nachrichten bietet dem Versender viele Vorteile gegenüber der bewährten Email-Methode. Zum einen ist eine direkte Antwort vom Server möglich, d.h. es muss nicht auf eine Bestätigungsemail gewartet werden, sondern direkt nach Bearbeitung des Auftrags wird eine entsprechende Bestätigung erstellt. Dadurch ist wesentlich einfacher festzustellen ob eine SMS Nachricht bei uns eingegangen ist oder nicht, und wenn nicht, warum nicht.

Das XML-Interface arbeitet besonders schnell, da wesentlich mehr Aufträge gleichzeitig bearbeitet werden können als durch den Email-to-SMS Gateway.

## Versenden von SMS Nachrichten

Um eine Nachricht über den Gateway zu verschicken muss zunächst eine gültige XML Datei erstellt werden. XML-Dateien sind einfach Textdateien, die entweder mit jedem beliebigen Editor erstellt werden können, oder die man auch ganz einfach mit Hilfe von Programmen erzeugen kann. Dabei ist man auch nicht auf eine bestimmte Programmiersprache festgelegt, sondern kann jede Programmiersprache verwenden, die mit Texten umgehen kann.

Für das versenden der XML-Datei über das HTTP Protokoll ist es allerdings notwendig eine Programmiersprache zu verwenden, die Netzwerkverbindungen mindestens auf Socket-Basis unterstützt, besser aber gleich Funktionen oder Methoden für einen HTTP Aufruf mitbringt. Für Java und Perl haben beispielsweise eine solche HTTP Funktionalität. PHP hat leider keine Standardfunktionen um auf einem anderen Server einen HTTP Request durchzuführen. Hier ist es notwendig das HTTP Protokoll rudimentär selbst zu implementieren und dann eine Socketverbindung aufzubauen.

Für die genannten Programmiersprachen existiert im Kapitel Beispiele jeweils ein Listing um eine einfache SMS Nachricht zu versenden. Damit soll die Verfahrensweise eines HTTP-Post Aufrufs, in der jeweiligen Programmiersprache, verdeutlicht werden.

### Versand einer einfachen SMS Nachricht

Eine einfache SMS übermittelt einen bis zu 160 Zeichen langen Text ohne besondere Absenderkennung auf das Mobiltelefon eines oder mehrerer Mobilfunkteilnehmer. Neben der Angabe des Textes und der Zielrufnummer(n) ist es notwendig die BenutzerID und das Passwort ihres Kontos bei BEYOND THE NET anzugeben. Bei einer Nachricht wie dieser, in der keine Priorität angegeben wird, wird in dem Tarif versendet, den wir mit ihnen im Vertrag vereinbart haben. Haben wir keinen besonderen Tarif vereinbart, wird die Nachricht im Standard Tarif versendet. Den Preis für eine solche Nachricht finden Sie in ihrem Vertrag bzw. in unserer gesonderten Preistabelle, die auf unserer Internetseite (<http://www.btn.de/>) zum Download bereit steht.

Die XML-Datei muss zunächst mit einem gültigen XML-Header beginnen. Dieser sieht beispielsweise folgendermaßen aus:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Beachten sie bitte, dass der Wert des Attributs `encoding` auf die von ihnen benutzte Zeichencodierung gesetzt werden muss, damit in der Nachricht evtl. vorhandene Umlaute oder andere Sonderzeichen korrekt dargestellt werden können. Ein typischer Wert für ein Java-Programm welches SMS Nachrichten versenden soll ist `UTF-8`. Andere Programme auf Unix Umgebungen verwenden meist den Zeichensatz `ISO-8859-1`. Weitere Informationen hierzu sind auf Seite 3 Umlaute zu finden.

Auf den XML-Header folgt die Angabe des sogenannten `DOCTYPE`. Diese verweist auf die sogenannte `DTD`, welche als eine Art Schablone, für den Aufbau einer XML-Datei ist. Anhand des `DOCTYPE` und der `DTD` kann ermittelt werden ob eine XML-Datei gültig ist oder nicht. Wenn sie ohne `DOCTYPE` Angabe eine ansonsten gültige XML-Datei erstellen, wird unser Server diese Datei trotzdem zurückweisen, da er ihre XML-Datei nicht gegen die `DTD`

validieren kann. Für das versenden von SMS Nachricht über BEYOND THE NET lautet der DOCTYPE stets folgendermaßen:

```
<!DOCTYPE btn-sms-send SYSTEM "http://www.btn.de/dtd/btn-sms-send.dtd">
```

In einer XML-Datei existiert immer ein sogenanntes Root-Element. Dieses umschließt alle anderen Elemente der XML-Datei. In unserer XML-Datei heißt dieses Root-Element `btn-sms-send` und wird folgendermaßen notiert:

```
<btn-sms-send>
```

(Hier folgen die restlichen Elemente der XML-Datei)

```
</btn-sms-send>
```

Zwischen diesen beiden Einträgen sind noch drei weitere Elemente zwingend erforderlich um eine SMS Nachricht versenden zu können. Zunächst ist da das `sender` Element, welches an erster Stelle innerhalb des Root-Elementes stehen muss. Im `sender` Element müssen die Daten ihres Kontos bei BEYOND THE NET als Attribute angegeben werden. Das ist zum einen ihre BenutzerID, die als das Attribut `userid` angegeben werden muss, und zum anderen ihr Passwort, welches als Attribut `password` angegeben werden muss. Für den Fall, dass ihre BenutzerID „ABC00000“, und ihr Passwort „xyz0123“ lautet sähe dieses Element folgendermaßen aus:

```
<sender userid="ABC00000" password="xyz0123"/>
```

Auf das Element `sender` muss das Element `message` folgen. Dieses Element hat selbst keine Attribute und enthält nur andere Elemente. Bei einer einfachen SMS Nachricht enthält das Element `message` nur das Element `text`. Das Element `text` wiederum enthält dann den Text der SMS Nachricht, der versendet werden soll. Das sieht in XML dann folgendermaßen aus:

```
<message>
  <text>(Hier steht der Text der SMS Nachricht)</text>
</message>
```

Umlaute im Nachrichtentext können auf verschiedenen Wegen codiert werden. Normalerweise können die Umlaute einfach im Zeichensatz ISO-8859-1 codieren, mit dem dann auch die Übertragung durchgeführt wird. Soll der Zeichensatz UTF-8 verwendet werden, müssen alle Zeichen außerhalb des US-ASCII Zeichensatzes mithilfe numerischer Entities umschrieben werden. Es ist allerdings zu beachten, dass es nicht immer ausreicht in dem XML-Header einfach den Zeichensatz ISO-8859-1 anzugeben. Auch die verwendete Programmiersprache muss die Zeichen in der richtigen Codierung ausgeben. Hier ist teilweise eine explizite Angabe erforderlich. Die Umschreibung der Umlaute mit numerischen Entities kann folgender Tabelle entnommen werden:

Zeichen	numerische Entity
ä	&#228;
ö	&#246;

Zeichen	numerische Entity
ü	&#252;
Ä	&#196;
Ö	&#214;
Ü	&#220;
ß	&#223;

Auf das `message` Element muss mindestens ein `destination` Element folgen. In einem `destination` Element wird festgelegt an welchen Empfänger eine SMS Nachricht gesendet werden soll. Dabei können theoretisch beliebig viele `destination` Elemente angegeben werden. Die tatsächliche Anzahl ist nur durch einen eventuellen Timeout der Verbindung und der Auslastung des Servers begrenzt. Bei bis zu 5000 Empfängern pro SMS Nachricht in einer XML-Datei treten im Normalfall keine Probleme auf. Das `destination` Element muss die Mobilfunkrufnummer des Empfängers im Internationalen Format enthalten. Das Internationale Format hat folgenden Aufbau:

```
+<Länderkennung><Vorwahl ohne 0><Teilnehmernummer>
```

Ein `destination` Element mit der Mobilfunknummer, von einem deutschen Vodafone Mobilfunkteilnehmer, mit der Handy-Nummer 0172-1234567, würde dann folgendermaßen aussehen:

```
<destination>+491721234567</destination>
```

Damit ist die XML-Datei zum Versand einer einfachen SMS Nachricht vollständig. Es folgt nun die komplette XML-Datei, wie sie an den Gateway gesendet werden kann:

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE btn-sms-send SYSTEM "http://www.btn.de/dtd/btn-
sms-send.dtd">

<btn-sms-send>
  <sender userid="XXX00000" password="xyz0123"/>
  <message>
    <text>TEXT</text>
  </message>
  <destination>+491721234567</destination>
</btn-sms-send>
```

Diese Datei kann dann mit Hilfe eines HTTP Post Uploads an die folgende URL gesendet werden: <http://xml2sms1.btn.de:8080/sendSMS/sendSMS.do>

Bei diesem Upload ist es sehr wichtig darauf zu achten, dass die HTTP-Header richtig gesetzt sind. Insbesondere muss der `Content-Type-Header` auf `text/xml` gesetzt werden (also: „`Content-Type: text/xml`“). In dem Body des Posts befindet sich dann nur die reine XML-Datei. Wird kein `Content-Type-Header` angegeben, benutzen viele Client Bibliotheken bei einem HTTP Post Upload automatisch `application/x-www-form-urlencoded`. Dies ist **nicht korrekt** und darf **nicht** verwendet werden. Auch wenn diese Angabe im Moment noch funktioniert, wird die Unterstützung dafür in absehbarer Zeit eingestellt.

## Versand einer SMS Nachricht mit variablem Absender

Die Beschreibung, wie eine SMS Nachricht mit variablem Absender verschickt werden kann, baut auf dem Versand einer einfachen SMS Nachricht auf. Alle dort behandelten Dinge werden als bekannt vorausgesetzt.

Um der SMS Nachricht einen individuellen Absender zuzuweisen, muss in das `message` Element das Element `originator` eingefügt werden. Bei diesem Element muss das Attribut `type` angegeben werden. Es kann die Werte `text` oder `number` enthalten. Im Element selbst ist dann der gewünschte Absender anzugeben.

Falls Sie einen Absender verwenden möchten der Alphanumerische Zeichen (also Buchstaben) enthält geben sie unter dem Attribut `type` bitte `text` an. Sie müssen sich dann auf bis zu 11 Zeichen beschränken. Das Element `originator` sieht dann z.B. so aus:

```
<originator type="text">www.btn.de</originator>
```

Möchten Sie als Absender eine Telefonnummer angeben geben sie unter dem Attribut `type` bitte `number` an. Dann haben sie bis zu 16 Zeichen zur Verfügung. Beachten sie bitte, dass neben den Ziffern von 0 bis 9 auch das Plus-Zeichen „+“ in einem Absender vom Typ `number` erlaubt ist, damit Internationale Telefonnummern angegeben werden können. Das Element `originator` sieht dann z.B. so aus:

```
<originator type="number">+491779876543</originator>
```

Eine gültige XML-Datei, die eine SMS Nachricht mit einer Telefonnummer als Absender versendet, sieht wie folgt aus:

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE btn-sms-send SYSTEM "http://www.btn.de/dtd/btn-
sms-send.dtd">

<btn-sms-send>
  <sender userid="XXX00000" password="xyz0123"/>
  <message>
    <text>TEXT</text>
    <originator
type="number">+491779876543</originator>
  </message>
  <destination>+491721234567</destination>
</btn-sms-send>
```

## Versand einer SMS Nachricht mit Termin

Die Beschreibung, wie eine SMS Nachricht mit Termin versendet werden kann, baut auf dem Versand einer einfachen SMS Nachricht auf. Alle dort behandelten Dinge werden als bekannt vorausgesetzt.

Um eine SMS Nachricht zu einem bestimmten Zeitpunkt zu senden, muss in das `message` Element das Element `delivery` eingefügt werden. Wichtig ist dabei auch die Reihenfolge. Das Element `delivery` darf erst nach einem `text` Element eingefügt werden. Ist ein `originator` Element vorhanden, darf er erst nach diesem eingefügt werden.

Das Element `delivery` benötigt zwingend zwei Attribute. Mit dem Attribut `date` kann man festlegen an welchem Tag genau die SMS Nachricht gesendet werden soll. Dabei kann sowohl ein deutsches (TT.MM.JJJ), als auch ein amerikanisches (MM-TT-JJJJ) Datumsformat verwendet werden. Das Attribut `time` muss die Uhrzeit, zu der die SMS Nachricht versendet werden soll, enthalten. Hierbei muss das 24 Stunden Format (SS:MM) benutzt werden. Ein gültiges `delivery` Element sieht folgendermaßen aus:

```
<delivery date="21.09.2002" time="11:50"/>
```

Eine SMS Nachricht die dieses Element innerhalb des `message` Elementes beinhaltet wird versendet wenn der Zeitpunkt 21. September 2002, 11:50 Uhr erreicht ist, oder bereits in der Vergangenheit liegt. Das `delivery` Element kann in jede Art von SMS Nachricht eingefügt werden. Es können somit auch Logos, Klingeltöne und Flash SMS zu einem bestimmten Termin versendet werden.

Eine komplette XML-Datei einer einfachen SMS Nachricht mit Terminversand sieht folgendermaßen aus:

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE btn-sms-send SYSTEM "http://www.btn.de/dtd/btn-
sms-send.dtd">

<btn-sms-send>
  <sender userid="XXX00000" password="xyz0123"/>
  <message>
    <text>Dies ist ein XML Test mit einem anderen Text!
  </text>
    <delivery date="21.09.2002" time="11:50"/>
  </message>
  <destination>+491712345678</destination>
</btn-sms-send>
```

## Versand von WAP-Push-/Browser-Nachrichten

Oft kommt es vor, dass man dem Benutzer eines Handys möglichst einfach und komfortabel einen Link zum Download eines Programms, einer Musikdatei oder eines Videos anbieten möchte. Für diesen Zweck gibt es die WAP-Push oder auch Browser-Nachrichten genannten SMS basierten Informationsdienste. Dem Benutzer wird beim Lesen der Nachricht sofort angeboten die angegebene URL zu besuchen und die entsprechende Datei herunterzuladen.

Der Versand funktioniert ähnlich dem einer normalen Textnachricht, allerdings muss dem Element `text` das Element `WapPushMessage` vorausgehen. Dabei bekommt das Element `WapPushMessage` im Attribut `url` den Download-Link übergeben. Hierbei ist zu beachten, dass kein `http`-Protokoll-Präfix „`http://`“ angegeben werden darf, da dieses automatisch hinzugefügt wird.

Das Element `text`, welches auf das Element `WapPushMessage` folgen muss, enthält dann die Beschreibung des Download-Links.

Eine komplette XML-Datei die eine WAP-Push-Nachricht versendet sieht dann folgendermaßen aus:

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE btn-sms-send SYSTEM "http://www.btn.de/dtd/btn-
sms-send.dtd">

<btn-sms-send>
  <sender userid="XXX00000" password="xyz0123"/>
  <message>
    <WapPushMessage url="www.btn.de/download.jar"/>
    <text>Bitte das Programm runterladen.</text>
  </message>
  <destination>+491712345678</destination>
</btn-sms-send>
```

## Versand eines Nokia Operator Logos

Die Beschreibung, wie ein Nokia Operator Logo versendet werden kann, baut auf dem Versand einer einfachen SMS Nachricht auf. Alle dort behandelten Dinge werden als bekannt vorausgesetzt.

Um Logos, Klingeltöne und evtl. andere Binärdaten zu versenden sind zwei grundsätzliche verschiedene Möglichkeiten gegeben. Zum einen können die Binärdaten direkt in die XML-Datei integriert werden. Für die zweite benötigen sie zunächst ein FTP Konto auf unserem Server. Über einen FTP Upload müssen sie dann die Dateien, die sie versenden möchten hochladen. Diese Dateien können dann hier über ihre Dateinamen benutzt und versendet werden.

Nokia Logos müssen im sogenannten „OTA-Bitmap“ Format vorliegen. Die Dateien haben dann typischerweise die Endung „.otb“. Eine solche Logo Datei kann dann als Operator Logo an ein Mobiltelefon gesendet werden.

Die XML-Datei einer einfachen SMS Nachricht muss dazu verändert werden. Innerhalb des `message` Elementes gibt es nun kein `text` Element mehr. Stattdessen wird das Element `NokiaOperatorLogo` hinzugefügt.

Um die Binärdaten direkt mit der XML-Datei zu versenden, müssen diese zwischen das Anfang- und Ende-Tag des `NokiaOperatorLogo` Elementes geschrieben werden. Dabei müssen die Binärdaten in einen hexadezimalen String umgewandelt werden. Ein solcher String sollte ungefähr so aussehen:

```
09F5E965C7FC54... (usw.)
```

Mithilfe dieser Methode können alle Binärzeichen von 00h bis FFh dargestellt werden. Der tatsächliche String sollte keine Umbrüche oder Leerzeichen enthalten. Ein solcher String sollte also so **nicht** aussehen:

```
09 F5 E90D0A
65C7FC 54
```

Ein komplettes `NokiaOperatorLogo` Element sollte ungefähr so aussehen:

```
<NokiaOperatorLogo>09F5E965C7 ... (usw.)... 0A0DFF0005FE</NokiaOperatorLogo>
```

Bei der Umwandlung ist zu beachten, dass jedes Bytes in der originalen Binärdatei genau zwei Zeichen im hexadezimalen String entsprechen. Deswegen gilt für Binärwerte die kleiner als 15 (0Fh) sind, dass diese mit führenden Nullen zu versehen sind. Aus einem Wert wie 10 (Ah) muss bei der Umwandlung also 0A werden. Sollte die Umwandlung falsch durchgeführt werden, kann das Handy das Bild nicht verstehen. Viele Nokia Handys zeigen dies aber nicht an, sondern werfen die Nachricht(en) stillschweigend fort. **Tipp:** Das Handy kann zu Testzwecken an ein leise eingeschaltetes Radio gehalten werden. Man hört dann durch das bekannte klopfen den Empfang der Nachrichten.

Um eine auf dem FTP-Server hochgeladene Datei zu benutzen, muss das `NokiaOperatorLogo` Element leer bleiben. Stattdessen wird das Attribut `filename` benötigt. Mit dessen Hilfe kann angegeben werden, welche Logo-Datei versendet werden soll. Bitte auf Groß- und Kleinschreibung des Dateinamens achten! Ein solches `NokiaOperatorLogo` Element sieht dann so aus:



```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE btn-sms-send SYSTEM "http://www.btn.de/dtd/btn-
sms-send.dtd">

<btn-sms-send>
  <sender userid="XXX00000" password="xyz0123"/>
  <message>
    <NokiaOperatorLogo filename="Compaq.otb"/>
  </message>
  <destination network="26202">+491721234567</destination>
</btn-sms-send>
```

## Versand eines Nokia Gruppensymbols

Die Beschreibung, wie ein Nokia Gruppensymbol versendet werden kann, baut auf dem Versand eines Nokia Operator Logos auf. Alle dort behandelten Dinge werden als bekannt vorausgesetzt.

Die XML-Datei die zur Versendung eines Nokia Gruppensymbols erstellt werden muss, ist der XML-Datei zum Versand eines Nokia Operator Logos sehr ähnlich. Es sind nur geringfügige Änderungen vorzunehmen.

Das Element `NokiaOperatorLogo` wird durch das Element `NokiaGroupLogo` ersetzt. Dabei ist auch wieder, analog zum `NokiaOperatorLogo`, ein hexadezimaler String oder das `filename` Attribut anzugeben.

Da Gruppensymbole nicht Netzbetreiberspezifisch sind, sondern angezeigt werden, wenn ein Teilnehmer von einem Anrufer aus einer bestimmten Gruppe seines Telefonbuchs angerufen wird, ist eine Angabe des `network` Attributs im `destination` Element **nicht** notwendig.

Eine gültige XML-Datei, die ein Logo als hexadezimaler String enthält und dieses an einen Mobilfunkteilnehmer sendet, könnte so aussehen:





## Versand von Nokia Klingeltönen

Die Beschreibung, wie ein Nokia Klingelton versendet werden kann, baut auf dem Versand eines Nokia Gruppensymbols auf. Alle dort behandelten Dinge werden als bekannt vorausgesetzt.

Ein Nokia Klingelton wird fast genauso versendet wie ein Nokia Gruppensymbol. Dabei ist das Element `NokiaGroupLogo` durch das Element `NokiaRingtone` zu ersetzen. Innerhalb des Elements kann dann direkt eine Angabe im RTTTF Format folgen. Alternativ dazu kann auch hier das Attribut `filename` den Dateinamen der zu versendenden Datei enthalten. Dabei muss es sich auch hierbei um eine Datei handeln, die einen Klingelton im RTTTF Format enthält.

Eine gültige XML-Datei zum Versand eines Nokia Klingeltons mit der direkten Angabe der Tonfolge im RTTTF Format sähe demnach so aus:

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE btn-sms-send SYSTEM "http://www.btn.de/dtd/btn-
sms-send.dtd">

<btn-sms-send>
  <sender userid="XXX00000" password="xyz0123"/>
  <message>

<NokiaRingtone>Underberg:d=4,o=5,b=140:8g,8e,4p.,8e,8f,8g,8e
6,8p,8e6,8p,2c6,16p,8g,8e,4p.,8e,8f,8e,8g,8p,8g,8p,2f,16p,8f
,8d,4p.,8d,8e,8f,8g,8e,4p.,8e,8f#,8e,8d,8g,8p,8e,8f#,8d,8p,8
a,g</NokiaRingtone>
  </message>
  <destination>+491721234567</destination>
</btn-sms-send>
```

Eine gültige XML-Datei zum Versand eines Nokia Klingeltons mit Hilfe einer Datei auf dem FTP Server sähe demnach so aus:

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE btn-sms-send SYSTEM "http://www.btn.de/dtd/btn-
sms-send.dtd">

<btn-sms-send>
  <sender userid="XXX00000" password="xyz0123"/>
  <message>
    <NokiaRingtone filename="song.txt"/>
  </message>
  <destination>+491721234567</destination>
</btn-sms-send>
```

## Versand von Siemens Daten

Die Beschreibung, wie ein Siemens Datenobjekt versendet werden kann, baut auf dem Versand eines Nokia Gruppensymbols auf. Alle dort behandelten Dinge werden als bekannt vorausgesetzt.

Siemens Handys der 40er Serie und aufwärts können Daten in verschiedenen Formaten empfangen und diese im sogenannten „FlexMemory“ speichern. Das können Bilder im BMP-Format oder Audio-Dateien im MIDI-Format sein. Bilder können als Logos oder als Bildschirmschoner benutzt werden, Audio-Dateien können als Klingeltöne benutzt werden. Neuere Siemens Handys können auch andere Formate verarbeiten. Das S55 kann z.B. auch JPEG-Bilder empfangen.

Um solche Daten an ein Siemens Handy zu schicken, muss im `message` Element das Element `SiemensData` angegeben werden. Wenn man von der XML-Datei eines Nokia Gruppensymbols ausgeht, braucht also das Element `NokiaGroupLogo` nur durch `SiemensData` ersetzt zu werden. Die Daten können auch hier als hexadezimaler String zwischen die Tags des Elements geschrieben werden. Allerdings ist dabei die Angabe des Dateityps notwendig, damit das Siemens Handy weiß, was es mit den Daten anfangen kann. Deswegen muss dem `SiemensData` Element das Attribut `type` hinzugefügt werden, in dem einfach die übliche Endung einer Datei angegeben werden muss. Für ein Bild also `bmp` und für einen Klingelton `mid`. Alternativ zum Versand als hexadezimaler String kann auch hier das Element `SiemensData` das Attribut `filename` enthalten, in welchem der Dateiname des zu versendenden Bildes oder des zu versendenden Klingeltons angegeben werden muss. Die Bestimmung des Dateityps wird dabei anhand der Endung der Datei vorgenommen, so dass die Angabe `type` hier nicht mehr notwendig ist. Da die Art Übertragung sowohl für Bilder, als auch für Audio-Dateien dieselbe ist, braucht ansonsten keine Unterscheidung angegeben werden.

Beachten sie bitte, dass BMP-Dateien mit vielen Farben recht groß sein können. Versuchen sie BMP-Dateien die sie an Siemens Handys verschicken möchten mit maximal 2 Farben zu speichern, da das Handy alle Bilder mit mehr Farben sowieso auf 2 Farben reduziert. Durch die Beschränkung auf 2 Farben bereits vor dem Versand wird das Bild wesentlich kleiner. Das spart ihnen Geld, da je nach Größe mehrere SMS Nachrichten verschickt werden müssen, die sie bezahlen.

Eine gültige XML-Datei die eine BMP-Datei, die als hexadezimaler String codiert wurde, an ein Siemens Handy schickt sieht dann folgendermaßen aus:

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE btn-sms-send SYSTEM "http://www.btn.de/dtd/btn-
sms-send.dtd">

<btn-sms-send>
  <sender userid="XXX00000" password="xyz0132"/>
  <message>
    <SiemensData
type="bmp">424D8E0100000000000003E0000000280000000480000001C000
00001000100000000000500100000000000000000000000000002000000000000000
00000000FFFFFF007FFFFFFFFFFFFFAAAAAA0000003FFFFFFFFFFFF5555FF000
000BFFFFFFFFFABEAAFF00000005FFFFFFFFD55F5FFF050000009FFFFFFFFFA
AAFFFF0AB0000004FFFFFFFFD41FFF855F000000AFFFFFFFFB2A7F82BFE000
00057FFFFFFFFED5BE55FC100000027FAAFFFEAABCBF83F00000093F557FFD
5FDDE07FF000000ABEA83FFCBFDE1FFF000000D5F400FFD7FDFFFFFF000
000C9E8787FCFFDFFFFFF000000E5F4CC7FD7FDFFFFFF000000EAE9B63FE
FFBFFFFFF000000F4F9CE3FEFFBFFFFFF000000F279B63FF3E7FFFFFF000
000F478CC3FFC1FFFFFFFF000000FA38783FFFFFFFFFFFF000000FA3C017FF
FFFFFFFF000000FC1C067FFFFFFFFFFFF000000F80E38FFFFFFFFFFFF000
000000383FFFFFFFFFFFF0000004000FFFFFFFFFFFF000000A00000000
0000000000000054003FFFFFFFFFFFF000000AAAABFFFFFFFFFFFF000
000D5553FFFFFFFFFFFF000000</SiemensData>
    </message>
    <destination>+491721234567</destination>
</btn-sms-send>
```

Eine gültige XML-Datei die eine MIDI-Datei, die auf dem FTP Server gespeichert ist, an ein Siemens Handy schickt sieht dann folgendermaßen aus:

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE btn-sms-send SYSTEM "http://www.btn.de/dtd/btn-
sms-send.dtd">

<btn-sms-send>
  <sender userid="XXX00000" password="xyz0123"/>
  <message>
    <SiemensData filename="zauberfloete.mid"/>
  </message>
  <destination>+491721234567</destination>
</btn-sms-send>
```

## Versand von reinen Binärdaten

Die Beschreibung, wie reine Binärdaten versendet werden können, baut auf dem Versand einfacher Nachrichten auf. Alle dort behandelten Dinge werden als bekannt vorausgesetzt.

Mit Hilfe des `RawBinaryData` Elements kann der binäre Inhalt einer SMS Nachricht direkt festgelegt werden. Dazu muss das `message` Element das `RawBinaryData` Element beinhalten. Genauso wie bei den anderen Binärdaten kann der Inhalt in Form einer Datei auf dem FTP-Server angegeben werden oder als Hexadezimaler String direkt in die XML-Datei geschrieben werden. Die Größe der Daten darf dabei 140 Bytes (280 Zeichen als Hexadazimaler String) nicht überschreiten, da dies die Größe einer SMS Nachricht ist.

Zusätzlich ist es möglich über das Attribut `udh` anzugeben, ob die übergebenen Daten einen User Data Header enthalten. Dieser muss im Inhalt der Nachricht vorne anstehen und dem allgemein üblichen Format nach GSM-Standard entsprechen. Der Inhalt des `udh` Attributs sollte mit dem String „true“ angegeben werden, wenn ein User Data Header vorhanden ist. Wird das Attribut `udh` nicht angegeben oder enthält es den String „false“ wird die Nachricht ohne User Data Header versendet.

Es ist zu beachten, dass der User Data Header in die Größe der SMS Nachricht von 140 Bytes einzuberechnen ist. Ist der User Data Header z.B. 11 Bytes groß, bleiben für die Daten noch 129 Bytes übrig.

Eine XML-Datei die reine Binärdaten, die in der XML-Datei mitgesendet werden, mit einem User Data Header versendet sieht folgendermaßen aus:

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE btn-sms-send SYSTEM "http://www.btn.de/dtd/btn-
sms-send.dtd">

<btn-sms-send>
  <sender userid="XXX00000" password="xyz0132"/>
  <message>
    <RawBinaryData
udh="true">0605041582158200424D8E0100000000000003E00000028000
0004800000001C00000000100010000000000050010000000000000000000
20000000000000000000000000000000000000000000000000000000000000
FFFFFFFF007FFFFFFFFFFFFFFFAAAAAA00000003FFFF
FFFFFFFF5555FF000000BFFFFFFFFFABEAAFFF00000005FFFFFFFFD55F5FFF0
50000009FFFFFFFFFAAFFF0AB0000004FFFFFFFFD41FFF855F000000AFFFF
FFFB2A7F82BFE00000057FFFFFFFFED5BE55FC100000027FAAFFFEAABCBF83
F00000093F557FFD5FDDE07FF000000ABEA83FFCBFDE1FFF000000D5F40
0FFD7FDFFFFFFFF000000C9E8787FCFFDFFFFFFFF000000E5F4CC7FD7FDFFFFF
F000000EAE9B63FEFFBFFFFFFFF000000F4F9CE3FEFFBFFFFFFFF000000F279B
63FF3E7FFFFFFFF000000F478CC3FFC1FFFFFFFF000000FA38783FFFFFFFFFFF
F000000FA3C017FFFFFFFFFFFFFF000000FC1C067FFFFFFFFFFFFFF000000F80E3
8FFFFFFFFFFFFFF000000000383FFFFFFFFFFFFFF0000004000FFFFFFFFFFFFFF
F000000A000000000000000000000000054003FFFFFFFFFFFFFF000000AAAAB
FFFFFFFFFFFFFF000000D5553FFFFFFFFFFFFFF000000</RawBinaryData>
    </message>
  <destination>+491721234567</destination>
</btn-sms-send>
```

Eine gültige XML-Datei die reine Binärdaten, die auf dem FTP Server gespeichert sind, ohne User Data Header an ein Handy schickt sieht dann folgendermaßen aus:

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE btn-sms-send SYSTEM "http://www.btn.de/dtd/btn-
sms-send.dtd">

<btn-sms-send>
  <sender userid="XXX00000" password="xyz0123"/>
  <message>
    <RawBinaryData filename="message.bin"/>
  </message>
  <destination>+491721234567</destination>
</btn-sms-send>
```

## Zusätzliche Optionen

Neben den bereits beschriebenen Typen von SMS Nachrichten gibt es noch zusätzliche Optionen, die angegeben werden können.

### Tarif festlegen

Jede SMS Nachricht wird mit einem bestimmten Tarif verschickt. Wird kein Tarif angegeben, wird die Nachricht automatisch im Standard Tarif versendet. Folgende Angaben bei dem Tarif sind analog zu unserer Preisliste möglich:

Tarif	Dezimalwert
Premium	30
Standard	20
Eco	15

Um den Tarif einer SMS Nachricht festzulegen, müssen sie dem Element `message` ein Attribut mit dem Namen `tarif` hinzufügen. Dabei kann als Wert entweder der Alphanumerische Code des Tarifs (z.B. „Premium“) oder der Dezimalwert (z.B. „30“) angegeben werden. Ein solches verändertes `message` Element einer einfachen Text SMS Nachricht, die im Premium Tarif gesendet werden soll, würde dann so aussehen:

```
<message tarif="30">  
  <text>(Hier steht der Text der SMS Nachricht)</text>  
</message>
```

Es ist dringend zu beachten, dass Bestimmte Nachrichtenarten automatisch in einem anderen Tarif versendet werden, auch wenn ein Tarif angegeben wurde. So werden Flash SMS, SMS mit mehr als 160 Zeichen und alle binären SMS Nachrichten automatisch im Tarif Premium gesendet.

### Priorität festlegen

Um auch weiterhin mit bereits existierenden Programmen kompatibel zu sein, wird die bisherige Angabe der Priorität in die neuen Tarife umgewandelt. Dabei gilt folgende Tabelle:

Tarif	Priorität
Premium	1 und 2

Standard	5, 6, 7 und 8
Eco	9

Die Preise der einzelnen Tarife entnehmen sie bitte unser jeweils aktuellen Preisliste.

Um die Priorität einer SMS Nachricht festzulegen, müssen sie dem Element `message` ein Attribut mit dem Namen `priority` hinzufügen. Ein solches verändertes `message` Element einer einfachen Text SMS Nachricht, die im Premium Tarif gesendet werden soll, würde dann so aussehen:

```
<message priority="1">
  <text>(Hier steht der Text der SMS Nachricht)</text>
</message>
```

Sollte sowohl ein Tarif, als auch eine Priorität angegeben worden sein, so hat die Angabe des Tarifes in jedem Fall Vorrang vor der Angabe der Priorität.

## Personalisierte Nachrichten versenden

Es ist möglich die zu versendenden SMS Nachrichten mit einer persönlichen Anrede zu versehen. Hierbei wird ein Wort oder ein Teil des Textes in der Nachricht mit einem Namen oder eine Anrede ersetzt, die für jede einzelne Empfängernummer angegeben werden kann. Angenommen der Text einer Nachricht lautet „Hallo #ANREDE#, wir möchten Dir gerne Produkt XYZ verkaufen.“. Diese Nachricht wird an zwei oder mehrere verschiedene Empfänger gesendet. Für jeden Empfänger kann nun das Wort „#ANREDE#“ durch den Namen des jeweiligen Empfängers ersetzt werden. Angenommen die Empfängerin „Ute“ hätte die Telefonnummer +491709999999 und „Willi“ hätte die Telefonnummer +491711111111. So würde an die Empfängernummer +491709999999 die Nachricht „Hallo Ute, wir möchten Dir gerne Produkt XYZ verkaufen.“ gesendet und an die Empfängernummer +491711111111 die Nachricht „Hallo Willi, wir möchten Dir gerne Produkt XYZ verkaufen.“

In der XML Datei muss dazu zunächst das zusätzliche Attribut `replacetext="..."` im `<text>` Element hinzugefügt werden. Das Textelement für die Beispielnachricht von oben würde dann folgendermaßen aussehen.

```
<text replacetext="#ANREDE#">Hallo #ANREDE#, wir möchten Dir gerne Produkt
XYZ verkaufen.</text>
```

In den einzelnen `<destination>` Elementen müssen dann die Ersetzungen im Attribut `replace="..."` angegeben werden. Diese sähen so aus:

```
<destination replace="Ute">+491709999999</destination>
<destination replace="Willi">+491711111111</destination>
```

## Statusreport anfordern

Mithilfe des Statusreports besteht die Möglichkeit zu überprüfen welche Nachrichten letztlich wirklich auf dem Handy des Empfängers angekommen sind. Eine Nachricht kann aus verschiedenen Gründen nicht den Empfänger erreichen. Zum einen kann es sein, dass es den

Mobilfunkanschluss nicht mehr gibt. Ein anderer typischer Fall ist, dass der Anschluss zwar noch besteht, das Handy aber fortwährend ausgeschaltet ist. Dann werden aber dennoch die SMS Nachrichten bei dem Provider zwischengespeichert.

Um einen Statusreport für die versendeten Nachrichten anzufordern muss im `<message>` Element zusätzliches das Element `<status-report>` hinzugefügt werden. Dabei ist zu beachten, dass das `<status-report>` Element immer als letztes Element im `<message>` Element auftaucht. Es können nur Statusreports für Textnachrichten angefordert werden, für Binärdaten (Klingeltöne, Logos, etc.) ist kein Statusreport möglich.

Das `<status-report>` Element hat keinen Inhalt, aber kann ein oder zwei Attribute aufweisen. Wird kein weiteres Attribut angegeben, wird kein automatischer Statusreport zugeschickt. Stattdessen kann über eine andere URL jederzeit der Status der versendeten Nachrichten selbst abgefragt werden (siehe dazu die gesonderte Dokumentation).

Wenn der Statusreport per Email zugestellt werden soll, muss mindestens das Attribut `email` angegeben werden. Mit diesem Attribut wird festgelegt an welche Email Adresse der Statusreport geschickt wird. Optional kann das Attribut `delay` angegeben werden. Dieses Attribut bestimmt die Zeit in Stunden die nach dem Versand der Nachrichten gewartet wird, bis eine Statusabfrage stattfindet und der Statusreport erstellt wird. Wird das Attribut `delay` nicht angegeben, wird der Statusreport nach 12 Stunden erstellt.

Der Ausschnitt einer entsprechenden XML-Datei, die einen Statusreport anfordert, der per Email 24 Stunden nach Versand zugestellt wird, sieht folgendermaßen aus:

```
<message>
  (Hier stehen die anderen Elemente)
  <status-report email="none@email.com" delay="24"/>
</message>
```

Der Ausschnitt einer entsprechenden XML-Datei, die einen Statusreport anfordert, der später selbst per HTTP/XML-Request abgefragt wird, sieht folgendermaßen aus:

```
<message>
  (Hier stehen die anderen Elemente)
  <status-report/>
</message>
```

## Auswertung der Rückantwort

Wie die XML-Dateien, die sie erstellen müssen um SMS Nachrichten zu versenden, aussehen müssen, ist ihnen jetzt bekannt. In diesem Abschnitt soll es nun darum gehen, wie die XML-Datei, die ihnen als Antwort gesendet wird aussieht, und was sie daraus erkennen können.

Auch die Antwort-XML-Datei hat eine Schablone, also eine DTD. Diese finden sie ebenfalls im Anhang.

Die XML-Datei, die sie als Antwort erhalten beginnt selbstverständlich mit einem gültigen XML-Header:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Darauf folgt auch hier eine gültige `!DOCTYPE` Deklaration. Diese beinhaltet natürlich eine andere DTD, nämlich diejenige die ein Antwort-Dokument definiert.

```
<!DOCTYPE btn-sms-response SYSTEM "http://www.btn.de/dtd/btn-sms-response.dtd">
```

Darauf folgt auch hier wieder das Root Element, welches in einer Antwort Datei `btn-sms-response` heißt. Dieses Element enthält alle weiteren Elemente:

```
<btn-sms-response>
```

(Hier folgen die restlichen Elemente der XML-Datei)

```
</btn-sms-response>
```

Welche Elemente im Root-Element enthalten sind, richtet sich danach um welche Art von Fehler es sich handelt. Wir unterscheiden grundsätzlich zwischen zwei Arten von Fehlern. Zum einen sind da die Bestätigungen, die ihnen den Eingang eines Auftrags zum Versand einer SMS Nachricht an einen einzelnen Empfänger bestätigen oder nicht. Zum anderen gibt es die Fatalen Fehler, die eine Bearbeitung der von ihnen übermittelten XML-Datei unmöglich machen.

### Bestätigungen

Bestätigungen enthalten im Root-Element eins oder mehrere `destination` Elemente. Die Anzahl der `destination` Elemente in der Antwort-XML-Datei stimmt immer mit der Anzahl der `destination` Elemente in der von ihnen übermittelten XML-Datei überein.

Der Aufbau des `destination` Elementes in der Antwort-XML-Datei ist jedoch ein anderer als in der von ihnen erstellten XML-Datei. Typischerweise sehen ein solche `destination` Elemente folgendermaßen aus:

```
<destination result="success" errorcode="0">+491721234567</destination>
```

```
<destination result="error" errorcode="1" message="Wrong Phone Number Format">01779876543</destination>
```

Das erste Element meldet einen Erfolgreichen Eingang einer SMS Nachricht an die Telefonnummer +491721234567. Das zweite Element meldet eine falsch formatierte

Telefonnummer 01779876543. Aus dem Attribut `result` können sie ablesen, ob die Übermittlung der Nachricht an uns für den im Element angegebenen Empfänger erfolgreich war oder nicht. War sie erfolgreich, enthält das Attribut `result` den Wert `success`. Ist speziell für den im Element angegebenen Empfänger ein Fehler aufgetreten finden sie im Attribut `result` den Wert `error`. Aus den Werten in den Attributen `errorcode` und `message` können sie dann den Grund für den Fehler ermitteln.

Eine Beispielantwort könnte so aussehen:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE btn-sms-response SYSTEM
"http://www.btn.de/dtd/btn-sms-response.dtd">
<btn-sms-response>
    <destination result="success" errorcode="0">
        +491729419388
    </destination>
    <destination result="success" errorcode="0">
        +491729419388
    </destination>
</btn-sms-response>
```

## Angabe des Kontostandes

Bezahlt der Kunde die Nachrichten im Voraus, wird am Ende der Liste noch ein `<account-balance>` Element angegeben. Dieses Element enthält den Kontostand nach der aktuellen Versendung. Dadurch ist es möglich frühzeitig ein zur Neige gehendes Konto zu erkennen. Diese Element sieht typischerweise so aus:

```
<account-balance>12.3400</account-balance>
```

Es werden immer vier Stellen hinter dem Dezimalpunkt angegeben, da Preise auch teilweise auch mit 1/10 oder 1/100 Cent berechnet werden können.

Eine komplette Antwort würde dann so aussehen:

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE btn-sms-response SYSTEM
"http://www.btn.de/dtd/btn-sms-response.dtd">

<btn-sms-response>

    <destination result="success" errorcode="0">
        +491729419388
    </destination>

    <destination result="success" errorcode="0">
        +491729419388
    </destination>
    <account-balance>12.3400</account-balance>

</btn-sms-response>
```

## Fatale Fehler

Fatale Fehler treten auf, wenn es ein Grundsätzliches Problem gibt, welches die Verarbeitung der von Ihnen übermittelten XML-Datei nicht erlaubt. Der simpelste Fall ist sicherlich der, dass ihre XML-Datei nicht wohlgeformt ist. Das kann an nicht geschlossenen Anführungszeichen, nicht geschlossenen Elementen oder sonstigen Schreibfehler liegen. Zum anderen kann es sein, dass ihre XML-Datei nicht gültig im Sinne der DTD war. Ihre XML-Datei kann ungültig werden, wenn zwingend erforderliche Elemente und/oder Attribute fehlen, an falscher Stelle stehen, oder zu oft vorkommen. Ein fataler Fehler gibt ihnen immer genau darüber Auskunft was nicht stimmte und hilft ihnen somit bei der Fehlerbeseitigung.

Im Root-Element der XML-Datei steht bei einem fatalen nur ein Element mit dem Namen fatal. Diese Element enthält die Attribute errorcode und message, die man auslesen kann um den Grund des Fehler herauszufinden.

Ein typisches solches Element sieht folgendermaßen aus:

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE btn-sms-response SYSTEM
"http://www.btn.de/dtd/btn-sms-response.dtd">

<btn-sms-response>

    <fatal errorcode="9" message="Error on line 9 of
document : The element type &quot;text&quot; must be
terminated by the matching end-tag
&quot;&lt;/text&gt;&quot;." />

</btn-sms-response>
```

## Fehlercodes

Welcher Fehler aufgetreten ist kann am besten anhand der Fehlercodes ausgewertet werden, die bei jedem Fehler in dem `errorcode` Attribut übergeben werden. Folgende Fehlercodes gibt es:

<b>Fehlercode</b>	<b>Bedeutung</b>
1	kein Prepaid Guthaben mehr vorhanden
2	BenutzerID und/oder Passwort falsch
3	interner Fehler
4	keine Berechtigung
5	Account gesperrt
6	IP-Check fehlgeschlagen
7	Ungültige Kombination von Versandoptionen
9	Fehler im Aufbau der XML Datei

## Nachrichten mit Termin stornieren

Nachrichten die mit einem Termin übermittelt wurden können vor dem tatsächlichen Versand storniert werden. Dabei ist zu beachten, dass diese Nachrichten trotzdem noch abgerechnet, aber nicht versendet werden. Diese Funktion macht daher nur Sinn, wenn bereits SMS an uns übermittelt wurden, der Empfänger diese aber nicht mehr wünscht.

Falls diese Funktion dazu benutzt wird, ein neu entwickeltes Programm zu testen, können Sie uns gerne nach dem Ende der Testphase kontaktieren. Für diesen Fall werden diese Nachrichten dann nicht berechnet.

Zum Versand von SMS existieren hier folgende Unterschiede:

- andere URL
- anderer Aufbau der zu sendenden XML-Datei
- anderer Aufbau der zu empfangenden XML-Datei

Die Unterschiede werden im folgenden beschrieben.

### URL

Die URL um bereits übermittelte Nachrichten mit Termin zu stornieren muss eine XML-Datei per http-Request an folgende URL gesendet werden:

```
http://xml2sms1.btn.de:8080/sendSMS/cancelSMS.do
```

### Aufbau der zu sendenden XML-Datei

Die XML-Datei zum stornieren von Nachrichten ist grundsätzlich anders, als der zum Versand von Nachrichten. Aus diesem Grund ist auch ein anderer DOCTYPE notwendig. Dieser lautet:

```
<!DOCTYPE btn-cancel-request SYSTEM "http://www.btn.de/dtd/btn-cancel-request.dtd">
```

Das Root-Element lautet: `<btn-cancel-request>`. Diese muss alle anderen Tags umschließen.

Als erstes Tag innerhalb dieses Block muss sich das `<auth>`-Element befinden. Dieses ist mit zwei Parametern `userid="..."` und `password="..."` anzugeben. Hier sind die Zugangsdaten anzugeben.

Auf dieses Element können in beliebiger Reihenfolge und Häufigkeit die Elemente `<destination>`, `<transaction>` und `<message>` folgen. Diese stornieren Nachrichten nach verschiedenen Kriterien.

Das `<destination>`-Tag verhindert den Versand von bereits eingestellten Nachrichten an eine bestimmte Telefonnummer. Im Inhalt des Tags muss die entsprechende Zielrufnummer

im Internationalen Format angegeben werden. Alle noch offenen Nachrichten des Benutzers an diese Telefonnummer werden damit storniert.

Mit dem `<transaction>`-Tag kann ein ganz bestimmter Versandvorgang annulliert werden. Dazu benötigt man die entsprechende Identifikationsnummer dieses Vorgangs, die in den Body des Elementes einzutragen ist. Diese ist aus der Antwort-XML-Datei des Versandvorgangs vorher auszulesen. Alle Nachrichten, die bei diesem Versandvorgang eingestellt wurden, werden dann nicht mehr versendet.

Das `<message>`-Tag löscht eine einzelne Nachricht. Dazu wird auch eine entsprechende Identifikationsnummer benötigt, die in den Body des Elementes einzutragen ist. Diese wird momentan leider noch nicht durch die Antwort-XML-Datei beim Versand mitgeteilt. Wenn dies in Zukunft geschieht, kann auch ganze gezielt eine einzelne Nachricht im Nachhinein verhindert werden.

Eine XML-Datei die alle offenen Nachrichten an einen Bestimmten Empfänger versendet, sieht dann folgendermaßen aus:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE btn-cancel-request SYSTEM
"http://www.btn.de/dtd/btn-cancel-request.dtd">

<btn-cancel-request>
  <auth userid=" XXX00000" password="xyz0123"/>
  <destination>+491712345678</destination>
</btn-cancel-request>
```

## Aufbau der zu empfangenden XML-Datei

Die Antwort XML-Datei besitzt das Root-Element `<btn-cancel-response>`. In diesem sind alle weiteren Elemente enthalten.

Auf dieses folgt immer ein `<result>`-Element. Dieses enthält Informationen darüber, ob die Anfrage grundsätzlich Erfolgreich verlaufen ist oder nicht. Dazu besitzt es immer ein `errorcode="..."` Attribut. Dieses enthält einen Fehlercode, der anhand der Tabelle für fatale Fehler ausgewertet werden kann. Sollte der dieser nicht null (0) sein, kann das Tag noch das zusätzliche Attribut `message="..."` enthalten, welches genauere Auskunft über den Fehler gibt.

War die Anfrage Grundsätzlich erfolgreich, folgen dann `<destination>`, `<transaction>` und `<message>` Elemente und zwar genau so viele, wie in der Anfrage vorhanden waren. Dabei kann sich allerdings die Reihenfolge verändern. Im Normalfall werden sind zunächst alle `<destination>`-Tags, dann alle `<transaction>`-Tags und zuletzt alle `<message>`-Tags zu finden. Im Body der Elemente findet sich immer die übergebene Nummer wieder, damit problemlos die Verbindung zur Anfrage hergestellt werden kann.

Die Ergebnisse finden sich dann in den Attributen dieser Tags. Jedes dieser Tags enthält die Attribute `errorcode="..."` und `count="..."`. Errorcode gibt darüber Auskunft, ob die Stornierung erfolgreich war oder nicht. Count gibt an, wieviele Nachrichten dadurch annulliert

wurden. Dabei kann auch eine Stornierung erfolgreich verlaufen, bei der keine Nachrichten storniert wurden. Es ist also immer darauf zu achten, ob in `count` die gewünschte Anzahl an Nachrichten steht.

Eine erfolgreiche Antwort könnte dann so aussehen:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE btn-cancel-response SYSTEM
"http://www.btn.de/dtd/btn-cancel-response.dtd">

<btn-cancel-response>
  <result errorcode="0" />
  <destination errorcode="0"
count="1">+491712345678</destination>
</btn-cancel-response>
```

# Beispiele

## Java

```
// -----  
// sendsms.java Version 1.0  
// -----  
// BEYOND THE NET - Internet Service GmbH  
// Giradetstr. 2-38  
// 45131 Essen  
// Germany  
// -----  
// This Application is a simple example how to send  
// sms messages via the http post interface at BTN.  
// -----  
// Created: 21.09.2002  
// Last Modified: 21.09.2002  
  
import java.net.*;  
import java.io.*;  
  
public class sendsms {  
  
    // userid used to send the message  
    private static String UserID = "your userid goes here";  
  
    // password matching the userid  
    private static String Password = "your password goes  
here";  
  
    // destination mobile phone number  
    private static String Destination = "your destination  
goes here";  
  
    // text of the message that will be sent  
    private static String Text = "This is a test sms message  
via the BTN sms gateway";  
  
    public static void main(String [] args) {  
        try {  
            // make a new URL for the BTN webservice "send sms"  
            URL myURL = new  
URL("http://xml2sms1.btn.de:8080/sendsMS/sendsMS.do");  
  
            // connect to this URL using a http connection  
            HttpURLConnection myHttpCon =  
(HttpURLConnection)myURL.openConnection();  
  
            // specify to use the POST method and that we want  
to send content  
            myHttpCon.setRequestMethod("POST");  
            myHttpCon.setDoOutput(true);  
  
            // get the output stream of the http connection  
            PrintWriter out = new
```

```
PrintWriter(myHttpCon.getOutputStream());

    // write the xml header and doctype into the stream
    out.println("<?xml version=\"1.0\" encoding=\"UTF-
8\"?>");
    out.println("<!DOCTYPE btn-sms-send
SYSTEM \"http://www.btn.de/dtd/btn-sms-send.dtd\">");

    // starting with the root element
    out.println("<btn-sms-send>");

    // specify the sender information
    out.println("<sender userid=\""+UserID+"\"
password=\""+Password+"\"/>");

    // we try to send a simple text message
out.println("<message><text>"+Text+"</text></message>");

    // specify the destination of the message
out.println("<destination>"+Destination+"</destination>");

    // closing the root element and the output stream
    out.println("</btn-sms-send>");
    out.close();

    // open the input stream to get the response
    InputStream in = myHttpCon.getInputStream();

    // read response from the input stream into a buffer
    byte[] myBuffer = new byte[2048];
    int i = in.read(myBuffer);
    while(i > 0) {
        System.out.print(new String(myBuffer, 0, i));
        i = in.read(myBuffer);
    }

    } catch (Exception e) {
        // catching all exceptions and putting them out
        e.printStackTrace();
    }
}

}
```

## Perl

```
#!/usr/bin/perl

# -----
# sendsms.pl Version 1.0
# -----
# BEYOND THE NET - Internet Service GmbH
# Giradetstr. 2-38
# 45131 Essen
# Germany
# -----
# This script is a simple example how to send sms
# messages via the http post interface at BTN.
# Perl modules HTTP::Request and LWP::UserAgent
# have to be installed to use this script.
# -----
# Created: 21.09.2002
# Last Modified: 06.12.2004

use HTTP::Request;
use LWP::UserAgent;

# Please modify this values to reach functionality

$UserID = 'your userid goes here';
>Password = 'your password goes here';

# You should customize this values to specify
# where to send a sample message.

$Destination = 'your destination mobile phone number goes
here';
$Text = 'This is a test via BTN SMS Gateway';

# creating new user agent
$useragent = LWP::UserAgent->new;

# creating new http post request
$request = HTTP::Request->new( POST =>
'http://xml2sms1.btn.de:8080/sendSMS/sendSMS.do');

# setting the content type of the request
$request->header('Content-Type' => 'text/xml');

# filling the content with standard header, doctype and root
element
$request->content($xmlfile = '<?xml version="1.0"
encoding="ISO-8859-1"?><!DOCTYPE btn-sms-send SYSTEM
"http://www.btn.de/dtd/btn-sms-send.dtd"><btn-sms-send>');

# adding user information
$request->add_content("<sender userid=\"\$UserID\"
password=\"\$Password\"/>");

# adding message text
$request-
```

```
>add_content("<message><text>$Text</text></message>");

# adding destination of the message
$request-
>add_content("<destination>$Destination</destination>");

# closing root element
$request->add_content("</btn-sms-send>");

# sending request
$response = $useragent->request($request);

# display response output
print $response->content();
```

## PHP

```
<html>
<head><title>send sms message via BTN gateway</title></head>
<body>
<h1>BEYOND THE NET - Internet Service GmbH</h1>
<p>This script is a small example demonstrating how to send
sms messages
via the HTTP Post interface of the BTN sms gateway</p>
<p>It shows the fundamentals in using HTTP Post in PHP. It
sends
only simple messages. Adding other send options can be
acomplished by altering the XML file
sent to the Server.</p>

<form action="sendsms.php" method="post" name="daten">
<p>userid:<br><input type="text" name="userid"></p>
<p>password:<br><input type="password" name="password"></p>
<p>destination:<br><input type="text"
name="destination"></p>
<p>text:<br><input type="text" name="text" value="Test via
BTN gateway"></p>
<p><input type="submit"></p>
<input type="hidden" name="action" value="doit">
</form>

<p>
<?php

// function to send simple XML content via http

function postHttp($host, $port, $path, $Content) {
    $fp = fsockopen($host, $port, $errno, $errstr, 30);
    fputs($fp, "POST $path HTTP/1.1\n");
    fputs($fp, "Host: $host\n");
    fputs($fp, "Content-type: text/xml\n");
```

```
fputs($fp, "Content-length: ".strlen($Content)."\n");
fputs($fp, "Connection: close\n\n");
fputs($fp, $Content);

while(!feof($fp)) {
    $RetVal .= fgets($fp,128);
}

return $RetVal;
}

if ($action == "doit") {

    $Content = '<?xml version="1.0" encoding="ISO-8859-1"?
>';
    $Content .= '<!DOCTYPE btn-sms-send SYSTEM
"http://www.btn.de/dtd/btn-sms-send.dtd">';
    $Content .= '<btn-sms-send>';
    $Content .= "<sender userid=\"\$userid\"
password=\"\$password\"/>";
    $Content .= "<message><text>$text</text></message>";
    $Content .= "<destination>$destination</destination>";
    $Content .= '</btn-sms-send>';

    echo
nl2br(htmlspecialchars(postHttp("xml2sms1.btn.de",8080,"/sen
dSMS/sendSMS.do", $Content)));
}

?>
</p>

</body>
</html>
```

# Anhang

## A. Übersicht über die Schlüsselemente

<b>Element</b>	<b>Beschreibung</b>								
<btn-sms-send>	Root-Element. Enthält alle weiteren Elemente der XML-Datei. Dieses Element darf keine Attribute haben.								
<sender>	Darf selbst keinen Inhalt haben, sondern hat nur Attribute. <table border="1"><thead><tr><th><b>Attribut</b></th><th><b>Beschreibung</b></th></tr></thead><tbody><tr><td>userid="..."</td><td>BenutzerID des Versenders der SMS Nachricht</td></tr><tr><td>password="..."</td><td>Passwort des Versenders der SMS Nachricht.</td></tr><tr><td>customnumber="..."</td><td>Optional. Selbst frei definierbare Kundennummer des Versenders der SMS Nachricht.</td></tr></tbody></table>	<b>Attribut</b>	<b>Beschreibung</b>	userid="..."	BenutzerID des Versenders der SMS Nachricht	password="..."	Passwort des Versenders der SMS Nachricht.	customnumber="..."	Optional. Selbst frei definierbare Kundennummer des Versenders der SMS Nachricht.
<b>Attribut</b>	<b>Beschreibung</b>								
userid="..."	BenutzerID des Versenders der SMS Nachricht								
password="..."	Passwort des Versenders der SMS Nachricht.								
customnumber="..."	Optional. Selbst frei definierbare Kundennummer des Versenders der SMS Nachricht.								

Element	Beschreibung	
<message>	<p>Enthält weitere Elemente, die den Inhalt der SMS Nachricht festlegen. Mögliche Kombinationen von Elementen sind:</p> <ul style="list-style-type: none"> <li>• &lt;text&gt;&lt;originator&gt;&lt;delivery&gt;&lt;status-report&gt;</li> <li>• &lt;WapPushMessage&gt;&lt;text&gt;&lt;delivery&gt;</li> <li>• &lt;NokiaOperatorLogo&gt;&lt;delivery&gt;</li> <li>• &lt;NokiaGroupLogo&gt;&lt;delivery&gt;</li> <li>• &lt;NokiaPictureMessage&gt;&lt;text&gt;&lt;originator&gt;&lt;delivery&gt;</li> <li>• &lt;NokiaRingtone&gt;&lt;delivery&gt;</li> <li>• &lt;SiemensData&gt;&lt;delivery&gt;</li> </ul>	
	Attribut	Beschreibung
	priority="..."	<ul style="list-style-type: none"> <li>• <u>1</u> Die Nachricht wird im Quick+ Tarif versendet.</li> <li>• <u>2</u> Die Nachricht wird im Termin+ Tarif versendet.</li> <li>• <u>5</u> Die Nachricht wird im Standard Tarif versendet</li> <li>• <u>7</u> Die Nachricht wird im AbendEco Tarif versendet.</li> <li>• <u>8</u> Die Nachricht wird im Mitternacht+ Tarif versendet.</li> <li>• <u>9</u> Die Nachricht wird im SuperWeekend Tarif versendet.</li> </ul>

Element	Beschreibung	
<text>	Enthält den Text der SMS Nachricht	
	Attribut	Beschreibung
	type="..."	<ul style="list-style-type: none"><li>• <u>normal</u> Der Text der Nachricht ist maximal 160 Zeichen lang. Sollte er länger sein, wird er bei 160 Zeichen abgeschnitten. Muss im Normalfall nicht angegeben werden, wird als Standard angenommen.</li><li>• <u>long</u> Die Nachricht wird bei einem Text mit mehr als 160 Zeichen in entsprechend viele SMS Nachrichten aufgeteilt.</li><li>• <u>flash</u> Der Text der Nachricht erscheint direkt auf dem Display des Empfängers. Maximal 160 Zeichen.</li></ul>

Element	Beschreibung							
<originator>	Enthält den Variablen Absender der SMS Nachricht. Der Typ des Absenders wird durch Attribute bestimmt.							
	<table border="1"> <thead> <tr> <th data-bbox="550 421 962 526">Attribut</th> <th data-bbox="962 421 1388 526">Beschreibung</th> </tr> </thead> <tbody> <tr> <td data-bbox="550 526 962 1064">type="..."</td> <td data-bbox="962 526 1388 1064"> <ul style="list-style-type: none"> <li data-bbox="1018 566 1345 779">• <u>number</u> Wenn der Absender nur aus Ziffern und dem plus-Zeichen (“+”) besteht. Maximal 16 Zeichen.</li> <li data-bbox="1018 824 1345 996">• <u>text</u> Der Absender kann aus beliebigen Zeichen bestehen. Maximal 11 Zeichen.</li> </ul> </td> </tr> </tbody> </table>	Attribut	Beschreibung	type="..."	<ul style="list-style-type: none"> <li data-bbox="1018 566 1345 779">• <u>number</u> Wenn der Absender nur aus Ziffern und dem plus-Zeichen (“+”) besteht. Maximal 16 Zeichen.</li> <li data-bbox="1018 824 1345 996">• <u>text</u> Der Absender kann aus beliebigen Zeichen bestehen. Maximal 11 Zeichen.</li> </ul>			
Attribut	Beschreibung							
type="..."	<ul style="list-style-type: none"> <li data-bbox="1018 566 1345 779">• <u>number</u> Wenn der Absender nur aus Ziffern und dem plus-Zeichen (“+”) besteht. Maximal 16 Zeichen.</li> <li data-bbox="1018 824 1345 996">• <u>text</u> Der Absender kann aus beliebigen Zeichen bestehen. Maximal 11 Zeichen.</li> </ul>							
<delivery>	Darf selbst keinen Inhalt haben, sondern hat nur Attribute.							
	<table border="1"> <thead> <tr> <th data-bbox="550 1160 962 1265">Attribut</th> <th data-bbox="962 1160 1388 1265">Beschreibung</th> </tr> </thead> <tbody> <tr> <td data-bbox="550 1265 962 1444">date="..."</td> <td data-bbox="962 1265 1388 1444">Datum zu dem die SMS Nachricht versendet werden soll.</td> </tr> <tr> <td data-bbox="550 1444 962 1641">time="..."</td> <td data-bbox="962 1444 1388 1641">Uhrzeit zu der die SMS Nachricht versendet werden soll</td> </tr> </tbody> </table>	Attribut	Beschreibung	date="..."	Datum zu dem die SMS Nachricht versendet werden soll.	time="..."	Uhrzeit zu der die SMS Nachricht versendet werden soll	
	Attribut	Beschreibung						
date="..."	Datum zu dem die SMS Nachricht versendet werden soll.							
time="..."	Uhrzeit zu der die SMS Nachricht versendet werden soll							

Element	Beschreibung	
<status-report>	Darf selbst keinen Inhalt haben, sondern hat nur Attribute.	
	Attribut	Beschreibung
	email="..."	Hier muss die Email Adresse angegeben werden, an die der Statusreport geschickt werden soll.
delay="..."	Die Zeit in Stunden, die zwischen Absenden der Nachrichten und Erstellung des Statusreports liegen soll. Dieses Attribut kann auch weggelassen werden, dann wird der Report 12 Stunden nach Versand erstellt.	
<WapPushMessage>	Dieses Element ist selbst immer leer. Als Attribut muss immer <code>url</code> angegeben werden.	
	Attribut	Beschreibung
	url="..."	Ziel des Download-Links ohne das http-Protokoll-Präfix „http://“.
<NokiaOperatorLogo>	Kann selbst als Inhalt das Logo als hexadezimalen String enthalten. Ist das Element leer muss es das Attribut <code>filename</code> besitzen.	
	Attribut	Beschreibung
	filename="..."	Dateiname des zu versendenden Logos.

Element	Beschreibung				
<NokiaGroupLogo>	Kann selbst als Inhalt das Logo als hexadezimalen String enthalten. Ist das Element leer muss es das Attribut <code>filename</code> besitzen.				
	<table border="1"> <thead> <tr> <th data-bbox="549 501 963 533">Attribut</th> <th data-bbox="963 501 1398 533">Beschreibung</th> </tr> </thead> <tbody> <tr> <td data-bbox="549 600 963 636">filename="..."</td> <td data-bbox="963 600 1398 680">Dateiname des zu versendenden Logos.</td> </tr> </tbody> </table>	Attribut	Beschreibung	filename="..."	Dateiname des zu versendenden Logos.
Attribut	Beschreibung				
filename="..."	Dateiname des zu versendenden Logos.				
<NokiaPictureMessage>	Kann selbst als Inhalt das Logo als hexadezimalen String enthalten. Ist das Element leer muss es das Attribut <code>filename</code> besitzen.				
	<table border="1"> <thead> <tr> <th data-bbox="549 949 963 981">Attribut</th> <th data-bbox="963 949 1398 981">Beschreibung</th> </tr> </thead> <tbody> <tr> <td data-bbox="549 1048 963 1084">filename="..."</td> <td data-bbox="963 1048 1398 1128">Dateiname des zu versendenden Logos.</td> </tr> </tbody> </table>	Attribut	Beschreibung	filename="..."	Dateiname des zu versendenden Logos.
Attribut	Beschreibung				
filename="..."	Dateiname des zu versendenden Logos.				
<NokiaRingtone>	Kann selbst als Inhalt den Klingelton als String im RTTTF Format enthalten. Ist das Element leer muss es das Attribut <code>filename</code> besitzen.				
	<table border="1"> <thead> <tr> <th data-bbox="549 1397 963 1429">Attribut</th> <th data-bbox="963 1397 1398 1429">Beschreibung</th> </tr> </thead> <tbody> <tr> <td data-bbox="549 1496 963 1532">filename="..."</td> <td data-bbox="963 1496 1398 1576">Dateiname des zu versendenden Klingeltons.</td> </tr> </tbody> </table>	Attribut	Beschreibung	filename="..."	Dateiname des zu versendenden Klingeltons.
Attribut	Beschreibung				
filename="..."	Dateiname des zu versendenden Klingeltons.				

Element	Beschreibung			
<SiemensData>	<p>Kann selbst als Inhalt das Logo oder den Klingelton als hexadezimalen String enthalten, wobei dann das <code>type</code> Attribut auch angegeben sein muss und die übliche Dateiergung (<code>bmp</code>, <code>mid</code>) für die Daten enthalten muss. Ist das Element leer muss es das Attribut <code>filename</code> besitzen, wobei dann die Dateiergung als Dateityp genommen wird.</p>			
	<table border="1"> <thead> <tr> <th data-bbox="550 577 962 680">Attribut</th> <th data-bbox="962 577 1394 680">Beschreibung</th> </tr> </thead> </table>	Attribut	Beschreibung	
	Attribut	Beschreibung		
<table border="1"> <tbody> <tr> <td data-bbox="550 680 962 819">filename="..."</td> <td data-bbox="962 680 1394 819">Dateiname der zu versendenden Datei.</td> </tr> <tr> <td data-bbox="550 819 962 987">type="..."</td> <td data-bbox="962 819 1394 987">Dateityp in Form einer entsprechenden Dateiergung.</td> </tr> </tbody> </table>	filename="..."	Dateiname der zu versendenden Datei.	type="..."	Dateityp in Form einer entsprechenden Dateiergung.
filename="..."	Dateiname der zu versendenden Datei.			
type="..."	Dateityp in Form einer entsprechenden Dateiergung.			
<RawBinaryData>	<p>Kann selbst als Inhalt die Binärdaten als hexadezimalen String enthalten. Ist das Element leer muss es das Attribut <code>filename</code> besitzen, wobei dann die Dateiergung als Dateityp genommen wird. Soll ein User Data Header versendet werden ist dieser den Daten voranzustellen und das Attribut <code>udh</code> auf „true“ zu setzen.</p>			
	<table border="1"> <thead> <tr> <th data-bbox="550 1272 962 1375">Attribut</th> <th data-bbox="962 1272 1394 1375">Beschreibung</th> </tr> </thead> </table>	Attribut	Beschreibung	
	Attribut	Beschreibung		
<table border="1"> <tbody> <tr> <td data-bbox="550 1375 962 1514">filename="..."</td> <td data-bbox="962 1375 1394 1514">Dateiname der zu versendenden Datei.</td> </tr> <tr> <td data-bbox="550 1514 962 1861">udh="..."</td> <td data-bbox="962 1514 1394 1861">Muss auf „true“ gesetzt werden, wenn die Daten einen User Data Header enthalten. Wird es nicht angegeben oder auf „false“ gesetzt wird kein User Data Header versendet.</td> </tr> </tbody> </table>	filename="..."	Dateiname der zu versendenden Datei.	udh="..."	Muss auf „true“ gesetzt werden, wenn die Daten einen User Data Header enthalten. Wird es nicht angegeben oder auf „false“ gesetzt wird kein User Data Header versendet.
filename="..."	Dateiname der zu versendenden Datei.			
udh="..."	Muss auf „true“ gesetzt werden, wenn die Daten einen User Data Header enthalten. Wird es nicht angegeben oder auf „false“ gesetzt wird kein User Data Header versendet.			

Element	Beschreibung	
<destination>	Enthält die Mobilfunkrufnummer eines Empfängers der SMS Nachricht. Kann außerdem noch ein Attribut enthalten.	
	Attribut	Beschreibung
network="..."	Enthält die Eindeutige Nummer des Netzbetreibers für den jeweiligen Empfänger. Muss beim Versand von Nokia Operator Logos angegeben werden.	

## B. DTD: btn-sms-send

```
<?xml version="1.0" encoding="UTF-8"?>

<!ELEMENT btn-sms-send (sender, message, destination)>
<!ATTLIST btn-sms-send
  test CDATA #IMPLIED
>

<!ELEMENT delivery EMPTY>
<!ATTLIST delivery
  date CDATA #REQUIRED
  time CDATA #REQUIRED
>

<!ELEMENT destination (#PCDATA)>
<!ATTLIST destination
  replace CDATA #IMPLIED
  network CDATA #IMPLIED
>

<!ELEMENT message ((text, originator?, delivery?, status-
report?) | (NokiaOperatorLogo, delivery?, status-report?) |
(NokiaPictureMessage, text?, originator?, delivery?, status-
report?) | (NokiaGroupLogo, delivery?, status-report?) |
(NokiaRingtone, delivery?, status-report?) | (SiemensData,
delivery?, status-report?))>
<!ATTLIST message
  priority (1 | 2 | 3 | 5 | 7 | 8 | 9 | 10) "5"
>

<!ELEMENT originator (#PCDATA)>
<!ATTLIST originator
  type (text | number) #REQUIRED
>

<!ELEMENT sender EMPTY>
<!ATTLIST sender
  userid CDATA #REQUIRED
  password CDATA #REQUIRED
  customnumber CDATA #IMPLIED
>

<!ELEMENT status-report EMPTY>
<!ATTLIST status-report
  delay CDATA "12"
  email CDATA #REQUIRED
>

<!ELEMENT NokiaOperatorLogo EMPTY>
<!ATTLIST NokiaOperatorLogo
  filename CDATA #REQUIRED
>

<!ELEMENT NokiaGroupLogo EMPTY>
<!ATTLIST NokiaGroupLogo
```

```
    filename CDATA #REQUIRED
  >

  <!ELEMENT NokiaPictureMessage EMPTY>
  <!ATTLIST NokiaPictureMessage
    filename CDATA #REQUIRED
  >

  <!ELEMENT NokiaRingtone EMPTY>
  <!ATTLIST NokiaRingtone
    filename CDATA #REQUIRED
  >

  <!ELEMENT SiemensData EMPTY>
  <!ATTLIST SiemensData
    filename CDATA #REQUIRED
  >

  <!ELEMENT text (#PCDATA)>
  <!ATTLIST text
    replacetext CDATA #IMPLIED
    type (normal | long | flash) "normal"
  >
```

## C. DTD: btn-sms-response

```
<?xml version="1.0" encoding="UTF-8"?>

<!ELEMENT btn-sms-response ( ( fatal ) | ( destination+,
account-balance? ) )>
<!ELEMENT fatal (#PCDATA)>
<!ATTLIST fatal
  errorcode (0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9)
#REQUIRED
  message CDATA #REQUIRED
>

<!ELEMENT destination (#PCDATA)>
<!ATTLIST destination
  result (success | error) #REQUIRED
  errorcode (0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9) "0"
  message CDATA "The Message was successfully sent."
>
```

# Index

<b>B</b>		NokiaOperatorLogo.....	8
BenutzerID.....	3	NokiaPictureMessage.....	12
Bildmitteilung.....	12	NokiaRingtone.....	13
BMP-Format.....	14	<b>O</b>	
Browser-Nachrichten.....	7	Operator Logo.....	8
btn-sms-response.....	21	originator.....	6
btn-sms-send.....	4	OTA-Bitmap.....	8
<b>D</b>		<b>P</b>	
date.....	7	password.....	4
delay.....	20	Premium.....	18
delivery.....	7	Priorität.....	18
destination.....	5	priority.....	19
DOCTYPE.....	3	<b>R</b>	
DTD.....	3	RawBinaryData.....	16
<b>E</b>		reine Binärdaten.....	16
Eco.....	18	Root-Element.....	4
email.....	20	RTTTF Format.....	13
Encoding.....	3	Rückantwort.....	21
<b>F</b>		<b>S</b>	
filename.....		sender.....	4
Bildmitteilung.....	12	Siemens Daten.....	14
Nokia Gruppensymbol.....	11	SiemensData.....	14
Nokia Klingelton.....	13	Standard.....	18
Nokia Operator Logo.....	9	Statusreport.....	19
Siemens Daten.....	14	<b>T</b>	
FlexMemory.....	14	Tarif.....	18
<b>G</b>		automatische Änderung.....	18
Gruppensymbol.....	10	Termin.....	7
<b>H</b>		text.....	4
hexadezimalen String.....	9	Bildmitteilung.....	12
HTTP Protokoll.....	3	time.....	7
<b>I</b>		type.....	
internationales Format.....	5	originator.....	6
<b>K</b>		Siemens Daten.....	14
Klingeltöne.....	13	<b>U</b>	
<b>M</b>		Umlaute.....	4
message.....	4	User Data Header.....	16
MIDI-Format.....	14	userid.....	4
<b>N</b>		<b>V</b>	
network.....	9	variabler Absender.....	6
NokiaGroupLogo.....	11	<b>W</b>	
		WAP-Push-Nachrichten.....	7

WapPushMessage.....	8
<b>X</b>	
XML-Header.....	3
<b>&lt;</b>	
<status-report>.....	20