smsBox® v6.4 -- HTTP XML API specifications

Table of Contents

Table of Contents

Table of Contents

smsBox® v6.4 -- HTTP XML API specifications

October 2012 -- applicable to smsBox® v6.4

1. About this document

1.1. Meta information

META START

DOC-TITLE	smsBox® HTTP XML API v6.4
DOC-ID	3DC 21148 1902 PBZZA
AUTHORS	GCC, CEC, PBB
OWNER	GCC
STATUS	Draft
KEYWORDS	SmsBox XML API

META END

1.2. Copyright Notice

MNC Mobile News Channel SA
Avenue de la Gare 10
1003 Lausanne
Switzerland

http://www.mnc.ch
info@mnc.ch
Phone: +41 21 317 50 20

2. Introduction

This document describes the HTTP XML API of the smsBox® Professional Platform. The goal of this API is to provide a straightforward, powerful, and secure interface to the smsBox® functionality for registered partners; it allows to remotely retrieve data and trigger actions, such as listing the subscribers of services and posting new content.

This document is the **complete edition** of the smsBox® API: it is the comprehensive reference to the smsBox® API. Partners needing to integrate SMS traffic in the simple way, e.g. just receive an XML document when a end-user sends an SMS, and be able to dynamically reply, should rather use the **simplified edition**.

2.1. Changes

Noticeable changes with previous versions:

2.1.1. In version 6.4 (October 2012)

- the long ago deprecated `message` element of **FORWARD** notifications was removed
- notifications may now be alternatively sent by HTTP GET
- notifications now contain `receptionDate`, the date/time the SMS was received

2.1.2. In version 6.3 (June 2012)

- API errors are now reported with empty `error` XML elements

2.1.3. In version 6.2 (January 2012)

2.1.3.1. Updated notifications

- it is now possible to specify long messages (concatenated SMS) in notification replies

2.1.3.2. Updated commands

- **SEND** and **WEBSEND**: when using multiple receivers in a single command, there must be maximum 1,000 receivers - applications must be updated
- **SEND**, **WEBSEND** and **POST**: it is now possible to specify the behaviour regarding UCS2 characters
- **WEBSUBSCRIBE**: pending double optin is reported with `<needdoubleoptin/>` instead of `<presubscribe/>`; the latter is kept for compatibility purpose in 6.2 but will be removed in the next version

2.1.3.3. MMS Layouts

It is now possible to specify the layout of MMS.

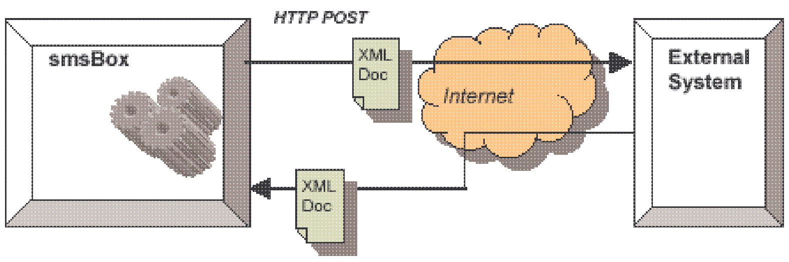
3. Overview

There are two types of interactions: *from* and *to* smsBox®.

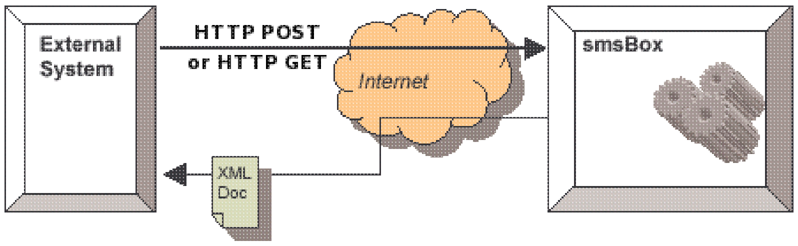
Interactions *from* smsBox® are notifications of events going from smsBox® to external platforms, typically triggered by mobile messages reception and processing, and mobile messages sent back to mobile users.

Interactions *to* smsBox® are ways for external applications to "remote control" functions or query information from smsBox®.

3.1. Interaction from smsBox®



3.2. Interaction to smsBox®



3.3. Technical principles

Interactions *to* smsBox® use the HTTP protocol, typically with the POST *method* to send data. Please refer to the HTTP RFC (latest as of this writing is RFC 2616) for technical documentation about HTTP POST. It is also possible to use the GET *method*, when accessing the main commands.

Interactions *from* smsBox® typically use HTTP POST as well, but can also use Internet mail ("email") with the drawback of unability for the partner to reply to smsBox® with messages to send back to the mobile user.

The data is sent and received as a UTF-8 encoded XML document (except in case of MMS or Mobile Internet page operations, where the data is multipart; however the first part is a UTF-8 encoded XML document as well). Please refer to Unicode documentation to learn more about how to ensure your documents are UTF-8 encoded. Please refer to XML documentation at w3.org to learn more about how to produce XML documents.

Important: failure to send data to smsBox® using properly UTF-8 encoded XML documents will lead to failure of command or incorrect data sent to mobile users! Don't hesitate to validate your applications with the support team.

3.4. Acronyms and terms used

The following "mobile" acronyms and terms are used in this document:

- **EMS**: *Enhanced Message Service* - "standard" binary SMS messages; well supported by Sony Ericsson phones, not supported by Nokia phones, often problematic with other manufacturers phones
- **Forward Lock**: a simple DRM (Digital Rights Management) system supported by reasonably recent phones (2004 onwards) forbidding users of mobile phones to forward multimedia content by MMS or save on unsecure external storage
- **instance**: smsBox® "engine" typically handling one short number

- **MMS**: *Multimedia Message Service* - a message containing a series of multimedia elements of any type (can be text, image, audio, video) a mobile phone can send or receive
- **MO**: *Mobile Originated message* - a message received from a mobile phone
- **MT**: *Mobile Terminated message* - a message sent to a mobile phone
- **MSISDN**: mobile phone number (note: we always express them in international format, that is beginning with + and the country code, for example `+41761234567` or `+33612345678`)
- **Nokia Smart Messaging**: Nokia-specific binary SMS messages; most popular of them are Operator Logo, Caller Line Identification Logo, Picture Message, Ringtone; well supported by Nokia and Sony Ericsson phones, often refused by other manufacturers phones
- **Octet**: ISO/IEC terminology for *byte* - an ordered sequence of 8 bits considered as a unit - often used in technical mobile documentation
- **SMS**: *Short Message Service* - a short text or binary message a mobile phone can send or receive. For text, limited to 160 characters when carrying GSM 7-bit default alphabet (western europe characters, more or less), or 70 characters when carrying UCS2 16-bit international text. For binary, limited to 140 octets.
- **UCS-2**: *Universal Character Set coded in 2 octets* - alternative character set available to transmit text in SMS (is a subset of UTF-16); useful when the characters to transmit aren't member of the GSM 7-bit default alphabet
- **UTF-8**: *UCS Transformation Format 8* - an Unicode encoding form in which characters are represented with a byte sequence of one to four bytes in length
- **Mobile Internet**: mobile phones browsing web pages (similarly to web pages browsed on a computer)

4. Notification XML API (interactions from smsBox®)

4.1. General Principle

An SMS sent to the smsBox® platform invokes a command in the SMS processing engine. This command executes its functionality, then determines if there are any notifications that must be sent to external applications in order to inform partners. If so, an XML notification document containing information about the command executed is sent by HTTP POST or by Internet mail ("email" - in that case, the XML document is simplified in favor of a series of *name: value* lines) to the specified external application. From now on, these notifications will be named "command notifications".

Only successful commands with status `ok` are sent as command notifications. Errors such as subscribing to a service the user is already subscribed to will return him an SMS with an error, and no command notifications will be sent.

Any SMS or MMS messages sent from smsBox® to mobile users, can also trigger notifications. These notifications are first aimed at advanced partners who need to be more fully informed. There can be multiple notifications per message, as the smsBox® platform receives delivery reports from the operators informing about the processing, delivery or failure of messages (availability depending on operators). From now on, these notifications will be named "sent message notifications".

Note that any number of applications can be "subscribed" to notifications of different types. For example, all users of the XML API should receive UNSUBSCRIBEALL notifications, as it indicates a full opt-out by the end-user and it is a contractual requirement to comply with this command.

In the following documentation, examples show a response sent back to smsBox®; this applies only to notifications sent by HTTP POST as there is no such response supported for notifications sent by Internet mail.

4.2. Command Notifications

To illustrate the way command notifications work, a couple of examples can be found in the following table:

Event	Command Notification	Service	Destination	Result
+41791231234 sends an SMS "START INFONEWS"	SUBSCRIBE	INFONEWS	URL1/email1	URL1 or email1 will be notified every time an end-user subscribes to service INFONEWS.
+41791231234 sends an SMS "STOP ALL"	UNSUBSCRIBEALL	(none)	URL2/email2	URL2 or email2 will be notified every time somebody sends a "stop all" command.
+41791231234 sends an SMS "PARTY HELLO EVERYBODY"	FORWARD	PARTY	URL3/email3	URL3 or email3 will be notified every time an SMS is received beginning with the word PARTY.

The command notifications contain the information data about the command processed.

The XML document contains elements describing the base command notification, as well as those that describe any special parameters of the command that generated this notification. The XML of the base command notification contains elements representing the instance, the sender, the sender's operator, the service, the language, the command, and a unique identifier for this command request (unique within a single instance). Specific command elements are represented as sub elements of the `parameters` element.

The XML document is made of a root element enclosing the base notification elements and command specific elements. The root element is named `Notification`. The base elements are `instance`, `sender`, `operator`, `service`, `language`, `command` and optionally `requestUid`. The command specific elements are sub elements of the `parameters` element. Command specific elements vary by type of command and thus each command has its own DTD.

4.2.1. HTTP GET alternative

Instead of sending an XML document by HTTP POST, it is also possible to use HTTP GET notifications. As this API documentation details all notifications using XML documents by HTTP POST, when using the HTTP GET method instead, all documented XML elements are sent as GET query string parameters (URL-encoded with the UTF-8 encoding).

Notice: the expected notification reply does not change. It is still a NotificationReply XML document.

4.2.2. Generic Command Notification DTD

Here is the base DTD "template" for all command notifications. As different parameters are transmitted among different commands, the `parameters` element is variable.

```
<!ELEMENT Notification (instance,sender,operator,service,language,
                            command,receptionDate,requestUid?,parameters)>
<!ELEMENT instance     (#PCDATA)>
<!ELEMENT sender       (#PCDATA)>
<!ELEMENT operator     (#PCDATA)>
<!ELEMENT service      (#PCDATA)>
<!ELEMENT language     (#PCDATA)>
<!ELEMENT command      (#PCDATA)>
<!ELEMENT receptionDate (#PCDATA)>
<!ELEMENT requestUid   (#PCDATA)>
<!ELEMENT parameters   ( -- unique to each command notification -- )>
```

4.2.2.1. Generic Parameters

Parameter	Description
instance	System instance the notification originates from. Normally, an instance corresponds to one short number.
sender	Phone number of the end-user. Normally an MSISDN in international format (+417xyyyzzzz, +336xxyyzztt, etc); it can also be an encrypted MSISDN if applicable to you and/or instance (normally a 32 characters hexadecimal series such as 10A74C23F1F5D1A8A84885E2005AE24F).
operator	Name of the operator.
service	Name of the service or keyword that triggered the notification. May be empty.
language	Language of the service or keyword that triggered the notification, as ISO 639-1 language code. If the notification is not about a service, it is the command language. By default: `en` for english.
command	Name of the command that triggered the notification.
receptionDate	The date/time the SMS was received, formatted as `YYYY-MM-DD hh:mm:ss.mmm`, e.g. for example `2012-07-27 15:58:21.581`, expressed in the instance timezone (constant, fetchable with the INSTANCEINFO command if needed). May be useful when the platform is busy, since the notification may be transmitted with a noticeable delay; e.g. some specific services may need a different reply when the reception date is noticeably far in the past.
requestUid	A value uniquely identifying the command request in the instance. Starts with a word identifying the media which generated the command, which can be "sms", "mms", "wap" (= Mobile Internet, kept for compatibility) or "xml", and is followed by a variable amount of digits. Is primarily used by partners receiving also sent message notifications, to match sent messages with command requests. May be missing, in case no command generated that notification (for example, internal cleanup daemons in smsBox® may generate AUTOUNSUBSCRIBEALL notifications).

4.2.3. Command Notification Reply DTD

When receiving a command notification XML document from smsBox® by HTTP POST, the external application is required to reply with an XML document. It can optionally contain a list of messages to send to the mobile user who triggered the command. For all types of command notifications, the reply structure is the same:

```
<!ELEMENT NotificationReply (message*)>
<!ELEMENT message (text,cost?,maximumSMSAmount?)>
<!ELEMENT text (#PCDATA)>
<!ELEMENT cost (#PCDATA)>
<!ELEMENT maximumSMSAmount (#PCDATA)>
```

There can be 0 or more `message` elements. One or several SMS message are sent per `message` element (details below). If no `message` elements are present in the reply XML document, there will not be an SMS sent from the notification (depending on the configuration of the notification on the smsBox® platform, a message may have already been sent from the SMS command; check with your support team).

There is a limit in the notification reply size. By default it is of 10 KB for SMS notifications, and 1 MB for MMS notifications.

When receiving a command notification from smsBox® by Internet mail ("email"), the external application cannot reply, thus it must use an XML API command if it needs to send a message to the mobile user.

Note: the `cost` element is needed for each message if MT billing is available in smsBox®.

4.2.3.1. Parameters

Parameter	Description
text	Text of the message to send to the end-user as confirmation of his/her command. Size is limited, see below.
cost	Cost of the confirmation SMS, expressed in cents. Needed if MT billing is available. Possible costs are limited by the contractual chargeable amount with the operators and the accesses configured for the partner by the support team.
maximumSMSAmount	The maximum number of SMS to generate for this message, in case concatenated SMS is needed.

If smsBox® is configured to not allow concatenated SMS, or if `maximumSMSAmount` is omitted, then only one SMS will be generated per `message` element; in that case, the message will be cut if it is longer than 160 characters and it contains only western latin characters or smsBox® is configured to disallow international characters, or 70 characters if there is at least one international character.

If `maximumSMSAmount` is specified and smsBox® allows concatenated SMS, then the message will be cut to 160 / 70 characters if `maximumSMSAmount`'s value is 1, else to 153*`maximumSMSAmount` / 65*`maximumSMSAmount` characters.

Warning, if concatenated SMS are generated, the operator will typically charge for each SMS independantly. In other words, if a message of 3 concatenated SMS is generated, operator charging will probably be tripled. For any question regarding that issue, please contact the support team for details.

4.2.4. Command Notification Types

4.2.4.1. FORWARD

The FORWARD command is a special command notification. It disables command processing of the smsBox® platform: it's a catch-all for any SMS content, provided it matches the regexp for which a given FORWARD command notification is configured.

In case of HTTP transport, the system **enforces a 20 seconds timeout** for the command notification reply.

The FORWARD command notification is triggered basing on a regular expression matching he two first words of the message. All MO-SMS beginning with keywords matching it will be forwarded to the external application. Only the support team can define the regular expression.

The smsBox® platform can be configured to respond to the end-user directly, or use the response of the external application (latter only in case of HTTP transport).

4.2.4.1.1. Forward Notification DTD

```
<!ELEMENT Notification (instance,sender,operator,service,language,
                        command,receptionDate,requestUid,parameters)>
<!ELEMENT instance     (#PCDATA)>
<!ELEMENT sender       (#PCDATA)>
<!ELEMENT operator     (#PCDATA)>
<!ELEMENT service      (#PCDATA)>
<!ELEMENT language     (#PCDATA)>
<!ELEMENT command      (#PCDATA)>
<!ELEMENT receptionDate (#PCDATA)>
<!ELEMENT requestUid   (#PCDATA)>
<!ELEMENT parameters   (text)>
<!ELEMENT text         (#PCDATA)>
```

4.2.4.1.2. Parameters

Parameter	Description
text	Text of the message as received from the end-user.

The `service` element contains the first word of the message.

4.2.4.1.3. Forward Notification Example

Let's suppose we have a forwarding configured on a message beginning with the word "CYCLE". The external application responds with two SMS with different pricing:

Notification:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Notification>
  <instance>pro</instance>
  <sender>+41761234567</sender>
  <operator>sunrise</operator>
  <service>CYCLE</service>
  <language>en</language>
  <command>FORWARD</command>
  <receptionDate>2012-07-27 15:58:21.581</receptionDate>
  <requestUid>sms9676205</requestUid>
  <parameters>
    <text>cycle This is a text message.</text>
  </parameters>
</Notification>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<NotificationReply>
  <message>
    <text>Thanks for sending the message!</text>
    <cost>20</cost>
  </message>
  <message>
    <text>And enjoy our service!</text>
    <cost>50</cost>
  </message>
</NotificationReply>
```

4.2.4.1.4. HTTP GET alternative

The same notification, when opting for the HTTP GET transport, would be:

```
...base url...?instance=pro&sender=%2B41761234567&operator=sunrise&service=CYCLE
&language=en&command=FORWARD&receptionDate=2012-07-27+15%3A58%3A21.581
&requestUid=sms9676205&text=cycle+This+is+a+text+message
```

Notice: the expected notification reply does not change. It is still a NotificationReply XML document.

4.2.4.2. Simple command notifications

Some of the command notifications are tied to smsBox® base commands. These are always executed on the platform and don't require external input. These command notifications are then in most cases informational only, in other words they do not require a response. The external application has to acknowledge them and react internally accordingly. These commands are:

Command Name	Description
SUBSCRIBE	The user has subscribed to a service. Also triggered in case of a subscription done through Command API (interactions *to* smsBox®).
UNSUBSCRIBE	The user has cancelled his/her subscription to the service. Also triggered in case of a subscription done through Command API.
GET	The user has retrieved content from a service in PULL mode (only once - no subscription involved).
OPLOGO	The user has downloaded a Nokia Operator Logo (PULL).
MYTHEMES	The user has retrieved the list of services he/she owns.
RETRIEVE	The user has retrieved the SMS password of one of his/her services.
SUBSCRIBERCOUNT	The user has retrieved the number of subscribers to one of his/her services.
VIEW	The user has retrieved the list of services he/she is subscribed to.
VOTE	The user has voted for a candidate.
UPDATE	The user has updated the base parameters of a service.

4.2.4.3. Simple command notification

In the generic case of the commands listed above, the command notification and reply always look the same:

4.2.4.3.1. Simple Command Notification DTD

```
<!ELEMENT Notification (instance,sender,operator,service,language,
                            command,receptionDate,requestUid?,parameters)>
<!ELEMENT instance     (#PCDATA)>
<!ELEMENT sender       (#PCDATA)>
<!ELEMENT operator     (#PCDATA)>
<!ELEMENT service      (#PCDATA)>
<!ELEMENT language     (#PCDATA)>
<!ELEMENT command      (#PCDATA)>
<!ELEMENT receptionDate (#PCDATA)>
<!ELEMENT requestUid   (#PCDATA)>
<!ELEMENT parameters   (text)>
<!ELEMENT text         (#PCDATA)>
```

4.2.4.3.2. Parameters

Parameter	Description
text	Text of the raw message as received from the end-user.

4.2.4.3.3. Subscribe Notification Example

Notification:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Notification>
  <instance>pro</instance>
  <sender>+41761234567</sender>
  <operator>sunrise</operator>
  <service>XCBNEWS</service>
  <language>en</language>
  <command>SUBSCRIBE</command>
  <receptionDate>2012-07-27 15:58:21.581</receptionDate>
  <requestUid>sms9676206</requestUid>
  <parameters>
```

```
      <text>START XCBNEWS</text>
    </parameters>
  </Notification>
```

Reply:

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

4.2.4.4. AUTOUNSUBSCRIBE(ALL)

The AUTOUNSUBSCRIBE command notification is very similar to the UNSUBSCRIBE command notification. It is however transmitted when the unsubscription event happens for "internal" reasons: spending credits reaches zero or a status indicated that the end-user has left the operator.

The system has to unsubscribe that end-user from a specific service. With the AUTOUNSUBSCRIBE command notification, the platform is able to notify external systems that the end-user's subscription must be removed from their databases.

AUTOUNSUBSCRIBEALL notifies in case of a status indicating that the end-user is not active anymore for the operator. All subscriptions have to be cancelled. Depending on smsBox® configuration, the AUTOUNSUBSCRIBEALL command notification may contain the list of services the end-user was subscribed to; it is transmitted as a semicolon (;) separated list of service names the end-user was subscribed to and the subscription date, surrounded by square brackets under the parameters element `subscriptions` (see documentation of UNSUBSCRIBEALL for details and examples). Additionally, a `reason` element may be provided to explain why this automatic unsubscription happened; current list of possible reasons:

Reason	Description
`selfblacklist`	The end-user has blacklisted himself from the shortnumber.
`blacklisted`	The end-user has been blacklisted from the shortnumber by an administrator or service owner.
`noCredit-LimitPerService`	End-user reached the configured limit per service.
`noCredit-LimitPerSubscription`	End-user reached the configured limit per subscription.
`tooManyFailedSentMessages`	The end-user has had too many failed sent messages over a recent period.
`operator-ported`	The end-user has changed mobile operator.
`AdC invalid`	The SMSC reports that the MSISDN is invalid.
`Unknown subscriber`	The SMSC reports that the MSISDN is unassigned.
`Illegal subscriber`	The SMSC reports that the MSISDN is illegal; often used by foreign operators to block SMS from other networks.
`Equipment not SM equipped`	The SMSC reports that receiving equipment is not a mobile phone.
`Barring service active`	The SMSC reports that end-user has blocked SMS.
`Operator barring`	The SMSC reports that operator has blocked SMS.
`No translation for an address of such nature`	The SMSC reports that the number range isn't known in the network.
`No translation for this specific address`	The SMSC reports that the number isn't known in the network.

Unequipped user	The SMSC reports that receiving equipment is not a mobile phone.
410-not-orange-customer	The Orange gateway reports the MSISDN is not from an Orange end-user.
failed|invalid-msisdn-format	The Swisscom gateway reports the MSISDN format is incorrect.
failed|no-scmn	The Swisscom gateway reports the MSISDN is not from a Swisscom end-user.
failed|all-premium-scm	The Swisscom gateway reports that Swisscom blocked premium content.
failed|all-premium-cust	The Swisscom gateway reports that end-user blocked premium content.
failed|adult-premium-scm	The Swisscom gateway reports that Swisscom blocked adult content.
failed|adult-premium-cust	The Swisscom gateway reports that end-user blocked adult content.
failed|reached-scm	The Swisscom gateway reports that end-user reached the Swisscom-set limit.
failed|reached-cust	The Swisscom gateway reports that end-user reached the end-user-set limit.
failed|age-verification-not-passed	The Swisscom gateway reports that end-user didn't pass age verification.
failed|age-verification-not-available	The Swisscom gateway reports that end-user age is unknown.
failed|age-verification-failed	The Swisscom gateway reports that end-user age verification failed.
failed|blocked-msisdn	The Swisscom gateway reports that the MSISDN is blocked.
failed|err-dst	The Swisscom gateway reports that the destination is not a standard MSISDN or a foreign MSISDN (premium shortnumbers only).
failed|invalid-msisdn	The Swisscom gateway reports that the MSISDN is invalid or not a Swisscom end-user.
failed|invalid-customer	The Swisscom gateway reports that the MSISDN is not a Swisscom end-user or is inactive.

4.2.4.5. CHAT

The CHAT command notification happens when a user sends an SMS to a chat service. The contents of the SMS is normally the name of the chat service followed by the chat message. The receiving and processing of such an SMS will trigger a CHAT command notification.

For the CHAT types other than 'moderated', the smsBox® platform will reply itself to the end-user request message: the NotificationReply has to be empty.

4.2.4.5.1. CHAT Notification DTD

```
<!ELEMENT Notification (instance,sender,operator,service,language,
                            command,receptionDate,requestUid,parameters)>
<!ELEMENT instance     (#PCDATA)>
<!ELEMENT sender       (#PCDATA)>
<!ELEMENT operator     (#PCDATA)>
<!ELEMENT service      (#PCDATA)>
<!ELEMENT language     (#PCDATA)>
<!ELEMENT command      (#PCDATA)>
<!ELEMENT receptionDate (#PCDATA)>
<!ELEMENT requestUid   (#PCDATA)>
```

```
<!ELEMENT parameters   (text,id?)>
<!ELEMENT text         (#PCDATA)>
<!ELEMENT id           (#PCDATA)>
```

4.2.4.5.2. Specific Parameters

Parameter	Description
text	Text of the chat message as submitted by the end-user (the chat service name is removed).
id	The identifier of the chat message used in the case of moderated chats. This id can be used to publish or delete the message using the XML API command MODERATEDCHAT. It is optional and present only for chat services of type `moderated`.

4.2.4.5.3. CHAT Notification Example 1

The user sends an SMS with "cool How are you guys?" and COOL is a chat service.

Notification:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Notification>
  <instance>pro</instance>
  <sender>+41761234567</sender>
  <operator>sunrise</operator>
  <service>COOL</service>
  <language>de</language>
  <command>CHAT</command>
  <receptionDate>2012-07-27 15:58:21.581</receptionDate>
  <requestUid>sms9676207</requestUid>
  <parameters>
    <text>How are you guys?</text>
  </parameters>
</Notification>
```

Reply:

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

4.2.4.5.4. CHAT Notification Example 2

The user sends an SMS with "disco How are you guys?" and DISCO is a moderated chat.

Notification:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Notification>
  <instance>pro</instance>
  <sender>+41761234567</sender>
  <operator>sunrise</operator>
  <service>DISCO</service>
  <language>de</language>
  <command>CHAT</command>
  <receptionDate>2012-07-27 15:58:21.581</receptionDate>
  <requestUid>sms9676208</requestUid>
  <parameters>
    <text>How are you guys?</text>
    <id>3267</id>
  </parameters>
</Notification>
```

Reply:

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

4.2.4.6. CREATE

The CREATE command notification is sent when a user creates a new service.

4.2.4.6.1. CREATE Notification DTD

```
<!ELEMENT Notification (instance,sender,operator,service,language,
                            command,receptionDate,requestUid,parameters)>
<!ELEMENT instance     (#PCDATA)>
<!ELEMENT sender       (#PCDATA)>
<!ELEMENT operator     (#PCDATA)>
<!ELEMENT service      (#PCDATA)>
<!ELEMENT language     (#PCDATA)>
<!ELEMENT command      (#PCDATA)>
<!ELEMENT receptionDate (#PCDATA)>
<!ELEMENT requestUid   (#PCDATA)>
<!ELEMENT parameters   (text,password,description?,keywords?,type?,
                                subscriptionType?,subscriptionTypeData?,...)>
<!ELEMENT text         (#PCDATA)>
<!ELEMENT password     (#PCDATA)>
<!ELEMENT description  (#PCDATA)>
<!ELEMENT keywords     (#PCDATA)>
<!ELEMENT type         (#PCDATA)>
<!ELEMENT subscriptionType (#PCDATA)>
<!ELEMENT subscriptionTypeData (#PCDATA)>
```

4.2.4.6.2. Common specific Parameters

Parameter	Description
password	The SMS service password as generated automatically by the platform.
text	Raw message from the end-user.
description	Description of the service.
keywords	Keywords associated to the service, for indexing use; comma-separated simple english words.
type	Type of the service: standard/info, mms, chatstandard, chatmoderated, chatkeywordless, page (= Mobile Internet), election.
subscriptionType	Subscription type (if subscription has a limit).
subscriptionTypeData	Limit details.

In addition to the parameters described above, all the parameters returned by the command SERVICEINFO (except the parameters: `service`, `language`, `type` and `keywords` already present) are included in the notification, at the end of the `<parameters>` element.

4.2.4.6.3. CREATE Notification Example 1: standard service

```
<?xml version="1.0" encoding="UTF-8" ?>
<Notification>
  <instance>pro</instance>
  <sender>105</sender>
  <operator>xml</operator>
  <service>MYSERVICE</service>
  <language>en</language>
  <command>CREATE</command>
  <receptionDate>2012-07-27 15:58:21.581</receptionDate>
  <requestUid>xml9676209</requestUid>
  <parameters>
    <type>standard</type>
    <description></description>
    <keywords>News,Football</keywords>
    <subscriptionType>0</subscriptionType>
    <subscriptionTypeData>0</subscriptionTypeData>
    <password>QWERTZ</password>
    <owner>108</owner>
    <subscribers>0</subscribers>
    <sendCurrentMessageOnSubscription>false</sendCurrentMessageOnSubscription>
    <requireSubscriptionCode>false</requireSubscriptionCode>
    <creationDate>2008-05-08 17:09:03</creationDate>
```

```
      <lastModified>2008-05-08 17:09:03</lastModified>
    </parameters>
  </Notification>
```

4.2.4.6.4. CREATE Notification Example 2: keywordless-chat service

```
  <?xml version="1.0" encoding="UTF-8" ?>
  <Notification>
    <instance>pro</instance>
    <sender>105</sender>
    <operator>xml</operator>
    <service>MYCHAT</service>
    <language>en</language>
    <command>CREATE</command>
    <receptionDate>2012-07-27 15:58:21.581</receptionDate>
    <requestUid>xml9676209</requestUid>
    <parameters>
      <type>chatkeywordless</type>
      <description>This is my chat!</description>
      <keywords></keywords>
      <subscriptionType>0</subscriptionType>
      <subscriptionTypeData>0</subscriptionTypeData>
      <password>LS4ZAXE</password>
      <owner>10010</owner>
      <subscribers>0</subscribers>
      <sendCurrentMessageOnSubscription>false</sendCurrentMessageOnSubscription>
      <requireSubscriptionCode>false</requireSubscriptionCode>
      <creationDate>2007-07-23 17:21:46</creationDate>
      <lastModified>2007-07-23 17:21:46</lastModified>
      <keywordless-chat>
        <bar>30</bar>
        <type>Keywordless</type>
        <autoSubscribe>true</autoSubscribe>
        <state>Active</state>
      </keywordless-chat>
    </parameters>
  </Notification>
```

Reply:

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

4.2.4.7. DELETE

The DELETE command notification happens when a user destroys a service.

4.2.4.7.1. DELETE Notification DTD

```
  <!ELEMENT Notification (instance,sender,operator,service,language,
                               command,receptionDate,requestUid,parameters)>
  <!ELEMENT instance     (#PCDATA)>
  <!ELEMENT sender       (#PCDATA)>
  <!ELEMENT operator     (#PCDATA)>
  <!ELEMENT service      (#PCDATA)>
  <!ELEMENT language     (#PCDATA)>
  <!ELEMENT command      (#PCDATA)>
  <!ELEMENT receptionDate (#PCDATA)>
  <!ELEMENT requestUid   (#PCDATA)>
  <!ELEMENT parameters   (text)>
  <!ELEMENT text         (#PCDATA)>
```

4.2.4.7.2. Specific Parameters

The `<text>` parameter indicates the full SMS content received to trigger the DELETE command.

4.2.4.7.3. DELETE Notification Example 1

Notification:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Notification>
  <instance>pro</instance>
  <sender>+41761234567</sender>
  <operator>sunrise</operator>
  <service>MOBOX XYZ</service>
  <language>en</language>
  <command>DELETE</command>
  <receptionDate>2012-07-27 15:58:21.581</receptionDate>
  <requestUid>sms9676210</requestUid>
  <parameters>
    <text>CANCEL MOBOX XYZ 1234</text>
  </parameters>
</Notification>
```

Reply:

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

Remark: CANCEL in this case is the SMS alias for the DELETE command.

4.2.4.8. ELECTION

The ELECTION command notification happens when some actions are happening on an election/voting service.

4.2.4.8.1. ELECTION Notification DTD

```
<!ELEMENT Notification (instance,sender,operator,service,language,
                            command,receptionDate,requestUid,parameters)>
<!ELEMENT instance     (#PCDATA)>
<!ELEMENT sender       (#PCDATA)>
<!ELEMENT operator     (#PCDATA)>
<!ELEMENT service      (#PCDATA)>
<!ELEMENT language     (#PCDATA)>
<!ELEMENT command      (#PCDATA)>
<!ELEMENT receptionDate (#PCDATA)>
<!ELEMENT requestUid   (#PCDATA)>
<!ELEMENT parameters   (subcommand,winner?)>
<!ELEMENT subcommand   (#PCDATA)>
<!ELEMENT winner       (#PCDATA)>
```

4.2.4.8.2. Specific Parameters

Parameter	Description
subcommand	The subcommand - same as the `action` field in the ELECTION command.
winner	If `subcommand` is LOTTERY or LUCKYWINNER, the MSISDN of the winner; in case of multiple winners, the comma-separated list of MSISDNs of the winners.

4.2.4.8.3. ELECTION Notification Example 1

Notification:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Notification>
  <instance>pro</instance>
  <sender>+41761234567</sender>
  <operator>sunrise</operator>
  <service>ELEC</service>
  <language>en</language>
  <command>ELECTION</command>
  <receptionDate>2012-07-27 15:58:21.581</receptionDate>
  <requestUid>sms9676210</requestUid>
  <parameters>
```

```
      <subcommand>LOTTERY</subcommand>
      <winner>+41767654321</winner>
    </parameters>
  </Notification>
```

Reply:

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

4.2.4.9. MODERATEDCHAT

The MODERATEDCHAT command notification happens when an action is triggered on a moderated chat (from XML API or from the web administration interface).

4.2.4.9.1. MODERATEDCHAT Notification DTD

```
<!ELEMENT Notification (instance,sender,operator,service,language,
                            command,receptionDate,requestUid,parameters)>
<!ELEMENT instance     (#PCDATA)>
<!ELEMENT sender       (#PCDATA)>
<!ELEMENT operator     (#PCDATA)>
<!ELEMENT service      (#PCDATA)>
<!ELEMENT language     (#PCDATA)>
<!ELEMENT command      (#PCDATA)>
<!ELEMENT receptionDate (#PCDATA)>
<!ELEMENT requestUid   (#PCDATA)>
<!ELEMENT parameters   (type,action,message*,send*,delete*)>
<!ELEMENT type         (#PCDATA)>
<!ELEMENT action       (#PCDATA)>
<!ELEMENT message      (id,msisdn,text,receivedDate,sent)>
<!ELEMENT id           (#PCDATA)>
<!ELEMENT msisdn       (#PCDATA)>
<!ELEMENT text         (#PCDATA)>
<!ELEMENT receivedDate (#PCDATA)>
<!ELEMENT sent         (#PCDATA)>
<!ELEMENT send         (#PCDATA)>
  <!ATTLIST send status CDATA #REQUIRED>
  <!ATTLIST send message CDATA #REQUIRED>
<!ELEMENT delete       (#PCDATA)>
  <!ATTLIST delete status CDATA #REQUIRED>
```

4.2.4.9.2. Specific Parameters

Parameter	Description
type	The type of chat (`chat` for SMS chat, `mmschat` for MMS chat).
action	The action triggered; see the MODERATEDCHAT command documentation for the list of possible actions.
message	The retrieved messages in case of RETRIEVE action.
id	The id of the message.
msisdn	The MSISDN who sent the message.
text	The text content of the message.
receivedDate	The date the message was received, in format `yyyy-MM-dd HH:mm:ss`.
sent	`true` is this message was sent (published) to the subscribers of the chat, or `false`.
send	The outcome of a message to send in case of SEND action; attribute `status` will contain `ok` on success or an error code, `message` the text content of the message, CDATA is the id of the message.
delete	The outcome of a message to delete in case of DELETE action; attribute `status` will contain `ok` on success or an error code, CDATA is the id of the message.

4.2.4.9.3. MODERATEDCHAT Notification Example, SEND action

Notification:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Notification>
  <instance>pro</instance>
  <sender></sender>
  <operator>xml</operator>
  <service>ONE</service>
  <language>en</language>
  <command>MODERATEDCHAT</command>
  <receptionDate>2012-07-27 15:58:21.581</receptionDate>
  <requestUid>xml9830827</requestUid>
  <parameters>
    <type>chat</type>
    <action>SEND</action>
    <send status="ok" message="Hello ONE FM &amp; greetings!">601940</send>
  </parameters>
</Notification>
```

4.2.4.9.4. MODERATEDCHAT Notification Example, RETRIEVE action

Notification:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Notification>
  <instance>pro</instance>
  <sender></sender>
  <operator>xml</operator>
  <service>ONE</service>
  <language>en</language>
  <command>MODERATEDCHAT</command>
  <receptionDate>2012-07-27 15:58:21.581</receptionDate>
  <requestUid>xml9830827</requestUid>
  <parameters>
    <type>chat</type>
    <action>RETRIEVE</action>
    <message>
      <id>601943</id>
      <msisdn>+41761234567</msisdn>
      <text>Hello ONE FM &amp; greetings!</text>
      <receivedDate>2008-01-11 14:31:21</receivedDate>
      <sent>true</sent>
    </message>
  </parameters>
</Notification>
```

4.2.4.10. PASSWORD

The PASSWORD command notification happens when a user changes the password of his/her service.

4.2.4.10.1. PASSWORD Notification DTD

```
<!ELEMENT Notification (instance,sender,operator,service,language,
                              command,receptionDate,requestUid,parameters)>
<!ELEMENT instance    (#PCDATA)>
<!ELEMENT sender      (#PCDATA)>
<!ELEMENT operator    (#PCDATA)>
<!ELEMENT service     (#PCDATA)>
<!ELEMENT language    (#PCDATA)>
<!ELEMENT command     (#PCDATA)>
<!ELEMENT receptionDate (#PCDATA)>
<!ELEMENT requestUid  (#PCDATA)>
<!ELEMENT parameters  (text,newPassword)>
<!ELEMENT text        (#PCDATA)>
<!ELEMENT newPassword (#PCDATA)>
```

4.2.4.10.2. Specific Parameters

Parameter	Description
newPassword	The new service password as defined by the user.

4.2.4.10.3. PASSWORD Notification Example

The old password for the service MOBOX XYZ was "EASY" and the end-user just changed it to "1234":

Notification:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Notification>
  <instance>pro</instance>
  <sender>+41761234567</sender>
  <operator>sunrise</operator>
  <service>MOBOX XYZ</service>
  <language>en</language>
  <command>PASSWORD</command>
  <receptionDate>2012-07-27 15:58:21.581</receptionDate>
  <requestUid>sms9676211</requestUid>
  <parameters>
    <text>PASS MOBOX XYZ EASY 1234</text>
    <newPassword>1234</newPassword>
  <parameters/>
</Notification>
```

Reply:

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

4.2.4.11. POST

The POST command notification happens when an end-user uploads content successfully to a service. This can happen with standard "info services", but also with chats.

4.2.4.11.1. POST Notification DTD

```
<!ELEMENT Notification (instance,sender,operator,service,language,
                            command,receptionDate,requestUid,parameters)>
<!ELEMENT instance     (#PCDATA)>
<!ELEMENT sender       (#PCDATA)>
<!ELEMENT operator     (#PCDATA)>
<!ELEMENT service      (#PCDATA)>
<!ELEMENT language     (#PCDATA)>
<!ELEMENT command      (#PCDATA)>
<!ELEMENT receptionDate (#PCDATA)>
<!ELEMENT requestUid   (#PCDATA)>
<!ELEMENT parameters   (message)>
<!ELEMENT text         (#PCDATA)>
```

4.2.4.11.2. Specific Parameters

Parameter	Description
text	The content as uploaded by the end-user (the service name and password are removed).

4.2.4.11.3. Post Notification Example

Notification:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Notification>
  <instance>pro</instance>
  <sender>+41761234567</sender>
  <operator>sunrise</operator>
  <service>HCXNEWS</service>
```

```
    <language>de</language>
    <command>POST</command>
    <receptionDate>2012-07-27 15:58:21.581</receptionDate>
    <requestUid>sms9676212</requestUid>
    <parameters>
      <text>Hockey Club Xenon has won the championship!</text>
    </parameters>
  </Notification>
```

Reply:

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

4.2.4.12. SESSIONMESSAGE

The SESSIONMESSAGE command notification happens when an end-user is currently within a session (by using the XML API command OPENSESSION) and sends an MO SMS which isn't handled by a command (e.g. which doesn't start with a common keyword such as START, STOP, VOTE etc).

4.2.4.12.1. SESSIONMESSAGE Notification DTD

```
<!ELEMENT Notification (instance,sender,operator,service,language,
                            command,receptionDate,requestUid,parameters)>
<!ELEMENT instance     (#PCDATA)>
<!ELEMENT sender       (#PCDATA)>
<!ELEMENT operator     (#PCDATA)>
<!ELEMENT service      (#PCDATA)>
<!ELEMENT language     (#PCDATA)>
<!ELEMENT command      (#PCDATA)>
<!ELEMENT receptionDate (#PCDATA)>
<!ELEMENT requestUid   (#PCDATA)>
<!ELEMENT parameters   (message)>
<!ELEMENT text         (#PCDATA)>
```

4.2.4.12.2. Specific Parameters

Parameter	Description
text	The content of the MO SMS.

4.2.4.12.3. Post Notification Example

Notification:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Notification>
  <instance>pro</instance>
  <sender>+41761234567</sender>
  <operator>sunrise</operator>
  <service>HCXNEWS</service>
  <language>de</language>
  <command>SESSIONMESSAGE</command>
  <receptionDate>2012-07-27 15:58:21.581</receptionDate>
  <requestUid>sms9676212</requestUid>
  <parameters>
    <text>I am interested in this offer.</text>
  </parameters>
</Notification>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<NotificationReply>
  <message>
    <text>Thanks, we registered your order.</text>
    <cost>20</cost>
  </message>
</NotificationReply>
```

4.2.4.13. SETSERVICEINFO

The SETSERVICEINFO command notification happens when a user defines or changes the description/keywords for a service (a feature not available on all instances). The command notification document provides the new description/keywords of the service.

4.2.4.13.1. SETSERVICEINFO Notification DTD

```
<!ELEMENT Notification (instance,sender,operator,service,language,
                             command,receptionDate,requestUid,parameters)>
<!ELEMENT instance     (#PCDATA)>
<!ELEMENT sender       (#PCDATA)>
<!ELEMENT operator     (#PCDATA)>
<!ELEMENT service      (#PCDATA)>
<!ELEMENT language     (#PCDATA)>
<!ELEMENT command      (#PCDATA)>
<!ELEMENT receptionDate (#PCDATA)>
<!ELEMENT requestUid   (#PCDATA)>
<!ELEMENT parameters   (description,keywords)>
<!ELEMENT description  (#PCDATA)>
<!ELEMENT keywords     (#PCDATA)>
```

4.2.4.13.2. Specific Parameters

Parameter	Description
description	The service description.
keywords	The service keywords, as comma-separated simple english words.

4.2.4.13.3. SETSERVICEINFO Notification Example

Notification:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Notification>
  <instance>sky</instance>
  <sender>+3361231234</sender>
  <operator>sfr</operator>
  <service>TARZAN</service>
  <language>en</language>
  <command>SETSERVICEINFO</command>
  <receptionDate>2012-07-27 15:58:21.581</receptionDate>
  <requestUid>sms9676213</requestUid>
  <parameters>
    <description>The jungle service</description>
    <keywords>Lion,Elephant,Zebra,Gorilla</keywords>
  </parameters>
</Notification>
```

Reply:

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

4.2.4.14. UNSUBSCRIBEALL

The UNSUBSCRIBEALL command notification is sent when the user has cancelled all his/her subscriptions to any service on the instance / short number. Depending on smsBox® configuration, the command notification document may contain the list of services the end-user was subscribed to; it is transmitted as a semicolon (;) separated list of service names the end-user was subscribed to and the subscription date, surrounded by square brackets under the parameters element `subscriptions`.

4.2.4.14.1. UNSUBSCRIBEALL Notification DTD

```
<!ELEMENT Notification (instance,sender,operator,service,language,
                            command,receptionDate,requestUid,parameters)>
<!ELEMENT instance      (#PCDATA)>
<!ELEMENT sender        (#PCDATA)>
<!ELEMENT operator      (#PCDATA)>
<!ELEMENT service       (#PCDATA)>
<!ELEMENT language      (#PCDATA)>
<!ELEMENT command       (#PCDATA)>
<!ELEMENT receptionDate (#PCDATA)>
<!ELEMENT requestUid    (#PCDATA)>
<!ELEMENT parameters    (text,subscriptions?)>
<!ELEMENT text          (#PCDATA)>
<!ELEMENT subscriptions (#PCDATA)>
```

4.2.4.14.2. Specific Parameters

Parameter	Description
text	The message sent by the end-user which triggered the UNSUBSCRIBEALL command. Typically "STOP" or "STOPALL".
subscriptions	A semicolon (;) separated list of service names the end-user was subscribed to and the subscription date, surrounded by square brackets.

4.2.4.14.3. UNSUBSCRIBEALL Notification Example

Notification:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Notification>
  <instance>pro</instance>
  <sender>+3361231234</sender>
  <operator>sfr</operator>
  <service/>
  <language>en</language>
  <command>UNSUBSCRIBEALL</command>
  <receptionDate>2012-07-27 15:58:21.581</receptionDate>
  <requestUid>sms9676214</requestUid>
  <parameters>
    <text>stop</text>
    <subscriptions>[FOOTBALL|2007-02-03 10:18:31];[HOCKEY|2007-09-19 12:12:34] </subscriptions>
  </parameters>
</Notification>
```

Reply:

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

4.2.4.15. UNKNOWN

When the system receives an SMS with a first word that is not a command and cannot be interpreted as a service, the "UNKNOWN" command is executed.

On such a circumstance, an external application can still receive a command notification to implement commands that are not known to smsBox®. For example, let's imagine an application that has the command TRANSPOSE (whatever that is). If a user sends an SMS with the text TRANSPOSE MAX, a notification of type UNKNOWN can be sent based on the "service" name TRANSPOSE. This lets the external application determine what to do with this TRANSPOSE "command".

This command notification is different from a FORWARD, because FORWARD is performed before normal command processing. The UNKNOWN command notification can be considered as a forwarding performed *after* normal command processing. For example, START MAX will not generate an UNKNOWN notification, as START is a defined command keyword.

4.2.4.15.1. Unknown Notification DTD

```
<!ELEMENT Notification (instance,sender,operator,service,language,
                            command,receptionDate,requestUid,parameters)>
<!ELEMENT instance     (#PCDATA)>
<!ELEMENT sender       (#PCDATA)>
<!ELEMENT operator     (#PCDATA)>
<!ELEMENT service      (#PCDATA)>
<!ELEMENT language     (#PCDATA)>
<!ELEMENT command      (#PCDATA)>
<!ELEMENT receptionDate (#PCDATA)>
<!ELEMENT requestUid   (#PCDATA)>
<!ELEMENT parameters   (text)>
```

4.2.4.15.2. Unknown Notification Example

Notification:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Notification>
  <instance>pro</instance>
  <sender>+41791112222</sender>
  <operator>swisscom</operator>
  <service>TRANSPOSE</service>
  <language>en</language>
  <command>UNKNOWN</command>
  <receptionDate>2012-07-27 15:58:21.581</receptionDate>
  <requestUid>sms9676215</requestUid>
  <parameters>
    <text>TRANSPOSE MAX</text>
  </parameters>
</Notification>
```

Reply:

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

4.2.4.16. UPDATE

The UPDATE notification is send when the user update the base parameters of a service.

4.2.4.16.1. UPDATE Notification DTD

```
<!ELEMENT Notification (instance,sender,operator,service,language,
                            command,receptionDate,requestUid,parameters)>
<!ELEMENT instance     (#PCDATA)>
<!ELEMENT sender       (#PCDATA)>
<!ELEMENT operator     (#PCDATA)>
<!ELEMENT service      (#PCDATA)>
<!ELEMENT language     (#PCDATA)>
<!ELEMENT command      (#PCDATA)>
<!ELEMENT receptionDate (#PCDATA)>
<!ELEMENT requestUid   (#PCDATA)>
<!ELEMENT parameters   (text)>
```

4.2.4.16.2. Update Notification Example

Notification:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Notification>
  <instance>pro</instance>
  <sender>+41791112222</sender>
  <operator>swisscom</operator>
  <service>TRANSPOSE</service>
  <language>en</language>
  <command>UPDATE</command>
  <receptionDate>2012-07-27 15:58:21.581</receptionDate>
  <requestUid>sms9676215</requestUid>
```

```
    <parameters>
    </parameters>
  </Notification>
```

Reply:

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

In addition to the parameters described above, all the parameters returned by the command SERVICEINFO (except the parameters: service and language already present) are included in the notification, in the <parameters> element.

4.3. Sent Message Notifications

Sent message notifications are sent when the smsBox® platform sends an SMS or MMS message to a mobile user, or when the status of a sent message is updated from operator's information (for example, when the operator informs smsBox® that a message has been received on the device of the mobile user).

Sent message notifications are independant from command notifications because zero to many sent messages can be sent after a command processing, credit checking can be performed to verify that we really want to send the messages to mobile users (spam preventing measures enforced by operators to limit the amount of money or messages received by a mobile user, for example) and different media can be used (an XML command can trigger the sending of an SMS or MMS message, an SMS command can trigger the sending of an MMS message).

Sent message notifications do not transmit different information for each command, as command notifications do, because they closely represent SMS and MMS messages, thus they don't need specific parameters per command.

Generally, the name of the command can be found in the sent message notification, with the following exceptions: the POST command will generate sent message(s) reported as PUSH (one for each subscribed mobile user of the service, if the POST was not silent); the CHAT command, in case of a non-moderated chat, will generate sent message(s) reported as CHATPUSH (one for each subscribed mobile user of the chat); the POSTMMS command will generate MMS sent message(s) reported as PUSHMMS (same applies).

4.3.1. New Message

A sent message notification is sent for each new sent message, with extensive information about the sent message.

4.3.1.1. SMS case

The XML document is made of the root element `DeliveryNotification`, with attribute `eventType="newMessage"`. The elements following are:

Parameter	Description
requestUid	the unique identifier of the command which generated the message; it begins with the name of the media used to trigger the command, and currently can be "sms" (for example, the mobile user sent the SMS "STOP" to 939), "mms" or "xml" (for example, the partner sent an XML API command UNSUBSCRIBEALL to the smsBox® platform)
sentMessageUid	the unique identifier of the sent message within the instance; useful essentially if a single request generated more than one sent message
instance	the instance name - *same as in command notifications*
receiver	the receiver MSISDN - *same as "sender" in command notifications*
operator	the operator - *same as in command notifications*
service	the service - *same as in command notifications*
command	the command name - *same as in existing notifications, except PUSH for POST commands and CHATPUSH for CHAT commands*

cost	the cost of the message, in cents (if MT billing is available)
status	the status of the message; would begin with "sent" when message is being processed by the operator, "delivered" when message has been received on the destination phone (if applicable), "failed" when there was a failure; a pipe character ("|") may follow these statuses with details depending on the operator
type	message type; can be "text", "oplogo", "clilogo", "pictmsg", "ring", "varpict", "usersound", "raw", "wappush_si_sms", "wappush_sl_sms"
message	the full message content
binaryHeaders	the binary headers if any (implies a binary message)
metaData	the meta data if any; these are information not directly related to the message content; for example on UNSUBSCRIBEALL it would contain the list of services the mobile user was subscribed to

<u>Notification:</u>

```
<?xml version="1.0" encoding="UTF-8" ?>
<DeliveryNotification eventType="newMessage">
  <requestUid>sms9676187</requestUid>
  <sentMessageUid>sms29282896</sentMessageUid>
  <instance>pro</instance>
  <receiver>+41761234567</receiver>
  <operator>sunrise</operator>
  <service/>
  <command>UNSUBSCRIBEALL</command>
  <cost>0</cost>
  <status>sent</status>
  <type>text</type>
  <message>You were successfully unsubscribed from all services. See you!</message>
  <binaryHeaders/>
  <metaData/>
</DeliveryNotification>
```

Note: the external application receiving a sent message notification do not need to reply with anything, even if HTTP POST transport was used. Replies are ignored by smsBox®.

4.3.1.2. MMS case

The XML document is made of the root element `DeliveryNotification`, with attribute `eventType="newMessage"`. The elements following are:

Parameter	Description
requestUid	*same as SMS case*
sentMessageUid	*same as SMS case*
instance	*same as SMS case*
receiver	*same as SMS case*
operator	*same as SMS case*
service	*same as SMS case*
command	*same as SMS case* - except PUSHMMS for POSTMMS commands
cost	*same as SMS case*
status	*same as SMS case*
type	message type; always "mms"
subject	the subject of the MMS, if any
contentBytes	the number of bytes of content in the MMS
metaData	*same as SMS case*

Note: the actual content of the MMS is not transmitted.

4.3.2. Update of Status

Another, shorter, sent message notification is sent whenever smsBox® receives an update of the status of a particular sent message from the operator (note: there can be zero to several updates of status per sent message). This can typically be about the successful delivery of the message, or failure of delivery because the customer is not reachable within the validity period of the message.

The XML document is made of the root element `DeliveryNotification`, with attribute `eventType="updateStatus"`. The elements following are `requestUid`, `sentMessageUid`, `instance`, `receiver`, `operator`, `service`, `command` and `status`, with the same meaning as in the "New Message" case.

Notification:

```
<?xml version="1.0" encoding="UTF-8" ?>
<DeliveryNotification eventType="updateStatus">
  <requestUid>sms9676187</requestUid>
  <sentMessageUid>sms29282896</sentMessageUid>
  <instance>pro</instance>
  <receiver>+41761234567</receiver>
  <operator>sunrise</operator>
  <service/>
  <command>UNSUBSCRIBEALL</command>
  <status>delivered</status>
</DeliveryNotification>
```

5. Command XML API (interactions to smsBox®)

5.1. Introduction

The principle of the command XML API is similar to the notification API: the external system uses an HTTP POST request to send a UTF-8 encoded XML document with the `Content-Type` header set to `text/xml` to the smsBox® platform, which will respond in the HTTP response using a Reply XML document containing the required acknowledge or information. For the trace of a complete XML API request down at the HTTP byte level, please refer to *Appendix D*.

It is also possible, for the main commands, to use an HTTP GET request instead (please refer to next subsection for details).

The Command XML API documents consist of a root element enclosing the base request elements and command specific elements. The root element is named `SMSBoxXMLRequest`. The base elements are `username`, `password` and `command`. The command specific elements are subelements of the `parameters` element. Command specific elements vary by type of command and thus each command has its own DTD.

Root Element:

```
  <SMSBoxXMLRequest>
```

Base Elements (example):

```
<username>myuser</username>
<password>mypass</password>
<command>VIEW</command>
```

Command Specific Elements (example):

```
<parameters>
  <msisdn>+41761234567</msisdn>
</parameters>
```

Closing Root Element:

```
</SMSBoxXMLRequest>
```

The `parameters` element may optionally be empty if the command does not require any special parameters. In this case the `parameters` element takes the form of a standard empty XML element: `<parameters/>`.

Please note that the Commands XML API document **must be encoded in UTF-8**. If it appears not to be, it will be immediately rejected with the error `wrongutf8`. However, we cannot detect all cases of wrong encoding, so don't trust the absence of `wrongutf8` if you're not sure whether your document is correctly encoded or not.

There is a limit in the XML API document size. By default it is of 15 MB; the error PARSEERROR with error type `requestoolong` is triggered if that size is exceeded.

The reply XML document consists of the root element `SMSBoxXMLReply`, a `status` element, a `command` element and a `requestUid` element. The `status` element will either be `ok`, or a specific error. The `command` element will contain the attribute `name` and subelements specific to the particular command. As these command specific elements vary by type of command, each command will have its own DTD for their response XML document. The `requestUid` element will contain a unique identifier corresponding to this command request. It is repeated in the command notification (if any) and, more interesting, it is repeated in the sent message(s) notification(s) so that they can be properly linked to the request which generated the sent message(s). Check the appropriate chapter for informations about notifications.

Root Element:

```
<SMSBoxXMLReply>
```

Status Element, ok status:

```
<ok/>
```

Status Element, example error status:

```
<error type="wrongutf8"/>
```

Command Specific Elements (example):

```
<command name="VIEW">
  <msisdn>+41761234567</msisdn>
  <service>FCBNEWS</service>
  <service>FCBTOTOMAT</service>
</command>
```

RequestUid Element (example):

```
<requestUid>xml9677287</requestUid>
```

Closing Root Element:

```
</SMSBoxXMLReply>
```

5.1.1. HTTP GET alternative

Instead of sending an XML document with HTTP POST, it is also possible to use HTTP GET requests (REST-style), for the main commands. As this document details all commands used with the HTTP POST method, to use the HTTP GET method instead, the following changes in the request must be done:

- the `/xml` ending part of the request URL must be changed to `/rest`; the command is specified by using the proper end of request URL, e.g. for example for the command USERINFO the ending part of the URL will be `/user/info` - refer to *Appendix F* to view the command mapping for all available commands
- the username and password must be provided as an HTTP Basic Authentication header (refer to the HTTP documentation, but basically depending on HTTP clients it might be as simple as specifying `username:password@` before the hostname in the URL)
- all the specific parameters are specified as GET parameters (in the query string); they must be URL-encoded with the UTF-8 encoding, except if an `encoding=` GET parameter is specified (for example, use `encoding=ISO-8859-15` to use traditional western latin encoding instead)

Please refer to *Appendix F* for a trace of an HTTP GET request.

5.1.2. Maximum SMS Message Length

If smsBox® is configured to not allow concatenated SMS, or if the special command parameter `maximumSMSAmount` is not specified (or not supported for a given command), then only one SMS will be generated per message; in that case, the message will be cut if it is longer than 160 characters and it contains only western latin characters or smsBox® is configured to disallow international characters, or 70 characters if there is at least one international character.

If `maximumSMSAmount` is specified and smsBox® allows concatenated SMS, then the message will be cut to 160 / 70 characters if `maximumSMSAmount`'s value is 1, else to 153*`maximumSMSAmount` / 65*`maximumSMSAmount` characters.

Warning, if the API command generates concatenated SMS, the operator will typically charge for each SMS independantly. In other words, if a message of 3 concatenated SMS is generated, operator charging will probably be tripled. For any question regarding that issue, please contact the support team for details.

5.2. Common Parameters

Each command typically has specific request parameters, response elements and error types, but it also shares the following list of common parameters:

5.2.1. Base Request Elements

Parameter	Description
username	Your username.
password	Your password. It is the same password as the web administration tool. If someone changes the password on the web, you will have to change your request accordingly!
command	Name of the command.

5.2.2. Response Elements

Element	Description
ok	If the command was executed properly.
error	If the command was not executed properly; contains a `type=` attribute indicating the error type.
command	Name of the command.
requestUid	A unique identifier for this command request/response.

5.2.3. Common Error Types

Error Type	Description
xmlparseerror	The submitted XML document is not valid.
dtdparseerror	The submitted XML document doesn't properly follow the DTD.
wrongutf8	The XML request document is not properly encoded in UTF-8.
userunknown	The specified username is unknown.
wrongpassword	The specified password is incorrect.
xmlapinoaccess	You do not have access to any commands of the XML API.
commandnoaccess	You do not have access to the requested command.
parammissing:<name>	One of the mandatory parameters is missing.
paramnomatch:<name>	One of the parameter values is incorrect (typically a range or data type problem).

Notice: partners should never blindly retry commands on errors, as on wrongly detected or reported errors, it is an open door for spamming end-users (send a large amount of actual SMS to end-users).

5.3. Command Types

5.3.1. ADDCREDIT

The ADDCREDIT command adds the given amount to the account of an end-user. This function is part of the Stored Value Account (SVA) module, where the end-user is billed by debiting his SVA on the platform.

The response to the command indicates the current amount of the SVA after crediting.

5.3.1.1. ADDCREDIT Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (msisdn,amount,operator)>
<!ELEMENT msisdn           (#PCDATA)>
<!ELEMENT amount           (#PCDATA)>
<!ELEMENT operator         (#PCDATA)>
```

5.3.1.2. Request Specific Parameters

Parameter	Description
msisdn	MSISDN owning the SVA to increment. If the MSISDN is not yet in the database, it is created using the provided operator.
amount	Amount expressed in cents (`1000` will increment by 10 base units).
operator	Operator corresponding to the MSISDN. If the MSISDN is already in the database, this information is ignored.

5.3.1.3. ADDCREDIT Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok    EMPTY>
<!ELEMENT error EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command EMPTY>
  <!ATTLIST command name CDATA #FIXED "ADDCREDIT">
<!ELEMENT requestUid (#PCDATA)>
```

5.3.1.3.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
blacklistmember	The specified end-user is blacklisted.
selfblacklistmember	The specified end-user has blacklisted himself.
missingoperator	The operator element is missing.
badoperator	The specified operator is unknown/incorrect.
badphone	The specified end-user's phone number is incorrect.
missingwhitelistmember	A whitelist entry is needed for the specified service, but is missing.
notadult	On an adult-only instance, the specified end-user has not previously certified being an adult.
expiredadult	On an adult-only instance, the specified end-user has previously certified being an adult, but his certification has expired.
invalidamount	The specified amount if not an integer.

5.3.1.4. ADDCREDIT Command Example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>ADDCREDIT</command>
  <parameters>
    <msisdn>+41765435432</msisdn>
    <amount>2000</amount>
    <operator>sunrise</operator>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="ADDCREDIT">
    <credits>2260</credits>
  </command>
  <requestUid>xml9677288</requestUid>
</SMSBoxXMLReply>
```

5.3.2. BILLINGTOKENVALIDATION

The BILLINGTOKENVALIDATION command is used to validate a billing token received after Mobile Internet billing has been performed by smsBox® on behalf of a partner. Refer to the document "smsBox® browsing external integration" for detailed integration process.

If the token is validated, the answer would be a regular `ok`, and contains the two parameters `cost` and `amount`, whose values the partner must verify are equal to the supposed values for the page; if negative, an error of type `invalidtoken` is answered.

On any error, the partner must not send the premium content to the end-user.

5.3.2.1. BILLINGTOKENVALIDATION Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (token)>
<!ELEMENT token            (#PCDATA)>
```

5.3.2.2. Request Specific Parameters

Parameter	Description
token	The token value as received via the `billingToken` GET parameter, after billing process.

5.3.2.3. BILLINGTOKENVALIDATION Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok EMPTY>
<!ELEMENT error EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command ((cost,amount)|EMPTY)>
  <!ATTLIST command name CDATA #FIXED "BILLINGTOKENVALIDATION">
<!ELEMENT cost (#PCDATA)>
<!ELEMENT amount (#PCDATA)>
<!ELEMENT requestUid (#PCDATA)>
```

5.3.2.3.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
invalidtoken	The specified token does not exist.

5.3.2.4. BILLINGTOKENVALIDATION example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>BILLINGTOKENVALIDATION</command>
  <parameters>
    <token>ABCDEFGH</service>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="BILLINGTOKENVALIDATION"/>
    <cost>50</cost>
    <amount>5</amount>
  </command>
  <requestUid>xml9677792</requestUid>
</SMSBoxXMLReply>
```

5.3.3. BLACKLIST

The BLACKLIST command adds an MSISDN to the blacklist for a given service. This will prevent the user from any action on the service.

5.3.3.1. BLACKLIST Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (msisdn,service,reason?)>
<!ELEMENT msisdn           (#PCDATA)>
<!ELEMENT service          (#PCDATA)>
<!ELEMENT reason           (#PCDATA)>
```

5.3.3.2. Specific Parameters

Parameter	Description
msisdn	MSISDN of the user to blacklist. The MSISDN must be already present in the database (the end-user must have had some activity in the past).
service	Name of the service to blacklist the user for (you must have access to that service to be able to blacklist someone).
reason	Reason for blacklisting this user. This information is stored by smsBox® to keep track of blacklists. It is optional.

5.3.3.3. BLACKLIST Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok             EMPTY>
<!ELEMENT error          EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command        EMPTY>
  <!ATTLIST command name CDATA #FIXED "BLACKLIST">
<!ELEMENT requestUid (#PCDATA)>
```

5.3.3.3.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
blacklistmember	The specified end-user is blacklisted.
selfblacklistmember	The specified end-user has blacklisted himself.
unknownmember	The specified end-user is unknown (never interacted with the instance).
missingwhitelistmember	A whitelist entry is needed for the specified service, but is missing.
notadult	On an adult-only instance, the specified end-user has not previously certified being an adult.
expiredadult	On an adult-only instance, the specified end-user has previously certified being an adult, but his certification has expired.

themenomatch	The specified service does not exist.
deactivatedtheme	The specified service is deactivated.
noaccess	You do not have access to the specified service.
alreadyblacklisted	This MSISDN is already blacklisted.

5.3.3.4. BLACKLIST Command Request Example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>BLACKLIST</command>
  <parameters>
    <msisdn>+41761234567</msisdn>
    <service>MEGACHAT</service>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="BLACKLIST"/>
  <requestUid>xml9677289</requestUid>
</SMSBoxXMLReply>
```

5.3.4. BLACKLISTEDUSERS

The BLACKLISTEDUSERS command returns the list of members that are blacklisted for the indicated service.

5.3.4.1. BLACKLISTEDUSERS Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (service)>
<!ELEMENT service          (#PCDATA)>
```

5.3.4.2. Specific Parameters

Parameter	Description
service	Name of the service for which the blacklisted users are requested (you must have access to that service).

5.3.4.3. BLACKLISTEDUSERS Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok             EMPTY>
<!ELEMENT error          EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command        (service,msisdn*)>
  <!ATTLIST command name CDATA #FIXED "BLACKLISTEDUSERS">
<!ELEMENT service        (#PCDATA)>
<!ELEMENT msisdn         (#PCDATA)>
 <!ATTLIST msisdn date CDATA #IMPLIED>
 <!ATTLIST msisdn reason CDATA #IMPLIED>
<!ELEMENT requestUid     (#PCDATA)>
```

5.3.4.4. Specific Parameters

Parameter	Description
msisdn	Phone number blacklisted for this service. The **optional** attribute `date` indicates the date of the blacklist if available, and the **optional** attribute `reason` indicates the reason for the blacklist if available (possibly empty).

5.3.4.4.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
themenomatch	The specified service does not exist.
deactivatedtheme	The specified service is deactivated.
noaccess	You do not have access to the specified service.

5.3.4.5. BLACKLISTEDUSERS Command Request Example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>BLACKLISTEDUSERS</command>
  <parameters>
    <service>MEGACHAT</service>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="BLACKLISTEDUSERS">
    <service>MEGACHAT</service>
    <msisdn>+41761234567</msisdn>
    <msisdn date="2006-04-04 09:53:35"
       reason="Requested by operator>+41761234568</msisdn>
  </command>
  <requestUid>xml9677291</requestUid>
</SMSBoxXMLReply>
```

5.3.5. BROWSINGVALIDATION

The BROWSINGVALIDATION command verifies that a end-user has paid for content not hosted by the smsBox® platform. If positive, the answer would be a regular `ok`; if negative, an error of type `noaccess` is answered (in such a case, the external application should redirect the end-user to the appropriate browsing location on smsBox®, with an HTTP header `cyote-ext-url` containing the page where to redirect the end-user after being invoiced - please check browsing specific documentation for more details).

5.3.5.1. BROWSINGVALIDATION Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (service,msisdn,path)>
<!ELEMENT service          (#PCDATA)>
<!ELEMENT msisdn           (#PCDATA)>
<!ELEMENT path             (#PCDATA)>
```

5.3.5.2. Request Specific Parameters

Parameter	Description
service	Name of the browsing service.
msisdn	End-user for which browsing validation is requested.
path	Page path in the specified browsing `service`; note that the path is **case-sensitive**.

5.3.5.3. BROWSINGVALIDATION Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok EMPTY>
<!ELEMENT error EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command EMPTY>
  <!ATTLIST command name CDATA #FIXED "BROWSINGVALIDATION">
<!ELEMENT requestUid (#PCDATA)>
```

5.3.5.3.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
blacklistmember	The specified end-user is blacklisted.
selfblacklistmember	The specified end-user has blacklisted himself.
unknownmember	The specified end-user is unknown (never interacted with the instance).
missingwhitelistmember	A whitelist entry is needed for the specified service, but is missing.
notadult	On an adult-only instance, the specified end-user has not previously certified being an adult.
expiredadult	On an adult-only instance, the specified end-user has previously certified being an adult, but his certification has expired.
themenomatch	The specified service does not exist.
deactivatedtheme	The specified service is deactivated.
noaccess	You do not have access to the specified service.
pagenomatch	The specified service is not a browsing service.
subpagenomatch	The specified path does not exist in the specified browsing service.
subpagenotactive	The specified page is inactive.
pageexternalintegration	The specified page is reserved to external integration.

5.3.5.4. BROWSINGVALIDATION example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>BROWSINGVALIDATION</command>
  <parameters>
    <service>MOBILE FOOT</service>
    <msisdn>+41761234567</msisdn>
    <path>/news/latest</path>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

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

```
<SMSBoxXMLReply>
  <ok/>
  <command name="BROWSINGVALIDATION"/>
  <requestUid>xml9677292</requestUid>
</SMSBoxXMLReply>
```

5.3.6. CHARGE

The CHARGE command asks to charge a user for some amount, without sending any SMS (when the operator's network allows to do so); mostly useful when integrated to other applications for which smsBox® is responsible for billing only.

5.3.6.1. CHARGE Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (service,msisdn,cost)>
<!ELEMENT service          (#PCDATA)>
<!ELEMENT msisdn           (#PCDATA)>
<!ELEMENT cost             (#PCDATA)>
```

5.3.6.2. Request Specific Parameters

Parameter	Description
service	Name of the service related to the charging.
msisdn	Customer to charge.
cost	Amount to charge for (in cents).

5.3.6.3. CHARGE Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok EMPTY>
<!ELEMENT error EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command EMPTY>
  <!ATTLIST command name CDATA #FIXED "CHARGE">
<!ELEMENT requestUid (#PCDATA)>
```

5.3.6.3.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
themenomatch	The specified service does not exist.
deactivatedtheme	The specified service is deactivated.
noaccess	You do not have access to the specified service.
costnomatch	The specified cost is incorrect or unauthorized.
blacklistmember	The specified end-user is blacklisted.
selfblacklistmember	The specified end-user has blacklisted himself.
unknownmember	The specified end-user is unknown (never interacted with the instance).
missingwhitelistmember	A whitelist entry is needed for the specified service, but is missing.
notadult	On an adult-only instance, the specified end-user has not previously certified being an adult.

expiredadult	On an adult-only instance, the specified end-user has previously certified being an adult, but his certification has expired.
notsubscribed	The specified end-user is not subscribed to the specified service.
failed|direct-billing-not-supported	The operator doesn't support direct billing.

5.3.6.4. CHARGE example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>CHARGE</command>
  <parameters>
    <service>MNC DL</service>
    <msisdn>+41761234567</msisdn>
    <cost>100</cost>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="CHARGE"/>
  <requestUid>xml9677293</requestUid>
</SMSBoxXMLReply>
```

5.3.7. CLOSESESSION

The CLOSESESSION command closes an open session previously opened with the OPENSESSION command.

5.3.7.1. CLOSESESSION Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (msisdn,service)>
<!ELEMENT msisdn           (#PCDATA)>
<!ELEMENT service          (#PCDATA)>
```

5.3.7.2. Request Specific Parameters

Parameter	Description
msisdn	MSISDN for which the session was opened.
service	Service on which the session was opened.

5.3.7.3. CLOSESESSION Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok    EMPTY>
<!ELEMENT error EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command EMPTY>
  <!ATTLIST command name CDATA #FIXED "CLOSESESSION">
<!ELEMENT requestUid (#PCDATA)>
```

5.3.7.3.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
themenomatch	The specified service does not exist.
deactivatedtheme	The specified service is deactivated.
noaccess	You do not have access to the specified service.
blacklistmember	The specified end-user is blacklisted.
selfblacklistmember	The specified end-user has blacklisted himself.
missingoperator	The operator element is missing.
badoperator	The specified operator is unknown/incorrect.
badphone	The specified end-user's phone number is incorrect.
missingwhitelistmember	A whitelist entry is needed for the specified service, but is missing.
notadult	On an adult-only instance, the specified end-user has not previously certified being an adult.
expiredadult	On an adult-only instance, the specified end-user has previously certified being an adult, but his certification has expired.
sessionnomatch	The specified MSISDN is not in a session or in a session related to another service.

5.3.7.4. CLOSESESSION Command Example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>CLOSESESSION</command>
  <parameters>
    <msisdn>+41765435432</msisdn>
    <service>MYSERVICE</service>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="CLOSESESSION"/>
  <requestUid>xml9677288</requestUid>
</SMSBoxXMLReply>
```

5.3.8. COSTS

The COSTS command is used to retrieve the costs associated with a service. Optional parameters `operator` and `action` can filter the output to provide a shorter reply to the sender. The `language` optional parameter can be used to specify the language when searching for costs. If not there, the service default language is used. If more than one default message is available for an action, both are returned in an ascending order by their position.

5.3.8.1. COSTS Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (service,language?,action?,operator?)>
```

```
<!ELEMENT service          (#PCDATA)>
<!ELEMENT language         (#PCDATA)>
<!ELEMENT action           (#PCDATA)>
<!ELEMENT operator         (#PCDATA)>
```

5.3.8.2. Request Specific Parameters

Parameter	Description
service	Mandatory parameter to query the costs of its related actions.
language	Optional language parameter, as ISO 639-1 language code. Must be one of the configured languages.
action	Optional parameter. Must be an existing action.
operator	Optional parameter. Must be an already existing operator.

5.3.8.3. COSTS Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok    EMPTY>
<!ELEMENT error EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command           (language*,cost*)>
  <!ATTLIST command name CDATA #FIXED "COSTS">
<!ELEMENT language          (#PCDATA)>
<!ELEMENT cost              (#PCDATA)>
  <!ATTLIST cost action CDATA>
  <!ATTLIST cost operator CDATA>
<!ELEMENT requestUid (#PCDATA)>
```

5.3.8.3.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
themenomatch	The specified service does not exist.
deactivatedtheme	The specified service is deactivated.
noaccess	You do not have access to the specified service.

5.3.8.4. COSTS Command Example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>COSTS</command>
  <parameters>
    <service>MNC INFO</service>
    <operator>orange</operator>
    <action>subscribe</action>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="COSTS">
    <language>en</language>
    <cost action="subscribe" operator="orange">80</cost>
  </command>
  <requestUid>xml9677294</requestUid>
</SMSBoxXMLReply>
```

5.3.9. CREATE

The CREATE command creates a new service. Depending on the configuration of smsBox®, you may be limited to creating services only under a root name belonging to you.

5.3.9.1. CREATE Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (service,servicePassword,language,owner?,silent?,billingSMS?,
                                days?,hours?,messages?,type?,description?,keywords?,
                                startDate?,endDate?,externalIntegration?,
                                sendCurrentMessageOnSubscription?,requireSubscriptionCode?,
                                askWinnersShippingAddress?,public?)>
<!ELEMENT service          (#PCDATA)>
<!ELEMENT servicePassword  (#PCDATA)>
<!ELEMENT language         (#PCDATA)>
<!ELEMENT days             (#PCDATA)>
<!ELEMENT hours            (#PCDATA)>
<!ELEMENT messages         (#PCDATA)>
<!ELEMENT owner            (#PCDATA)>
<!ELEMENT type             (#PCDATA)>
<!ELEMENT description      (#PCDATA)>
<!ELEMENT keywords         (#PCDATA)>
<!ELEMENT startDate        (#PCDATA)>
<!ELEMENT endDate          (#PCDATA)>
<!ELEMENT silent           EMPTY>
<!ELEMENT billingSMS       EMPTY>
<!ELEMENT externalIntegration (#PCDATA)>
<!ELEMENT askWinnersShippingAddress (#PCDATA)>
<!ELEMENT sendCurrentMessageOnSubscription EMPTY>
<!ELEMENT requireSubscriptionCode EMPTY>
<!ELEMENT public EMPTY>
```

5.3.9.2. Request Specific Parameters

Parameter	Description
service	Name of the service to create, including the root name.
servicePassword	Password to attribute to the service. The password is necessary for content upload or service removal when those commands are sent through SMS.
language	Language of service, as ISO 639-1 language code.
owner	MSISDN of the owner of the service. It has to be already present on the system database. If no owner is specified, the owner of the service will be the user of the CREATE command (username). If the MSISDN is valid, the confirmation SMS will be sent to that number at the normal cost for creating a service if the billingSMS parameter is set, unless the silent parameter is set.
silent	Determines if an SMS has to be sent for confirmation or not; need the billingSMS parameter if wanted.
billingSMS	Confirms that a billing SMS has to be sent for confirmation to the service owner; overriden with the silent parameter.
days & ***hours***	Indicates that a time limited service will be created and specifies the number of days and/or hours for subscriptions validity. The subscriptions to this service will be valid only the specified number of days/hours. After the specified period of time pass, subscribers will be automatically unsubscribed, and will receive a message informing them about this. These two parameters are mutual exclusive with `messages`.
messages	

	Indicates that a message counter limited service will be created, with subscriptions valid for the specified numbers of messages. The subscribers of this service will receive the specified number of messages before being automatically unsubscribed. Subscribers will be informed about the event by receiving a message. This parameter is mutual exclusive with `days`.
type	If present, allows to set the type of the service: standard (= regular), mms, page (= Mobile Internet), chatstandard, chatmoderated, chatkeywordless, election. Access to types of services is granted according to partner's profile. The type "election" needs the aditional parameters ***electionType*** and ***candidate***.
description	Allows to set the description of the service (may be used by end-users in a catalog site).
keywords	Allows to set the indexing keywords of the service (may be used by end-users to search a catalog site); comma-separated simple english words.
startDate	When creating an election, allows to set the start date (if necessary). Must be formatted as "YYYY-MM-DD HH:MM" and minutes must be a multiple of 5.
endDate	When creating an election, allows to set the end date (if necessary). Same formatting.
externalIntegration	When creating a Mobile Internet service, if the instance supports External Integration, allows to specify if the service is to be dedicated to external integration. Set to `true` or `false`. Defaults to `false`.
askWinnersShippingAddress	When creating an Election service, specify whether the winners should be asked to send their shipping address. Set to `true` or `false`. No default (mandatory parameter in case of elections).
sendCurrentMessageOnSubscription	Allows to specify whether the current content of a service must be sent on subscription. Useless for page and election service types.
requireSubscriptionCode	Specify whether the subscription to the service requires a unique code. Valid only for standard services.
public	Public services appear in the services directory and may be part of search results.

5.3.9.3. CREATE Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok    EMPTY>
<!ELEMENT error EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command EMPTY>
  <!ATTLIST command name CDATA #FIXED "CREATE">
<!ELEMENT requestUid (#PCDATA)>
```

5.3.9.3.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
blacklistmember	The specified end-user is blacklisted.
selfblacklistmember	The specified end-user has blacklisted himself.
unknownmember	The specified end-user is unknown (never interacted with the instance).
missingwhitelistmember	A whitelist entry is needed for the specified service, but is missing.
notadult	On an adult-only instance, the specified end-user has not previously certified being an adult.

expiredadult	On an adult-only instance, the specified end-user has previously certified being an adult, but his certification has expired.
rootnomatch	Either there is no root in specified service or the root name does not exist.
privateroot	The root in specified service is private.
forbiddentheme	The specified service name is forbidden.
reservedtheme	The specified service name is reserved.
alreadyowner	You are already the owner of the specified service.
themeexists	The specified service already exists.
themesizeerror	The size of the specified service is invalid.
passsizeerror	The size of the specified password is invalid.

5.3.9.4. CREATE Command Example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>CREATE</command>
  <parameters>
    <service>MNC INFO</service>
    <servicePassword>a8n34y</servicePassword>
    <language>en</language>
    <owner>+41765435432</owner>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <error type="alreadyexists"/>
  <command name="CREATE"/>
  <requestUid>xml9677295</requestUid>
</SMSBoxXMLReply>
```

5.3.10. CREATELOGIN

The CREATELOGIN command allows to create an account on the web administration interface remotely. The newly created login and password are sent by SMS to the specified MSISDN.

5.3.10.1. CREATELOGIN Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (login?,msisdn?,operator?,firstName?,lastName?,email?,comments?,hotline?,web?)>
<!ELEMENT msisdn           (#PCDATA)>
<!ELEMENT operator         (#PCDATA)>
<!ELEMENT login            (#PCDATA)>
<!ELEMENT firstName        (#PCDATA)>
<!ELEMENT lastName         (#PCDATA)>
<!ELEMENT email            (#PCDATA)>
<!ELEMENT comments         (#PCDATA)>
<!ELEMENT hotline          (#PCDATA)>
<!ELEMENT web              (#PCDATA)>
```

5.3.10.2. Request Specific Parameters

Parameter	Description

login	The login to create. If it is omitted, the MSISDN is used as login.
msisdn	MSISDN of the end-user who will be the owner of the account. Mandatory only if the field login is missing. If this field is not empty, the password is sent by SMS to that mobile number.
operator	The operator of this end-user.
firstName	First name of the end-user.
lastName	Last name of the end-user.
email	Email of the end-user.
comments	Comments for this end-user.
hotline	Hotline to call when future users of services managed by the end-user will have hotline enquiries. If it is omitted, the MSISDN is used as hotline.
web	Website of the end-user, can host some online help about the services for example.

5.3.10.3. CREATELOGIN Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok EMPTY>
<!ELEMENT error EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command EMPTY>
  <!ATTLIST command name CDATA #FIXED "CREATELOGIN">
<!ELEMENT requestUid (#PCDATA)>
```

5.3.10.3.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
loginexists	The login already exists.
blacklistmember	The specified end-user is blacklisted.
selfblacklistmember	The specified end-user has blacklisted himself.
missingoperator	The operator element is missing.
badoperator	The specified operator is unknown/incorrect.
badphone	The specified end-user's phone number is incorrect.
missingwhitelistmember	A whitelist entry is needed for the specified service, but is missing.
notadult	On an adult-only instance, the specified end-user has not previously certified being an adult.
expiredadult	On an adult-only instance, the specified end-user has previously certified being an adult, but his certification has expired.
alreadyblacklisted	This MSISDN is already blacklisted.

5.3.10.4. CREATELOGIN example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>CREATELOGIN</command>
  <parameters>
    <msisdn>+41761234567</msisdn>
    <operator>sunrise</operator>
    <firstName>John</firstName>
    <lastName>Doe</lastName>
    <email>john@yahoo.com</email>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="CREATELOGIN"/>
  <requestUid>xml9677296</requestUid>
</SMSBoxXMLReply>
```

5.3.11. CREATEROOT

The CREATEROOT command allows to create a root name in smsBox®.

5.3.11.1. CREATEROOT Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (name,private)>
<!ELEMENT name             (#PCDATA)>
<!ELEMENT private          (#PCDATA)>
```

5.3.11.2. Request Specific Parameters

Parameter	Description
name	The name of the root name to be created.
private	Boolean value indicating if the root name to be created is only accessible for the user that executed the creation.

5.3.11.3. CREATEROOT Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok EMPTY>
<!ELEMENT error EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command EMPTY>
  <!ATTLIST command name CDATA #FIXED "CREATEROOT">
<!ELEMENT requestUid (#PCDATA)>
```

5.3.11.3.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
failed	The creation of the root name failed for any reason.

5.3.11.4. CREATEROOT example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>CREATEROOT</command>
  <parameters>
    <name>MYROOT</name>
    <private>true</private>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="CREATEROOT"/>
  <requestUid>xml9677296</requestUid>
</SMSBoxXMLReply>
```

5.3.12. DELETE

The DELETE command deletes a specified service.

5.3.12.1. DELETE Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (service,owner?,silent?)>
<!ELEMENT service          (#PCDATA)>
<!ELEMENT owner            (#PCDATA)>
<!ELEMENT silent           EMPTY>
```

5.3.12.2. Request Specific Parameters

Parameter	Description
service	Name of the service to be deleted.
owner	MSISDN of the owner of the service. Optional. If provided, the owner will receive a confirmation of the deletion by SMS.
silent	Determines if an SMS has to be sent to the owner to confirm the deletion.

5.3.12.3. DELETE Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok    EMPTY>
<!ELEMENT error EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command EMPTY>
  <!ATTLIST command name CDATA #FIXED "DELETE">
<!ELEMENT requestUid (#PCDATA)>
```

5.3.12.3.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
themenomatch	The specified service does not exist.
deactivatedtheme	The specified service is deactivated.
noaccess	You do not have access to the specified service.
externaluse	The specified service is for external use.

5.3.12.4. DELETE Command Example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
```

```
    <command>DELETE</command>
    <parameters>
      <service>MNC INFO</service>
      <owner>+41765435432</owner>
    </parameters>
  </SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="DELETE"/>
  <requestUid>xml9677296</requestUid>
</SMSBoxXMLReply>
```

5.3.13. DELETELOGIN

The DELETELOGIN command deletes a client account in smsBox®.

5.3.13.1. DELETELOGIN Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (login|userId)>
<!ELEMENT login            (#PCDATA)>
<!ELEMENT userId           (#PCDATA)>
```

5.3.13.2. Request Specific Parameters

Parameter	Description
login	The login to be deleted.
userId	The userId to be deleted (must be set if and only if ***login*** is not set).

5.3.13.3. DELETELOGIN Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok    EMPTY>
<!ELEMENT error EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command EMPTY>
  <!ATTLIST command name CDATA #FIXED "DELETELOGIN">
<!ELEMENT requestUid (#PCDATA)>
```

5.3.13.3.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
loginnomatch	The specified login can not be found in the database.
useridnomatch	The specified userId can not be found in the database.

5.3.13.4. DELETELOGIN Command Example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>DELETELOGIN</command>
  <parameters>
```

```
      <login>tt10069</login>
    </parameters>
  </SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="DELETELOGIN"/>
  <requestUid>xml9677296</requestUid>
</SMSBoxXMLReply>
```

5.3.14. DELETEROOT

The DELETEROOT command deletes a root name in smsBox®.

5.3.14.1. DELETEROOT Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (name)>
<!ELEMENT name             (#PCDATA)>
```

5.3.14.2. Request Specific Parameters

Parameter	Description
name	The name of the root name to be deleted.

5.3.14.3. DELETEROOT Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok    EMPTY>
<!ELEMENT error EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command EMPTY>
  <!ATTLIST command name CDATA #FIXED "DELETEROOT">
<!ELEMENT requestUid (#PCDATA)>
```

5.3.14.3.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
rootnomatch	Either there is no root in specified service or the root name does not exist.
noaccess	The user has no access to the root name.

5.3.14.4. DELETEROOT Command Example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>DELETEROOT</command>
  <parameters>
    <name>MYROOT</name>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="DELETEROOT"/>
  <requestUid>xml9677296</requestUid>
</SMSBoxXMLReply>
```

5.3.15. ELECTION

The ELECTION command allows an external application to control an SMS election/voting. It allows starting and stopping the election. It also gives access to the lottery functions and election results. The username has to have access to that election.

5.3.15.1. ELECTION Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (election,action,magicNumber?,candidate?,
                                 amount?,silent?,startDate?,endDate?,
                                 multiAdditionalVote*)>
<!ELEMENT election         (#PCDATA)>
<!ELEMENT action           (#PCDATA)>
<!ELEMENT magicNumber      (#PCDATA)>
<!ELEMENT candidate        (#PCDATA)>
<!ELEMENT amount           (#PCDATA)>
<!ELEMENT silent           EMPTY>
<!ELEMENT startDate        (#PCDATA)>
<!ELEMENT endDate          (#PCDATA)>
<!ELEMENT multiAdditionalVote EMPTY>
 <!ATTLIST multiAdditionalVote voter CDATA #REQUIRED>
 <!ATTLIST multiAdditionalVote type CDATA #REQUIRED>
 <!ATTLIST multiAdditionalVote candidate CDATA>
```

5.3.15.2. ELECTION Request Specific Parameters

Parameter	Description
action	The action element is required and determines the action that will be taken for this election. Possible actions are: `START`: enables the election. Voting is allowed. `STOP`: disables the election. Voting is not possible and a specific system message informs the user who tries to vote anyway. `CLEAR`: deletes all votes. This action can be executed whatever the state of the election (started, stopped). `RESULTS`: lists the candidates with their scores. `LOTTERY`: selects one from the voters using the `magicNumber` seed. The voter will receive a congratulation message if the parameter `silent` is not set. `LUCKYWINNER`: same as LOTTERY, but select winner among users who have voted for the indicated `candidate`.
magicNumber	Integer used to select one of the voters "randomly". If magicNumber is n, then the nth voter will be selected, using a modulo function; required for LOTTERY and LUCKYWINNER.
candidate	Required for LUCKYWINNER; indicates the candidate to select potential winners from.
amount	Optional. If set, indicates the amount of winners to trigger, else one winner is triggered.

silent	For LOTTERY or LUCKYWINNER, can be specified to prevent the sending of the SMS message to the selected MSISDN.
startDate	For LOTTERY or LUCKYWINNER, allows to specify a date interval for votes if necessary. Must be formatted as "YYYY-MM-DD HH:MM".
endDate	For LOTTERY or LUCKYWINNER, allows to specify a date interval for votes if necessary. Must be formatted as "YYYY-MM-DD HH:MM".
multiAdditionalVote	For LOTTERY or LUCKYWINNER, additional votes to be taken into account; attribute ***voter*** is the voter identifier (can be a name, an address, an MSISDN...), ***type*** is the type of vote (to distinguish with SMS votes; cannot be `SMS` which is reserved), ***candidate*** is the candidate voted for (needed in case of LUCKYWINNER).

5.3.15.3. ELECTION Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok             EMPTY>
<!ELEMENT error          EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command        ((totalVotes?,distinctVoters?,candidate?,
                               winner*)|EMPTY)>
  <!ATTLIST command name CDATA #FIXED "ELECTION">
<!ELEMENT totalVotes     (#PCDATA)>
<!ELEMENT distinctVoters (#PCDATA)>
<!ELEMENT candidate      (name,totalVotes,percentVotes)>
<!ELEMENT name           (#PCDATA)>
<!ELEMENT totalVotes     (#PCDATA)>
<!ELEMENT percentVotes   (#PCDATA)>
<!ELEMENT winner         (#PCDATA)>
<!ELEMENT requestUid (#PCDATA)>
```

5.3.15.4. Reply Specific Parameters

Parameter	Description
totalVotes	Total number of valid votes in this election. Present for action RESULTS.
distinctVoters	Number of distinct voters in this election. Present for action RESULTS.
candidate	Structure containing the results per candidate. Present for action RESULTS.
name	Name of the candidate.
totalVotes	Total number of valid votes for that candidate.
percentVotes	Percentage of total votes for that candidate.
winner	MSISDN of the selected winner among the voters. Present for actions LOTTERY or LUCKYWINNER. If parameter `amount` was specified, there are `amount` numbers of `winner` elements in the reply.

5.3.15.4.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
themenomatch	The specified service does not exist.
deactivatedtheme	The specified service is deactivated.
noaccess	You do not have access to the specified service.
electionnomatch	The specified service is not an election/voting.
candidatenomatch	The specified candidate does not exist.
electionnotstarted	The specified election is not started.
electionover	The specified election is over (finished).

candidatenovotes	The specified candidate has no votes

5.3.15.5. ELECTION Request START Example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>ELECTION</command>
  <parameters>
    <election>FCBVOTE</election>
    <action>START</action>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="ELECTION"/>
  <requestUid>xml9677297</requestUid>
</SMSBoxXMLReply>
```

Actions STOP and CLEAR will have the same reply.

5.3.15.6. ELECTION Request RESULTS Example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>ELECTION</command>
  <parameters>
    <election>FCBVOTE</election>
    <action>RESULTS</action>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="ELECTION">
    <totalVotes>1295</totalVotes>
    <distinctVoters>973</distinctVoters>
    <candidate>
      <name>Hakin</name>
      <totalVotes>733</totalVotes>
      <percentVotes>56.61</percentVotes>
    </candidate>
    <candidate>
      <name>Atuba</name>
      <totalVotes>562</totalVotes>
      <percentVotes>43.39</percentVotes>
    </candidate>
  </command>
  <requestUid>xml9677298</requestUid>
</SMSBoxXMLReply>
```

5.3.15.7. ELECTION Request LUCKYWINNER Example

Request:

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

```
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>ELECTION</command>
  <parameters>
    <election>FCBVOTE</election>
    <action>LUCKYWINNER</action>
    <magicNumber>127</magicNumber>
    <candidate>ATUBA</candidate>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="ELECTION">
    <winner>+41761234567</winner>
  </command>
  <requestUid>xml9677299</requestUid>
</SMSBoxXMLReply>
```

5.3.15.8. ELECTION Request LUCKYWINNER Example, with additional votes

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>ELECTION</command>
  <parameters>
    <election>FCBVOTE</election>
    <action>LUCKYWINNER</action>
    <magicNumber>127</magicNumber>
    <candidate>ATUBA</candidate>
    <multiAdditionalVote voter="Robert Smith" type="Web" candidate="TULSA"/>
    <multiAdditionalVote voter="Roger Waters" type="Web" candidate="ATUBA"/>
    <multiAdditionalVote voter="John Ventimigliano" type="Web" candidate="VENEZIA"/>
  </parameters>
</SMSBoxXMLRequest>
```

5.3.16. EXPECTSHIPPINGADDRESS

The EXPECTSHIPPINGADDRESS command indicates the given phone number has been asked his shipping address and should soon send it to us; it helps properly recognizing shipping addresses received without the normally expected leading keyword.

5.3.16.1. EXPECTSHIPPINGADDRESS Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (msisdn,operator,language)>
<!ELEMENT msisdn           (#PCDATA)>
<!ELEMENT operator         (#PCDATA)>
<!ELEMENT language         (#PCDATA)>
```

5.3.16.2. EXPECTSHIPPINGADDRESS Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok             EMPTY>
<!ELEMENT error          EMPTY>
 <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command        EMPTY>
 <!ATTLIST command name CDATA #FIXED "EXPECTSHIPPINGADDRESS">
<!ELEMENT requestUid     (#PCDATA)>
```

5.3.16.2.1. Possible Error Types

The Common Error Types listed in **Common Parameters**.

5.3.16.3. EXPECTSHIPPINGADDRESS Command Example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
 <username>myuser</username>
 <password>mypass</password>
 <command>EXPECTSHIPPINGADDRESS</command>
 <parameters>
   <msisdn>+41761234567</msisdn>
   <operator>sunrise</operator>
   <language>de</language>
 </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="EXPECTSHIPPINGADDRESS"/>
  <requestUid>xml9677316</requestUid>
</SMSBoxXMLReply>
```

5.3.17. INSTANCEINFO

The INSTANCEINFO command returns various information about the smsBox® instance.

5.3.17.1. INSTANCEINFO Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       EMPTY>
```

5.3.17.2. INSTANCEINFO Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok             EMPTY>
<!ELEMENT error          EMPTY>
 <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command        (instance,shortnumber,timezone,currency,
                              operator+,version)>
 <!ATTLIST command name CDATA #FIXED "INSTANCEINFO">
<!ELEMENT instance       (#PCDATA)>
<!ELEMENT shortnumber      (#PCDATA)>
<!ELEMENT timezone       (#PCDATA)>
<!ELEMENT currency       (#PCDATA)>
<!ELEMENT operator       (#PCDATA)>
<!ELEMENT version        (#PCDATA)>
<!ELEMENT requestUid     (#PCDATA)>
```

5.3.17.3. Reply Specific Parameters

Parameter	Description
instance	The instance name.
shortnumber	The short number the instance is connected to.
timezone	The timezone in which dates are inputed and outputed.
currency	The base currency used by the instance

operator	A supported operator. Notice that there can be multiple such elements if the instance is connected to multiple operators.
version	The version of smsBox®.

5.3.17.3.1. Possible Error Types

The Common Error Types listed in **Common Parameters**.

5.3.17.4. INSTANCEINFO Command Example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
 <username>myuser</username>
 <password>mypass</password>
 <command>INSTANCEINFO</command>
 <parameters/>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="INSTANCEINFO">
    <instance>pro</instance>
    <shortnumber>939</shortnumber>
    <timezone>Europe/Zurich</timezone>
    <currency>CHF</currency>
    <operator>swisscom</operator>
    <operator>sunrise</operator>
    <operator>orange</operator>
    <version>5.2</version>
  </command>
  <requestUid>xml9677316</requestUid>
</SMSBoxXMLReply>
```

5.3.18. LISTSHIPPINGADDRESSES

The LISTSHIPPINGADDRESSES command allows to retrieve the list of accessible shipping addresses.

5.3.18.1. LISTSHIPPINGADDRESSES Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (owner?)>
<!ELEMENT owner            (#PCDATA)>
```

5.3.18.2. LISTSHIPPINGADDRESSES Request Specific Parameters

Parameter	Description
owner	The owner User Id for which the list should be retrieved (only possible to administrators).

5.3.18.3. LISTSHIPPINGADDRESSES Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok             EMPTY>
<!ELEMENT error          EMPTY>
 <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command        (shippingAddress*)>
 <!ATTLIST command name CDATA #FIXED "LISTSHIPPINGADDRESSES">
<!ELEMENT shippingAddress (#PCDATA)>
 <!ATTLIST shippingAddress firstName CDATA #REQUIRED>
 <!ATTLIST shippingAddress lastName CDATA #REQUIRED>
```

```
<!ATTLIST shippingAddress street CDATA #REQUIRED>
<!ATTLIST shippingAddress postcode CDATA #REQUIRED>
<!ATTLIST shippingAddress city CDATA #REQUIRED>
```

5.3.18.4. Reply Specific Parameters

Parameter	Description
shippingAddress	The shipping address of the user, if existing and available. The required attributes indicate the splitted information by part, and the CDATA contains the MSISDN of the end-user.

5.3.18.4.1. Possible Error Types

The Common Error Types listed in **Common Parameters**.

5.3.18.5. LISTSHIPPINGADDRESSES Command Example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
 <username>myuser</username>
 <password>mypass</password>
 <command>LISTSHIPPINGADDRESSES</command>
 <parameters/>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="LISTSHIPPINGADDRESSES">
    <shippingAddress firstName="John" lastName="Doe" street="1 Main Street"
         postcode="12345" city="Anytown">+41761234567</shippingAddress>
    <shippingAddress firstName="Jane" lastName="Doe" street="1 Main Street"
         postcode="12345" city="Anytown">+41769876543</shippingAddress>
  </command>
  <requestUid>xml9692381</requestUid>
</SMSBoxXMLReply>
```

5.3.19. LOGENTRY

The LOGENTRY command allows to arbitrary log an external event in the smsBox® platform. It can be useful for example, when an externally managed content-download succeeded.

5.3.19.1. LOGENTRY Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (service,msisdn,status)>
<!ELEMENT service          (#PCDATA)>
<!ELEMENT msisdn           (#PCDATA)>
<!ELEMENT status           (#PCDATA)>
```

5.3.19.2. Request Specific Parameters

Parameter	Description
service	Name of the service related to the log entry.
msisdn	MSISDN related to the log entry (note: it must already exist in our database).
status	Status for this log entry. Maximum 22 characters.

5.3.19.3. LOGENTRY Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok EMPTY>
<!ELEMENT error EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command EMPTY>
  <!ATTLIST command name CDATA #FIXED "LOGENTRY">
<!ELEMENT requestUid (#PCDATA)>
```

5.3.19.3.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
themenomatch	The specified service does not exist.
deactivatedtheme	The specified service is deactivated.
noaccess	You do not have access to the specified service.
blacklistmember	The specified end-user is blacklisted.
selfblacklistmember	The specified end-user has blacklisted himself.
unknownmember	The specified end-user is unknown (never interacted with the instance).
missingwhitelistmember	A whitelist entry is needed for the specified service, but is missing.
notadult	On an adult-only instance, the specified end-user has not previously certified being an adult.
expiredadult	On an adult-only instance, the specified end-user has previously certified being an adult, but his certification has expired.
notsubscribed	The specified end-user is not subscribed to the specified service.

5.3.19.4. LOGENTRY example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>LOGENTRY</command>
  <parameters>
    <service>MNC PICTURE</service>
    <msisdn>+41761234567</msisdn>
    <status>download:ok</status>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="LOGENTRY"/>
  <requestUid>xml9677300</requestUid>
</SMSBoxXMLReply>
```

5.3.20. MODERATEDCHAT

The MODERATEDCHAT command allows control of a moderated SMS or MMS chat service by an external application. It allows starting and stopping of the chat, as well as retrieval and management of the current list of chat messages, and triggering an end-user participation in an SMS moderated chat.

5.3.20.1. MODERATEDCHAT Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (service,action,(id?|multiId+)?,
                                msisdn?,operator?,text?,silent?)>
<!ELEMENT service          (#PCDATA)>
<!ELEMENT action           (#PCDATA)>
<!ELEMENT id               (#PCDATA)>
<!ELEMENT multiId          (#PCDATA)>
<!ELEMENT msisdn           (#PCDATA)>
<!ELEMENT operator         (#PCDATA)>
<!ELEMENT text             (#PCDATA)>
<!ELEMENT silent           EMPTY>
```

5.3.20.2. Request Specific Parameters

Parameter	Description
service	Name of the chat service.
action	The action element is required and determines what action will be taken for this chat. Possible actions are: `START`: enables the moderated chat. `STOP`: disables the moderated chat. A specific system message informs the user if he tries to chat. `CLEAR`: deletes all pending messages. `RETRIEVE`: reads all the pending messages. `SEND`: sends one or many pending messages to the subscribers of the chat, as specified in the id or multiId elements. `DELETE`: deletes one or many pending messages, as specified in the id or multiId elements. `PARTICIPATE`: trigger end-user participation in the chat.
id	Identifier of the message to SEND or DELETE.
multiId	Same, but for multiple messages are the same time.
msisdn	End-user MSISDN in case of PARTICIPATE action.
operator	End-user operator in case of PARTICIPATE action.
text	Text in case of PARTICIPATE action.
silent	An optional element preventing the sending of the confirmation SMS message in case of PARTICIPATE action.

Important: in case of PARTICIPATE action, depending on sites, smsBox® may need to generate a confirmation request SMS before the actual chat participation is performed (double optin case). That is stated in the XML response by the presence of the empty element `<needdoubleoptin/>` in the `<command>` element.

5.3.20.3. MODERATEDCHAT Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok             EMPTY>
<!ELEMENT error          EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command        (message*|send*|delete*|id?|needdoubleoptin|EMPTY)>
  <!ATTLIST command name CDATA #FIXED "MODERATEDCHAT">
<!ELEMENT message        (id,msisdn,text?,receivedDate,sent)>
<!ELEMENT id             (#PCDATA)>
<!ELEMENT msisdn         (#PCDATA)>
<!ELEMENT text           (#PCDATA)>
<!ELEMENT receivedDate   (#PCDATA)>
```

```
<!ELEMENT sent           (#PCDATA)>
<!ELEMENT send           (#PCDATA)>
  <!ATTLIST send status CDATA #REQUIRED>
<!ELEMENT delete           (#PCDATA)>
  <!ATTLIST delete status CDATA #REQUIRED>
<!ELEMENT requestUid (#PCDATA)>
<!ELEMENT needdoubleoptin EMPTY>
```

5.3.20.4. Reply Specific Parameters, RETRIEVE action

Parameter	Description
message	Holds details about a message in the chat.
id	Identifier of the pending message.
msisdn	MSISDN of the author of the pending message.
text	Text of the submitted message, in case of SMS chat (the name of the chat service has been removed).
receivedDate	Timestamp of the reception of the message, in the timezone of the instance.
sent	Status indicating if the message was sent in the chat or not.

Note: in case of MMS chat, multimedia content of MMS messages is not transmitted.

5.3.20.5. Reply Specific Parameters, SEND action

send	Container for the status of a message. Contains the submitted id, and the outcome as `status` attribute which can be `ok` for success, `messagenotfound` when the id doesn't correspond to a pending message in the chat, or `messagealreadysent` when the id refers to an already sent message.

5.3.20.6. Reply Specific Parameters, DELETE action

delete	Container for the status of a message. Contains the submitted id, and the outcome as `status` attribute which can be `ok` for success, or `messagenotfound` when the id doesn't correspond to a pending message in the chat.

5.3.20.7. Reply Specific Parameters, PARTICIPATE action

id	The message id in case of successful participation.

5.3.20.8. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
themenomatch	The specified service does not exist.
deactivatedtheme	The specified service is deactivated.
noaccess	You do not have access to the specified service.
chatnomatch	The specified service is not a chat.
chatnotactive	The specified chat is not active.
chatnotmoderated	The specified chat is not a moderated chat.
chatnotactive	In case of PARTICIPATE action, the chat must be active.
idnotaninteger	The specified `id` is not an integer.
messagenotfound	No message corresponding to the specified `id`.
messagealreadysent	In case of SEND action, the message corresponding to the specified `id` was already sent.

5.3.20.9. MODERATEDCHAT Request START Example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>MODERATEDCHAT</command>
  <parameters>
    <service>FCBCHAT</service>
    <action>START</action>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="MODERATEDCHAT"/>
  <requestUid>xml9677301</requestUid>
</SMSBoxXMLReply>
```

Actions STOP and CLEAR will have the same reply.

5.3.20.10. MODERATEDCHAT Request SEND Example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>MODERATEDCHAT</command>
  <parameters>
    <service>FCBCHAT</service>
    <action>SEND</action>
    <id>121</id>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="MODERATEDCHAT">
    <send status="ok" message="Hello to all FCBCHAT people!">121</send>
  </command>
  <requestUid>xml9677302</requestUid>
</SMSBoxXMLReply>
```

Note: the pending chat message with id `121` will be sent to all current subscribers of the chat service.

5.3.20.11. MODERATEDCHAT Request DELETE Example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>MODERATEDCHAT</command>
  <parameters>
    <service>FCBCHAT</service>
    <action>DELETE</action>
    <multiId>121</multiId>
    <multiId>122</multiId>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="MODERATEDCHAT">
    <delete status="ok">121</delete>
    <delete status="messagenotfound">122</delete>
  </command>
  <requestUid>xml9677303</requestUid>
</SMSBoxXMLReply>
```

Note: the pending chat messages with id `121` will be deleted (without being sent to anybody). Nothing is done with a message with id `122` as it was not found in the chat.

5.3.20.12. MODERATEDCHAT Request RETRIEVE Example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>MODERATEDCHAT</command>
  <parameters>
    <service>FCBCHAT</service>
    <action>RETRIEVE</action>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="MODERATEDCHAT">
    <message>
      <id>121</id>
      <msisdn>+41761234567</msisdn>
      <text>
        I think that FCB is the greatest fbclub in the world!
        :-) Hey to all my friends!
      </text>
      <receivedDate>2003-03-20 15:41:26</receivedDate>
      <sent>true</sent>
    </message>
    <message>
      <id>122</id>
      <msisdn>+41767654321</msisdn>
      <text>I think FCB sucks! Can't even beat ManU. Loosers!</text>
      <receivedDate>2003-03-20 15:42:03</receivedDate>
      <sent>false</sent>
    </message>
  </command>
  <requestUid>xml9677304</requestUid>
</SMSBoxXMLReply>
```

5.3.20.13. MODERATEDCHAT Request PARTICIPATE Example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
 <username>myuser</username>
 <password>mypass</password>
 <command>MODERATEDCHAT</command>
 <parameters>
   <service>FCBCHAT</service>
   <action>PARTICIPATE</action>
   <msisdn>+41761234567</msisdn>
   <operator>sunrise</operator>
```

```
    <text>FCB best club da world</text>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="MODERATEDCHAT">
    <id>123</id>
  </command>
  <requestUid>xml9918440</requestUid>
</SMSBoxXMLReply>
```

5.3.21. OPENSESSION

The OPENSESSION command alows to open session for a given MSISDN and service. For detailed scenario using sessions, please refer to the specialized document **sessions handling on top of HTTP XML API**.

5.3.21.1. OPENSESSION Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (msisdn,service)>
<!ELEMENT msisdn           (#PCDATA)>
<!ELEMENT service          (#PCDATA)>
```

5.3.21.2. Request Specific Parameters

Parameter	Description
msisdn	MSISDN for which the session should be opened.
service	Service on which the session should be opened.

5.3.21.3. OPENSESSION Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok    EMPTY>
<!ELEMENT error EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command EMPTY>
  <!ATTLIST command name CDATA #FIXED "OPENSESSION">
<!ELEMENT requestUid (#PCDATA)>
```

5.3.21.3.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
themenomatch	The specified service does not exist.
deactivatedtheme	The specified service is deactivated.
noaccess	You do not have access to the specified service.
blacklistmember	The specified end-user is blacklisted.
selfblacklistmember	The specified end-user has blacklisted himself.
missingoperator	The operator element is missing.
badoperator	The specified operator is unknown/incorrect.
badphone	The specified end-user's phone number is incorrect.

missingwhitelistmember	A whitelist entry is needed for the specified service, but is missing.
notadult	On an adult-only instance, the specified end-user has not previously certified being an adult.
expiredadult	On an adult-only instance, the specified end-user has previously certified being an adult, but his certification has expired.
insession	The specified MSISDN is already in a session.

5.3.21.4. OPENSESSION Command Example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>OPENSESSION</command>
  <parameters>
    <msisdn>+41765435432</msisdn>
    <service>MYSERVICE</service>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="OPENSESSION"/>
  <requestUid>xml9677288</requestUid>
</SMSBoxXMLReply>
```

5.3.22. POST

The POST command allows an application to upload new text content to a service, and optionally have that content sent to the current service subscribers.

The service's password is not required to post content as partner service access is used.

Posting to multiple services is possible using `multiService` parameters in place of the `service` parameter. The `silent` parameter specifies that the new content should be uploaded but not sent to the current service subscribers.

5.3.22.1. POST Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (((service,servicePassword?)|multiService+),
                                 text,releaseDate?,silent?,fullLength?,
                                 samecontentOk?,forceUseUcs2?)>
<!ELEMENT service          (#PCDATA)>
<!ELEMENT servicePassword  (#PCDATA)
<!ELEMENT multiService     (#PCDATA)>
<!ELEMENT text             (#PCDATA)>
<!ELEMENT releaseDate      (#PCDATA)>
<!ELEMENT silent           EMPTY>
<!ELEMENT fullLength       EMPTY>
<!ELEMENT samecontentOk    EMPTY>
<!ELEMENT forceUseUcs2     (#PCDATA)>
<!ELEMENT maximumSMSAmount (#PCDATA)>
```

5.3.22.2. Specific Parameters

Parameter	Description

service	Name of the service to upload new content to.
servicePassword	Password of the service - optional: if provided, it will be checked and the content upload will take place only if the password is correct; if it is not provided, the upload will take place if the username has the necessary access rights to the service. Notice that the service password is not accepted for multiService uploads.
multiService	To be used if the content is to be posted to several services at once.
text	Text of the message. Will be cut to the maximum length possible depending on type of characters, value of the `maximumSMSAmount` parameter, and smsBox® configuration.
releaseDate	To be used if the message content needs to be released at a given future date/time. The format is `YYYY-MM-dd HH:mm` and the minutes must be a multiple of 5.
silent	Optional parameter. If present, no distribution of the new content will be executed. This is useful to post a message like "There is no information currently", which users would receive by performing a GET request.
fullLength	Optional parameter. If present, the entire message length is available, instead of the usual case: `SERVICE NAME>message`, where service name and text separator appear. The option will affect only this command, future requests will not be affected.
samecontentOk	Optional parameter. If present, no automatic cancelation of the post is performed if the content is the same as previous content.
forceUseUcs2	Optional parameter. If present, `true` indicates non-GSM characters should go through (but then SMS length is reduced), `false` indicates non-GSM characters should not go through (accents are removed from alphabetical characters, other characters are substituted by spaces). Notice: if absent, behaviour depends on platform configuration.
maximumSMSAmount	The maximum number of SMS to generate for this message, in case concatenated SMS is needed.

Please read the chapter related the maximum SMS length in the Command XML API Introduction.

Important: the text must be encoded in UTF-8, and can not contain any of the 5 reserved XML characters: `&` `<` `>` `'` and `"`. Here are the corresponding replacement entities which must be used instead:

Character	Replacement to use (XML entity)
&	&amp;
<	&lt;
>	&gt;
'	&#39;
"	&quot;

For example, to post a text "`André says: Brazil is champ now & then.`", one may send:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>POST</command>
  <parameters>
    <service>FOOTBALL</service>
    <text>André says: Brazil is champ now &amp; then.</text>
  </parameters>
</SMSBoxXMLRequest>
```

Notice that all other (printable) characters are allowed, and other entities are forbidden. For example, the HTML entity for non breaking space `&nbsp;` **cannot** be used (the XML document will be rejected), you should directly use the non breaking space character itself; same for accented characters such as é, e.g. `&eacute;` **cannot** be used, the character itself should be used directly.

5.3.22.3. POST Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok      EMPTY>
<!ELEMENT error   EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command (service+|EMPTY)>
  <!ATTLIST command name CDATA #FIXED "POST">
<!ELEMENT service (#PCDATA)>
  <!ATTLIST service status CDATA #REQUIRED>
<!ELEMENT requestUid (#PCDATA)>
```

5.3.22.3.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
themenomatch	The specified service does not exist.
deactivatedtheme	The specified service is deactivated.
noaccess	You do not have access to the specified service.
passnomatch	The specified password is incorrect.
samecontent	The specified content is identical to the previous one.
postsuspended	There was too many posts recently. Posting is temporarily suspended for security reason.
invalidreleasedate	The specified release date has not the valid format `YYYY-MM-dd HH:mm`.
pastreleasedate	The specified release date is already passed.
noncompliantreleasedate	The specified release date minute is not a multiple of 5.
uniqueconstraintviolation	A message content is already queued for the given service at the specified date and time.

5.3.22.4. POST Command Request Example 1 (single service)

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>POST</command>
  <parameters>
    <service>FCBTOTOMAT</service>
    <text>Goal for ManU in the first minute!</text>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="POST">
    <service status="ok">FCBTOTOMAT</service>
  </command>
  <requestUid>xml9677305</requestUid>
</SMSBoxXMLReply>
```

5.3.22.5. POST Command Request Example 2 (multiple services, & and > characters)

Request:

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

```
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>POST</command>
  <parameters>
    <multiService>FCBTOTOMAT</multiService>
    <multiService>MNC FOOT</multiService>
    <text>ManU &gt; Arsenal &amp; Chelsea!</text>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <error type="noaccess"/>
  <command name="POST">
    <service status="ok">FCBTOTOMAT</service>
    <service status="noaccess">MNC FOOT</service>
  </command>
  <requestUid>xml9677306</requestUid>
</SMSBoxXMLReply>
```

5.3.23. POSTBINARY

The POSTBINARY command allows an application to upload new binary content to one or more services.

The service's password is not required to post content as partner service access is used.

Posting to multiple services is possible using `multiService` parameters in place of the `service` parameter.

Note: the content is never automatically sent to subscribers of the target service(s). If you want to send binary content to end-users, use the SENDBINARY command.

5.3.23.1. POSTBINARY Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       ((service|multiService+),class,hex)>
<!ELEMENT service          (#PCDATA)>
<!ELEMENT servicePassword  (#PCDATA)>
<!ELEMENT multiService     (#PCDATA)>
<!ELEMENT class            (#PCDATA)>
<!ELEMENT hex              (#PCDATA)>
```

5.3.23.2. Specific Parameters

Parameter	Description
service	Name of the service to upload new content to.
servicePassword	Password of the service - optional: if provided, it will be checked and the content upload will take place only if the password is correct; if it is not provided, the upload will take place if the username has the necessary access rights to the service. Notice that the service password is not accepted for multiService uploads.
multiService	To be used if the content is to be posted to several services at once.
class	Class of binary message. See SENDBINARY command for details.
hex	Hexadecimal encoded binary content (uppercase).

5.3.23.3. POSTBINARY Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok      EMPTY>
<!ELEMENT error   EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
```

```
<!ELEMENT command (service+|EMPTY)>
  <!ATTLIST command name CDATA #FIXED "POSTBINARY">
<!ELEMENT service (#PCDATA)>
  <!ATTLIST service status CDATA #REQUIRED>
<!ELEMENT requestUid (#PCDATA)>
```

5.3.23.3.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
themenomatch	The specified service does not exist.
deactivatedtheme	The specified service is deactivated.
noaccess	You do not have access to the specified service.
passnomatch	The specified password is incorrect.

5.3.23.4. POSTBINARY Command Request Example (EMS Variable Picture)

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>POSTBINARY</command>
  <parameters>
    <service>MNC EMSLOGO</service>
    <class>varpict</class>
    <hex>090E000001000000000000000000020000000000000000000...</hex>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="POSTBINARY">
    <service status="ok">MNC EMSLOGO</service>
  </command>
  <requestUid>xml9677307</requestUid>
</SMSBoxXMLReply>
```

Please refer to *Appendix E* for examples of different classes of binary messages.

5.3.24. POSTWAPPUSH

The Wap Push SMS are deprecated because they are not supported by modern smartphones, thus smsBox® will automatically convert Wap Push SMS to a text SMS containing equivalent text and URL.

POSTWAPPUSH is used to transmit a Wap Push SMS to the subscribers of a service. The Service Indication (SI) will propose a URL to the end-user. Due to the different policies from the operators, this feature is available only on some instances. Please check with the support group.

5.3.24.1. POSTWAPPUSH Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (service,text,url,silent?,forceRealWapPush?)>
<!ELEMENT service          (#PCDATA)>
<!ELEMENT text             (#PCDATA)>
<!ELEMENT url              (#PCDATA)>
<!ELEMENT silent           EMPTY)>
```

```
<!ELEMENT forceRealWapPush EMPTY)>
```

5.3.24.2. Specific Parameters

Parameter	Description
text	This is the text of the message to display on the phone (size is limited, see note below).
url	URL where the content can be retrieved from.
silent	(optional) Only edit Wap Push of service, don't send to subscribers.
forceRealWapPush	(optional) By default, the wap pushes are converted to standard sms. Force sending a real wap push by setting this parameter.

Note: the text plus the URL must not account for more than 100 UTF-8 encoded bytes (e.g. 100 characters if you're not sending accented characters, a little less if that's the case).

5.3.24.3. POSTWAPPUSH Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok      EMPTY>
<!ELEMENT error   EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command EMPTY>
  <!ATTLIST command name CDATA #FIXED "POSTWAPPUSH">
<!ELEMENT requestUid (#PCDATA)>
```

5.3.24.3.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
themenomatch	The specified service does not exist.
deactivatedtheme	The specified service is deactivated.
noaccess	You do not have access to the specified service.
samecontent	The specified content is identical to the previous one.

5.3.24.4. POSTWAPPUSH Command Request Example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>POSTWAPPUSH</command>
  <parameters>
    <service>GAME133</service>
    <text>Super Gazilla Jumper</text>
    <url>http://www.siteexample.com/downloads/game133.jad</url>
  </parameters>
</SMSBoxXMLRequest>
```

Note: total size of text plus URL is 68 bytes, POSTWAPPUSH will be accepted.

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="POSTWAPPUSH"/>
  <requestUid>xml9677308</requestUid>
</SMSBoxXMLReply>
```

5.3.25. REQUESTSUBSCRIPTIONCODE

The REQUESTSUBSCRIPTIONCODE command allocates a unique and temporary code identifier. It's useful for identifying and linking a SMS transaction with a member/account on a different channel (such as web), especially if there is a msisdn "aliasing" on the network of the phone operator. It can also be used for validating the phone number entered in the creation process of a member account on a different channel.

A code request has a very short validity and usually expires in less than 5 minutes. Such codes can only be used in the case of a subscription to a service. All other usage of the generated codes are not permitted.

NB: in case of "aliasing" of the msisdn on the operator network, this function is not secure as there is no way to check if the member who consumed the code is the same as the one who requested it. More generally, if the instance isn't configured with a strict mode of consumption of the subscription codes, the check of the requester and consumer isn't executed (ask our support team for more information).

5.3.25.1. REQUESTSUBSCRIPTIONCODE Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (msisdn,service,codeLength?,sendCode?,operator?)>
<!ELEMENT msisdn           (#PCDATA)>
<!ELEMENT service          (#PCDATA)>
<!ELEMENT codeLength       (#PCDATA)>
<!ELEMENT sendCode         EMPTY>
<!ELEMENT operator         (#PCDATA)>
```

5.3.25.2. Specific Parameters

Parameter	Description
msisdn	The msisdn that requests the code.
service	The service for which the code is requested.
codeLength	Optional: the number of characters of the required code (default value = 4).
sendCode	Optional: if present, the code is automatically sent to the msisdn and the validation is executed on another channel (web for instance) - otherwise the member needs to send the generated code to a given short number for executing the validation/identification process.
operator	Optional: the operator of the msisdn. If not present, smsBox® tries to guess it (only if supported by the instance). Unused if no message is sent.

5.3.25.3. REQUESTSUBSCRIPTIONCODE Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply  ((ok|error),command,requestUid)>
<!ELEMENT ok              EMPTY>
<!ELEMENT error           EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT requestUid      (#PCDATA)>
<!ELEMENT command         (subscriptionCode)>
  <!ATTLIST command name  CDATA #FIXED "REQUESTSUBSCRIPTIONCODE">
<!ELEMENT subscriptionCode            (#PCDATA)>
  <!ATTLIST subscriptionCode msisdn   CDATA #REQUIRED>
```

5.3.25.4. Reply Specific Parameters

Parameter	Description
subscriptionCode	The unique code generated by smsBox® for the specified msisdn.

5.3.25.4.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
themenomatch	The specified service does not exist.
deactivatedtheme	The specified service is deactivated.
noaccess	You do not have access to the specified service.
missingoperator	The `operator` parameter is missing.
badoperator	The specified operator is invalid or unsupported.
badphone	The phone number has a bad syntax (bad prefix, missing digits...).
appnoaccess	You do not have access to the smartphone app related to the phone number (only for smartphone app messaging).
blacklistmember	The specified end-user is blacklisted.
selfblacklistmember	The specified end-user has blacklisted himself.
missingwhitelistmember	A whitelist entry is needed for the specified service, but is missing.
notadult	On an adult-only instance, the specified end-user has not previously certified being an adult.
expiredadult	On an adult-only instance, the specified end-user has previously certified being an adult, but his certification has expired.
responsedisabled	The command is currently disabled for this msisdn was the target of too many requests.

5.3.25.5. RequestSubscriptionCode Command Request Example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>REQUESTSUBSCRIPTIONCODE</command>
  <parameters>
    <msisdn>+41780001122</msisdn>
    <service>FOOT</service>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="REQUESTSUBSCRIPTIONCODE">
    <subscriptionCode msisdn="+41780001122">4DFP</subscriptionCode>
  </command>
  <requestUid>xml12993</requestUid>
</SMSBoxXMLReply>
```

5.3.26. REQUESTINFO

The REQUESTINFO command gives information about a request, and the list of responses it generated (if any). It is ideal to check the status of SMS MT.

5.3.26.1. REQUESTINFO Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (requestUid)>
<!ELEMENT requestUid       (#PCDATA)>
```

5.3.26.2. Specific Parameters

Parameter	Description
requestUid	The unique identifier of the request to check. When checking XML API requests, corresponds to the value of the `<requestUid>` previously received back.

5.3.26.3. REQUESTINFO Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply  ((ok|error),command,requestUid)>
<!ELEMENT ok              EMPTY>
<!ELEMENT error           EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command         (date,status,service,command,sentMessage*,mmsSentMessage*)>
  <!ATTLIST command name CDATA #FIXED "REQUESTINFO">
<!ELEMENT sentMessage     (operator?,msisdn,service,action,status,cost?,date,message)>
<!ELEMENT mmsSentMessage  (operator?,msisdn,service,action,status,cost?,date)>
<!ELEMENT date            (#PCDATA)>
<!ELEMENT status          (#PCDATA)>
<!ELEMENT service         (#PCDATA)>
<!ELEMENT command         (#PCDATA)>
<!ELEMENT operator        (#PCDATA)>
<!ELEMENT msisdn          (#PCDATA)>
<!ELEMENT action          (#PCDATA)>
<!ELEMENT cost            (#PCDATA)>
<!ELEMENT message         (#PCDATA)>
<!ELEMENT requestUid      (#PCDATA)>
```

5.3.26.4. Reply Specific Parameters

Parameter	Description
date	The date the request was received / the sent message was sent.
status	The status of the request / sent message.
service	The service name of the request / sent message.
command	The command corresponding to the request.
operator	The operator of the destination MSISDN of the sent message (present only if multiple operators are active).
msisdn	The destination MSISDN of the sent message.
action	The action corresponding to the sent message.
cost	The cost of the sent message (present only if MT billing is active).
message	The message content of the sent message.

5.3.26.4.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
nosuchrequest	Request not found.
requestnoaccess	You have no access to the selected request.

5.3.26.5. Requestinfo Command Request Example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>REQUESTINFO</command>
  <parameters>
    <requestUid>xml12706</requestUid>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="REQUESTINFO">
    <date>2010-01-15 10:50:31</date>
    <status>ok</status>
    <service>MONITORING</service>
    <command>WEBSEND</command>
    <sentMessage>
      <operator>swisscom</operator>
      <msisdn>+41791234567</msisdn>
      <service>MONITORING</service>
      <action>WEBSEND</action>
      <status>delivered</status>
      <cost>20</cost>
      <date>2010-01-15 10:50:32</date>
      <message>Warning, the monitoring component is unavailable!</message>
    </sentMessage>
  </command>
  <requestUid>xml12993</requestUid>
</SMSBoxXMLReply>
```

5.3.27. SEARCHSERVICES

The SEARCHSERVICES command returns the list of services the API user owns or has access to. The searchString parameter allows the search to be fine tuned. Leaving searchString empty means all accessible services will be returned. The maximum output size is 1,000 services.

5.3.27.1. SEARCHSERVICES Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (searchString)>
<!ELEMENT searchString     (#PCDATA)>
```

5.3.27.2. Specific Parameters

Parameter	Description
searchString	Specifications of the services to search for; SQL LIKE pattern wildcards (%, _) are available.

5.3.27.3. SEARCHSERVICES Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply  ((ok|error),command,requestUid)>
<!ELEMENT ok              EMPTY>
<!ELEMENT error           EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command         (service*|EMPTY)>
  <!ATTLIST command name CDATA #FIXED "SEARCHSERVICES">
<!ELEMENT service         (name,description,keywords,text,
                               language,owner,creationDate,
                               lastModified,subscriberCount,
                               available,public)>
<!ELEMENT name            (#PCDATA)>
<!ELEMENT description     (#PCDATA)>
<!ELEMENT keywords        (#PCDATA)>
<!ELEMENT text            (#PCDATA)>
<!ELEMENT language        (#PCDATA)>
<!ELEMENT owner           (#PCDATA)>
<!ELEMENT creationDate    (#PCDATA)>
<!ELEMENT lastModified    (#PCDATA)>
<!ELEMENT subscriberCount (#PCDATA)>
<!ELEMENT available       (#PCDATA)>
<!ELEMENT public          (#PCDATA)>
```

```
<!ELEMENT requestUid      (#PCDATA)>
```

5.3.27.4. Reply Specific Parameters

Parameter	Description
name	Name of the service.
description	Description of the service.
keywords	Comma-separated list of keywords of the service.
text	Current message in the service.
language	Language definition of the service, as ISO 639-1 language code.
owner	Identifier of the owner of the service. Can be a MSISDN or a userid on the platform (such as 10001).
creationDate	Timestamp of the service creation, in the timezone of the instance. Format is `YYYY-MM-DD HH:MM:SS`.
lastModified	Timestamp of the last content modification (last upload), in the timezone of the instance. Format is `YYYY-MM-DD HH:MM:SS`.
subscriberCount	Number of subscribers at the moment of the request.
available	Whether the service is available or not.
public	Whether the service is public or not.

5.3.27.4.1. Possible Error Types

The Common Error Types listed in **Common Parameters**.

5.3.27.5. SearchServices Command Request Example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>SEARCHSERVICES</command>
  <parameters>
    <searchString>ROM H%</searchString>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="SEARCHSERVICES">
    <service>
      <name>ROM HOCK</name>
      <text>LNA Play-offs: Berne - Servette 1-0</text>
      <language>fr</language>
      <owner>10999</owner>
      <creationDate>2001-11-19 10:11:05</creationDate>
      <lastModified>2004-03-04 22:47:38</lastModified>
      <subscriberCount>252</subscriberCount>
    </service>
    <service>
      <name>ROM HUM</name>
      <text>Quel est le comble du malheur? Se jeter a
        l'eau pour noyer son chagrin!</text>
      <language>fr</language>
      <owner>10999</owner>
      <creationDate>2001-10-12 09:55:43</creationDate>
      <lastModified>2004-03-04 12:30:01</lastModified>
      <subscriberCount>155</subscriberCount>
    </service>
  </command>
  <requestUid>xml9677309</requestUid>
</SMSBoxXMLReply>
```

5.3.28. SEND

The SEND command allows an application to spontaneously send text messages to individual MSISDNs. Messages can be sent only to MSISDNs that are currently subscribed to one of your managed services.

The message can be sent to multiple receivers in only one request by using the `multiReceiver` element instead of `receiver`, but to maximum 1,000 receivers in a single command.

When allowed by smsBox® (normally, only allowed for non-premium instances), it is possible to generate messages longer than a single/normal SMS (this is called "concatenated SMS"); to activate that feature, the `maximumSMSAmount` parameter must be specified (see details below).

In the reply XML document, the status element will be `ok` if the message could be sent to **all** receivers. The status element will be error if **one or more** of the receivers were not validated. The receiver element in the reply XML document contains a status attribute (with values: `ok` or `error` type). The status of the last invalid receiver will be the error type contained in the `type` attribute of the `error` status element.

5.3.28.1. SEND Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       ((receiver|multiReceiver+),service,text,
                                 cost?,maximumSMSAmount?,forceUseUcs2?,
                                 test?)>
<!ELEMENT receiver         (#PCDATA)>
<!ELEMENT multiReceiver    (#PCDATA)>
<!ELEMENT service          (#PCDATA)>
<!ELEMENT text             (#PCDATA)>
<!ELEMENT cost             (#PCDATA)>
<!ELEMENT maximumSMSAmount (#PCDATA)>
<!ELEMENT forceUseUcs2     (#PCDATA)>
<!ELEMENT test             EMPTY>
```

5.3.28.2. Specific Parameters

Parameter	Description
receiver	MSISDN of the destination user, in international format (+417xyyyzzzz, +336xxyyzztt, etc).
multiReceiver	//Same as receiver, but to be used for sending to a list of users. There can be maximum 1,000 receivers in a single command. //
service	Name of the service in which context the message is sent. This is for logging and statistics purposes, as well as to determine if the user is still subscribed to this service.
text	Text of the message. Will be cut to the maximum length possible depending on type of characters, value of the `maximumSMSAmount` parameter, and smsBox® configuration.
cost	Price to charge for the message in cents, if MT billing is available. The cost needs to be authorized for your account.
maximumSMSAmount	The maximum number of SMS to generate for this message, in case concatenated SMS is needed.
forceUseUcs2	Optional parameter. If present, `true` indicates non-GSM characters should go through (but then SMS length is reduced), `false` indicates non-GSM characters should not go through (accents are removed from alphabetical characters, other characters are substituted by spaces). Notice: if absent, behaviour depends on platform configuration.
test	Flag to set when the actual sending process should not occur, instead a feedback about the actual SMS generated will be returned.

Please read the chapter related to maximum SMS length in the Command XML API Introduction.

5.3.28.3. SEND Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok        EMPTY>
<!ELEMENT error     EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command  (receiver+|EMPTY)>
  <!ATTLIST command name CDATA #FIXED "SEND">
<!ELEMENT receiver (#PCDATA)>
  <!ATTLIST receiver status CDATA #REQUIRED>
<!ELEMENT requestUid (#PCDATA)>
```

Additionally, the `receiver` element contains the following elements when the `test` flag is set and the phone number is valid:

- `message`: the actual message, potentially modified, for example to remove unsupported accents if UCS2 is not activated
- `coding`: the message coding, `0` for GSM 7-bit alphabet or `2` for UCS2
- `operator`: the operator
- `concatenatedSequence`: if the SMS is part of a concatenated SMS, the sequence number
- `concatenatedTotal`: if the SMS is part of a concatenated SMS, the total number of concatenated SMS

5.3.28.3.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
themenomatch	The specified service does not exist.
deactivatedtheme	The specified service is deactivated.
noaccess	You do not have access to the specified service.
costnomatch	The specified cost is incorrect or unauthorized.
blacklistmember	The specified end-user is blacklisted.
selfblacklistmember	The specified end-user has blacklisted himself.
unknownmember	The specified end-user is unknown (never interacted with the instance).
missingwhitelistmember	A whitelist entry is needed for the specified service, but is missing.
notadult	On an adult-only instance, the specified end-user has not previously certified being an adult.
expiredadult	On an adult-only instance, the specified end-user has previously certified being an adult, but his certification has expired.
notsubscribed	The specified end-user is not subscribed to the specified service.
duplicate	The specified end-user is specified more than once.

5.3.28.3.2. Per-receiver Possible Erroneous Statuses

Erroneous Status	Description
blacklistmember	The specified end-user is blacklisted.
selfblacklistmember	The specified end-user has blacklisted himself.
unknownmember	The specified end-user is unknown (never interacted with the instance).
missingwhitelistmember	A whitelist entry is needed for the specified service, but is missing.
notadult	On an adult-only instance, the specified end-user has not previously certified being an adult.

expiredadult	On an adult-only instance, the specified end-user has previously certified being an adult, but his certification has expired.
notsubscribed	The specified end-user is not subscribed to the specified service.
duplicate	The specified end-user is specified more than once.
nogateway	There is no gateway (no sending component) for the operator of the specified end-user.
nomessage	The message for this end-user has been deactivated on the platform.

5.3.28.4. SEND Command Request Example (single receiver, MT billing)

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>SEND</command>
  <parameters>
    <receiver>+41761234567</receiver>
    <service>ULTIMATE</service>
    <text>This is a message from us!</text>
    <cost>20</cost>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="SEND">
    <receiver status="ok">+41761234567</receiver>
  </command>
  <requestUid>xml9677310</requestUid>
</SMSBoxXMLReply>
```

5.3.28.5. SEND Command Request Example (single receiver, large account)

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>SEND</command>
  <parameters>
    <receiver>+41761234567</receiver>
    <service>ULTIMATE</service>
    <text>This is a message from us!</text>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="SEND">
    <receiver status="ok">+41761234567</receiver>
  </command>
  <requestUid>xml9677310</requestUid>
</SMSBoxXMLReply>
```

5.3.28.6. SEND Command Request Example (multiple receivers)

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>SEND</command>
  <parameters>
    <multiReceiver>+41761234567</multiReceiver>
    <multiReceiver>+41798765432</multiReceiver>
    <service>ULTIMATE</service>
    <text>This is a message from us!</text>
    <cost>20</cost>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <error type="notsubscribed"/>
  <command name="SEND">
    <receiver status="ok">+41761234567</receiver>
    <receiver status="notsubscribed">+41798765432</receiver>
  </command>
  <requestUid>xml9677311</requestUid>
</SMSBoxXMLReply>
```

5.3.28.7. SEND Command Request Example (concatenated SMS, large account)

To allow a message of maximum 459 western latin characters e.g. generating maximum 3 SMS (**warning**, generally the charging is tripled if 3 concatenated SMS are generated for a single message!), you may send:

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>SEND</command>
  <parameters>
    <receiver>+41761234567</receiver>
    <service>ULTIMATE</service>
    <text>This is a quite long message (...)</text>
    <maximumSMSAmount>3</maximumSMSAmount>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="SEND">
    <receiver status="ok">+41761234567</receiver>
  </command>
  <requestUid>xml9677310</requestUid>
</SMSBoxXMLReply>
```

5.3.28.8. SEND Command Request Example (test mode)

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>SEND</command>
  <parameters>
```

```
        <receiver>+41761234567</receiver>
        <service>ULTIMATE</service>
        <text>In french, René would say: Joyeux Noël</text>
        <cost>20</cost>
        <test/>
    </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="SEND">
    <receiver message="In french, René would say: Joyeux Noel" operator="sunrise" coding="0">+41761234567<
  </command>
  <requestUid>xml9677310</requestUid>
</SMSBoxXMLReply>
```

Notice "Noël" is modified to "Noel" on this smsBox® instance with UCS2 not activated.

5.3.29. SENDBINARY

SENDBINARY is used to transmit a binary SMS to an end-user. It is compatible to specific types of binary SMSes (see below).

5.3.29.1. SENDBINARY Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (receiver,service,class,hex,cost?)>
<!ELEMENT receiver         (#PCDATA)>
<!ELEMENT service          (#PCDATA)>
<!ELEMENT class            (#PCDATA)>
<!ELEMENT hex              (#PCDATA)>
<!ELEMENT cost             (#PCDATA)>
```

5.3.29.2. Specific Parameters

Parameter	Description
receiver, ***service***, ***cost***	*same as SEND command.*
class	Type of transmitted binary SMS. See below.
hex	Content of the binary SMS, encoded in hexadecimal (uppercase).

Supported classes are:

- `oplogo`: Nokia operator logo
- `clilogo`: Nokia Caller Line Identification logo
- `pictmsg`: Nokia Picture Message
- `ringtone`: Nokia Smart Messaging Ringtone
- `varpict`: EMS Variable Picture
- `usersound`: EMS User Defined Sound
- `raw`: Anything (you send DCS, UDH and UD)

Note: to get details and explanations about data to send for each class, refer to *Appendix E*.

5.3.29.3. SENDBINARY Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok       EMPTY>
<!ELEMENT error    EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command  (receiver+|EMPTY)>
  <!ATTLIST command name CDATA #FIXED "SENDBINARY">
<!ELEMENT receiver (#PCDATA)>
  <!ATTLIST receiver status CDATA #REQUIRED>
<!ELEMENT requestUid (#PCDATA)>
```

5.3.29.3.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
themenomatch	The specified service does not exist.
deactivatedtheme	The specified service is deactivated.
noaccess	You do not have access to the specified service.
costnomatch	The specified cost is incorrect or unauthorized.
blacklistmember	The specified end-user is blacklisted.
selfblacklistmember	The specified end-user has blacklisted himself.
unknownmember	The specified end-user is unknown (never interacted with the instance).
missingwhitelistmember	A whitelist entry is needed for the specified service, but is missing.
notadult	On an adult-only instance, the specified end-user has not previously certified being an adult.
expiredadult	On an adult-only instance, the specified end-user has previously certified being an adult, but his certification has expired.
notsubscribed	The specified end-user is not subscribed to the specified service.

5.3.29.4. SENDBINARY Command Request Example (EMS Variable Picture)

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>SENDBINARY</command>
  <parameters>
    <receiver>+41761234567</receiver>
    <service>MNC EMSLOGO</service>
    <hex>090E000001000000000000000002000000000000000000...</hex>
    <class>varpict</class>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="SENDBINARY">
    <receiver status="ok">+41761234567</receiver>
  </command>
  <requestUid>xml9677312</requestUid>
</SMSBoxXMLReply>
```

5.3.30. SENDSERVICE

The SENDSERVICE command will trigger the sending of an SMS to a specific end-user. The current content of the indicated SMS service will be sent to the end-user at the current price for the corresponding pull request. *It is as if the end-user had requested the SMS service directly with an MO-SMS.*

5.3.30.1. SENDSERVICE Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (receiver,service)>
<!ELEMENT receiver         (#PCDATA)>
<!ELEMENT service          (#PCDATA)>
```

5.3.30.2. Request Specific Parameters

Parameter	Description
receiver	MSISDN of the receiving end-user. As for the SEND command, the user has to be already present in the smsBox® database.
service	Name of the SMS service whose content is to be sent to the end-user.

5.3.30.3. SENDSERVICE Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok EMPTY>
<!ELEMENT error EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command (receiver|EMPTY)>
  <!ATTLIST command name CDATA #FIXED "SENDSERVICE">
<!ELEMENT receiver (#PCDATA)>
  <!ATTLIST receiver status CDATA #REQUIRED>
<!ELEMENT requestUid (#PCDATA)>
```

5.3.30.3.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
themenomatch	The specified service does not exist.
deactivatedtheme	The specified service is deactivated.
noaccess	You do not have access to the specified service.
blacklistmember	The specified end-user is blacklisted.
selfblacklistmember	The specified end-user has blacklisted himself.
unknownmember	The specified end-user is unknown (never interacted with the instance).
missingwhitelistmember	A whitelist entry is needed for the specified service, but is missing.
notadult	On an adult-only instance, the specified end-user has not previously certified being an adult.
expiredadult	On an adult-only instance, the specified end-user has previously certified being an adult, but his certification has expired.

5.3.30.4. SENDSERVICE example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>SENDSERVICE</command>
  <parameters>
    <service>MNC NEWS</service>
    <receiver>+41765435432</receiver>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="SENDSERVICE"/>
  <requestUid>xml9677313</requestUid>
</SMSBoxXMLReply>
```

5.3.31. SERVICEINFO

The SERVICEINFO command allows to retrieve information about a service.

Notice: the reply is slightly different depending on the type of service specified (standard SMS service, MMS service, etc). See below for details.

5.3.31.1. SERVICEINFO Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (service)
<!ELEMENT service          (#PCDATA)>
```

5.3.31.2. Request Specific Parameters

Parameter	Description
service	Name of the service whose information is queried.

5.3.31.3. SERVICEINFO Command Reply DTD (standard SMS service)

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok             EMPTY>
<!ELEMENT error          EMPTY>
 <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command        (replyMessage*)>
 <!ATTLIST command name CDATA #FIXED "SERVICEINFO">
<!ELEMENT replyMessage     (service,password,message?,owner,language,subscribers,
                            sendCurrentMessageOnSubscription,requireSubscriptionCode,
                            description?,keywords?,creationDate,lastModified,limited?,
                            type,messageSpool?,public)>
<!ELEMENT service          (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT type             (#PCDATA)>
<!ELEMENT message          (#PCDATA)>
<!ELEMENT owner            (#PCDATA)>
<!ELEMENT language         (#PCDATA)>
<!ELEMENT subscribers      (#PCDATA)>
<!ELEMENT sendCurrentMessageOnSubscription (#PCDATA)>
<!ELEMENT requireSubscriptionCode          (#PCDATA)>
<!ELEMENT public           (#PCDATA)>
<!ELEMENT description      (#PCDATA)>
<!ELEMENT keyword          (#PCDATA)>
<!ELEMENT creationDate     (#PCDATA)>
<!ELEMENT lastModified     (#PCDATA)>
<!ELEMENT activated        EMPTY>
<!ELEMENT deactivated      EMPTY>
<!ELEMENT limited          (#PCDATA)>
 <!ATTLIST limited pushes days hours CDATA>
<!ELEMENT messageSpool     (message+)>
<!ELEMENT message          (#PCDATA)>
  <!ATTLIST message releaseDate CDATA>
```

5.3.31.4. Reply Parameters

Parameter	Description
service	The name of the service.
password	The password of the service.
message	The message of the service; Optional.
owner	The owner's user id of the service.
language	The language of the service, as ISO 639-1 language code (`en` for english, `de` for german, etc).
subscribers	Subscribers count.
sendCurrentMessageOnSubscription	Specify if message is send on new subscription. Can be `true` or `false`.
requireSubscriptionCode	Specify if the subscription to this service require a code. Can be `true` or `false`.
public	Public services appear in the services directory and may be part of search results. Can be `true` or `false`.
description	Description of the service; optional.
keyword	Search keywords of the service; optional.
creationDate	Creation date of the service.
lastModified	Last modification date of the service.
type	Type of service. Can be: `standard`, `standard-chat`, `moderated-chat`, `keywordless-chat`, `election`, `mms`, `mms-chat-moderated`, `wap-page` (= Mobile Internet, kept for compatibility) and `wappush`.
limited	Subscription limit of the service; Optional.
activated	Present if service is activated.
deactivated	Present if service is deactivated.
pushes	Push amount limited subscription.
days	Time limited subscription.
hours	Time limited subscription.
messageSpool	The list of messages in the sending spool. Optional.
message	The message from the sending spool with its release date.

5.3.31.4.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
themenomatch	The specified service does not exist.
deactivatedtheme	The specified service is deactivated.
noaccess	You do not have access to the specified service.

5.3.31.5. SERVICEINFO example (standard SMS service)

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
```

```
    <command>SERVICEINFO</command>
    <parameters>
      <service>MNC NEWS</service>
    </parameters>
  </SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="SERVICEINFO">
    <service>TEST</service>
    <password>LS4ZAXE</password>
    <message>text of SMS</message>
    <owner>10010</owner>
    <language>de</language>
    <subscribers>5</subscribers>
    <sendCurrentMessageOnSubscription>false</sendCurrentMessageOnSubscription>
    <requireSubscriptionCode>false</requireSubscriptionCode>
    <public>false</public>
    <creationDate>2007-07-23 17:21:46</creationDate>
    <lastModified>2007-08-21 10:20:58</lastModified>
    <activated/>
    <limited pushes="10"/>
    <type>standard</type>
    <messageSpool>
      <message releaseDate="2009-07-31 17:25:00>This is a first message to be released in July</message>
      <message releaseDate="2009-08-23 12:45:00>This is a second message to be released in August</message
    </messageSpool>
  </command>
  <requestUid>xml353</requestUid>
</SMSBoxXMLReply>
```

5.3.31.6. SERVICEINFO Reply DTD extract - SMS chat services

```
<!ELEMENT standard-chat    (bar,type,autoSubscribe,state)>
<!ELEMENT moderated-chat   (bar,type,autoSubscribe,state)>
<!ELEMENT keywordless-chat (bar,type,autoSubscribe,state)>
<!ELEMENT bar              (#PCDATA)>
<!ELEMENT type             (#PCDATA)>
<!ELEMENT autoSubscribe    (#PCDATA)>
<!ELEMENT state            (#PCDATA)>
```

5.3.31.6.1. Reply Specific Parameters

Parameter	Description
bar	Maximum of subscription to the service.
type	Type of the service. Can be: `Standard`, `Moderated` or `Keywordless`.
autosubscribe	Send current message on subscription. Can be `true` or `false`.
state	State of the chat service. Can be: `Active` or `Inactive`.

5.3.31.6.2. Reply Example

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="SERVICEINFO">
    <service>TEST</service>
    <password>LS4ZAXE</password>
    <message>text of SMS</message>
    <owner>10010</owner>
    <language>de</language>
    <subscribers>5</subscribers>
    <sendCurrentMessageOnSubscription>false</sendCurrentMessageOnSubscription>
    <requireSubscriptionCode>false</requireSubscriptionCode>
    <public>false</public>
    <creationDate>2007-07-23 17:21:46</creationDate>
    <lastModified>2007-08-21 10:20:58</lastModified>
```

```
      <limited pushes="10"/>
      <type>keywordless-chat</type>
      <keywordless-chat>
        <bar>30</bar>
        <type>Keywordless</type>
        <autoSubscribe>true</autoSubscribe>
        <state>Active</state>
      </keywordless-chat>
    </command>
    <requestUid>xml353</requestUid>
  </SMSBoxXMLReply>
```

5.3.31.7. SERVICEINFO Reply DTD extract - MMS-chat-moderated service

```
<!ELEMENT mms-chat-moderated   (bar,autoSubscribe,state)>
<!ELEMENT bar                  (#PCDATA)>
<!ELEMENT autoSubscribe        (#PCDATA)>
<!ELEMENT state                (#PCDATA)>
```

5.3.31.7.1. Reply Specific Parameters

Parameter	Description
bar	Maximum of subscription to the service.
autosubscribe	Send current message on subscription. Can be `true` or `false`.
state	State of the chat service. Can be: `Active` or `Inactive`.

5.3.31.7.2. Reply Example

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="SERVICEINFO">
    <service>TEST</service>
    <password>LS4ZAXE</password>
    <message>text of SMS</message>
    <owner>10010</owner>
    <language>de</language>
    <subscribers>5</subscribers>
    <sendCurrentMessageOnSubscription>false</sendCurrentMessageOnSubscription>
    <requireSubscriptionCode>false</requireSubscriptionCode>
    <public>false</public>
    <creationDate>2007-07-23 17:21:46</creationDate>
    <lastModified>2007-08-21 10:20:58</lastModified>
    <limited pushes="10"/>
    <type>mms-chat-moderated</type>
    <mms-chat-moderated>
      <bar>30</bar>
      <autoSubscribe>true</autoSubscribe>
      <state>Active</state>
    </mms-chat-moderated>
  </command>
  <requestUid>xml353</requestUid>
</SMSBoxXMLReply>
```

5.3.31.8. SERVICEINFO Reply DTD extract - MMS service

```
<!ELEMENT mms                  (dataBytes,forwardLock,billingSMS)>
<!ELEMENT dataBytes            (#PCDATA)>
<!ELEMENT forwardLock          (#PCDATA)>
```

5.3.31.8.1. Reply Specific Parameters

Parameter	Description
dataBytes	Size of content (in bytes).

forwardLock	If `true`, indicate that the receiver mobile phone will not allow that any download parts be forwarded to another mobile phone. Can be `true` or `false`.

5.3.31.8.2. Reply Example

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="SERVICEINFO">
    <service>TEST</service>
    <password>LS4ZAXE</password>
    <message>text of SMS</message>
    <owner>10010</owner>
    <language>de</language>
    <subscribers>5</subscribers>
    <sendCurrentMessageOnSubscription>false</sendCurrentMessageOnSubscription>
    <requireSubscriptionCode>false</requireSubscriptionCode>
    <public>false</public>
    <creationDate>2007-07-23 17:21:46</creationDate>
    <lastModified>2007-08-21 10:20:58</lastModified>
    <limited pushes="10"/>
    <type>mms</type>
    <mms>
      <dataBytes>559</dataBytes>
      <forwardLock>false</forwardLock>
      <billingSMS>true</billingSMS>
    </mms>
  </command>
  <requestUid>xml353</requestUid>
</SMSBoxXMLReply>
```

5.3.31.9. SERVICEINFO Reply DTD extract - election service

```
<!ELEMENT election           (type,state,start?,end?,autoWinnerNumer,autoWinnerAmount,candidates)>
<!ELEMENT candidates         (candidate*)>
<!ELEMENT type               (#PCDATA)>
<!ELEMENT state              (#PCDATA)>
<!ELEMENT start              (#PCDATA)>
<!ELEMENT end                (#PCDATA)>
<!ELEMENT autoWinnerNumber   (#PCDATA)>
<!ELEMENT autoWinnerAmount   (#PCDATA)>
<!ELEMENT candidate          (#PCDATA)>
 <!ATTLIST candidate vote CDATA>
```

5.3.31.9.1. Reply Specific Parameters

Parameter	Description
type	Type of the election. Can be: `MultipleVotes`, `OneVote`, `OneChangeableVote` or `OneVoteperCandidate`.
state	State of the election. Can be: `notstarted`, `running` or `finished`.
start	Start date of the election; Optional.
end	End date of the election; Optional.
autoWinnerNumber	Originator of the vote number.
autoWinnerAmount	Amount of winners.
candidate	Name of a candidate.
vote	Number of vote for a candidate.

5.3.31.9.2. Reply Example

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="SERVICEINFO">
    <service>TEST</service>
```

```
      <password>LS4ZAXE</password>
      <message>text of SMS</message>
      <owner>10010</owner>
      <language>de</language>
      <subscribers>5</subscribers>
      <sendCurrentMessageOnSubscription>false</sendCurrentMessageOnSubscription>
      <requireSubscriptionCode>false</requireSubscriptionCode>
      <public>false</public>
      <creationDate>2007-07-23 17:21:46</creationDate>
      <lastModified>2007-08-21 10:20:58</lastModified>
      <limited pushes="10"/>
      <type>election</type>
      <election>
        <type>MultipleVotes</type>
        <state>finished</state>
        <start>2007-08-10 00:00:00</start>
        <end>2007-08-21 00:00:00</end>
        <autoWinnerNumber>125</autoWinnerNumber>
        <autoWinnerAmount>1</autoWinnerAmount>
        <candidates>
          <candidate vote="5">VOTE4</candidate>
          <candidate vote="3">VOTE1</candidate>
          <candidate vote="1">VOTE2</candidate>
          <candidate vote="1">VOTE3</candidate>
        </candidates>
      </election>
    </command>
    <requestUid>xml353</requestUid>
  </SMSBoxXMLReply>
```

5.3.31.10. SERVICEINFO Reply DTD extract, Wap Push service

```
<!ELEMENT wappush              (text,url)>
<!ELEMENT text                 (#PCDATA)>
<!ELEMENT url                  (#PCDATA)>
```

5.3.31.10.1. Reply Specific Parameters

Parameter	Description
`text`	Text of the SMS wappush.
`url`	URL of the SMS wappush.

5.3.31.10.2. Reply Example

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="SERVICEINFO">
    <service>TEST</service>
    <password>LS4ZAXE</password>
    <owner>10010</owner>
    <language>de</language>
    <subscribers>5</subscribers>
    <sendCurrentMessageOnSubscription>false</sendCurrentMessageOnSubscription>
    <requireSubscriptionCode>false</requireSubscriptionCode>
    <public>false</public>
    <creationDate>2007-07-23 17:21:46</creationDate>
    <lastModified>2007-08-21 10:20:58</lastModified>
    <limited pushes="10"/>
    <type>wappush</type>
    <wappush>
      <text>dfdfdf</text>
      <url>http://wqerrt.ch</url>
    </wappush>
  </command>
  <requestUid>xml353</requestUid>
</SMSBoxXMLReply>
```

5.3.31.11. SERVICEINFO Reply DTD extract, Mobile Internet service

```
<!ELEMENT wap-page            (deletable,externalIntegration,pages)>
<!ELEMENT pages               (page*)>
<!ELEMENT page                (deletable,active,foward_lock,databytes,billingType,
                       billingCost,billingId,billingParam?,inputDestination?,type?)>
<!ELEMENT deletable           (#PCDATA)>
<!ELEMENT externalIntegration (#PCDATA)>
<!ELEMENT active              (#PCDATA)>
<!ELEMENT foward_lock         (#PCDATA)>
<!ELEMENT databytes           (#PCDATA)>
<!ELEMENT billingType         (#PCDATA)>
<!ELEMENT billingCost         (#PCDATA)>
<!ELEMENT billingId           (#PCDATA)>
<!ELEMENT billingParam        (#PCDATA)>
<!ELEMENT InputDestination    (#PCDATA)>
<!ELEMENT type                (#PCDATA)>
```

5.3.31.11.1. Reply Specific Parameters

Parameter	Description
deletable	Indicate if the service/page can be removed. Can be `true` or `false`.
externalIntegration	If `true`, no page can be created as the service is fully dedicated. Can be `true` or `false`.
page	Path of a Mobile Internet page.
foward_lock	If `true`, indicate that the receiver mobile phone will not allow that any download parts be forwarded to another mobile phone. Can be `true` or `false`.
active	Indicate if the page is active. Can be `true` or `false`.
dataBytes	Size of content (in bytes).
billingType	Type of billing of the page. Can be `free`, `ntimes`, `nhours`, `referrer` or `subscriptionbased`.
billingCost	Cost of billing of the page.
billingId	Amount associated with the billing. In case of `ntimes`, the number of times; in case of `nhours`, the number of hours; in case of `referrer`, the id of the referred page.
billingParam	Billing specific parameter; only used for `subscriptionbased`, where it contains the name of the service for which a subscription is needed to access this page.
inputDestination	URL/email destination of input elements; only used in case the page contains input (form) elements.
type	Type of the page. Can be: `ordered`, `download` or `forward`.

5.3.31.11.2. Reply Example

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="SERVICEINFO">
    <service>TEST</service>
    <password>LS4ZAXE</password>
    <owner>10010</owner>
    <language>de</language>
    <subscribers>5</subscribers>
    <sendCurrentMessageOnSubscription>false</sendCurrentMessageOnSubscription>
    <requireSubscriptionCode>false</requireSubscriptionCode>
    <public>false</public>
    <creationDate>2007-07-23 17:21:46</creationDate>
    <lastModified>2007-08-21 10:20:58</lastModified>
    <limited pushes="10"/>
    <type>wap-page</type>
    <wap-page>
      <deletable>true</deletable>
      <externalIntegration>false</externalIntegration>
      <pages>
        <page forwardLock="false" billingCost="0" billingId="-1" type="ordered" deletable="false"
```

```
                billingType="free" databytes="51" active="true">/</page>
          <page forwardLock="false" billingCost="0" billingId="0" deletable="true" billingType="free"
                databytes="0" active="true">/page1</page>
          <page forwardLock="false" billingCost="0" billingId="0" deletable="true" billingType="free"
                databytes="0" active="true">/page1/page11</page>
          <page forwardLock="false" billingCost="0" billingId="0" deletable="true" billingType="free"
                databytes="0" active="true">/page2</page>
         </pages>
      </wap-page>
    </command>
    <requestUid>xml353</requestUid>
  </SMSBoxXMLReply>
```

5.3.32. SERVICEUPDATE

The SERVICEUPDATE command allows to update the major data of a service.

5.3.32.1. SERVICEUPDATE Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (service,newService?,owner?,language?,servicePassword?,subscriptionType?,subscriptionTy
<!ELEMENT service          (#PCDATA)>
<!ELEMENT newService       (#PCDATA)>
<!ELEMENT owner            (#PCDATA)>
<!ELEMENT language         (#PCDATA)>
<!ELEMENT servicePassword  (#PCDATA)>
<!ELEMENT subscriptionType (#PCDATA)>
<!ELEMENT subscriptionTypeData (#PCDATA)>
```

5.3.32.2. Request Specific Parameters

Parameter	Description
service	The service for which will set the new data.
newService	The new service name (service renaming).
owner	The new service's owner.
language	The new service language.
servicePassword	The new service password.
subscriptionType	The new subscription type (if the subscription has a limit).
subscriptionTypeData	The limit detail related to the subscription type.

5.3.32.3. SERVICEUPDATE Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok EMPTY>
<!ELEMENT error EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command EMPTY>
  <!ATTLIST command name CDATA #FIXED "SERVICEUPDATE">
<!ELEMENT requestUid (#PCDATA)>
```

5.3.32.3.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
themenomatch	The specified service does not exist.
deactivatedtheme	The specified service is deactivated.

noaccess	You do not have access to the specified service.
namealreadyexists	The new service name that is requested already exists in the database.

5.3.32.4. SERVICEUPDATE example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>SERVICEUPDATE</command>
  <parameters>
    <service>TARZAN</service>
    <newService>JANE</newService>
    <owner>10009</owner>
    <language>en</language>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="SERVICEUPDATE"/>
  <requestUid>xml9677314</requestUid>
</SMSBoxXMLReply>
```

5.3.33. SERVICEREPLYMESSAGE

The SERVICEREPLYMESSAGE command allows to view, create and delete service reply messages; additionally, it allows to list current reply messages.

5.3.33.1. SERVICEREPLYMESSAGE Command Request DTD, VIEW task

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (task,action?,service,language?,operator?)
<!ELEMENT task             (#PCDATA)>
<!ELEMENT action           (#PCDATA)>
<!ELEMENT service          (#PCDATA)>
<!ELEMENT language         (#PCDATA)>
<!ELEMENT operator         (#PCDATA)>
```

5.3.33.2. Command Specific Parameters, VIEW task

Parameter	Description
task	The task to perform; must be VIEW.
action	The action of the reply message; Optional.
service	The service of the reply message.
language	The language of the reply message, as ISO 639-1 language code (optional).
operator	The operator of the reply message; Optional.

5.3.33.3. SERVICEREPLYMESSAGE Command Reply DTD, VIEW task

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok              EMPTY>
<!ELEMENT error           EMPTY>
 <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command         (replyMessage*)>
 <!ATTLIST command name CDATA #FIXED "SERVICEREPLYMESSAGE">
```

```
<!ELEMENT replyMessage     (action,service,language,format,cost,validity,position,operator)>
<!ELEMENT requestUid       (#PCDATA)>
<!ELEMENT action           (#PCDATA)>
<!ELEMENT service          (#PCDATA)>
<!ELEMENT language         (#PCDATA)>
<!ELEMENT format           (#PCDATA)>
<!ELEMENT cost             (#PCDATA)>
<!ELEMENT validity         (#PCDATA)>
<!ELEMENT position         (#PCDATA)>
<!ELEMENT operator         (#PCDATA)>
```

5.3.33.4. Reply Specific Parameters, VIEW task

Parameter	Description
action	The action of the reply message.
service	The service of the reply message.
language	The language of the reply message, as ISO 639-1 language code.
format	The format of the reply message.
cost	The cost of the reply message, in cents.
validity]	The validity of the reply message, in minutes.
position	The position of the reply message (integer starting at 1).
operator	The operator of the reply message.

5.3.33.4.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
themenomatch	The specified service does not exist.
deactivatedtheme	The specified service is deactivated.
noaccess	You do not have access to the specified service.

5.3.33.5. SERVICEREPLYMESSAGE Command Example, VIEW task

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
 <username>myuser</username>
 <password>mypass</password>
 <command>SERVICEREPLYMESSAGE</command>
 <parameters>
   <task>VIEW</task>
   <service>MNCNEWS</service>
 </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
 <ok/>
 <command name="SERVICEREPLYMESSAGE">
   <replyMessage>
     <action>get</action>
     <service>MNCNEWS</service>
     <language>fr</language>
     <format>%t&gt;%m</format>
     <cost>20</cost>
     <validity>360</validity>
     <position>1</position>
     <operator>swisscom</operator>
```

```
            </replyMessage>
            <replyMessage>
              <action>get</action>
              <service>MNCNEWS</service>
              <language>fr</language>
              <format>%t&gt;%m</format>
              <cost>20</cost>
              <validity>360</validity>
              <position>1</position>
              <operator>sunrise</operator>
            </replyMessage>
          </command>
          <requestUid>xml9677316</requestUid>
        </SMSBoxXMLReply>
```

5.3.33.6. SERVICEREPLYMESSAGE Command Request DTD, CREATE task

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (task,action,service,language,format,cost,validity,position,operator?)
<!ELEMENT task             (#PCDATA)>
<!ELEMENT action           (#PCDATA)>
<!ELEMENT service          (#PCDATA)>
<!ELEMENT language         (#PCDATA)>
<!ELEMENT format           (#PCDATA)>
<!ELEMENT cost             (#PCDATA)>
<!ELEMENT validity         (#PCDATA)>
<!ELEMENT position         (#PCDATA)>
<!ELEMENT operator         (#PCDATA)>
```

5.3.33.7. Command Specific Parameters, CREATE task

Parameter	Description
task	The task to perform; must be CREATE.
action	The action of the reply message.
service	The service of the reply message.
language	The language of the reply message, as ISO 639-1 language code.
format	The format of the reply message
cost	The cost of the reply message, in cents.
validity	The validity of the reply message, in minutes.
position	The position of the reply message (integer starting at 1).
operator	The operator of the reply message; Optional.

5.3.33.8. SERVICEREPLYMESSAGE Command Reply DTD, CREATE task

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok             EMPTY>
<!ELEMENT error          EMPTY>
 <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command        (created)>
 <!ATTLIST command name CDATA #FIXED "SERVICEREPLYMESSAGE">
<!ELEMENT created        (#PCDATA)>
```

5.3.33.8.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
themenomatch	The specified service does not exist.

deactivatedtheme	The specified service is deactivated.
noaccess	You do not have access to the specified service.
exists	A service reply message already exists by the same parameters.

5.3.33.9. SERVICEREPLYMESSAGE Command Example, CREATE task

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
 <username>myuser</username>
 <password>mypass</password>
 <command>SERVICEREPLYMESSAGE</command>
 <parameters>
   <task>CREATE</task>
   <action>get</action>
   <service>MNCNEWS</service>
   <language>fr</language>
   <format>%t&gt;%m</format>
   <cost>20</cost>
   <validity>360</validity>
   <position>1</position>
 </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
 <ok/>
 <command name="SERVICEREPLYMESSAGE">
   <created>4</created>
 </command>
 <requestUid>xml9677316</requestUid>
</SMSBoxXMLReply>
```

5.3.33.10. SERVICEREPLYMESSAGE Command Request DTD, DELETE task

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (task,action,service,language,operator)
<!ELEMENT task             (#PCDATA)>
<!ELEMENT action           (#PCDATA)>
<!ELEMENT service          (#PCDATA)>
<!ELEMENT language         (#PCDATA)>
<!ELEMENT operator         (#PCDATA)>
```

5.3.33.11. Command Specific Parameters, DELETE task

Parameter	Description
task	The task to perform; must be DELETE.
action	The action of the reply message.
service	The service of the reply message.
language	The language of the reply message, as ISO 639-1 language code.
operator	The operator of the reply message.

5.3.33.12. SERVICEREPLYMESSAGE Command Reply DTD, DELETE task

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok             EMPTY>
<!ELEMENT error          EMPTY>
 <!ATTLIST error type CDATA #REQUIRED>
```

```
<!ELEMENT command        (deleted)>
 <!ATTLIST command name CDATA #FIXED "SERVICEREPLYMESSAGE">
<!ELEMENT deleted        (#PCDATA)>
```

5.3.33.12.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
themenomatch	The specified service does not exist.
deactivatedtheme	The specified service is deactivated.
noaccess	You do not have access to the specified service.
notexists	No such service reply message.

5.3.33.13. SERVICEREPLYMESSAGE Command Example, DELETE task

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
 <username>myuser</username>
 <password>mypass</password>
 <command>SERVICEREPLYMESSAGE</command>
 <parameters>
   <task>DELETE</task>
   <action>get</action>
   <service>MNCNEWS</service>
   <language>fr</language>
   <operator>swisscom</operator>
 </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
 <ok/>
 <command name="SERVICEREPLYMESSAGE">
   <deleted>1</deleted>
 </command>
 <requestUid>xml9677316</requestUid>
</SMSBoxXMLReply>
```

5.3.33.14. SERVICEREPLYMESSAGE Command Request DTD, CURRENT task

The CURRENT task allows to display the list of reply messages that will be used given a service and an action; it displays service reply messages if there are, otherwise client reply messages if there are, otherwise default reply messages.

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (task,action,service)
<!ELEMENT task             (#PCDATA)>
<!ELEMENT action           (#PCDATA)>
<!ELEMENT service          (#PCDATA)>
```

5.3.33.15. Command Specific Parameters, CURRENT task

Parameter	Description
task	The task to perform; must be CURRENT.
action	The action of the reply message.

service	The service of the reply message.

5.3.33.16. SERVICEREPLYMESSAGE Command Reply DTD, CURRENT task

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok             EMPTY>
<!ELEMENT error          EMPTY>
 <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command        (serviceReplyMessage*,clientReplyMessage*,defaultReplyMessage*)>
 <!ATTLIST command name CDATA #FIXED "SERVICEREPLYMESSAGE">
<!ELEMENT *replyMessage    (action,service,language,format,cost,validity,position,operator)>
<!ELEMENT requestUid     (#PCDATA)>
<!ELEMENT action         (#PCDATA)>
<!ELEMENT service        (#PCDATA)>
<!ELEMENT language       (#PCDATA)>
<!ELEMENT format         (#PCDATA)>
<!ELEMENT cost           (#PCDATA)>
<!ELEMENT validity       (#PCDATA)>
<!ELEMENT position       (#PCDATA)>
<!ELEMENT operator       (#PCDATA)>
```

5.3.33.17. Reply Specific Parameters, CURRENT task

Parameter	Description
action	The action of the reply message.
service	The service of the reply message.
language	The language of the reply message, as ISO 639-1 language code.
format	The format of the reply message.
cost	The cost of the reply message, in cents.
validity]	The validity of the reply message, in minutes.
position	The position of the reply message (integer starting at 1).
operator	The operator of the reply message.

5.3.33.17.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
themenomatch	The specified service does not exist.
deactivatedtheme	The specified service is deactivated.
noaccess	You do not have access to the specified service.

5.3.33.18. SERVICEREPLYMESSAGE Command Example, CURRENT task

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
 <username>myuser</username>
 <password>mypass</password>
 <command>SERVICEREPLYMESSAGE</command>
 <parameters>
   <task>CURRENT</task>
   <service>MNCNEWS</service>
   <action>get</action>
 </parameters>
</SMSBoxXMLRequest>
```

Reply:

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

```
<SMSBoxXMLReply>
 <ok/>
 <command name="SERVICEREPLYMESSAGE">
   <serviceReplyMessage>
     <action>get</action>
     <service>MNCNEWS</service>
     <language>fr</language>
     <format>%t&gt;%m</format>
     <cost>20</cost>
     <validity>360</validity>
     <position>1</position>
     <operator>swisscom</operator>
   </serviceReplyMessage>
   <serviceReplyMessage>
     <action>get</action>
     <service>MNCNEWS</service>
     <language>fr</language>
     <format>%t&gt;%m</format>
     <cost>20</cost>
     <validity>360</validity>
     <position>1</position>
     <operator>sunrise</operator>
   </serviceReplyMessage>
 </command>
 <requestUid>xml9677316</requestUid>
</SMSBoxXMLReply>
```

5.3.34. SETSERVICEINFO

The SETSERVICEINFO command allows to set the description and keywords of a service.

5.3.34.1. SETSERVICEINFO Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (service,description,keywords)>
<!ELEMENT service          (#PCDATA)>
<!ELEMENT description      (#PCDATA)>
<!ELEMENT keywords         (#PCDATA)>
```

5.3.34.2. Request Specific Parameters

Parameter	Description
service	The service for which will set the new description/keywords.
description	The new description for this service.
keywords	The new keywords for this service; comma-separated simple english words.

Note: the description are keywords are both accessible by SMS, hence the length of each of these two parameters should be maximum 160 chars if only western latin chars are used. If there is at least one non western latin character in the parameters values, the maximum length will drop to 70 chars. The command will silently trim the params to fit one SMS.

5.3.34.3. SETSERVICEINFO Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok EMPTY>
<!ELEMENT error EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command EMPTY>
  <!ATTLIST command name CDATA #FIXED "SETSERVICEINFO">
<!ELEMENT requestUid (#PCDATA)>
```

5.3.34.3.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
themenomatch	The specified service does not exist.
deactivatedtheme	The specified service is deactivated.
noaccess	You do not have access to the specified service.

5.3.34.4. SETSERVICEINFO example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>SETSERVICEINFO</command>
  <parameters>
    <service>TARZAN</service>
    <description>The jungle service</description>
    <keywords>Lion,Zebra,Gorilla,Elephant</keywords>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="SETSERVICEINFO"/>
  <requestUid>xml9677314</requestUid>
</SMSBoxXMLReply>
```

5.3.35. STATISTICS

The STATISTICS command extracts statistics from the database and sends them back.

Notice: this command may induce large computations on smsBox® side. Hence, the following rules must be adopted:

- partners should never perform several STATISTICS command in parallel
- partners should not abandon waiting for the response too soon (the so called "data timeout" must be configured at least to 5 minutes); response time may increase if platform is busy, or the response may be "toobusy" if the platform is too busy

A good strategy to collect sent messages statuses for a large campaign is:

- a first STATISTICS request about 15 minutes after the end of the campaign, limited by either start/end dates or request_uid list
- a second STATISTICS request 1 hour later, limited to the sent messages with the "sent" status in the first response, using the sent_message_uid parameter
- another STATISTICS request 3 more times over night, limited to the sent messages that still remain in "sent" status

5.3.35.1. STATISTICS Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest     (username,password,command,parameters)>
<!ELEMENT username             (#PCDATA)>
<!ELEMENT password             (#PCDATA)>
<!ELEMENT command              (#PCDATA)>
<!ELEMENT parameters           (type,request_command?,message?,msisdn?,service?,
                                    service_owner_userid?,status?,requester_userid?,
                                    operator?,from?,to?,detailed?,aggregated?,
                                    request_uid?,sent_message_uid?,offset?,
                                    offset_requests?,offset_sent_messages?)>
```

```
<!ELEMENT type                 (#PCDATA)>
<!ELEMENT request_command      (#PCDATA)>
<!ELEMENT message              (#PCDATA)>
<!ELEMENT msisdn               (#PCDATA)>
<!ELEMENT service              (#PCDATA)>
<!ELEMENT service_owner_userid (#PCDATA)>
<!ELEMENT status               (#PCDATA)>
<!ELEMENT requester_userid     (#PCDATA)>
<!ELEMENT operator             (#PCDATA)>
<!ELEMENT from                 (#PCDATA)>
<!ELEMENT to                   (#PCDATA)>
<!ELEMENT detailed             EMPTY>
<!ELEMENT aggregated           EMPTY>
<!ELEMENT request_uid          (#PCDATA)>
<!ELEMENT sent_message_uid     (#PCDATA)>
<!ELEMENT offset               (#PCDATA)>
<!ELEMENT offset_requests      (#PCDATA)>
<!ELEMENT offset_sent_messages (#PCDATA)>
```

5.3.35.2. Request Specific Parameters

Parameter	Description
type	The type of statistics. Can be ***requests*** to request SMS/API requests, ***sent_messages*** to request SMS responses or ***requests_and_sent_messages*** to request both (requests and sent messages).
request_command	Additional filter: filter the results by the smsBox® command (only for `requests`).
response_action	Additional filter: filter the results by the smsBox® action (only for `sent_messages`).
message	Additional filter: only return results that match this message.
msisdn	Additional filter: only return results that match this msisdn.
service	Additional filter: only return results that match this service.
service_owner_userid	Additional filter: only return results that match this service owner userid - irrelevant for SA profile.
status	Additional filter: only return results that match this status.
requester_userid	Additional filter: only return results that match this userid (of the SMS/API requests - only `requests` type) - irrelevant for SA profile.
operator	Additional filter: only return results that match this operator.
from	Additional filter: only return results that are more recent than this date.
to	Additional filter: filter the results that are older that this date.
detailed	Request the detailed statistics (including the complete messages) if this parameter is set.
aggregated	Only compute the number of requests or sent messages that match the filtering critera if this parameter is set.
request_uid	Optional parameter allowing to return only the sent messages generated by the mentioned request unique id(s). Can be a list, comma-separated, of maximum 1'000 elements. Example value with one request only: `xml12776`. Example with a few requests: `xml12776,xml12779,xml12780,xml12781`. Only relevant for type = `sent_messages`.
sent_message_uid	Optional parameter allowing to return only the sent messages with the specified unique id(s). Can be a list, comma-separated, of maximum 1'000 elements. Example value with one request only: `9958`. Example with a few requests: `9958,9959,9962,9965`. Only relevant for type = `sent_messages`.
offset	Optional parameter allowing to return offset data from the database. Notice that maximum number of entries is normally limited, hence subsequent requests with this parameter may be needed to extract all data (only if `requests` or

	`sent_messages` are requested, but not both)
`offset_requests`	Idem as ***`offset`*** but this parameter is applied on the `requests` if both `requests` and `sent_messages` are requested (type `requests_and_sent_messages`)
`offset_sent_messages`	Idem as ***`offset`*** but this parameter is applied on the `sent_messages` if both `requests` and `sent_messages` are requested (type `requests_and_sent_messages`)

5.3.35.3. STATISTICS Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok EMPTY>
<!ELEMENT error EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT requestUid (#PCDATA)>
<!ELEMENT command (request*,sent_message*,requests?,sent_messages?,next_offset?,next_offset_requests?,next_offset_
  <!ATTLIST command name CDATA #FIXED "STATISTICS">
<!ELEMENT request EMPTY>
  <!ATTLIST request operator CDATA>
  <!ATTLIST request msisdn   CDATA>
  <!ATTLIST request requester_userid CDATA>
  <!ATTLIST request service  CDATA>
  <!ATTLIST request service_owner_userid CDATA>
  <!ATTLIST request command  CDATA>
  <!ATTLIST request status   CDATA>
  <!ATTLIST request date     CDATA>
  <!ATTLIST request message  CDATA>
<!ELEMENT sent_message EMPTY>
  <!ATTLIST sent_message uid CDATA>
  <!ATTLIST sent_message request_uid CDATA>
  <!ATTLIST sent_message service_owner_userid CDATA>
  <!ATTLIST sent_message status   CDATA>
  <!ATTLIST sent_message msisdn   CDATA>
  <!ATTLIST sent_message action  CDATA>
  <!ATTLIST sent_message service  CDATA>
  <!ATTLIST sent_message date     CDATA>
  <!ATTLIST sent_message cost     CDATA>
  <!ATTLIST sent_message operator CDATA>
  <!ATTLIST sent_message message  CDATA>
<!ELEMENT next_offset (#PCDATA)>
<!ELEMENT next_offset_requests (#PCDATA)>
<!ELEMENT next_offset_sent_messages (#PCDATA)>
<!ELEMENT requests (#PCDATA)>
<!ELEMENT sent_messages (#PCDATA)>
```

5.3.35.4. Reply Specific Parameters

Parameter	Description
`request`	A request matching the filtering options. The attributes contain the data related to each request.
`sent_message`	A sent message matching the filtering options. The attributes contain the data related to each request.
`requests`	The number of requests that match the filtering options (only if the parameter aggregated is set).
`sent_messages`	The number of sent messages that match the filtering options (only if the parameter aggregated is set).
`status`	The status of sent messages either start with "failed" to indicate a definitive failure, start with "sent" to indicate the message was successfully sent from smsBox® and we expect a definitive status later (if applicable), or is "delivered" to indicate a successful delivery on the handset.
`next_offset`	If the maximum number of entries is reached, not all data will be sent back and this parameter must be used in a subsequent request (in parameter `offset`) to retrieve the remaining (or part of the remaining) of data. Only sent when either the `requests` or the `sent_messages` are requested.

next_offset_requests	Idem as ***next_offset***, but this parameter is applied on `requests` when both `requests` and `sent_messages` are requested (type `requests_and_sent_messages`)
next_offset_sent_messages	Idem as ***next_offset***, but this parameter is applied on `sent_messages` when both `requests` and `sent_messages` are requested (type `requests_and_sent_messages`)

5.3.35.4.1. Possible Error Types

The Common Error Types listed in **Common Parameters** and:

Error Type	Description
themenomatch	The specified service does not exist.
deactivatedtheme	The specified service is deactivated.
noaccess	You do not have access to the specified service.
toobusy	smsBox® is currently too busy to perform statistics. Please retry later (at least 15 minutes).

5.3.35.5. STATISTICS example, small amount of data

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>STATISTICS</command>
  <parameters>
    <type>requests</type>
    <service>TARZAN</service>
    <from>2010-02-01 00:00</from>
    <to>2010-02-28 23:59</to>
    <detailed/>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <requestUid>xml9677314</requestUid>
  <command name="STATISTICS">
    <request operator="sunrise" msisdn="+41780001122" requester_userid="" service="TARZAN"
             service_owner_userid="10009" command="GETINFO" status="ok"
             date="2010-02-05 13:34:32.00123" message="TARZAN hello world!"/>
    <request operator="sunrise" msisdn="+41780001123" requester_userid="" service="TARZAN"
             service_owner_userid="10009" command="GETINFO" status="ok"
             date="2010-02-07 10:32:12.02345" message="TARZAN hello world 2"/>
    <request operator="sunrise" msisdn="+41780001124" requester_userid="" service="TARZAN"
             service_owner_userid="10009" command="GETINFO" status="ok"
             date="2010-02-12 09:01:34.09876" message="TARZAN hello world!"/>
  </command>
</SMSBoxXMLReply>
```

5.3.35.6. STATISTICS example, large amount of data

Notice `next_offset` and `offset` elements.

Request #1:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>STATISTICS</command>
```

```
    <parameters>
      <type>requests</type>
      <from>2010-02-01 00:00</from>
      <to>2010-08-28 23:59</to>
      <detailed/>
    </parameters>
  </SMSBoxXMLRequest>
```

Reply #1:

```
  <?xml version="1.0" encoding="UTF-8" ?>
  <SMSBoxXMLReply>
    <ok/>
    <requestUid>xml9677314</requestUid>
    <command name="STATISTICS">
      <request operator="sunrise" msisdn="+41780001122" requester_userid="" service="TARZAN"
               service_owner_userid="10009" command="GETINFO" status="ok"
               date="2010-02-05 13:34:32.00123" message="TARZAN hello world!"/>
      <request operator="sunrise" msisdn="+41780001123" requester_userid="" service="TARZAN"
               service_owner_userid="10009" command="GETINFO" status="ok"
               date="2010-02-07 10:32:12.02345" message="TARZAN hello world 2"/>
      ...
      <request operator="sunrise" msisdn="+41780001124" requester_userid="" service="TARZAN"
               service_owner_userid="10009" command="GETINFO" status="ok"
               date="2010-02-12 09:01:34.09876" message="TARZAN hello world!"/>
      <next_offset>10000</next_offset>
    </command>
  </SMSBoxXMLReply>
```

Request #2:

```
  <?xml version="1.0" encoding="UTF-8" ?>
  <SMSBoxXMLRequest>
    <username>myuser</username>
    <password>mypass</password>
    <command>STATISTICS</command>
    <parameters>
      <type>requests</type>
      <from>2010-02-01 00:00</from>
      <to>2010-08-28 23:59</to>
      <offset>10000</offset>
      <detailed/>
    </parameters>
  </SMSBoxXMLRequest>
```

Reply #2:

```
  <?xml version="1.0" encoding="UTF-8" ?>
  <SMSBoxXMLReply>
    <ok/>
    <requestUid>xml9677314</requestUid>
    <command name="STATISTICS">
      <request operator="sunrise" msisdn="+41780001122" requester_userid="" service="TARZAN"
               service_owner_userid="10009" command="GETINFO" status="ok"
               date="2010-02-15 13:34:32.00123" message="TARZAN hello world!"/>
      <request operator="sunrise" msisdn="+41780001123" requester_userid="" service="TARZAN"
               service_owner_userid="10009" command="GETINFO" status="ok"
               date="2010-02-17 10:32:12.02345" message="TARZAN hello world 2"/>
      ...
      <request operator="sunrise" msisdn="+41780001124" requester_userid="" service="TARZAN"
               service_owner_userid="10009" command="GETINFO" status="ok"
               date="2010-02-22 09:01:34.09876" message="TARZAN hello world!"/>
      <next_offset>20000</next_offset>
    </command>
  </SMSBoxXMLReply>
```

etc (until `next_offset` is no longer present in response).

Notice: packets of 10000 entries here is an example only, actual value will depend on smsBox® configuration; is typically configured to 10000 or 1000.

5.3.35.7. STATISTICS example, aggregated data

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>STATISTICS</command>
  <parameters>
    <type>requests</type>
    <service>TARZAN</service>
    <from>2010-02-01 00:00</from>
    <to>2010-02-28 23:59</to>
    <aggregated/>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <requestUid>xml9677314</requestUid>
  <command name="STATISTICS">
    <requests>254</requests>
  </command>
</SMSBoxXMLReply>
```

5.3.36. SUBSCRIBE

The SUBSCRIBE command subscribes an end-user to a service. If the end-user MSISDN is not already in the platform database, an error of type `unknownmember` will be returned and the user will not be subscribed. If the end-user is already subscribed to the service, an error of type `insubscription` will be returned.

The `silent` parameter indicates if the system has to send a confirmation SMS message to the end-user informing them that they have been subscribed to the service. If the `silent` parameter exists, no confirmation message will be sent.

Note: some operators force a mandatory opt-in process handled by a subscription server on their side; in such a case, the SUBSCRIBE command merely sends the necessary request to the operator (and no SUBSCRIBE command notification is generated); no real subscription is performed until the operator acknowledges the subscription.

5.3.36.1. SUBSCRIBE Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (msisdn,service,silent?)>
<!ELEMENT msisdn           (#PCDATA)>
<!ELEMENT service          (#PCDATA)>
<!ELEMENT silent           EMPTY>
```

5.3.36.1.1. Specific Parameters

Parameter	Description
msisdn	MSISDN of the user to subscribe. Normally an MSISDN in international format (+417xyyyzzzz, +336xxyyzztt, etc); it can also be an encrypted MSISDN if applicable to you and/or instance (normally a 32 characters hexadecimal series such as 10A74C23F1F5D1A8A84885E2005AE24F).
service	Name of the service to subscribe the user to; you must have access to that service to be able to subscribe someone.
silent	Optional parameter. If present, no confirmation of the subscription will be sent by SMS. Because

	we have to comply with strict opt-in rules in the case of premium services, 'silent' subscribe is only possible from an environment where the user is rigorously identified.

5.3.36.2. SUBSCRIBE Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok             EMPTY>
<!ELEMENT error          EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command        EMPTY>
  <!ATTLIST command name CDATA #FIXED "SUBSCRIBE">
<!ELEMENT requestUid (#PCDATA)>
```

5.3.36.2.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
blacklistmember	The specified end-user is blacklisted.
selfblacklistmember	The specified end-user has blacklisted himself.
unknownmember	The specified end-user is unknown (never interacted with the instance).
missingwhitelistmember	A whitelist entry is needed for the specified service, but is missing.
notadult	On an adult-only instance, the specified end-user has not previously certified being an adult.
expiredadult	On an adult-only instance, the specified end-user has previously certified being an adult, but his certification has expired.
themenomatch	The specified service does not exist.
deactivatedtheme	The specified service is deactivated.
noaccess	You do not have access to the specified service.
blacklistmember	The specified end-user is blacklisted.
selfblacklistmember	The specified end-user has blacklisted himself.
unknownmember	The specified end-user is unknown (never interacted with the instance).
missingwhitelistmember	A whitelist entry is needed for the specified service, but is missing.
notadult	On an adult-only instance, the specified end-user has not previously certified being an adult.
expiredadult	On an adult-only instance, the specified end-user has previously certified being an adult, but his certification has expired.
themenotavailable	The specified service is currently marked as not available.
insubscription	The specified end-user is currently subscribed to the specified service.

5.3.36.3. SUBSCRIBE Command Request Example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>SUBSCRIBE</command>
  <parameters>
    <msisdn>+41761234567</msisdn>
    <service>FCBTOTOMAT</service>
    <silent/>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="SUBSCRIBE"/>
  <requestUid>xml9677315</requestUid>
</SMSBoxXMLReply>
```

5.3.37. SUBSCRIBERS

The SUBSCRIBERS command returns a list of the MSISDNs which are subscribed to a particular service. Logins are limited to their managed services.

5.3.37.1. SUBSCRIBERS Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (service)>
<!ELEMENT service          (#PCDATA)>
```

5.3.37.2. SUBSCRIBERS Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok             EMPTY>
<!ELEMENT error          EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command        ((count,msisdn*)|EMPTY)>
  <!ATTLIST command name CDATA #FIXED "SUBSCRIBERS">
<!ELEMENT count          (#PCDATA)>
<!ELEMENT msisdn         (#PCDATA)>
<!ELEMENT requestUid     (#PCDATA)>
```

5.3.37.2.1. Reply Specific Parameters

Parameter	Description
count	Count of subscribers to the requested service.
msisdn	MSISDN of a subscriber of the service (repeating).

5.3.37.2.2. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
themenomatch	The specified service does not exist.
deactivatedtheme	The specified service is deactivated.
noaccess	You do not have access to the specified service.

5.3.37.3. SUBSCRIBERS Command Example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>SUBSCRIBERS</command>
  <parameters>
    <service>FCBTOTOMAT</service>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="SUBSCRIBERS">
    <count>3</count>
    <msisdn>+41761234567</msisdn>
    <msisdn>+41798765432</msisdn>
    <msisdn>+41789995555</msisdn>
  </command>
  <requestUid>xml9677316</requestUid>
</SMSBoxXMLReply>
```

5.3.38. UNBLACKLIST

The UNBLACKLIST command removes an MSISDN from the blacklist of a given service. It is the opposite of the BLACKLIST command, it uses the same parameters as the BLACKLIST command does.

5.3.38.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
themenomatch	The specified service does not exist.
deactivatedtheme	The specified service is deactivated.
noaccess	You do not have access to the specified service.
notblacklisted	The specified end-user is not blacklisted.

5.3.39. UNSUBSCRIBE

The UNSUBSCRIBE command unsubscribes an MSISDN from a service. Its parameters and functioning is the same as the SUBSCRIBE command, only reversed.

5.3.39.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
unknownmember	The specified end-user is unknown (never interacted with the instance).
missingwhitelistmember	A whitelist entry is needed for the specified service, but is missing.
themenomatch	The specified service does not exist.
noaccess	You do not have access to the specified service.
notsubscribed	The specified end-user is not subscribed to the specified service.

5.3.40. UNSUBSCRIBEALL

The UNSUBSCRIBEALL command removes all subscriptions for the user.

5.3.40.1. UNSUBSCRIBEALL Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (msisdn,silent?)>
<!ELEMENT msisdn           (#PCDATA)>
<!ELEMENT silent           EMPTY>
```

5.3.40.2. Specific Parameters

Parameter	Description
msisdn	MSISDN of the user to unsubscribe from all services. Normally an MSISDN in international format (+417xyyyzzzz, +336xxyyzztt, etc); it can also be an encrypted MSISDN if applicable to you and/or instance (normally a 32 characters hexadecimal series such as 10A74C23F1F5D1A8A84885E2005AE24F).
silent	Optional parameter. If present, no confirmation of the unsubscription will be sent by SMS.

5.3.40.3. UNSUBSCRIBEALL Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok             EMPTY>
<!ELEMENT error          EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command        EMPTY>
  <!ATTLIST command name CDATA #FIXED "UNSUBSCRIBEALL">
<!ELEMENT requestUid     (#PCDATA)>
```

5.3.40.4. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
unknownmember	The specified end-user is unknown (never interacted with the instance).

5.3.40.5. UNSUBSCRIBEALL Request Example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>UNSUBSCRIBEALL</command>
  <parameters>
    <msisdn>+41761234567</msisdn>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="UNSUBSCRIBEALL"/>
  <requestUid>xml9677317</requestUid>
</SMSBoxXMLReply>
```

5.3.41. UPDATEUSERDATA

The UPDATEUSERDATA allows to modify the user data. A user in the smsbox context is somebody with an access on an application managed by the smsbox.

5.3.41.1. UPDATEUSERDATA Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest      (username,password,command,parameters)>
<!ELEMENT username              (#PCDATA)>
<!ELEMENT password              (#PCDATA)>
<!ELEMENT command               (#PCDATA)>
<!ELEMENT parameters            (login,msisdn?,operator?,newPassword?,firstName?,lastName?,email?,
                                    companyName?,web?,hotline?,language?,multiDynamicParameter*)>
<!ELEMENT login                 (#PCDATA)>
<!ELEMENT msisdn                (#PCDATA)>
<!ELEMENT operator              (#PCDATA)>
```

```
  <!ELEMENT newPassword          (#PCDATA)>
  <!ELEMENT firstName            (#PCDATA)>
  <!ELEMENT lastName             (#PCDATA)>
  <!ELEMENT email                (#PCDATA)>
  <!ELEMENT companyName          (#PCDATA)>
  <!ELEMENT web                  (#PCDATA)>
  <!ELEMENT language             (#PCDATA)>
  <!ELEMENT hotline              (#PCDATA)>
  <!ELEMENT multiDynamicParameter (#PCDATA)>
    <!ATTLIST multiDynamicParameter name CDATA #REQUIRED>
    <!ATTLIST multiDynamicParameter type CDATA #REQUIRED "String">
```

5.3.41.2. Request Specific Parameters

Parameter	Description
login	The login name of the user whose data needs to be updated.
msisdn	The MSISDN of the user to request info for.
operator	The operator of the MSISDN number (not necessary for instances with a trusted correlation between the phone number and the operator such as the swiss instances).
newPassword	The password to update, plain text.
firstName	The user first name.
lastName	The user last name.
email	The user email address.
companyName	The user company name.
web	The user web url
hotline	The user hotline address
language	The user preferred language (format ISO 639-1)
multiDynamicParameter	A dynamic parameter associated to the user. The dynamic parameter is identified by its ***name*** and ***type***. Currently only the ***type*** *string*, *integer* or *bytea* are supported. If a parameter is of type *bytea*, it needs to be encoded in base 64. Moreover many dynamic parameters with the same ***name*** are not supported yet.

5.3.41.3. UPDATEUSERDATA Command Reply DTD

```
  <!ELEMENT SMSBoxXMLReply    ((ok|error),command,requestUid)>
  <!ELEMENT ok                EMPTY>
  <!ELEMENT error             EMPTY>
    <!ATTLIST error type CDATA #REQUIRED>
  <!ELEMENT command            EMPTY)>
    <!ATTLIST command name CDATA #FIXED "UPDATEUSERDATA">
  <!ELEMENT requestUid        (#PCDATA)>
```

5.3.41.4. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
novaliduserlogin	The login parameter of the user whose data need an update is not valid.

5.3.41.5. UPDATEUSERDATA Request Example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
```

```
    <command>UPDATEUSERDATA</command>
    <parameters>
      <login>st10010</login>
      <msisdn>+41761234567</msisdn>
      <firstName>Roger</firstName>
      <lastName>Federer</lastName>
      <email>roger.fereder@bluewin.ch</email>
      <companyName>MNC SA</companyName>
      <web>www.federer.ch</web>
      <newPassword>1234</newPassword>
      <multiDynamicParameter name="atpBestranking" type="integer">1</multiDynamicParameter>
      <multiDynamicParameter name="lastGrandChelemWon" type="String">Wimbledon</multiDynamicParameter>
    </parameters>
  </SMSBoxXMLRequest>
```

Reply:

```
  <?xml version="1.0" encoding="UTF-8" ?>
  <SMSBoxXMLReply>
    <ok/>
    <command name="UPDATEUSERINFO"/>
    <requestUid>xml9677318</requestUid>
  </SMSBoxXMLReply>
```

5.3.42. USERINFO

The USERINFO command returns detailed information concerning an MSISDN. Information such as the MSISDN's operator and timestamp of last request, the list of services to which the MSISDN is subscribed, as well as the list of services owned by the MSISDN and list of blacklisted services are returned.

5.3.42.1. USERINFO Command Request DTD

```
  <!ELEMENT SMSBoxXMLRequest    (username,password,command,parameters)>
  <!ELEMENT username            (#PCDATA)>
  <!ELEMENT password            (#PCDATA)>
  <!ELEMENT command             (#PCDATA)>
  <!ELEMENT parameters          (msisdn,includeRequests?,includeSentMessages?)>
  <!ELEMENT msisdn              (#PCDATA)>
  <!ELEMENT includeRequests     EMPTY>
  <!ELEMENT includeSentMessages EMPTY>
```

5.3.42.2. Request Specific Parameters

Parameter	Description
msisdn	The MSISDN of the user to request info for.
includeRequests	Whether to include the 10 last SMS received from the specified MSISDN over the past week in the response.
includeSentMessages	Whether to include the 10 last SMS sent messages to the specified MSISDN over the past week in the response.

5.3.42.3. USERINFO Command Reply DTD

```
  <!ELEMENT SMSBoxXMLReply    ((ok|error),command,requestUid)>
  <!ELEMENT ok                EMPTY>
  <!ELEMENT error             EMPTY>
    <!ATTLIST error type CDATA #REQUIRED>
  <!ELEMENT command           ((msisdn,operator,lastRequest,subscribedService*,
                                    ownedService*,blacklistedService*,blacklisted?,
                                    request*,sentMessage*,shippingAddress?)|EMPTY)>
    <!ATTLIST command name CDATA #FIXED "USERINFO">
  <!ELEMENT msisdn            (#PCDATA)>
  <!ELEMENT operator          (#PCDATA)>
  <!ELEMENT lastRequest       (#PCDATA)>
  <!ELEMENT subscribedService (#PCDATA)>
   <!ATTLIST subscribedService date CDATA>
  <!ELEMENT ownedService      (#PCDATA)>
   <!ATTLIST ownedService date CDATA>
  <!ELEMENT blacklistedService (#PCDATA)>
   <!ATTLIST blacklistedService date CDATA #IMPLIED>
   <!ATTLIST blacklistedService reason CDATA #IMPLIED>
  <!ELEMENT blacklisted        EMPTY>
   <!ATTLIST blacklisted date CDATA #IMPLIED>
   <!ATTLIST blacklisted reason CDATA #IMPLIED>
  <!ELEMENT request           (message,date,status,service,command)>
  <!ELEMENT message           (#PCDATA)>
  <!ELEMENT date              (#PCDATA)>
  <!ELEMENT status            (#PCDATA)>
  <!ELEMENT service           (#PCDATA)>
  <!ELEMENT command           (#PCDATA)>
  <!ELEMENT sentMessage       (message?,binaryMessage?,date,status,service,cost?)>
  <!ELEMENT binaryMessage     EMPTY>
  <!ELEMENT cost              (#PCDATA)>
  <!ELEMENT action            (#PCDATA)>
  <!ELEMENT shippingAddress   (#PCDATA)>
   <!ATTLIST shippingAddress firstName CDATA #REQUIRED>
   <!ATTLIST shippingAddress lastName CDATA #REQUIRED>
   <!ATTLIST shippingAddress street CDATA #REQUIRED>
   <!ATTLIST shippingAddress postcode CDATA #REQUIRED>
   <!ATTLIST shippingAddress city CDATA #REQUIRED>
  <!ELEMENT requestUid        (#PCDATA)>
```

5.3.42.4. Specific Parameters

Parameter	Description
operator	Operator the user is connected to.
lastRequest	Timestamp of the last request received from the user (last MO-SMS), in the timezone of the instance. Format is YYYY-MM-DD HH:MM:SS
subscribedService	Name of a service the user is subscribed to. Only services accessible for the login will be displayed. The attribute `date` indicates the date of the subscription to the service.
ownedService	Name of a service the user owns. Only services accessible for the login will be displayed. The attribute `date` indicates the date of the creation of the service.
blacklistedService	Name of a service the user is blacklisted to. The optional attribute `date` indicates the date of the blacklist to the service if available, and the optional attribute `reason` indicates the reason for the blacklist if available (possibly empty).
blacklisted	When the user is blacklisted for all services. The optional attribute `date` indicates the date of the blacklist to the service if available, and the optional attribute `reason` indicates the reason for the blacklist if available (possibly empty).
request	Information about a message received from the user (if `<includeRequests/>` was specified):
... ***message***	Text content of the received message.
... ***date***	Timestamp of the received message. Format is YYYY-MM-DD HH:MM:SS and it is expressed in the timezone of the instance.
... ***status***	Status of the received message. Is typically `ok` when the command was properly executed, else an error code.
... ***service***	Service related to the received message.
... ***command***	Command name triggered by the received message.
sentMessage	Information about a message sent to the user (if `<includeSentMessages/>` was specified):
... ***message***	Text content of the sent message, if applicable (otherwise, `<binaryMessage/>` will be replied).
... ***binaryMessage***	Indicates the message was binary (Wap Push, Operator Logo, etc).
... ***date***	Timestamp of the sent message. Format is YYYY-MM-DD HH:MM:SS and it is expressed in the timezone of the instance.
... ***status***	

	Status of the sent message (at the precise timestamp this request is performed - can be subsequently updated if operator sends us delivery notifications). Starts by `sent` if sent message is successfully sent, `delivered` if delivered on user's phone, `failed` if a failure occurred, and can be followed by operator-specific additional information.
... ***service***	Service related to this sent message.
... ***action***	Action name of this sent message.
... ***cost***	Cost of the sent message, in cents, if MT billing is available.
shippingAddress	The shipping address of the user, if existing and available. The required attributes indicate the splitted information by part, and the CDATA contains a human readable concatenated view of the address.

5.3.42.5. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
unknownmember	The specified end-user is unknown (never interacted with the instance).

5.3.42.6. USERINFO Request Example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>USERINFO</command>
  <parameters>
    <msisdn>+41761234567</msisdn>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="USERINFO">
    <msisdn>+41761234567</msisdn>
    <operator>sunrise</operator>
    <lastRequest>2006-03-16 16:07:01</lastRequest>
    <subscribedService date="2006-01-04 08:57:12">FCBNEWS</subscribedService>
    <subscribedService date="2006-02-07 15:01:27">FCBTOTOMAT</subscribedService>
    <ownedService date="2006-01-02 18:25:33">ROM HIPPY</ownedService>
    <ownedService date="2006-01-16 08:57:56">MOBOX HIPPY</ownedService>
    <blacklisted date="2006-03-11 15:01:27" reason="Just a test"/>
    <shippingAddress firstName="John" lastName="Doe" street="1 Main Street" postcode="12345"
         city="Anytown">John Doe, 1 Main Street, 12345 Anytown</shippingAddress>
  </command>
  <requestUid>xml9677318</requestUid>
</SMSBoxXMLReply>
```

5.3.42.7. USERINFO Request Example, including requests and sent messages

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>USERINFO</command>
  <parameters>
    <msisdn>+41761234567</msisdn>
    <includeRequests/>
    <includeSentMessages/>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="USERINFO">
    <msisdn>+41761234567</msisdn>
    <operator>sunrise</operator>
    <lastRequest>2006-03-16 16:07:01</lastRequest>
    <subscribedService date="2006-01-04 08:57:12">FCBNEWS</subscribedService>
    <subscribedService date="2006-02-07 15:01:27">FCBTOTOMAT</subscribedService>
    <ownedService date="2006-01-02 18:25:33">ROM HIPPY</ownedService>
    <ownedService date="2006-01-16 08:57:56">MOBOX HIPPY</ownedService>
    <request>
      <message>Stop TOPNEWS</message>
      <date>2006-03-16 16:07:01</date>
      <status>ok</status>
      <service>TOPNEWS</service>
      <command>UNSUBSCRIBE</command>
    </request>
    <request>
      <message>MNC LOGO</message>
      <date>2006-03-14 15:30:08</date>
      <status>ok</status>
      <service>MNC LOGO</service>
      <command>GET</command>
    </request>
    <sentMessage>
      <message>You have been unsubscribed from TOPNEWS. See you soon!</message>
      <date>2006-03-16 16:07:01</date>
      <status>sent</status>
      <service>TOPNEWS</service>
      <cost>0</cost>
      <action>UNSUBSCRIBE</action>
    </sentMessage>
    <sentMessage>
      <message>TOPNEWS&lt; Queen&apos;s birthday today!
          Don&apos;t forget to send flowers. Usual shipping address.</message>
      <date>2006-03-16 10:03:12</date>
      <status>sent</status>
      <service>TOPNEWS</service>
      <cost>50</cost>
      <action>PUSH</action>
    </sentMessage>
    <sentMessage>
      <binaryMessage/>
      <date>2006-03-14 15:30:08</date>
      <status>sent</status>
      <service>MNC LOGO</service>
      <cost>40</cost>
      <action>OPLOGO</action>
    </sentMessage>
  </command>
  <requestUid>xml9677318</requestUid>
</SMSBoxXMLReply>
```

5.3.43. VOTE

The VOTE command allows an external application to trigger an end-user vote in an SMS election/voting.

Important: depending on sites, smsBox® may need to generate a confirmation request SMS before the actual voting participation is performed (double optin case). That is stated in the XML response by the presence of the empty element `<needdoubleoptin/>` in the `<command>` element.

5.3.43.1. VOTE Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (election,candidate,msisdn,operator,silent?)>
```

```
<!ELEMENT election          (#PCDATA)>
<!ELEMENT candidate         (#PCDATA)>
<!ELEMENT msisdn            (#PCDATA)>
<!ELEMENT silent            EMPTY>
```

5.3.43.2. VOTE Request Specific Parameters

Parameter	Description
election	The name of the voting.
candidate	The name of the candidate. It must belong to the voting specified in `election`.
msisdn	The end-user for which the new vote will be added.
operator	The operator of the end-user, used in case the MSISDN is not yet known.
silent	An optional element preventing the sending of the confirmation SMS message to the selected MSISDN.

5.3.43.3. VOTE Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok        EMPTY>
<!ELEMENT error     EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command  (needdoubleoptin|EMPTY)>
  <!ATTLIST command name CDATA #FIXED "VOTE">
<!ELEMENT requestUid (#PCDATA)>
<!ELEMENT needdoubleoptin EMPTY>
```

5.3.43.3.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
themenomatch	The specified service does not exist.
deactivatedtheme	The specified service is deactivated.
noaccess	You do not have access to the specified service.
electionnomatch	The specified service is not an election/voting.
candidatenomatch	The specified candidate does not exist.
electionnotstarted	The specified election is not started.
electionover	The specified election is over (finished).
votedfor	The specified end-user already voted for this candidate (multionce and one vote voting type).
votedinelection	The specified end-user already voted in this voting (one vote voting type).

5.3.43.4. VOTE Request Example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>VOTE</command>
  <parameters>
    <election>FCBVOTE</election>
    <candidate>DERDIYOK</candidate>
    <msisdn>+41781234567</msisdn>
    <operator>orange</operator>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="VOTE"/>
  <requestUid>xml9677297</requestUid>
</SMSBoxXMLReply>
```

5.3.44. WAPPUSH

The Wap Push SMS are deprecated because they are not supported by modern smartphones, thus smsBox® will automatically convert Wap Push SMS to a text SMS containing equivalent text and URL.

WAPPUSH is used to transmit a Wap Push SMS to a mobile user. This Service Indication (SI) will offer the end-user to open a URL. Notice that the URL can point to a Wap/XHTML page, or to a download (which can be of any type: image, java application, ringtone, etc). Due to the different policies from the operators, this feature is available only on some instances. Please check with the support group.

The message can be sent to multiple receivers in only one request by using the `multiReceiver` element instead of `receiver`.

In the reply XML document, the status element will be `ok` if the message could be sent to **all** receivers. The status element will be error if **one or more** of the receivers were not validated. The receiver elements in the reply XML document contain a status attribute (with value `ok`, or a specific error). The status of the last invalid receiver will be the error type contained in the `type` attribute of the `error` status element.

5.3.44.1. WAPPUSH Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       ((receiver|multiReceiver+),service,text,url,cost?,forceRealWapPush?)>
<!ELEMENT receiver         (#PCDATA)>
<!ELEMENT multiReceiver    (#PCDATA)>
<!ELEMENT service          (#PCDATA)>
<!ELEMENT text             (#PCDATA)>
<!ELEMENT url              (#PCDATA)>
<!ELEMENT cost             (#PCDATA)>
<!ELEMENT forceRealWapPush EMPTY>
```

5.3.44.2. Specific Parameters

Parameter	Description
receiver, ***multiReceiver***, ***service***, ***cost***	*same as SEND command.*
text	This is the text of the message to display on the phone (size is limited, see note below).
url	URL where the content can be retrieved from.
forceRealWapPush	(optional) By default, the wap pushes are converted to standard sms. Force sending a real wap push by setting this parameter.

Note: the text plus the URL must not account for more than 100 UTF-8 encoded bytes (e.g. 100 characters if you're not sending accented characters, a little less if that's the case).

5.3.44.3. WAPPUSH Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok       EMPTY>
<!ELEMENT error    EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command  (receiver+|EMPTY)>
```

```
    <!ATTLIST command name CDATA #FIXED "WAPPUSH">
  <!ELEMENT receiver (#PCDATA)>
    <!ATTLIST receiver status CDATA #REQUIRED>
  <!ELEMENT requestUid (#PCDATA)>
```

5.3.44.3.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
themenomatch	The specified service does not exist.
deactivatedtheme	The specified service is deactivated.
noaccess	You do not have access to the specified service.
costnomatch	The specified cost is incorrect or unauthorized.
blacklistmember	The specified end-user is blacklisted.
selfblacklistmember	The specified end-user has blacklisted himself.
unknownmember	The specified end-user is unknown (never interacted with the instance).
missingwhitelistmember	A whitelist entry is needed for the specified service, but is missing.
notadult	On an adult-only instance, the specified end-user has not previously certified being an adult.
expiredadult	On an adult-only instance, the specified end-user has previously certified being an adult, but his certification has expired.
notsubscribed	The specified end-user is not subscribed to the specified service.
duplicate	The specified end-user is specified more than once.

5.3.44.3.2. Per-receiver Possible Erroneous Statuses

Erroneous Status	Description
blacklistmember	The specified end-user is blacklisted.
selfblacklistmember	The specified end-user has blacklisted himself.
unknownmember	The specified end-user is unknown (never interacted with the instance).
missingwhitelistmember	A whitelist entry is needed for the specified service, but is missing.
notadult	On an adult-only instance, the specified end-user has not previously certified being an adult.
expiredadult	On an adult-only instance, the specified end-user has previously certified being an adult, but his certification has expired.
notsubscribed	The specified end-user is not subscribed to the specified service.
duplicate	The specified end-user is specified more than once.

5.3.44.4. WAPPUSH Command Request Example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>WAPPUSH</command>
  <parameters>
    <receiver>+41761234567</receiver>
    <service>GAME133</service>
    <text>Super Gazilla Jumper</text>
    <url>http://www.siteexample.com/ downloads/game133.jad</url>
    <cost>20</cost>
  </parameters>
</SMSBoxXMLRequest>
```

Note: total size of text plus URL is 68 bytes, WAPPUSH will be accepted.

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="WAPPUSH">
    <receiver status="ok">+41761234567</receiver>
  </command>
  <requestUid>xml9677319</requestUid>
</SMSBoxXMLReply>
```

5.3.44.5. WAPPUSH Command Request Example with receiver errors

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>WAPPUSH</command>
  <parameters>
    <multiReceiver>+41761234567</multiReceiver>
    <multiReceiver>+41761234568</multiReceiver>
    <multiReceiver>+41761234567</multiReceiver>
    <multiReceiver>+41761234569</multiReceiver>
    <service>GAME133</service>
    <text>Super Gazilla Jumper</text>
    <url>http://www.siteexample.com/ downloads/game133.jad</url>
    <cost>20</cost>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <error type="duplicate"/>
  <command name="WAPPUSH">
    <receiver status="ok">+41761234567</receiver>
    <receiver status="notsubscribed">+41761234568</receiver>
    <receiver status="duplicate">+41761234567</receiver>
    <receiver status="ok">+41761234569</receiver>
  </command>
  <requestUid>xml9677319</requestUid>
</SMSBoxXMLReply>
```

5.3.45. WEBCHARGE

The WEBCHARGE command asks to charge an unknown user for some amount, thanks to the additional `operator` parameter.

The request is similar to the one you would perform for CHARGE, the only difference is that you need to specify the operator in `<parameters>`, for example `<operator>sunrise</operator>`, except on sites where it's not mandatory.

5.3.45.1. WEBCHARGE Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (service,operator,msisdn,cost)>
<!ELEMENT service          (#PCDATA)>
<!ELEMENT operator         (#PCDATA)>
<!ELEMENT msisdn           (#PCDATA)>
<!ELEMENT cost             (#PCDATA)>
```

5.3.45.2. Request Specific Parameters

Parameter	Description
service	Name of the service related to the charging.
msisdn	Customer to charge.
operator	Operator of this customer.
cost	Amount to charge for (in cents).

The WEBCHARGE command may enforce the same restrictions as the CHARGE command, depending on configuration.

5.3.45.3. WEBCHARGE Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok EMPTY>
<!ELEMENT error EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command EMPTY>
  <!ATTLIST command name CDATA #FIXED "WEBCHARGE">
<!ELEMENT requestUid (#PCDATA)>
```

5.3.45.3.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
themenomatch	The specified service does not exist.
deactivatedtheme	The specified service is deactivated.
noaccess	You do not have access to the specified service.
costnomatch	The specified cost is incorrect or unauthorized.
missingoperator	The `operator` parameter is missing.
badoperator	The specified operator is invalid or unsupported.
badphone	The phone number has a bad syntax (bad prefix, missing digits...).
appnoaccess	You do not have access to the smartphone app related to the phone number (only for smartphone app messaging).
blacklistmember	The specified end-user is blacklisted.
selfblacklistmember	The specified end-user has blacklisted himself.
missingwhitelistmember	A whitelist entry is needed for the specified service, but is missing.
notadult	On an adult-only instance, the specified end-user has not previously certified being an adult.
expiredadult	On an adult-only instance, the specified end-user has previously certified being an adult, but his certification has expired.
failed|direct-billing-not-supported	The operator doesn't support direct billing.

5.3.45.4. WEBCHARGE example

Request:

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

```
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>WEBCHARGE</command>
  <parameters>
    <service>MNC DL</service>
    <msisdn>+41761234567</msisdn>
    <operator>orange</operator>
    <cost>100</cost>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="WEBCHARGE"/>
  <requestUid>xml9677320</requestUid>
</SMSBoxXMLReply>
```

5.3.46. WEBSEND

The WEBSEND command is similar to the SEND command, except the SMS can be sent to any end-user, even if they are not already in the platform database and not subscribed to the service. An additional `guessOperator` or `operator` parameter is required to know through which operator the SMS is to be sent when the receiver's MSISDN is not already in the platform database.

5.3.46.1. WEBSEND Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       ((receiver|multiReceiver+),service,text,
                                operator?,guessOperator?,cost?,
                                maximumSMSAmount?,forceUseUcs2?,test?)>
<!ELEMENT receiver         (#PCDATA)>
<!ELEMENT multiReceiver    (#PCDATA)>
<!ELEMENT service          (#PCDATA)>
<!ELEMENT text             (#PCDATA)>
<!ELEMENT guessOperator    EMPTY>
<!ELEMENT operator         (#PCDATA)>
<!ELEMENT cost             (#PCDATA)>
<!ELEMENT maximumSMSAmount (#PCDATA)>
<!ELEMENT forceUseUcs2     (#PCDATA)>
<!ELEMENT test             EMPTY>
```

5.3.46.2. Specific Parameters

Parameter	Description
receiver, ***multiReceiver***, ***service***, ***text***, ***cost***, ***maximumSMSAmount***, ***test***	Please see definition in SEND command.
guessOperator	Instruct to guess the correct/best operator to use to send the message, if the receiver's MSISDN is unknown. Should be used always except when otherwise stated.
operator	Specific operator to use to send the message, if the receiver's MSISDN is unknown.

5.3.46.3. WEBSEND Command Reply

Same as SEND command.

5.3.46.3.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
themenomatch	The specified service does not exist.
deactivatedtheme	The specified service is deactivated.
noaccess	You do not have access to the specified service.
costnomatch	The specified cost is incorrect or unauthorized.
missingoperator	The `operator` parameter is missing.
badoperator	The specified operator is invalid or unsupported.
badphone	The phone number has a bad syntax (bad prefix, missing digits...).
appnoaccess	You do not have access to the smartphone app related to the phone number (only for smartphone app messaging).
blacklistmember	The specified end-user is blacklisted.
selfblacklistmember	The specified end-user has blacklisted himself.
missingwhitelistmember	A whitelist entry is needed for the specified service, but is missing.
notadult	On an adult-only instance, the specified end-user has not previously certified being an adult.
expiredadult	On an adult-only instance, the specified end-user has previously certified being an adult, but his certification has expired.
duplicate	The specified end-user is specified more than once.

5.3.46.3.2. Per-receiver Possible Erroneous Statuses

Erroneous Status	Description
missingoperator	The `operator` parameter is missing.
badoperator	The specified operator is invalid or unsupported.
badphone	The phone number has a bad syntax (bad prefix, missing digits...).
appnoaccess	You do not have access to the smartphone app related to the phone number (only for smartphone app messaging).
blacklistmember	The specified end-user is blacklisted.
selfblacklistmember	The specified end-user has blacklisted himself.
missingwhitelistmember	A whitelist entry is needed for the specified service, but is missing.
notadult	On an adult-only instance, the specified end-user has not previously certified being an adult.
expiredadult	On an adult-only instance, the specified end-user has previously certified being an adult, but his certification has expired.
duplicate	The specified end-user is specified more than once.
nogateway	There is no gateway (no sending component) for the operator of the specified end-user.
nomessage	The message for this end-user has been deactivated on the platform.

5.3.46.4. WEBSEND Command Request Example

Request:

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

```
  <username>myuser</username>
  <password>mypass</password>
  <command>WEBSEND</command>
  <parameters>
    <receiver>+41761234567</receiver>
    <service>ULTIMATE</service>
    <text>This is a message from us!</text>
    <cost>20</cost>
    <guessOperator/>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="WEBSEND">
    <receiver status="ok">+41761234567</receiver>
  </command>
  <requestUid>xml9677321</requestUid>
</SMSBoxXMLReply>
```

5.3.46.5. WEBSEND Command Request Example (multiple receivers)

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>WEBSEND</command>
  <parameters>
    <multiReceiver>+41790000001</multiReceiver>
    <multiReceiver>+41790000002</multiReceiver>
    <multiReceiver>+41790000003</multiReceiver>
    <service>TEST</service>
    <text>This is a test message.</text>
    <cost>50</cost>
    <guessOperator/>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="WEBSEND">
    <receiver status="ok">+41790000001</receiver>
    <receiver status="ok">+41790000002</receiver>
    <receiver status="ok">+41790000003</receiver>
  </command>
  <requestUid>xml9677322</requestUid>
</SMSBoxXMLReply>
```

5.3.47. WEBSENDBINARY

WEBSENDBINARY is almost identical to SENDBINARY. It doesn't require the end-user to be already present in the database and subscribed to a service. Just like WEBSEND, it is only activable for non-premium short numbers.

The request is similar to the one you would perform for SENDBINARY, the only difference is that you need to specify the operator in `<parameters>`, for example `<operator>sunrise</operator>`, except on sites where it's not mandatory.

Please refer to *Appendix E* for examples of different classes of binary messages.

5.3.47.0.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
themenomatch	The specified service does not exist.
deactivatedtheme	The specified service is deactivated.
noaccess	You do not have access to the specified service.
costnomatch	The specified cost is incorrect or unauthorized.
missingoperator	The `operator` parameter is missing.
badoperator	The specified operator is invalid or unsupported.
badphone	The phone number has a bad syntax (bad prefix, missing digits...).
appnoaccess	You do not have access to the smartphone app related to the phone number (only for smartphone app messaging).
blacklistmember	The specified end-user is blacklisted.
selfblacklistmember	The specified end-user has blacklisted himself.
missingwhitelistmember	A whitelist entry is needed for the specified service, but is missing.
notadult	On an adult-only instance, the specified end-user has not previously certified being an adult.
expiredadult	On an adult-only instance, the specified end-user has previously certified being an adult, but his certification has expired.

5.3.48. WEBSUBSCRIBE

The WEBSUBSCRIBE command is similar to the SUBSCRIBE command, but it can be sent to any end-user, even if they are not already in the platform database and not subscribed to a service.

The request is similar to the one you would perform for SUBSCRIBE, the only difference is that you need to specify the operator in `<parameters>`, for example `<operator>sunrise</operator>`, except on sites where it's not mandatory and on non-premium short numbers, in which case smsBox® makes best guess out of the MSISDN specified by `receiver`.

Important: depending on sites, smsBox® may need to generate a subscription confirmation SMS before the actual subscription (double optin case). In such a case, the end-user will not be immediately subscribed after that command; only after the end-user sends a confirmation SMS he will be really subscribed. That is stated in the XML response by the presence of the empty element `<needdoubleoptin/>` in the `<command>` element. Applications needing to act on actual subscription must use the SUBSCRIBE notification.

5.3.48.1. WEBSUBSCRIBE Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (msisdn,operator,service,silent?)>
<!ELEMENT msisdn           (#PCDATA)>
<!ELEMENT operator         (#PCDATA)>
<!ELEMENT service          (#PCDATA)>
<!ELEMENT silent           EMPTY>
```

5.3.48.2. WEBSUBSCRIBE Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok             EMPTY>
<!ELEMENT error          EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command        (needdoubleoptin|EMPTY)>
  <!ATTLIST command name CDATA #FIXED "WEBSUBSCRIBE">
```

```
<!ELEMENT requestUid (#PCDATA)>
<!ELEMENT needdoubleoptin EMPTY>
```

5.3.48.2.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
themenomatch	The specified service does not exist.
deactivatedtheme	The specified service is deactivated.
noaccess	You do not have access to the specified service.
costnomatch	The specified cost is incorrect or unauthorized.
missingoperator	The `operator` parameter is missing.
badoperator	The specified operator is invalid or unsupported.
badphone	The phone number has a bad syntax (bad prefix, missing digits...).
appnoaccess	You do not have access to the smartphone app related to the phone number (only for smartphone app messaging).
blacklistmember	The specified end-user is blacklisted.
selfblacklistmember	The specified end-user has blacklisted himself.
missingwhitelistmember	A whitelist entry is needed for the specified service, but is missing.
notadult	On an adult-only instance, the specified end-user has not previously certified being an adult.
expiredadult	On an adult-only instance, the specified end-user has previously certified being an adult, but his certification has expired.
themenotavailable	The specified service is currently marked as not available.
insubscription	The specified end-user is currently subscribed to the specified service.

5.3.48.3. WEBSUBSCRIBE Command Request Example, need double optin case

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>WEBSUBSCRIBE</command>
  <parameters>
    <msisdn>+41761234567</msisdn>
    <operator>sunrise</operator>
    <service>FCBTOTOMAT</service>
    <silent/>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="WEBSUBSCRIBE">
    <needdoubleoptin/>
  </command>
  <requestUid>xml9677323</requestUid>
</SMSBoxXMLReply>
```

5.3.48.4. WEBSUBSCRIBE Command Request Example, regular case

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>WEBSUBSCRIBE</command>
  <parameters>
    <msisdn>+41761234567</msisdn>
    <operator>sunrise</operator>
    <service>FCBTOTOMAT</service>
    <silent/>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="WEBSUBSCRIBE"/>
  <requestUid>xml9677323</requestUid>
</SMSBoxXMLReply>
```

5.3.49. WEBWAPPUSH

WEBWAPPUSH is the same as WAPPUSH, but allows sending to end-users who are not in the database and not subscribed to a service. The service parameter is required but the actual service doesn't necessary exist.

The request is similar to the one you would perform for WAPPUSH, the only difference is that you need to specify the operator in `<parameters>`, for example `<operator>sunrise</operator>`, except on sites where it's not mandatory.

5.3.49.0.1. Possible Error Types

The Common Error Types listed in **Common Parameters**, and:

Error Type	Description
themenomatch	The specified service does not exist.
deactivatedtheme	The specified service is deactivated.
noaccess	You do not have access to the specified service.
costnomatch	The specified cost is incorrect or unauthorized.
missingoperator	The `operator` parameter is missing.
badoperator	The specified operator is invalid or unsupported.
badphone	The phone number has a bad syntax (bad prefix, missing digits...).
appnoaccess	You do not have access to the smartphone app related to the phone number (only for smartphone app messaging).
blacklistmember	The specified end-user is blacklisted.
selfblacklistmember	The specified end-user has blacklisted himself.
missingwhitelistmember	A whitelist entry is needed for the specified service, but is missing.
notadult	On an adult-only instance, the specified end-user has not previously certified being an adult.
expiredadult	On an adult-only instance, the specified end-user has previously certified being an adult, but his certification has expired.
duplicate	The specified end-user is specified more than once.

6. MMS Functions

The following sections describe the MMS functionality available in the smsBox® platform. The smsBox® platform supports command notifications for end-user originated MMS messages (MO MMS). For a generic explanation on how MMS is supported, please refer to *Appendix A*. For the trace of a complete MMS request down at the HTTP byte level, please refer to *Appendix D*.

6.1. MMS Notification XML API (interactions from smsBox®)

As with the standard command notifications sent by smsBox® for SMS messages, mobile users originated MMS messages (MO MMS) received by the smsBox® platform invoke various commands which may in turn notify registered applications.

The command notification process is similar to that used for SMS messages. In case of HTTP POST transport, however, with MMS messages, the HTTP request is a `multi-part/form-data` POST request containing the notification XML document plus any pertinent content elements from the original MO MMS (such as images, videos, audio files, or text). In case of Internet mail transport, MMS content is sent as attachments.

6.1.1. FORWARD

The end-user originated MMS message (MO MMS) is converted into a `multipart/form-data` HTTP POST request (this request adheres to **RFC-1867**: *"Form-based File Upload in HTML"* which defines `multipart/form-data`), or into an Internet mail, and sent to the external application that is registered to receive this content. The first part of this request is a `<Notification>` XML document, exactly like the document used for the standard FORWARD command notification. In case of HTTP POST transport, the `content-disposition` header contains the attribute `name="Notification"`. The other parts of the `multipart/form-data` request correspond to the individual parts of the MMS message sent by the end-user. In case of Internet mail transport, the notification document is the first part and does not contain a `Content-Disposition` header. The other parts follow and contain a `Content-Disposition` header indicating they are attachments.

Each subsequent part contains a `content-type` and a `content-disposition` header. The `content-type` describes the type of content of this part (such as `text/plain` or `image/jpeg`). The `content-disposition` header specifies the original name and filename of this part given by the end-user's telephone. As each mobile telephone operates in a slightly different manner, the exact values of the `content-type` and `content-disposition` headers will vary for each request.

In case of HTTP POST transport, the external application must respond with either a `text/xml` content-type containing a `<NotificationReply>` XML document or a `multipart/form-data` content-type containing a `<NotificationReply>` XML document in the first part (named "`SMSBoxXMLResponse`") plus any other parts containing other MMS content. The `<NotificationReply>` document has exactly the same format as the response to a standard Notification XML request (one invoked by an SMS message). The text of each `<message>` element in the `<NotificationReply>` will be added as a text part of the resulting MMS message that is sent to the end-user. The cost for this MT MMS is the value of the last `<cost>` tag from the `<NotificationReply>` document (if MT billing is available).

If the external application choses to send a `multipart/form-data` response, each part of that response must contain the following headers:

- `content-type`

  ```
  Content-Type: text/xml; charset=UTF-8
  ```

- `content-disposition`

  ```
  Content-Disposition: Attachment; Name="SMSBoxXMLResponse";
  Filename="SMSBoxXMLResponse.xml"
  ```

Other headers are allowed, but the afore mentioned headers are mandatory. The `content-disposition` header must be of type "`Attachment`" and must contain a "`Filename`" parameter and a "`Name`" parameter. Any text content parts must have their `content-type` header set to "`text/plain`" and be ***encoded*** in UTF-8 (`charset=utf-8`).

In case of Internet mail transport, the external application must use the WEBSENDMMS XML API command if it needs to send back an MMS to the mobile user.

6.2. MMS Command XML API (interactions to smsBox®)

All MMS commands of the XML API have to adhere to a request format that is slightly different than that for standard XML API commands. The MMS commands require the XML request document plus the MMS contents.

In order to send all the media content representing the MMS, an HTTP POST with a `Content-Type: multipart/form-data` header is required. This request must adhere to **RFC-1867**: *"Form-based File Upload in HTML"* which defines `multipart/form-data`.

The multipart request is structured as follows:

- The first part of the multipart request must be the **XML request document**. Its field name must be `SMSBoxXMLRequest`. This XML request document is exactly the same as other XML API request documents (described previously) - it must be encoded in UTF-8 (or is immediately rejected). The DTD and examples for the currently supported commands can be found in the sections below.
- The subsequent parts in the multipart request are the **MMS content parts**. There is no restriction on the field names for these parts. `Content-Type` must be declared for all parts, **including a mandatory `charset`** for text parts (UTF-8 strongly advised).
- An optional **last part** in the multipart request is the **MMS Slides Definition** document; its `Content-Type` must be declared as `application/cyote-mms-slides-definition; charset=UTF-8`.

All content-types are accepted by the platform, the partner is responsible for verifying that the operator's network will accept carrying such content, and that the mobile phone of the end-user will be able to display it. The only limitation is the ability of the platform to guess the generic type of the content received: knowing if the part is text, audio, image or video is mandatory. To achieve that, declared content-type is first used; if that process fails, file extension is then used; if that process fails too, the part is silently discarded (e.g. the MMS will contain all the rest but not this part); if you need that a new content-type or file extension is supported, please contact the support people, as this can easily be added to the configuration of the smsBox® platform.

Here's a list of well-known content-types that should be ok in most situations:

Content-Type	File Ending
`text/plain; charset=UTF-8`	`.txt`
`application/smil`	`.smil`
`image/jpeg`	`.jpg, .jpeg`
`image/gif`	`.gif`
`audio/mpeg`	`.mp3`
`audio/amr`	`.amr`
`audio/x-midi`	`.midi`
`application/java-archive`	`.jar`

All operators do not yet accept all content-types. In Switzerland, currently Swisscom and Sunrise MMSCs do not accept `.jar`.

Note: the file names must be pure alphanumeric ASCII (no accented chars, no special/punctuation characters except underscore, dash and dot) and contain no spaces.

6.2.1. MMS Slides Definition

The MMS Slides Definition optional document describes the layout of the slides to create with the content parts. With that document, a proper SMIL document is generated and sent to the devices.

The MMS Slides Definition document is an XML document with the following structure:

```
<!ELEMENT slides (slide+)>
<!ELEMENT slide (element+)>
```

```
  <!ATTLIST slide duration CDATA #IMPLIED>
<!ELEMENT element EMPTY>
  <!ATTLIST element cid CDATA #REQUIRED>
```

Each specified slide may have a duration (in seconds). Each element must refer to a content, by using the `cid=` attribute, whose value refers to the field names in the multipart HTTP POST.

6.2.1.1. MMS Slides Definition example

An example file describing two slides, the first slide with a duration of 10 seconds using two elements, the second slide with an unspecified duration using one element.

```
<?xml version="1.0" encoding="UTF-8" ?>
<slides>
  <slide duration="10">
    <element cid="1"/>
    <element cid="2"/>
  </slide>
  <slide>
    <element cid="3"/>
  </slide>
</slides>
```

Notice: slides with unspecified duration are useful for audio and video parts.

6.2.1.2. Compatibily warning

More than two elements per slide, or more than one element of the same type per slide (e.g. two texts on a single slide, or two images on a single slide, etc), are not adviced because on most phones this layout will not be properly supported/displayed. If you use the MMS Slides Definition feature, it is **strongly adviced** to test the results of your MMS towards different types of phone models/manufacturers/networks, since some of them may adapt or not support the expected result.

6.2.2. AutoSMIL feature

After the compulsory XML request document `SMSBoxXMLRequest`, the part you can **optionally** provide is the SMIL document for the MMS.

If no SMIL is provided, the platform will generate a SMIL document; if you provided an MMS Slides Definition file, the generated SMIL document will match its content; else it will be made of one slide per element provided in your request in the same order, with a default display time (5 seconds for images, 3 + text length / 25 seconds for texts, unspecified for audios and videos).

If you want your MMS to be sent without a SMIL document to the device, you have to set the `noAutoSmil` element in your XML document request, but in our experience it's not a wise decision because it is not well supported by phones.

6.2.3. Forward Lock feature

Reasonably recent mobile phones (2004 onwards) support some DRM (Digital Rights Management) features, among which the Forward Lock is supported by smsBox®. When an MMS is marked with the Forward Lock flag, the receiver mobile phone will not allow that any multimedia parts contained in this MMS be forwarded to another mobile phone by MMS or saved on unsecure external storage. Note that if the mobile phone doesn't support the Forward-Lock feature, it will not be able to display/use the multimedia content at all.

If you want your MMS to be forward-locked, you have to set the `forwardLock` empty element in your XML document request.

Note that the Forward Lock feature may not be supported by smsBox® depending on capabilities of the operators MMS gateways. Check with your support team.

6.2.4. POSTMMS

The POSTMMS command will update the content of an existing MMS service, distributing the content to all subscribers. **The service must already exist as an MMS service**.

6.2.4.1. POSTMMS Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (service,silent?,noAutoSmil?,forwardLock?,subject?)>
<!ELEMENT service          (#PCDATA)>
<!ELEMENT silent           EMPTY>
<!ELEMENT noAutoSmil       EMPTY>
<!ELEMENT forwardLock      EMPTY>
<!ELEMENT subject          (#PCDATA)>
```

6.2.4.2. POSTMMS Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok    EMPTY>
<!ELEMENT error EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command (service|EMPTY)>
  <!ATTLIST command name CDATA #FIXED "POSTMMS">
<!ELEMENT service (#PCDATA)>
  <!ATTLIST service status CDATA #REQUIRED>
<!ELEMENT requestUid (#PCDATA)>
```

6.2.4.3. Request Specific Parameters

Parameter	Description
service	Name of the service to upload content to.
silent	Optional parameter. If present, no MMS push will happen. The content will be uploaded and will be available. If an end-user sends a GETMMS command per SMS, he will receive that content. If absent, the upload will trigger an MMS push to all end-users currently subscribed to the service.
noAutoSmil	Optional parameter. If present and no SMIL document was sent, the platform will not provide a SMIL structure with the MMS.
forwardLock	Optional parameter. If present, multimedia contents of the MMS will be forward-locked when sent to mobile phones.
subject	Optional. If present, sets the subject of the MMS. If absent, the subject is automatically set to be the service name. Maximum 40 characters.

6.2.4.4. POSTMMS Command Example

<u>1st element:</u>

◊ Name: `SMSBoxXMLRequest`
◊ Content-Type: `text/xml; charset=UTF-8`
◊ Contents:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>POSTMMS</command>
  <parameters>
    <service>mnc info</service>
  </parameters>
</SMSBoxXMLRequest>
```

<u>2nd element:</u>

◊ Name: `image.jpg`
◊ Content-Type: `image/jpeg`
◊ Contents:

<u>3rd element:</u>

◊ Name: `text.txt`
◊ Content-Type: `text/plain; charset=UTF-8`

◊ Contents: `This is the MNC info service by MMS.`

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="POSTMMS"/>
  <requestUid>xml9677324</requestUid>
</SMSBoxXMLReply>
```

Note: if you want to validate the data you send, you may want to send your data to a test server of yours and compare the content the server actually receives to traces you can find in **Appendix D**.

6.2.5. SENDMMS

The SENDMMS command will send an MMS to a specific end-user. Only the `SMSBoxXMLRequest` document is different; the attachments are defined in the same way as for the POSTMMS command.

6.2.5.1. SENDMMS Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (receiver,service,cost?,noAutoSmil?,forwardLock?,subject?)>
<!ELEMENT receiver         (#PCDATA)>
<!ELEMENT service          (#PCDATA)>
<!ELEMENT cost             (#PCDATA)>
<!ELEMENT noAutoSmil       EMPTY>
<!ELEMENT forwardLock      EMPTY>
<!ELEMENT subject          (#PCDATA)>
```

6.2.5.2. SENDMMS Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok EMPTY>
<!ELEMENT error EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command (receiver|EMPTY)>
  <!ATTLIST command name CDATA #FIXED "SENDMMS">
<!ELEMENT receiver (#PCDATA)>
  <!ATTLIST receiver status CDATA #REQUIRED>
<!ELEMENT requestUid (#PCDATA)>
```

6.2.5.3. Request Specific Parameters

Parameter	Description
receiver	MSISDN of the receiving end-user. As for the SEND command, the user has to be already present in the smsBox® database and has to be subscribed the `service` indicated.
service	Name of the service in which context the message is sent. This is for logging and statistics purposes, as well as to determine if the user is still subscribed to this service.
cost	Amount to charge the end-user for the content, if MT billing is available. Expressed in cents.
noAutoSmil	Optional parameter. If present and no SMIL document was sent, the platform will not provide a SMIL structure with the MMS.
forwardLock	Optional parameter. If present, multimedia contents of the MMS will be forward-locked when sent to mobile phones.
subject	Optional. If present, sets the subject of the MMS. If absent, the subject is automatically set to be the service name. Maximum 40 characters.

Note: multiple recipients are currently not supported.

6.2.5.4. SENDMMS Command Example

1st element:

◊ Name: `SMSBoxXMLRequest`
◊ Content-Type: `text/xml; charset=UTF-8`
◊ Contents:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>SENDMMS</command>
  <parameters>
    <receiver>+41761231234</receiver>
    <service>mnc alt</service>
    <cost>300</cost>
  </parameters>
</SMSBoxXMLRequest>
```

2nd element:

◊ Name: `rose.jpg`
◊ Content-Type: `image/jpeg`
◊ Contents:

3rd element:

◊ Name: `text.txt`
◊ Content-Type: `text/plain; charset=UTF-8`
◊ Contents: `Some text between the images.`

4th element:

◊ Name: `saturn.jpg`
◊ Content-Type: `image/jpeg`
◊ Contents:

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="SENDMMS">
    <receiver status="ok">+41761231234</receiver>
  </command>
  <requestUid>xml9677325</requestUid>
</SMSBoxXMLReply>
```

6.2.6. SENDMMSSERVICE

The SENDMMSSERVICE command will trigger the sending of an MMS to a specific end-user. The current content of the indicated MMS service will be sent to the end-user at the current price for the corresponding pull request. *It is as if the end-user had requested the MMS service directly with an MO-SMS.*

6.2.6.1. SENDMMSSERVICE Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
```

```
<!ELEMENT parameters        (receiver,service)>
<!ELEMENT service           (#PCDATA)>
<!ELEMENT receiver          (#PCDATA)>
```

6.2.6.2. SENDMMSSERVICE Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok EMPTY>
<!ELEMENT error EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command (receiver|EMPTY)>
  <!ATTLIST command name CDATA #FIXED "SENDMMSSERVICE">
<!ELEMENT receiver (#PCDATA)>
  <!ATTLIST receiver status CDATA #REQUIRED>
<!ELEMENT requestUid (#PCDATA)>
```

6.2.6.3. Request Specific Parameters

Parameter	Description
receiver	MSISDN of the receiving end-user. As for the SEND command, the user has to be already present in the smsBox® database.
service	Name of the MMS service whose content is to be sent to the end-user.

6.2.6.4. SENDMMSSERVICE example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>SENDMMSSERVICE</command>
  <parameters>
    <service>MNC PICTURE</service>
    <receiver>+41765435432</receiver>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="SENDMMSSERVICE"/>
  <requestUid>xml9677326</requestUid>
</SMSBoxXMLReply>
```

6.2.7. WEBSENDMMS

The WEBSENDMMS command is very similar to the SENDMMS command. The MMS can be sent to any end-user, even if they are not already in the platform database and not subscribed to a service. An additional operator parameter is required to know through which operator the MMS is to be sent.

6.2.7.1. WEBSENDMMS Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username          (#PCDATA)>
<!ELEMENT password          (#PCDATA)>
<!ELEMENT command           (#PCDATA)>
<!ELEMENT parameters        (receiver,service,cost?,operator,noAutoSmil?,forwardLock?,subject?)>
<!ELEMENT receiver          (#PCDATA)>
<!ELEMENT service           (#PCDATA)>
<!ELEMENT cost              (#PCDATA)>
<!ELEMENT operator          (#PCDATA)>
<!ELEMENT noAutoSmil        EMPTY>
<!ELEMENT forwardLock       EMPTY>
<!ELEMENT subject           (#PCDATA)>
```

6.2.7.2. WEBSENDMMS Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok EMPTY>
<!ELEMENT error EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command (receiver|EMPTY)>
  <!ATTLIST command name CDATA #FIXED "WEBSENDMMS">
<!ELEMENT receiver (#PCDATA)>
  <!ATTLIST receiver status CDATA #REQUIRED>
<!ELEMENT requestUid (#PCDATA)>
```

6.2.7.3. Specific Parameters

Parameter	Description
receiver, ***service***, ***text***, ***cost***, ***noAutoSmil***, ***forwardLock***, ***subject***	Please see definition in SENDMMS command.
operator	Operator to use to send the message.

6.2.7.4. WEBSENDMMS Command Request Example

1st element:

◊ Name: `SMSBoxXMLRequest`
◊ Content-Type: `text/xml; charset=UTF-8`
◊ Contents:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>WEBSENDMMS</command>
  <parameters>
    <receiver>+41761231234</receiver>
    <service>mnc alt</service>
    <cost>300</cost>
    <operator>sunrise</operator>
  </parameters>
</SMSBoxXMLRequest>
```

Multimedia elements:

Same as SENDMMS example.

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="WEBSENDMMS">
    <receiver status="ok">+41761231234</receiver>
  </command>
  <requestUid>xml9677327</requestUid>
</SMSBoxXMLReply>
```

7. Mobile Internet browsing Functions

The following sections describe the Mobile Internet browsing functionality available in the smsBox® platform. The smsBox® platform can associate Mobile Internet pages to services, and mobile users can browse the Mobile Internet pages with their phone. Notice that Mobile Internet pages available in smsBox® are intentionally simple (no complex formatting or content) to make easy for partners to create and update their own pages. Commercial availability depends on the possible integration with operators, especially concerning billing and MSISDN communication. For the trace of a complete Mobile Internet request down at the HTTP byte level, please refer to *Appendix D*.

There can be interactions *from* smsBox®, to notify an external partner that some form data content was sent from a Mobile Internet page hosted by smsBox®, and there can be interactions *to* smsBox® to trigger Mobile Internet pages related commands in smsBox®.

7.1. Mobile Internet Notification XML API (interactions from smsBox®)

Notifications comprising form data content sent from a Mobile Internet page hosted by smsBox®, are sent using the same mechanism as with MMS Notifications: either the transport is HTTP, in which case the HTTP request is a `multi-part/form-data` POST request containing the notification XML document plus any form data sent from the Mobile Internet page, or the transport is Internet mail ("email") and content is sent as attachments.

Please refer to the MMS Notifications chapter for details. Regarding HTTP transport, the differences with MMS Notifications are:

- the added `<path>` element in the notification XML document indicating the path of the Mobile Internet page from which content was sent
- the `<command>` reported in the notification XML document is always `WAP-INPUT`
- the response must be a UTF-8 XML reply document, with `text/xml` content-type, with root `<NotificationReply>` and either one element `<url>` containing the URL to redirect the end-user to, or one element `<text>` containing text to display in response (same formatting rules as for text content are available)

7.2. Mobile Internet XML API commands (interactions to smsBox®)

The principle is similar to the one used for MMS Functions: smsBox® receives a `multipart/form-data` HTTP POST request. This request must adhere to **RFC-1867**: *"Form-based File Upload in HTML"* which defines `multipart/form-data`.

The multipart request is structured as follows:

- The first part of the multipart request must be the **XML request document**. Its field name must be `SMSBoxXMLRequest`. This XML request document is exactly the same as other XML API request documents (described previously) - it must be encoded in UTF-8 (or is immediately rejected). The DTD and examples for the currently supported commands can be found in the sections below.
- The subsequent parts in the multipart request are the **Mobile Internet content parts**, sent top-to-bottom and left-to-right; depending on the type of layout chosen, there must be only one, or can be multiple parts there. There is no restriction on the field names for these parts. `Content-Type` must be declared for all parts, **including a mandatory `charset`** for text parts (UTF-8 strongly advised).

Note: the file names must be pure alphanumeric ASCII (no accented chars, no special/punctuation characters except underscore, dash and dot) and contain no spaces.

7.3. Layout types

There are currently three types of page layout supported:

- The `ordered` layout type is used to present a top-to-bottom and left-to-right "traditional" page made of one or many content parts
- The `download` layout type is used to trigger a download; only one subsequent part must be used in that case, which must be a download content (description below)
- The `forward` layout type is used to trigger a forward (HTTP 302 Moved Temporarily) to a specific URL; only one subsequent part must be used in that case, which must be a link content (description below)

7.4. Subsequent parts

A Mobile Internet page of layout type `ordered` can contain text, image, link, download, action, auto-provisioning, and input items. Each item's data is encapsulated inside an XML document, comprising common data (layout information) and specific data (the image bytes for images, for example). The following sections describe the necessary encapsulation format.

7.4.1. Common data

Common data determines layout and is specific to layout type `ordered` only. It can be omitted for layout types `download` and `forward`. The two following parameters are supported:

7.4.1.1. Location

The `<location>` element indicates location information for the item. The valid values are:

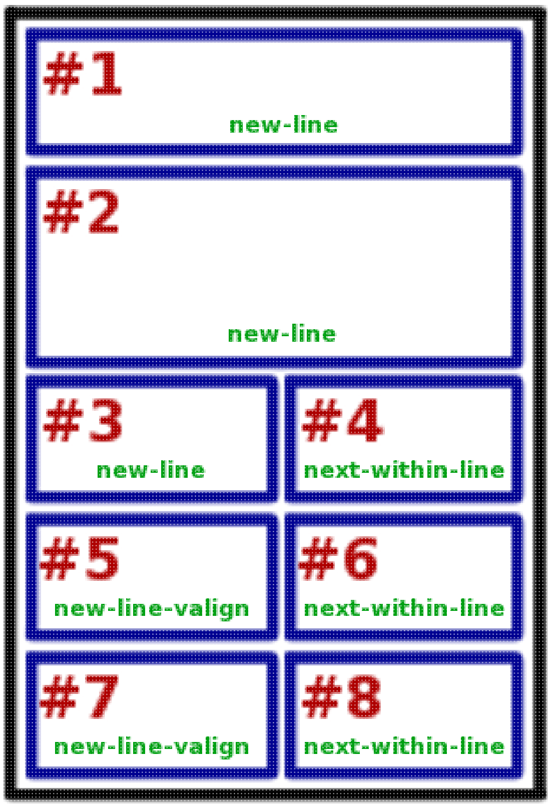
- `new-line`: indicate the item is beginning a new line
- `new-line-valign`: indicate the item is beginning a new line, vertically aligned on upper line; this value is not valid within first line
- `next-within-line`: indicate the item is continuing a line (this item will be rendered to the right of previous item); this value is not valid for the first item

7.4.1.2. Width

The `<width>` element indicates the relative size of the item within a line. The value must be an integer between 1 and 100 followed by the percent sign. This element is optional for items singles in a line and for items inside a vertically aligned line. The sum of widths within a line must be equal to 100%.

7.4.1.3. Complex layout example

The following diagram illustrates a complex layout example in a page of layout type `ordered`.



We want to have an image banner on top of the page, then some explanatory text, then three lines each one containing a link on the left for 50% of width of the page, and an image on the right for the 50% rest of width.

The image banner is transmitted as first item (#1) with location `new-line`; then explanatory text is the second item (#2) with location `new-line`; then one line: third item (#3) with location `new-line` and width `50%` and fourth item (#4) with location `next-within-line` and width `50%`; then the second aligned line: fifth item (#5) with location `new-line-valign` and sixth item (#6) with location `next-within-line`; then third aligned line similar to second aligned line.

7.4.2. Specific data

7.4.2.1. Text

Text parts may be sent with the `Content-Type: text/plain; charset=UTF-8` header. The data must be a well-formed XML document with root name `<text>`, the common data, the mandatory element `<content>` containing the actual text content, and the optional element `<fontSize>` indicating the wanted font size among three fixed values: `smaller`, `medium` or `larger`.

Within the actual text content, simple formatting may be used, as follows:

- keyletter b = bold - syntax: `[b text in bold]`
- keyletter i = italic - syntax: `[i text in italic]`
- keyletter u = underline - syntax: `[u underlined text]`
- keyletter # = color - syntax `[#red text in red]` or `[#FF0000 text in red]`

Formatting can be nested, for example `[b [i text in bold and italic] ]`, `[u underlined text [b now also bold [i now also italic] ] ]`, or `[b [#red bold red warning!] ]`.

Trace example of a text part:

```
Content-Disposition: form-data; name="part1"
Content-Type: text/plain; charset=UTF-8

<text>
  <location>new-line</location>
  <content>Hello Steve and [b happy birthday]! Good bye.</content>
  <fontSize>medium</fontSize>
</text>
```

7.4.2.2. Images

Image parts may be sent with a `Content-Type` depending on the image format used (`image/jpeg` or `image/gif` are best choices). The data must be a well-formed XML document with root name `<image>`, the common data, and the element `<content>` containing the base64-encoded image data. Refer to *Appendix C* for base64 transfer encoding information.

When uploading images, it is safe to stick to the well supported *jpeg* or *gif* formats. You should use a large enough size suitable for larger phone screens, so that a good quality is obtained for all phones, knowing that smsBox® will automatically downscale images according to phone screen size (400 pixels width adviced).

Trace example of an image part:

```
Content-Disposition: form-data; name="part2"
Content-Type: image/jpeg

<image>
  <location>new-line</location>
  <content>/9j/4AAQSkZJRgABAQIAHAAcAAD/2wBDAAgGBgcGBQgHBwcJCQgKDBQNDAsLDBkSEw8UHRofHh0a
HBwgJC4nICIsIxwcKDcpLDAxNDQ0Hyc5PTgyPC4zNDL/2wBDAQkJCQwLDBgNDRgyIRwhMjIyMjIy
     [...more base64 data...]
  </content>
</image>
```

7.4.2.3. Links

Link parts may be sent with the `Content-Type: application/cyote-link; charset=UTF-8` header. The data must be a well-formed XML document with root name `<link>`, the common data, element `<text>` containing the text of the link, and element `<url>` containing the relative or absolute URL where this link will point to.

Notice that the `<text>` element is meaningless for the link content transmitted in case of a page of layout type `forward`, yet the element is mandatory and should be transmitted as an empty element, `<text/>`.

Trace example of a link part:

```
Content-Disposition: form-data; name="part3"
Content-Type: application/cyote-link; charset=UTF-8
```

```
<link>
  <location>new-line</location>
  <text>click me!</text>
  <url>/HOME/NEWS</url>
</link>
```

7.4.2.4. Action items

Action item parts may be sent with the `Content-Type: application/cyote-action; charset=UTF-8` header. The data must be a well-formed XML document with root name `<action>`, the common data, element `<name>` containing the name of the action, and element `<service>` containing the service on which the action will be performed. The following action names are currently supported:

- **GET**: performs an SMS GET request on the service; sends the current content of the specified service as an SMS to the mobile user "clicking" this action item
- **GETMMS**: performs an MMS GET request on the service; sends the current MMS content of the specified service to the mobile user "clicking" this action item
- **SUBSCRIBE**: performs a subscription or unsubscription to the service; subscribes the mobile user "clicking" this action item to the specified service when not subscribed, else unsubscribes

Note: in a Mobile Internet page, an action item will appear as a clickable link. The text of the link is standardized in the smsBox® platform; only the support team can modify them to your particular needs, if any.

Trace example of an action item part:

```
Content-Disposition: form-data; name="part5"
Content-Type: application/cyote-action; charset=UTF-8

<action>
  <location>new-line</location>
  <name>GETMMS</name>
  <service>BECKHAM</service>
</action>
```

7.4.2.5. Auto-provisioning items

Auto-provisioning item parts may be sent with the `Content-Type: application/cyote-auto-provisioning; charset=UTF-8` header. The data must be a well-formed XML document with root name `<autoprovisioning>`, the common data, element `<service>` containing the name of the service from where to extract history messages, and element `<amount>` containing the amount of history messages to extract (only for SMS services).

An auto-provisioning item is a "bridge" between SMS or MMS information services and Mobile Internet pages: when a mobile user loads a Mobile Internet page containing such an item, the content of the page is dynamically created with the latest specified amount of messages from the SMS service, or with the current content from the MMS service (depending if the service is MMS or not).

Trace example of an auto-provisioning item part:

```
Content-Disposition: form-data; name="part6"
Content-Type: application/cyote-auto-provisioning; charset=UTF-8

<autoprovisioning>
  <location>new-line</location>
  <service>TOP NEWS</service>
  <amount>10</amount>
</autoprovisioning>
```

7.4.2.6. Downloads

Download parts may be sent with the `Content-Type: application/cyote-download; charset=UTF-8` header. The data must be a well-formed XML document with root name `<download>`, the common data, element `<name>` containing the name of the download; then either `<dynamic>true</dynamic>` and the element `<url>` containing the URL where to dynamically fetch the content at each end-user request, or `<dynamic>false</dynamic>` and the elements `<contentType>` containing the content-type of the download data and `<content>` containing the base64-encoded download data. Refer to *Appendix C* for base64 transfer encoding information.

Downloads inside `ordered` pages appear as links, the text of the link being the "name" of the download.

Reasonably recent mobile phones (2004 onwards) support some DRM (Digital Rights Management) features, among which the Forward Lock is supported by smsBox®. When a Mobile Internet page is marked with the Forward Lock flag, the receiver mobile phone will not allow that any download parts of the Mobile Internet page be forwarded to another mobile phone by MMS or saved on unsecure external storage. Note that if the mobile phone doesn't support the Forward-Lock feature, it will not be able to display/use the download at all. If you want the download parts of your Mobile Internet page to be forward-locked, you have to set the `forwardLock` empty element in your XML document request. Note that the Forward Lock feature may not be supported by smsBox® depending on configuration. Check with your support team.

Trace example of a non-dynamic download part:

```
Content-Disposition: form-data; name="part7"
Content-Type: application/cyote-download; charset=UTF-8

<download>
  <location>new-line</location>
  <name>Click to receive the official anthem of our club!</name>
  <dynamic>false</dynamic>
  <contentType>audio/mpeg</contentType>
  <content>/9j/4AAQSkZJRgABAQIAHAAcAAD/2wBDAAgGBgcGBQgHBwcJCQgKDBQNDAsLDBkSEw8UHRofHh0a
HBwgJC4nICIsIxwcKDcpLDAxNDQ0Hyc5PTgyPC4zNDL/2wBDAQkJCQwLDBgNDRgyIRwhMjIyMjIy
     [...more base64 data...]
  </content>
</download>
```

Trace example of a dynamic download part:

```
Content-Disposition: form-data; name="part7"
Content-Type: application/cyote-download; charset=UTF-8

<download>
  <location>new-line</location>
  <name>Click to receive the official anthem of our club!</name>
  <dynamic>true</dynamic>
  <url>http://www.partnersite.com/path1/path2?param1=value1&amp;param2=value2</url>
</download>
```

In case of dynamic download, for each end-user request, the specified URL is reverse-proxied by smsBox® (a redirect would break access control). The HTTP request as received by the web server contains (nearly) all original HTTP request headers from the end-user, so that per phone model rendering / content adaptation is possible. Additionally, the header `X-smsbox-real-remote-address` is added with the IP address from the end-user, and the header `X-smsbox-real-request-path` is added with the requested path in the Mobile Internet site. For a trace example of such a request, please refer to *Appendix D*.

7.4.2.7. Inputs

Input parts may be sent with the `Content-Type: application/cyote-input; charset=UTF-8` header. They create form data elements within the Mobile Internet page. The data must be a well-formed XML document with root name `<input>`, the common data, element `<name>` containing the name of the field, and element `<type>` indicating one of the following possible types:

- `text`: a simple text field
- `checkbox`: an option to check or not
- `radio`: a series of values shown as radio buttons
- `dropdown`: a series of values shown as a dropdown list
- `textarea`: a multiline text area
- `file`: a file field
- `submit`: a submission button

For `checkbox` and `submit`, the element `<value>` must be added to specify the text to display for the element. For `radio` and `dropdown`, the element `<value>` must be added to specify the series of values, as a semicolon-separated list. For `textarea`, the elements `<rows>` and `<cols>` must be added to specify the size of the text area.

Notice that once a mobile user fills up form data, and submits the form, the inputs are then transmitted with a specific notification document. Check the first part of this chapter. A page containing at least one input content will associate the destination (URL or email address) where to send the notification document to; you need to specify it with the `inputDestination` parameter in the command XML document.

Trace example of an input part:

```
Content-Disposition: form-data; name="part8"
Content-Type: application/cyote-input; charset=UTF-8

<input>
  <location>new-line</location>
  <name>firstname</name>
  <type>text</type>
</input>
```

7.5. Command Types

7.5.1. POSTWAP

The POSTWAP command creates, updates or deletes a specific sub-page in an existing Mobile Internet page enabled service.

7.5.1.1. POSTWAP Command Request DTD

```
<!ELEMENT SMSBoxXMLRequest (username,password,command,parameters)>
<!ELEMENT username         (#PCDATA)>
<!ELEMENT password         (#PCDATA)>
<!ELEMENT command          (#PCDATA)>
<!ELEMENT parameters       (service,path,create?,update?,delete?,layout?,billingType?,cost?,amount?,
                                forwardLock?,silent?,inputDestination?)>
<!ELEMENT service          (#PCDATA)>
<!ELEMENT path             (#PCDATA)>
<!ELEMENT create           EMPTY>
<!ELEMENT update           EMPTY>
<!ELEMENT delete           EMPTY>
<!ELEMENT layout           (#PCDATA)>
<!ELEMENT billingType      (#PCDATA)>
<!ELEMENT cost             (#PCDATA)>
<!ELEMENT amount           (#PCDATA)>
<!ELEMENT forwardLock      EMPTY>
<!ELEMENT silent           EMPTY>
<!ELEMENT inputDestination (#PCDATA)>
```

7.5.1.2. POSTWAP Command Reply DTD

```
<!ELEMENT SMSBoxXMLReply ((ok|error),command,requestUid)>
<!ELEMENT ok    EMPTY>
<!ELEMENT error EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command EMPTY>
  <!ATTLIST command name CDATA #FIXED "POSTWAP">
<!ELEMENT requestUid (#PCDATA)>
```

7.5.1.3. Request Specific Parameters

Parameter	Description
service	Name of the service to upload content to.
path	Path of the sub-page where to upload content to.
create	Optional parameter. If present, the page specified by service/path is created.
update	Optional parameter. If present, the page specified by service/path is updated.
delete	Optional parameter. If present, the page specified by service/path is deleted. In such a case, layout can be omitted and no subsequent parts are needed beyond the XML

	request document itself.
layout	Indicates the page layout wanted. May be `ordered`, `download` or `forward`.
billingType	Optional parameter. - If present, indicates the billing type of the Mobile Internet page; valid values: `free` for free access, `ntimes` for one billing every such amount of page-views, `nhours` for one billing every such amount of hours, `subscriptionbased` for limiting access to subscribers of a given Wap Push service - If absent, uses the default billing when creating a page, don't update current billing when updating a page.
cost	Optional parameter. If `billingType` is `ntimes` or `nhours`, needed to specify the cost (in cents). Useful only if Mobile Internet billing is available.
amount	Optional parameter. If `billingType` is `ntimes` or `nhours`, needed to specify the amount of times or hours.
billingService	Mandatory if `billingType` is set to `subscriptionbased`, specify the service to subscribers of which page access will be limited.
forwardLock	Optional parameter. If present, when creating or updating a Mobile Internet page, download elements of the Mobile Internet page will be marked forward-locked when later sent to mobile phones.
silent	Optional parameter. If present, when creating or updating a Mobile Internet page with `billingType` set to `subscriptionbased`, no Wap Push will be sent to the subscribers of the service.
inputDestination	Mandatory parameter when page layout is `ordered` and there is at least one input content within the page. Indicates the URL or email address to notify when form data will be sent from this page.

Any of the three elements: ***create***, ***update*** or ***delete*** must be present.

7.5.1.4. POSTWAP Command Example for an Ordered page

<u>1st element:</u>

◊ Name: `SMSBoxXMLRequest`
◊ Contents:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>POSTWAP</command>
  <parameters>
    <service>MNC WAPNEWS</service>
    <action>create</action>
    <path>/latest</path>
     <layout>ordered</layout>
  </parameters>
</SMSBoxXMLRequest>
```

<u>2nd element:</u>

◊ Content-Type: `image/jpeg`
◊ Contents:

```
<image>
  <location>new-line</location>
  <content>/9j/4AAQSkZJRgABAQIAHAAcAAD/2wBDAAgGBgcGBQgHBwcJCQgKDBQNDAsLDBkSEw8UHRofHh0a
HBwgJC4nICIsIxwcKDcpLDAxNDQ0Hyc5PTgyPC4zNDL/2wBDAQkJCQwLDBgNDRgyIRwhMjIyMjIy
     [...more base64 data...]
  </content>
</image>
```

<u>3rd element:</u>

◊ Content-Type: `text/plain; charset=UTF-8`
◊ Contents:

```
<text>
  <location>new-line</location>
  <content>[#blue This is the latest news from [b MNC:</content>
</text>
```

4th element:

◊ Content-Type: `application/cyote-auto-provisioning; charset=UTF-8`
◊ Contents:

```
<autoprovisioning>
  <location>new-line</location>
  <service>MNC SMSNEWS</service>
  <amount>5</amount>
</autoprovisioning>
```

5th element:

◊ Content-Type: `text/plain; charset=UTF-8`
◊ Contents:

```
<text>
  <location>new-line</location>
  <content>[b Note!] You can also receive these news by SMS:</content>
</text>
```

6th element:

◊ Content-Type: `application/cyote-action; charset=UTF-8`
◊ Contents:

```
<action>
  <location>new-line</location>
  <name>SUBSCRIBE</name>
  <service>MNC SMSNEWS</service>
</action>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="POSTWAP"/>
  <requestUid>xml9677328</requestUid>
</SMSBoxXMLReply>
```

7.5.1.5. POSTWAP Command Example for a Download page

1st element:

◊ Name: `SMSBoxXMLRequest`
◊ Contents:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>POSTWAP</command>
  <parameters>
    <service>MNC WAPNEWS</service>
    <action>create</action>
    <path>/latest</path>
    <layout>download</layout>
  </parameters>
</SMSBoxXMLRequest>
```

2nd element:

◊ Content-Type: `audio/mpeg`
◊ Contents:

```
<binary>
  <content>/9j/4AAQSkZJRgABAQIAHAAcAAD/2wBDAAgGBgcGBQgHBwcJCQgKDBQNDAsLDBkSEw8UHRofHh0a
HBwgJC4nICIsIxwcKDcpLDAxNDQ0Hyc5PTgyPC4zNDL/2wBDAQkJCQwLDBgNDRgyIRwhMjIyMjIy
```

```
        [...more base64 data...]
    </content>
  </binary>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="POSTWAP"/>
  <requestUid>xml9677328</requestUid>
</SMSBoxXMLReply>
```

7.5.1.6. POSTWAP Command Example for a Forward page

1st element:

◊ Name: `SMSBoxXMLRequest`
◊ Contents:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>POSTWAP</command>
  <parameters>
    <service>MNC WAPNEWS</service>
    <action>create</action>
    <path>/latest</path>
    <layout>forward</layout>
  </parameters>
</SMSBoxXMLRequest>
```

2nd element:

◊ Content-Type: `Content-Type: application/cyote-link; charset=UTF-8`
◊ Contents:

```
<link>
  <text/>
  <url>http://m.example.com/</url>
</link>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="POSTWAP"/>
  <requestUid>xml9677328</requestUid>
</SMSBoxXMLReply>
```

8. Command XML API Errors

8.1. Unknown Reply

The UNKNOWN command is never used directly by an external application, but is invoked if the command parameter in the request XML document does not match one of the existing command names.

8.1.0.1. Unknown Reply DTD

```
<!ELEMENT SMSBoxXMLReply (error,command,requestUid)>
<!ELEMENT error   EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command EMPTY>
  <!ATTLIST command name CDATA #FIXED "UNKNOWN">
<!ELEMENT requestUid (#PCDATA)>
```

8.1.0.2. Unknown Reply Example - Bad/Invalid Request

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>myuser</username>
  <password>mypass</password>
  <command>BLAHBLAH</command>
  <parameters>
    <service>LOVE</service>
  </parameters>
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <error type="unknown"/>
  <command name="UNKNOWN"/>
  <requestUid>xml9677329</requestUid>
</SMSBoxXMLReply>
```

8.2. ParseError Reply

The PARSEERROR command is never used directly by an external application, but is invoked if there was an error parsing the XML request document, including when the byte contents of the document appears not to be valid UTF-8.

8.2.0.1. ParseError Reply DTD

```
<!ELEMENT SMSBoxXMLReply (error,command,requestUid)>
<!ELEMENT error EMPTY>
  <!ATTLIST error type CDATA #REQUIRED>
<!ELEMENT command EMPTY>
  <!ATTLIST command name CDATA #FIXED "PARSEERROR">
<!ELEMENT requestUid (#PCDATA)>
```

8.2.0.2. ParseError Reply Example - Incorrect XML Request (incorrect opening root tag)

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest
  <username>myuser</username>
  <password>mypass</password>
  <command>SUBSCRIBERS</command>
  <parameters>
    <service>FCBTOTOMAT</service>
  </parameters>
```

```
</SMSBoxXMLRequest>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <error type="xmlparseerror"/>
  <command name="PARSEERROR"/>
  <requestUid>xml9677330</requestUid>
</SMSBoxXMLReply>
```

8.2.0.3. ParseError Reply Example - Non UTF-8 contents

Reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <error type="wrongutf8"/>
  <command name="PARSEERROR"/>
  <requestUid>xml9677331</requestUid>
</SMSBoxXMLReply>
```

9. SSL Support

The HTTP XML API offers the possibility to protect the data exchanged with SSL; when using normal HTTP, eavesdropping is possible by third parties or attackers; opposedly, when using SSL, the data is transmitted confidentially. All the commands and parameters are identical in SSL and normal mode.

In case of using SSL for sending Commands (*to* smsBox®), the URL to use for HTTP POST is different. Please contact the support. Notice that the smsBox® SSL server certificate has to be configured to be trusted on your side and will be transmitted to you separately.

In case of using SSL for receiving Notifications (*from* smsBox®), the partner must communicate the SSL server certificate in order for smsBox® to be able to trust it. Please contact the support.

It is possible to use SSL client authentication (2-way SSL) for increased security, both for Command API and Notifications. The setup involves a bit more work. Please contact the support.

10. General Remarks

10.1. HTTP POST Requests

10.1.1. Notifications (interactions from smsBox®)

Notifications are sent in the form of an XML document as the body of an HTTP POST request to the URL configured for a particular external application. In case of command notifications, the external application must respond to the notification **in the body of the response** (the response cannot be made in a separate, new connection). The response to the request is used either to simply acknowledge the notification or to submit a response/billing SMS message to the end-user. In case of sent message notifications, the external application's response is ignored by the smsBox® platform.

The external application must be able to receive HTTP POST requests in order to use smsBox® Notifications functionality.

10.1.2. Commands (interactions to smsBox®)

The **Command XML API** expects the XML request document in the body of an HTTP POST request. The external application must be able to send the XML request document in an HTTP POST request as well as receive and extract the XML reply document from the response. The information it contains is extracted by parsing the XML document.

10.2. HTTP POST body format

The body of HTTP POST requests must contain only the XML document (Notification or Command API request). The XML document **must not** be preceded by a variable demarcation (such as when sending variables as part of an HTTP GET request or sending a web form in HTTP POST), and **may not** be URL-encoded.

Incorrect:

```
POST /Pro/sms/xml HTTP/1.1
HOST: pro.smsbox.ch
CONTENT-LENGTH: 132
Content-type: text/xml; charset=UTF-8

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

In the excerpt above, **myxmldoc=** is wrong and need to be removed.

Incorrect:

```
POST /Pro/sms/xml HTTP/1.1
HOST: pro.smsbox.ch
CONTENT-LENGTH: 132
Content-type: text/xml; charset=UTF-8

%3C%3Fxml%20version%3D%221.0%22%20encoding%3D%22UTF-8%22%20%3F%3E....
```

In the excerpt above, URL-encoding is wrong.

Correct:

```
POST /Pro/sms/xml HTTP/1.1
HOST: pro.smsbox.ch
CONTENT-LENGTH: 123
Content-type: text/xml; charset=UTF-8

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

Note: there must be exactly **one** empty line between the HTTP header and the beginning of the XML document.

10.3. Command Notification Timeout

External applications are required to completely send their response to Command Notification HTTP POST requests within **20 seconds** of receiving the command notification. After that delay, the network connection will be automatically closed by smsBox®. Notifications are sent only once.

10.4. UTF-8 encoding (XML API)

All XML documents sent by smsBox® (Notifications and Command API responses) are encoded using the UTF-8 character set. All XML documents sent from external applications must also be encoded using the **UTF-8** character set.

Note: setting the encoding marker (encoding="UTF-8") is not enough: the content must be intrinsically encoded as indicated (simply marking "UTF-8" will not work if the content is in another encoding). Please consult the documentation of your operating system, programming language and database software to learn more about unicode and UTF-8, and how to make sure you are able to correctly send and receive UTF-8 encoded documents.

10.5. URL to use (XML API)

For Command API usage, use URLs built in the following way:
`http://<hostname>:<port>/<instance>/sms/xml`.

Example (939 in Switzerland): `http://pro.smsbox.ch:8047/Pro/sms/xml`.

10.5.1. If using HTTP GET

When using HTTP GET, the trailing `/xml` must be substituted by `/rest`, which creates such URLs:
`http://<hostname>:<port>/<instance>/sms/rest`.

Example (939 in Switzerland): `http://pro.smsbox.ch:8047/Pro/sms/rest`.

If the HTTP client you use supports username/password directly in URL, that would make them:
`http://<hostname>:<password>@<machine>:<port>/<instance>/sms/rest`.

Example (939 in Switzerland): `http://username:password@pro.smsbox.ch:8047/Pro/sms/rest`.

The proper ending part of URL to select the command, and the proper GET parameters, must be added too:
`http://<hostname>:<password>@<machine>:<port>/<instance>/sms/rest<commandpath>?<parameters>`.

Example (939 in Switzerland, USERINFO command on +41760000000):
`http://username:password@pro.smsbox.ch:8047/Pro/sms/rest/user/info?msisdn=%2B41760000000`.

10.6. IP address of your machine

Because of the security policy of the smsBox® platform, we need to configure specifically the firewall to accept requests from your machine. It means you have to provide us with one or two IP addresses for the machine(s) that will send the HTTP commands.

Note 1: if your local network accesses the Internet through a single machine which performs address translation, an HTTP request from one machine will appear as coming from the IP address of the machine that is connected to the Internet. Internal IP addresses can be in three forms: 10.x.x.x, 172.16.x.x - 172.31.x.x, or 192.168.x.x.

Therefore, the IP address you transmit to us should not begin with 10, 172.16 to 172.31, or 192.168. Please contact your network administrator if you have any doubts.

Note 2: if you access the Internet with dial-up or with ADSL, chances are you don't have a fixed IP address, as a different one is attributed to you each time you connect.

To be able to use the smsBox® API, you need a machine with a fixed IP address.

10.7. IP address of our server

Depending on the level of security **your system uses**, you might have to enter the smsBox® platform server IP address in your firewall. There are two cases when you need to do this:

- When using the API to send a command, some firewalls will not let the HTTP response come back to your system.
- When receiving notifications on one of your machines.

The necessary IP addresse(s) will be provided to you in those cases.

10.8. HTTP Request visibility

If you want to trigger some command of the smsBox® API in a public web page, you must take care that the HTTP request is not part of the page that the users view in their browsers. It is especially forbidden to put the HTTP request as a link or under a button. Even if the request is not visible in the page, any user can do a "View Source" in his browser and see the content of the page. This would show your confidential user/password information.

It is absolutely essential to put the code that actually executes the HTTP request in a different component on your server.

11. Appendix A: MMS Implementation Notes

This section explains how the MMS functions work on the smsBox® platform.

11.1. MO/MT MMS

The smsBox® platform is able to receive MO MMS messages from operators that support this functionality. Currently, only Swisscom and Sunrise in Switzerland support this functionality.

MT MMS can be ordered using SMS or MMS. The typical command to request an MMS is `MMS` *`SERVICE`*. For example, to request an MMS from service NICE, send an SMS (or an MMS) with content `MMS NICE`.

The MMS content of a service can be updated by three methods:

- using the MMS wizard in the web administration application;
- using the XML API `POSTMMS`;
- using an MO MMS with the first text part containing the keyword POST, the service name, and the service's password (if your operator supports MO MMS!); the remaining text parts or photos being used to replace the current content of the MMS.

It is possible to subscribe to an MMS service just like for a text service, using the command `START` *`SERVICE`*.

11.2. Billing

The billing is done normally while the MMS is sent for most operators. Orange Switzerland and Digitel Philippines don't support MMS billing currently and a separate SMS has to be sent for billing; the MMS delivery itself is not billed in this case.

It means that, for Orange Switzerland or Digitel Philippines end-users, there will always be an SMS accompanying the MMS for the charged amount. The SMS will arrive first as it is faster.

11.3. MMS Content

There are several issues related to MMS content that require some explaining. Please check the **Open Mobile Alliance's** *MMS Conformance Document* for full details on current industry guidelines.

The basic recommendations are:

- Maximum size for the whole MMS: **100kB** to **300kB** - make sure to test towards all the targeted operators because they usually enforce different maximum sizes!
- Maximum image size: **320 x 240**
- Image formats: **Baseline JPEG** (JFIF exchange), **GIF87a** & **GIF89a** (animated GIF).
- Audio formats: **AMR**, **MP3**.
- Video formats: **3GPP**.
- Text encoding: **UTF-8**

Some devices will of course support larger images, other formats, bigger multi-element MMS. The key point is that an MMS complying with the above limitations will have the widest compatibility.

We can further propose the following guidelines to choose the best format:

- **GIF** is best for drawings, cartoons and images where there are large areas of the same color (this is where the LZ compression will work best). It is the only image format supporting animations.
- **JPEG** is ideal for generic/photo images.

11.4. Layout

In an MMS, usually a SMIL document is transmitted alongside the content; it describes the layout of the MMS, made of one to many slides, each slide containing one to many content elements.

By default, smsBox® will generate a SMIL document describing one slide per element. It is possible to control the layout by sending an MMS Slides Definition file. **Warning:** more than two elements per slide, or more than one element of the same type per slide (e.g. two texts on a single slide, or two images on a single slide, etc), are not adviced because on most phones this layout will not be properly supported/displayed.

11.5. Device Configuration

Devices have to be configured for MMS before they can receive them. Operator and device manufacturers often have on their website a page where the end-user can type his/her MSISDN and receive the configuration. Sending an MMS to a device without the proper configuration will fail.

11.6. Device Considerations

The appearance of the images will largely depend on the device's displaying capacities.

Screen sizes and color support can vary a lot. Smartphones and PDAs with MMS capacity usually have large screens. It is adviced to test your MMS by sending it towards multiple phone types/manufacturers.

Some phones will resize images and others will allow viewing the full image by moving on it.

11.7. Distribution speed and throughput

MMS are slower than SMS. It is due first to the size of the message, which is 10-100 times larger than an SMS and also to the processing overhead at the operator's MMSC. We have seen times from 20 seconds to a couple of minutes.

The operators do not currently allow a high throughput and we currently measure speeds ranging from 1/10 to 2 MMS per seconds.

11.8. Encoding issues

The standard for content encoding with MMS is UTF-8. This is why the XML API and the smsBox® platform are using UTF-8 for content encoding. UTF-8 supports a wide variety of characters and symbols and smsBox® will receive and transmit them correctly, even non-latin characters. The real issue is that end-user's device will determine how to display the characters.

For most mobile phones in Switzerland, it is recommended to use only the GSM default character set (ETSI GSM 03.38).

Note: as a general rule, if you can type a character on the target phone, you will be able to send that character via MMS.

12. Appendix B: Code examples

Several examples of code for notification reception and command generation can be provided upon request. These examples are provided "as is", without any guarantee or support.

The available examples are available in several different programming languages and should probably fit one of your favourites.

Please contact the support group to obtain these examples: smsboxpro-support@mnc.ch

13. Appendix C: References

Follows a list of places to look for information about specific technical matters related to this document:

- **Base64 content transfer encoding**: Multipurpose Internet Mail Extensions Part One: Format of Internet Message Bodies (RFC 2045), section 6.8

  http://www.ietf.org/rfc/rfc2045.txt

- **DCS (Data Coding Scheme)**: 3GPP TS 23.038: Alphabets and language-specific information.

  http://www.3gpp.org/ftp/Specs/html-info/23038.htm

- **EMS (Enhanced Message Service)**: 3GPP TS 23.040: Technical realization of the Short Message Service (SMS).

  http://www.3gpp.org/ftp/Specs/html-info/23040.htm

- **HTTP POST**: Hypertext Transfer Protocol - HTTP/1.1.

  http://www.w3.org/Protocols/rfc2616/rfc2616.html

- **MMS**: Open Mobile Alliance's MMS Conformance Document

- **Nokia Smart Messaging**: Smart Messaging Specifications, Nokia Mobile Phones Ltd.

  http://forum.nokia.com/

- **UCS-2**: ISO/IEC 10646-1 International Standard - Information technology - Universal Multiple-Octet Coded Character Set (UCS)

- **UTF-8**: Unicode standard, version 4.0.

  http://www.unicode.org/versions/Unicode4.0.0/ http://www.unicode.org/faq/utf_bom.html

- **XML**: Extensible Markup Language (XML), W3C consortium.

  http://www.w3.org/XML/

- **Forward Lock**: Part of OMA Digital Rights Management V1.0.

  http://www.openmobilealliance.org/release_program/drm_v1_0.html

14. Appendix D: HTTP traces

To help trouble-shooting problems, you can find below example traces of various HTTP requests, either *from* or *to* smsBox®.

Note: these traces should help you debug your software when connecting to the smsBox® platform, but in general you won't have to build such traces for yourself down to each and every byte. We strongly advice to use an HTTP library that will do that for you, handling the intrinsics of the HTTP protocol, especially when it comes to HTTP POST in multipart. Also, we have written working code examples, they should be a good inspiration source for you, and possibly a starting point for your software; see *Appendix B*.

14.1. A simple XML API command

Follows the trace of a correct SEND command performed with the XML API. You will notice the HTTP command in the first line, then 5 lines of HTTP headers, then an empty line, then the XML API document being sent to smsBox®.

```
POST /SMSBox/sms/xml HTTP/1.1
Connection: close
Host: www.your-server-domain.com
User-Agent: libwww-perl/5.79
Content-Type: text/xml; charset=UTF-8
Content-Length: 307

<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>user</username>
  <password>pass</password>
  <command>SEND</command>
  <parameters>
    <receiver>+41761234567</receiver>
    <service>MNC</service>
    <text>Hello world.</text>
    <cost>20</cost>
  </parameters>
</SMSBoxXMLRequest>
```

14.2. A multipart XML API command accessing MMS functions

Here's the trace of a correct POSTMMS command performed with the XML API. This is a multipart request, so it is slightly more complex than the previous trace. You will notice the HTTP command and headers first, with the `Content-Type` set to `multipart/form-data` as expected, specifying a boundary used to delimit parts. Then, the XML API document being sent to smsBox®, followed by three data parts; each part is separated by the boundary, and has itself some headers to specify name, length and type of data.

```
POST /SMSBox/sms/xml HTTP/1.1
Connection: close
Host: www.your-server-domain.com
User-Agent: libwww-perl/5.79
Content-Length: 1741
Content-Type: multipart/form-data; boundary=xYzZY

--xYzZY
Content-Disposition: form-data; name="SMSBoxXMLRequest"
Content-Type: text/xml; charset=UTF-8

<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>user</username>
  <password>pass</password>
  <command>POSTMMS</command>
  <parameters>
    <service>MNC</service>
  </parameters>
</SMSBoxXMLRequest>

--xYzZY
Content-Disposition: form-data; name="part1"
Content-Type: text/plain; charset=UTF-8
```

```
Hello! Wanted to send you this funny penguin image..
--xYzZY
Content-Disposition: form-data; name="part2"; filename="fb.jpg"
Content-Length: 982
Content-Type: image/jpeg

     [...binary data...]

--xYzZY
Content-Disposition: form-data; name="part3"
Content-Type: text/plain; charset=UTF-8

Hope to send you soon,
Your best friend.
--xYzZY--
```

14.2.1. Using an MMS Slides Definition file

Here's the trace of the same POSTMMS command with an MMS Slides Definition file. The resulting layout will be a first slide showing the text and the image during 8 seconds, and a second slide showing the second text during 3 seconds.

```
POST /SMSBox/sms/xml HTTP/1.1
Connection: close
Host: www.your-server-domain.com
User-Agent: libwww-perl/5.79
Content-Length: 1741
Content-Type: multipart/form-data; boundary=xYzZY

--xYzZY
Content-Disposition: form-data; name="SMSBoxXMLRequest"
Content-Type: text/xml; charset=UTF-8

<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>user</username>
  <password>pass</password>
  <command>POSTMMS</command>
  <parameters>
    <service>MNC</service>
  </parameters>
</SMSBoxXMLRequest>

--xYzZY
Content-Disposition: form-data; name="part1"
Content-Type: text/plain; charset=UTF-8

Hello! Wanted to send you this funny penguin image..
--xYzZY
Content-Disposition: form-data; name="part2"; filename="fb.jpg"
Content-Length: 982
Content-Type: image/jpeg

     [...binary data...]

--xYzZY
Content-Disposition: form-data; name="part3"
Content-Type: text/plain; charset=UTF-8

Hope to send you soon,
Your best friend.

--xYzZY
Content-Disposition: form-data; name="__cyote_slides"
Content-Type: application/cyote-mms-slides-definition; charset=UTF-8

<?xml version="1.0" encoding="UTF-8" ?>
<slides>
  <slide duration="8">
    <element cid="part1"/>
    <element cid="part2"/>
  </slide>
  <slide duration="3">
    <element cid="part3"/>
  </slide>
```

```
</slides>
--xYzZY--
```

14.3. MMS Command Notification XML API - FORWARD (Request/Response)

This is a trace of the HTTP POST `multipart/form-data` request sent by the smsBox® platform to a registered external application, as a result of the MMS Forward Command Notification.

```
POST /mms-forward-receiver HTTP/1.1
connection: close
user-agent: Jakarta Commons-HttpClient/3.0-rc2
host: www.your-server-domain.com
content-length: 3568
content-type: multipart/form-data; boundary=hxq5eJnfdh6jhNEmdeTZtVYgqNwkgdVKGXPXt

--hxq5eJnfdh6jhNEmdeTZtVYgqNwkgdVKGXPXt
Content-Disposition: form-data; name="Notification"
Content-Type: text/xml; charset=UTF-8
Content-Transfer-Encoding: 8bit

<?xml version="1.0" encoding="UTF-8" ?>
<Notification>
  <instance>default</instance>
  <sender>+41793253780</sender>
  <operator>swisscom</operator>
  <service>DUDE</service>
  <language>en</language>
  <command>FORWARDMMS</command>
  <requestUid>mms9676204</requestUid>
  <parameters>
    <text>dude Welcome to my world! 123456789</text>
    <message>dude Welcome to my world! 123456789</message>
  </parameters>
</Notification>

--hxq5eJnfdh6jhNEmdeTZtVYgqNwkgdVKGXPXt
Content-Disposition: form-data; name="296Zhx"; filename="test.jpeg"
Content-Type: image/jpeg; charset=ISO-8859-1
Content-Transfer-Encoding: binary

     [...binary data...]

--hxq5eJnfdh6jhNEmdeTZtVYgqNwkgdVKGXPXt
Content-Disposition: form-data; name="zdLWKU"
Content-Type: text/plain; charset=utf-8
Content-Transfer-Encoding: 8bit

dude Welcome to my world! 123456789
--hxq5eJnfdh6jhNEmdeTZtVYgqNwkgdVKGXPXt--
```

This is an example of the response returned by the external application.

```
content-type: multipart/mixed; type="text/xml"; boundary="----=_Part_3403_25243398.1111675861055"

------=_Part_3403_25243398.1111675861055
Content-Type: text/xml; charset=UTF-8
Content-Disposition: form-data; Name="SMSBoxXMLResponse"; Filename="smsBox®XMLResponse.xml"

<?xml version="1.0" encoding="UTF-8"?>
<NotificationReply>
  <message>
    <text>Thanks for ordering the service, friend!</text>
    <cost>50</cost>
  </message>
</NotificationReply>
------=_Part_3403_25243398.1111675861055
Content-Type: image/jpeg
Content-Transfer-Encoding: base64
Content-Disposition: Attachment; Filename="hello.jpg"; Name="HelloImg"

     [...binary data...]
```

```
------=_Part_3403_25243398.1111675861055
Content-Type: text/plain; charset=utf-8
Content-Disposition: Attachment; Filename="IGdfF.txt"; Name="IGdfF.txt"

This is a test image. :-)
------=_Part_3403_25243398.1111675861055--
```

14.4. A multipart XML API command accessing Mobile Internet functions

Here's the trace of a correct POSTWAP command performed with the XML API. This is a multipart request, so it is slightly more complex than the regular trace. You will notice the HTTP command and headers first, with the `Content-Type` set to `multipart/form-data` as expected, specifying a boundary used to delimit parts. Then, the XML API document being sent to smsBox®, followed by three data parts; each part is separated by the boundary, and has itself some headers to specify name, length and type of data.

```
POST /SMSBox/sms/xml HTTP/1.1
User-Agent: Jakarta Commons-HttpClient/3.0-rc2
Host: www.your-server-domain.com
Content-Length: 36708
Content-Type: multipart/form-data; boundary=QO_-nZaC9RNcFHH4UA9SRxBwjDvVhH3

--QO_-nZaC9RNcFHH4UA9SRxBwjDvVhH3
Content-Disposition: form-data; name="SMSBoxXMLRequest"
Content-Type: text/xml; charset=UTF-8
Content-Transfer-Encoding: 8bit

<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLRequest>
  <username>user</username>
  <password>pass</password>
  <command>POSTWAP</command>
  <parameters>
    <service>MNC WAPNEWS</service>
    <path>/latest</path>
    <layout>ordered</layout>
  </parameters>
</SMSBoxXMLRequest>

--QO_-nZaC9RNcFHH4UA9SRxBwjDvVhH3
Content-Disposition: form-data; name="part2"
Content-Type: image/jpeg

<image>
  <content>
    /9j/4AAQSkZJRgABAQIAHAAcAAD/2wBDAAgGBgcGBQgHBwcJCQgKDBQNDAsLDBkSEw8UHRofHh0a
    [...more base64 data...]
  </content>
  <location>new-line</location>
  <width>100%</width>
</image>
--QO_-nZaC9RNcFHH4UA9SRxBwjDvVhH3
Content-Disposition: form-data; name="part3"
Content-Type: application/cyote-action; charset=UTF-8

<action>
  <name>GETMMS</name>
  <service>FCBLOGO</service>
  <location>new-line</location>
  <width>100%</width>
</action>
--QO_-nZaC9RNcFHH4UA9SRxBwjDvVhH3
Content-Disposition: form-data; name="part4.mp3"
Content-Type: application/cyote-download; charset=UTF-8

<download>
  <name>download my music!</name>
  <contentType>audio/mpeg</contentType>
  <content>
    //tQxAAAAAAAAAAAAAAAAAAAAAASW5mbwAAAA8AAAB3AABh9AAEBggKDBETFRcZHiAiJCYrLS8x
    [...more base64 data...]
  </content>
  <location>new-line</location>
  <width>100%</width>
</download>
```

```
--QO_-nZaC9RNcFHH4UA9SRxBwjDvVhH3--
```

14.5. Fetching a dynamic download content for a Mobile Internet site

Follows the trace of an HTTP GET request used to fetch some dynamic download content *from* smsbox. Refer to the **Mobile Internet browsing Functions** chapter for details about dynamic download content.

In this trace, the URL associated with the dynamic download content is `http://www.partnersite.com/path1/path2?param1=value1&param2=value2`, and the phone used is a Sony Ericsson K800i.

You can notice in the trace that the headers sent by the phone are sent over by smsBox® (in particular, `user-agent` and `x-wap-profile` are interesting for phone identification), and the headers `X-smsbox-real-remote-address` is added with the source IP address of the phone and `X-smsbox-real-request-path` with the path within the Wap site.

```
GET /path1/path2?param1=value1&param2=value2 HTTP/1.1
accept: multipart/mixed, application/vnd.wap.multipart.mixed, application/vnd.wap.xhtml+xml (...)
accept-charset: utf-8, utf-16, iso-8859-1, iso-10646-ucs-2, Shift_JIS, Big5
accept-language: en
x-wap-profile: "http://wap.sonyericsson.com/UAprof/K800iR101.xml"
user-agent: SonyEricssonK800i/R1CB Browser/NetFront/3.3 Profile/MIDP-2.0 Configuration/CLDC-1.1
accept-encoding: deflate, gzip
X-smsbox-real-remote-address: 1.2.3.4
X-smsbox-real-request-path: /HOME/news
Connection: close
Host: www.partnersite.com
```

As response, the server should send an HTTP 200 status code, and the content to be then sent to the phone. A proper `Content-Type` should not be forgotten in order for the phone to properly decide how to handle the content.

15. Appendix E: Binary SMS interface

The following explains the various types of binary content supported by smsBox®. To transmit binary content, you must use the XML API commands `SENDBINARY`, `WEBSENDBINARY` or `POSTBINARY` - please refer to the appropriate chapters for details about these commands.

Pre-requesites: be aware that you cannot send images or sound using "regular" format of computers (`jpeg` image, `midi` music, etc) to smsBox® binary SMS interface. You have to use format of data defined by Nokia (for Nokia Smart Messaging) or 3GPP (for EMS). You need to have basic understanding of these formats, and always refer to official documentation for them first.

How to read this chapter: each type of binary content is described globally, then all classes of each type are explained with first the description of what data format is expected, then an example of data. Within examples, colors are used to help functionally understanding the stream of data; everything in the `<hex>` element is what the client is supposed to send to smsBox® within the desired XML API command.

Compatibility warning: EMS and Nokia Smart Messaging are two competing formats to transmit roughly equivalent data. EMS is more or less the "standard" for binary SMS messages; it is well supported by Sony Ericsson phones, not supported by Nokia phones, often problematic with other manufacturers phones. Nokia Smart Messaging is Nokia-specific binary SMS messages; most popular of them are Operator Logo, Caller Line Identification Logo, Picture Message, Ringtone; well supported by Nokia and Sony Ericsson phones, often refused by other manufacturers phones.

15.1. Nokia content (Nokia Smart Messaging)

For Nokia Smart Messaging, `class` value can be `oplogo`, `clilogo`, `pictmsg`, or `ringtone`.

Data sent is made of content data only (no UDH header containing Nokia ports - the actual UDH sent is inferred from the `class` value). Data is encoded in hexadecimal (uppercase).

15.1.1. oplogo, clilogo

Frame sent to smsBox®:	OTA format (starts typically by `00480E01` when a 72x14 black and white image)

Example of such a frame:	`<class>`***`oplogo`***`</class>`
	`<hex>00480E010000010000000000000000000200000000000000000040`
	`0000000000000000C77381E0F00000000387FFC3F1FC0000000707FF`
	`E7FBFC0000000F07BDEFFF8C0000000E0F39FE7F000000000E0F39FC`
	`7F000000000F0E39BCF700000000078E7B38F79800000007CE7E38E7`
	`F800000003EE7C38E3F000000000FFF838E1E0000</hex>`
	...

Explanation of example:	`00`: OTA bitmap header, octet n. 1: infofield `48`: OTA bitmap header, octet n. 2: width (72 pixels) `0E`: OTA bitmap header, octet n. 3: height (14 pixels) `01`: OTA bitmap header, octet n. 4: color depth (2 colors - black and white) `00000100...8E1E0000`: OTA bitmap data, please refer to Nokia Smart Messaging specifications

15.1.2. pictmsg

Frame sent to smsBox®:	Nokia Smart Messaging format, Multipart (starts with the version octet, typically `30` for version 0)

Example of such a frame:	`<class>`***`pictmsg`***`</class>`
	`<hex>3000000454657374020070000480C0100000100000000000000000`
	`002000000000000000000400000000000000000C77381E0F000000003`
	`87FFC3F1FC0000000707FFE7FBFC0000000F07BDEFFF8C0000000E0F`
	`39FE7F000000000E0F39FC7F000000000F0E39BCF700000000078E7B`
	`38F79800000007CE7E38E7F8000</hex>`
	...

Explanation of example:	30: Version (version "0") 00: ISO-8859-1 Item 0004: Item length (4 octets) 54657374: Item data ("Test") 02: OTA bitmap Item 0070: Item length (112 octets) 00: OTA bitmap header, octet n. 1: infofield 48: OTA bitmap header, octet n. 2: width (72 pixels) 0C: OTA bitmap header, octet n. 3: height (12 pixels) 01: OTA bitmap header, octet n. 4: color depth (2 colors - black and white) 00000100...8E7F8000: OTA bitmap data, please refer to Nokia Smart Messaging specifications

15.1.3. ringtone

Frame sent to smsBox®:	Nokia Smart Messaging format (starts with a `command-length` octet)

Example of such a frame:	`<class>`***`ringtone`***`</class>`
	`<hex>024A3A4D554D040400871CCAEA22849A61641061A4106288A12`
	`418A2C8490628D20C30B212418690418710418A2284906288A08B0D2`
	`12418A308490628B212418A2A84906289A0AB0B212418A2C84906288`
	`A12418690590000</hex>`
	...

Explanation of example:	02: Command length (2 commands present) 4A3A4D55...: Rest of ringing tone data, please refer to Nokia Smart Messaging specifications

15.2. EMS

For EMS, `class` value can be `varpict` or `usersound`.

Data sent is made of UDH data representing content only. No UDHL octet, no UDH headers for concatenation, for EMS data type or position in message (these are inferred by smsBox®). Data is encoded in hexadecimal (uppercase).

15.2.1. varpict (EMS Variable Picture, IEI 0x12)

Frame sent to smsBox®:	Width/8, Height, and PDU (octets 2, 3, and 4-n of IED)

Example of such a frame:	`<class>`***`varpict`***`</class>`
	`<hex>090E000001000000000000000002000000000000000000400000`
	`000000000000C77381E0F00000000387FFC3F1FC0000000707FFE7FB`
	`FC0000000F07BDEFFF8C0000000E0F39FE7F000000000E0F39FC7F00`
	`0000000F0E39BCF700000000078E7B38F79800000007CE7E38E7F800`
	`000003EE7C38E3F000000000FFF838E1E0000</hex>`
	...

Explanation of example:	09: Width/8 (width is 72 pixels) 0E: Height (14 pixels) 00000100...8E1E0000: bitmap data, please refer to 3GPP specifications (23040 document)

FYI, corresponding UDH sent by smsBox® to end-user (front red part is added by smsBox® to build a valid UDH):	`831281000900E0000010000000000000000020000000000000000000400`
	`000000000000000C77381E0F00000000387FFC3F1FC0000000707FFE`
	`7FBFC0000000F07BDEFFF8C0000000E0F39FE7F000000000E0F39FC7`
	`F000000000F0E39BCF700000000078E7B38F79800000007CE7E38E7F`
	`800000003EE7C38E3F000000000FFF838E1E0000`

15.2.2. usersound (EMS User Defined Sound, IEI 0x0C)

Frame sent to smsBox®:	PDU iMelody (octets 2-n of IED)

Example of such a frame:	`<class>`***`usersound`***`</class>`
	`<hex>2A352364322364322A342361312364322364332366322361326133236132 2A352364322A34236131703223643323663223613361322361322361326131703223673223673323673223673223663223643123643223633223643323643 22A347033236432236332236433236 4300D0A</hex>`
	...

Explanation of example:	`2A352364...64300D0A`: iMelody, please refer to 3GPP specifications (23040 document)

FYI, corresponding UDH sent by smsBox® to end-user (front red part is added by smsBox® to build a valid UDH):	`720C70002A3523643223643 22A342361312364322364332366322361`
	`3261332361322A352364322A342361317032236433236632236133 61`
	`322361322361326131703223673223673323673223673223663223 64`
	`312364322363322364332364322A3470332364322363322364332364`
	`300D0A`

15.3. Generic

For "generic" type, `class` value is `raw`.

This class is targeted at advanced clients who want to precisely control the data, send more complex Smart Messaging or EMS content not supported by other classes, or send any other types of SMS by specifying any needed data for UDH and UD. Clients send the coding scheme (0 for 7bit, 1 for 8bit, or 2 for UCS2), the complete UDH, and the UD to be transmitted, in hexadecimal format (uppercase), separated by a tilde character. No transformation is made to data, smsBox® acts as a pure transport layer.

Frame sent to smsBox®:	coding scheme, ~ (tilde character), UDH, ~ (tilde character), UD

Example of such a frame:	`<class>`***`raw`***`</class>`
	`<hex>1~720C70002A352364322364322A342361312364322364332366`
	`6322361326133236132 2A352364322A342361317032236433236632 2`
	`3613361322361322361326131703223673223673323673223673223 6`
	`632236431236432236332236433236432 2A3470332364322363322 36`
	`4332364300D0A~666F6F626172</hex>`
	...

Explanation of example:	`1`: coding scheme (1 -> 8bit) ~: separator between coding scheme and UDH `720C7000...364300D0A`: UDH (contains one EMS User Defined Sound) ~: separator between UDH and UD `666F6F626172`: UD (text reading "foobar")

16. Appendix F: HTTP GET alternative

To send commands towards smsBox®, instead of sending an XML document with HTTP POST, it is also possible to use HTTP GET requests (REST-style), for the main commands. It may prove to be easier, if only the main commands are needed.

16.1. Commands mapping

Command name	Ending part of base request URL
CREATE	`/service/create`
DELETE	`/service/delete`
ELECTION	`/service/election`
MODERATEDCHAT	`/service/moderatedchat`
SERVICEINFO	`/service/info`
SERVICEUPDATE	`/service/update`
SUBSCRIBERS	`/service/subscribers`
POST	`/service/post`
BLACKLIST	`/user/blacklist`
UNBLACKLIST	`/user/unblacklist`
UNSUBSCRIBE	`/user/unsubscribe`
USERINFO	`/user/info`
SEND	`/user/send`
WEBSEND	`/user/websend`
WEBSUBSCRIBE	`/user/websubscribe`
OPENSESSION	`/session/open`
CLOSESESSION	`/session/close`
INSTANCEINFO	`/misc/instanceinfo`
REQUESTINFO	`/misc/requestinfo`
BILLINGTOKENVALIDATION	`/misc/billingtokenvalidation`

16.2. Trace of a successful USERINFO command

```
# curl -v http://username:password@pro.smsbox.ch:8047/Pro/sms/rest/user/info?msisdn=%2B41760000000

> GET /Pro/sms/rest/user/info?msisdn=%2B41760000000 HTTP/1.1
> Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQ=
> User-Agent: curl/7.19.6 (i586-mandriva-linux-gnu) libcurl/7.19.6 OpenSSL/0.9.8k zlib/1.2.3 libidn/1.15 libssh2/1
> Host: pro.smsbox.ch:8047
> Accept: */*
>

< HTTP/1.1 200 OK
< Server: Apache-Coyote/1.1
< Content-Type: text/xml;charset=UTF-8
< Content-Length: 281
<
<?xml version="1.0" encoding="UTF-8" ?>
<SMSBoxXMLReply>
  <ok/>
  <command name="USERINFO">
    <msisdn>+41760000000</msisdn>
    <operator>sunrise</operator>
    <lastRequest>2011-01-28 11:32:18</lastRequest>
  </command>
  <requestUid>xml10156182</requestUid>
```

```
</SMSBoxXMLReply>
```

16.3. Encoding consideration

The GET parameters must be URL-encoded, using the UTF-8 encoding. For example, posting the text "René" would be done with (notice the é character is represented by the two bytes 0xC3 and 0xA9):

```
# curl http://username:password@pro.smsbox.ch:8047/Pro/sms/rest/service/post?
    service=TEST&text=Ren%C3%A9
```

Notice: the above URL is split because of documentation constraint, but of course it must be specified in a single line.

If needed, it is also possible to use another standard encoding, using the `encoding=` GET parameter. For example, posting the text "René" using the ISO-8859-15 encoding (notice the é character is represented by the byte 0xE9):

```
# curl http://username:password@pro.smsbox.ch:8047/Pro/sms/rest/service/post?
    encoding=ISO-8859-15&service=TEST&text=Ren%E9
```