Dokumentation XML/HTTP-Post Interface

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

Inhaltsverzeichnis

Einleitung

Neben dem Email-to-SMS Gateway bietet BEYOND THE NET 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 das 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.

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 bereitsteht.
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 3Umlaute 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-smssend 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“ lauten 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 ein eventuelles 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 das Gateway gesendet werden kann:

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

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

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/btnsms-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/btnsms-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/btnsms-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 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 FTPServer angegeben werden oder als Hexadezimaler String direkt in die XML-Datei geschrieben werden.
Die Größe der Daten darf dabei 140 Bytes (280 Zeichen als Hexadezimaler 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/btnsms-send.dtd">

<btn-sms-send>
    <sender userid="XXX00000" password="xyz0132"/>
    <message>
        <RawBinaryData
            udh="true">060504158215820042 4D8E010000000000003E00000028000
            000480000001C0000000100010000000000500100000000000000000000000
2000000000000000000000000FFFFFF007FFFFFFFFFFFAAAAAA0000003FFFF
FFFFFFF5555FF000000BFFFFFFFFABEAAFFF00000005FFFFFFFD55F5FFF0
50000009FFFFFFFAAAFFFF0AB0000004FFFFFFFD41FFF855F000000AFFFF
FFFB2A7F82BFE00000057FFFFFFED5BE55FC100000027FAAFFFEAABCBF83
F00000093F557FFD5FDDE07FF000000ABEA83FFCBFDE1FFFF000000D5F40
0FFD7FDFFFFFF000000C9E8787FCFFDFFFFFF000000E5F4CC7FD7FDFFFFF
F000000EAE9B63FEFFBFFFFFF000000F4F9CE3FEFFBFFFFFF000000F279B
63FF3E7FFFFFF000000F478CC3FFC1FFFFFFF000000FA38783FFFFFFFFF
F000000FA3C017FFFFFFFFFFF000000FC1C067FFFFFFFFFFF000000F80E3
8FFFFFFFFFFFF000000000383FFFFFFFFFFF0000004000FFFFFFFFFFFF
F000000A000000000000000000000000054003FFFFFFFFFFFF000000AAAAB
FFFFFFFFFFFF000000D5553FFFFFFFFFFFF000000</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/btnsms-send.dtd">

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

Sichere Verbindung SSL

Im Normalfall wird der HTTP-Aufruf im Klartext übermittelt. Alternativ dazu kann der Aufruf auch als HTTPS-Aufruf stattfinden. Hier wird die sogenannte Secure Socket Layer (SSL) verwendet, die eine entsprechende Sicherheit durch Verschlüsselung garantiert. Wenn die verwendete Programmiersprache dies unterstützt und eine entsprechende Programmierung auf Kundenseite erfolgt, reicht es meist aus die Protokollangabe „http://“ durch „https://“ in der URL zu ersetzen und den Port von 8080 auf 443 zu ändern.

Hier ein Beispiel in PHP für den SSL Aufruf:

```
<?php

$request_xml = '<?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="Benutzername*" password="Passwort*"/>
<message tarif="30"><text>Hier steht ihr text</text><originator type="text">Absender*</originator></message>
<destination>Empfänger*</destination></btn-sms-send>';

$ch = curl_init();
curl_setopt($ch, CURLOPT_URL,'https://freiburg.btn.de:443/sendSMS/sendSMS.do');
curl_setopt($ch, CURLOPT_HTTPAUTH, CURLAUTH_ANY);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_TIMEOUT, 4);
curl_setopt($ch, CURLOPT_POSTFIELDS, $request_xml);
curl_setopt($ch, CURLOPT_HTTPHEADER, array('Connection: close'));
$start = array_sum(explode(' ', microtime()));
$result = curl_exec($ch);
$stop = array_sum(explode(' ', microtime()));
$totalTime = $stop - $start;
if ( curl_errno($ch) ) {
   $result = 'ERROR -> ' . curl_errno($ch) . ': ' . curl_error($ch);
   } else {
   $returnCode = (int)curl_getinfo($ch, CURLINFO_HTTP_CODE);
   switch($returnCode){
   case 200:
         break;
         default:
         $result = 'HTTP ERROR -> ' . $returnCode;
         break;
         }
   }
curl_close($ch);
echo 'Total time for request: ' . $totalTime . "\n";
echo $result;
?>
```

Benutzername * = hier tragen Sie den Benutzernamen ein, mit dem Sie sich auch bei unseren SMS-Gateway anmelden (https://www.sms-gateway.net)

Passwort *= hier tragen Sie das Passwort ein, mit dem Sie sich auch bei unseren SMS-Gateway anmelden (https://www.sms-gateway.net)

Absender * = hier können Sie den Absender eintrage.

type= "number"
Wenn der Absender nur aus Ziffern und dem plus-Zeichen (“+”) besteht.

type= "text"
Der Absender kann aus beliebigen Zeichen bestehen.

Empfänger * = hier tragen Sie die Handynummer des Empfängers ein.

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 unserer 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. Wir 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 XMLDatei, 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-smsresponse.dtd">
```

Darauf folgt auch hier wieder das Root Element, welches in einer Antwort Datei btn-smsresponse 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 <accountbalance>

Element angegeben. Dieses Element enthält den Kontostand nach der aktuellen Versendung. Dadurch ist es möglich frühzeitig ein zur Neige gehendes Konto zu erkennen. Dieses 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 XMLDatei 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. Dieses Element enthält die Attribute errorcode und message, die man auslesen kann um den Grund des Fehlers 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 Attribute übergeben werden. Folgende Fehlercodes gibt es:

Fehlercode	Bedeutung
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-cancelrequest.dtd">
```

Das Root-Element lautet: <btn-cancel-request>. Diese muss alle anderen Tags umschließen.
Als erster Tag innerhalb dieses Blocks 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, welche 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, wie viele 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
// Bonnerstr. 31
// 50389 Wesseling
// 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
```

Perl

```
#!/usr/bin/perl

# ----------------------------------------------------
# sendsms.pl Version 1.0
```

```
# ---------------------------------------------------
# BEYOND THE NET - Internet Service GmbH
# Bonnerstr. 31
# 50389 Wesseling
# 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-
```

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 Schlüsselelemente

<table>
<tr><th>Element</th><th>Beschreibung</th></tr>
<tr><td><btn-sms-send></td><td>Root-Element. Enthalten alle weiteren Elemente der XML-Datei.
Dieses Element darf keine Attribute haben.</td></tr>
<tr><td><sender></td><td>Darf selbst keinen Inhalt haben, sondern hat nur Attribute.
<table>
<tr><th>Attribut</th><th>Beschreibung</th></tr>
<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>
</table></td></tr>
<tr><th>Element</th><th>Beschreibung</th></tr>
<tr><td><message></td><td>Enthält weitere Elemente, die den Inhalt der SMS Nachricht festlegen. Mögliche Kombinationen von Elementen sind:
• <text><originator><delivery><status-report></td></tr>
</table>

	Attribut	Beschreibung
	priority="..."	• 1 Die Nachricht wird im Quick+ Tarif versendet. • 2 Die Nachricht wird im Termin+ Tarif versendet. • 5 Die Nachricht wird im Standard Tarif versendet • 7 Die Nachricht wird im AbendEco Tarif versendet. • 8 Die Nachricht wird um Mitternacht+ Tarif versendet. • 9 Die Nachricht wird im SuperWeekend Tarif

<table>
<tr><th>Element</th><th>Beschreibung</th></tr>
<tr><td><text></td><td>Enthält den Text der SMS Nachricht
<table>
<tr><th>Attribut</th><th>Beschreibung</th></tr>
<tr><td>type=”...”</td><td>• normal 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.
• long
Die Nachricht wird bei einem Text mit mehr als 160 Zeichen in entsprechend viele SMS Nachrichten aufgeteilt.
• flash
Der Text der Nachricht erscheint direkt auf dem Display des Empfängers. Maximal 160 Zeichen.</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Element</th><th>Beschreibung</th></tr>
<tr><td><originator></td><td>Enthält den Variablen Absender der SMS Nachricht. Der Typ des Absenders wird durch Attribute bestimmt.
<table>
<tr><th>Attribut</th><th>Beschreibung</th></tr>
<tr><td>type=”...”</td><td>• number
Wenn der Absender nur aus Ziffern und dem plus-Zeichen (“+”) besteht.
Maximal 16 Zeichen.
• text
Der Absender kann aus beliebigen
Zeichen bestehen.
Maximal 11 Zeichen.</td></tr>
</table></td></tr>
<tr><td><delivery></td><td>Darf selbst keinen Inhalt haben, sondern hat nur Attribute.
<table>
<tr><th>Attribut</th><th>Beschreibung</th></tr>
<tr><td>date=“...“</td><td>Datum zu dem die SMS Nachricht versendet werden soll.</td></tr>
<tr><td>time=“...“</td><td>Uhrzeit zu der die SMS Nachricht versendet werden soll</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Element</th><th>Beschreibung</th></tr>
<tr><td><status-report></td><td>Darf selbst keinen Inhalt haben, sondern hat nur Attribute.
<table>
<tr><th>Attribut</th><th>Beschreibung</th></tr>
<tr><td>email="..."</td><td>Hier muss die E-Mail-Adresse angegeben werden, an die der Statusreport geschickt werden soll.</td></tr>
<tr><td>delay="..."</td><td>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.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Element</th><th>Beschreibung</th></tr>
<tr><td><RawBinaryData></td><td>Kann selbst als Inhalt die Binärdaten als hexadezimalen String enthalten. Ist das Element leer muss es das Attribut filename besitzen, wobei dann die Dateiendung als Dateityp genommen wird. Soll ein User Data Header versendet werden ist dieser den Daten voranzustellen und das Attribut udh auf „true“ zu setzen.
<table>
<tr><th>Attribut</th><th>Beschreibung</th></tr>
<tr><td>filename=“...“</td><td>Dateiname der zu versendenden Datei.</td></tr>
<tr><td>udh=“...“</td><td>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>
</table>
</td></tr>
</table>

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? statusreport?) | (NokiaOperatorLogo, delivery? status-report?) |
(NokiaPictureMessage, text? originator?, delivery?, statusreport?) | (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 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