CPA WAP - Content Billing Service

- Implementation Guide for Content Provider

Version: 1.1

Distribution list

Name	Received date
Nygaard Arild Tarjei Ivar Røstengen	

Revision log

Version	Date	Description	Authors	Approved
0.1	26 Oct. 04	Created	Xinli Wang	
1.0	17 Jan. 05	Finalized	Arild T. Nygaard	

Table of contents

1 Introduction

CPA WAP is a web service interface offered by Telenor Mobile Nordic (TMN). This service allows content providers to charge the mobile phone user for utilizing wireless premium contents. Examples of such contents can be games, wallpapers, ring-tones, and video-clips. Telenor will charge/invoice the mobile user and content provider will deliver the content through the Telenor Network. The content provider receives a revenue share based on the generated transactions.

This document is assumed to be a guideline for those who want to integrate their WAP or wireless services with TMN's billing solution. The audience of this document may be technical architectures and software developers of Internet services.

1.1 Abbreviations

CP	Content Provider(s)
CPA	Content Provider Access
TMN	Telenor Mobile Nordic
WG	WAP Gateway

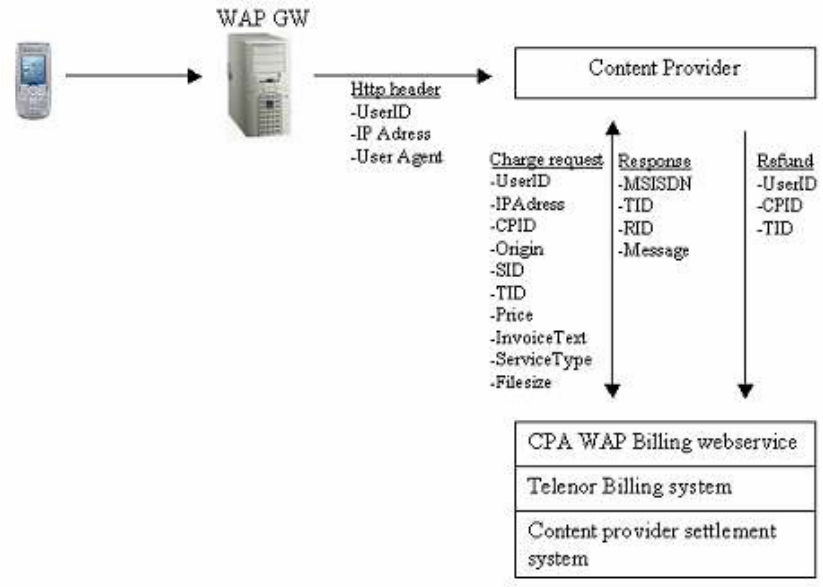
1.2 Related Documents

The reader should also view the website of Telenor's CPA services at: http://cpa.telenor.no/. Following procedures have to be finished before making the implementation to use CPA Content Billing Service:

1. Fill out the contact information form and mail the form to cpa@telenor.com
2. Sign up a standard CPA agreement with TMN.
3. Sign up the CPA WAP supplement agreement with TMN
4. Fill out the technical details on the technical form
5. Mail the form to cpa@telenor.com

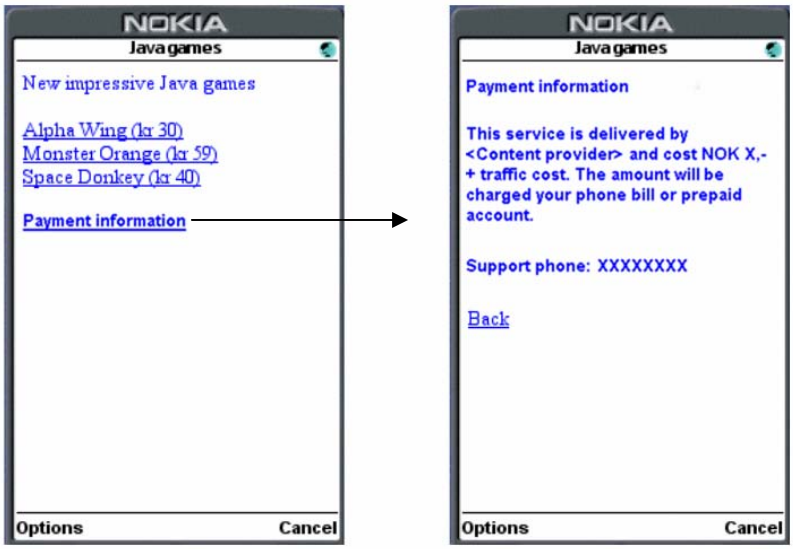
2 Solution Overview

Telenor providers a SOAP-based web-service interface for content providers to do charging and refunding (if download failed). In addition, a SOAP based WAP push interface is also supplied.



3 Implementation requirements

1. All charging of content downloaded via WAP and wireless internet shall be charged using CPA WAP.
2. The exact price of the service and an additional page containing payment information has to be made available before the user is actually using the service. The content providers must include the price of the service and payment information explicitly in the page that shows the list of contents offered to the user. The payment information must include price, name of the service provider and customer support phone number.



3. If the end-user clicks on the purchase link several times the Content Provider has to ensure that this only generates one charge request. In other words, the unique transaction ID (the TID parameter) generated by the Content provider shall be generated before the purchase page is displayed to the end-user to avoid several transactions ID (and multiple charge requests) for the same download.
4. When an end-user orders content, the Content Provider must check the User-Agent in the HTTP-header to ensure that the end-user receives content which is tested and designed for the terminal in use.
5. The Content Provider has to check that the request originates from Telenor's WAP gateway or Telenor's wireless gateway by checking from which IP address the request is originated. Telenor's gateways initiate request via Internet from the following domain-name/IP-addresses:

   WAPGW-VIP.mobil.telenor.no

   212.17.144.230

6. All WAP page interaction, streaming download, and content/application download must be tested with a Telenor's subscription via Telenor's WAP or wireless gateways before the launching of the service.

   Location of CPA WAP's WSDL file:

   Test site: http://billing-pilot.mobil.telenor.no/wbg/jsp/wsdl.jsp

   Production: http://access.mobil.telenor.no/wbg/jsp/wsdl.jsp

7. The Content Provider needs to recognize each user. Therefore Telenor adds a user ID as an HTTP header (X-Nokia-Alias parameter) in the request. This user ID will be static in a shout-term, and may be changed for a specific end-user. Hence, the user ID shall be used for billing purpose only. The

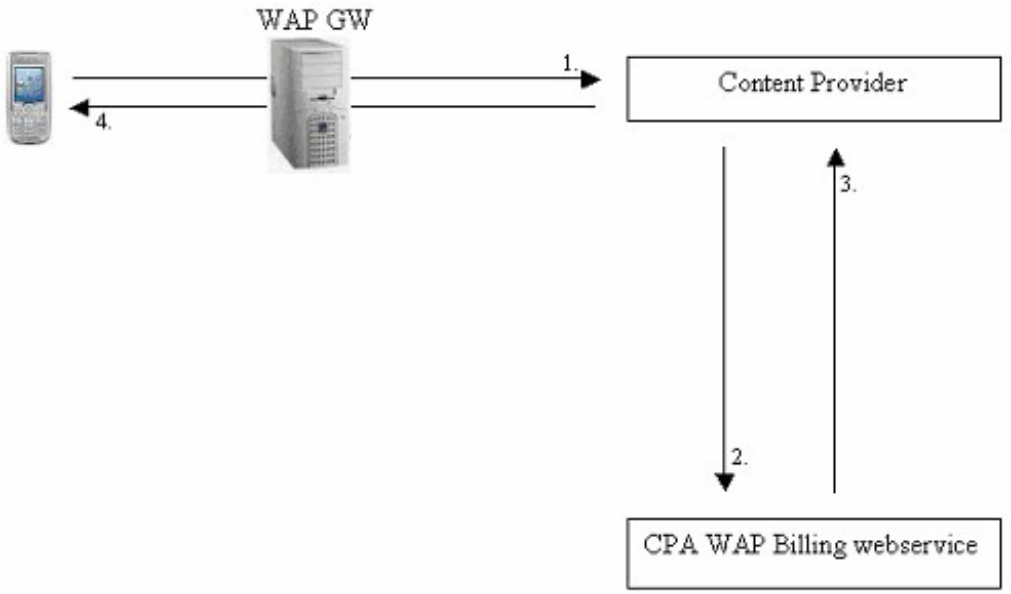
user ID shall not be used to personalize the service or in any other way be treated as a static ID for a specific end-user or terminal.

8. Reliable download is crucial for the commercialisation of digital content. The Content Provider shall enable and check if the content is successfully downloaded, whenever possible.
9. If the Content Provider does not receive a confirmation of successful download, or the end-user complains about the unsuccessful downloading, the Content Provider has to either offer the user a new free downloading or refund the user for previous content charging. In other words, the Content Provider needs to keep logging of the end-user's user ID, transaction ID and the billing result.
10. The refunding of the end-user towards Telenor's CPA WAP can only be done within a limited time, 24 hours at this moment. If the refunding failed by one or another reason (transaction too old, customer is barred, etc.), the Content Provider has to inform CPA Support (supportcpa@telenor.com) in order to do manual refunding.

4 Billing scenarios

4.1 Charge

To bill a Telenor subscriber CP must use the CPA WAP Charge method through a SOAP based WEB Services API. The figure below illustrates how the Web Service Charge Request works.



1. End user press the WAP link initiating CPA WAP billing ex.
   Download Space Donkey (kr 40)

   Http header
   -X-Nokia-alias
   -X-Nokia-ipaddress
   -User Agent

2. The content provider initiates a Web Service Request towards Telenor

   Charge request
   -UserID
   -IPAdress
   -CPID
   -SID
   -Price
   -InvoiceText
   -Origin
   -ServiceType
   -Filesize
   -TID

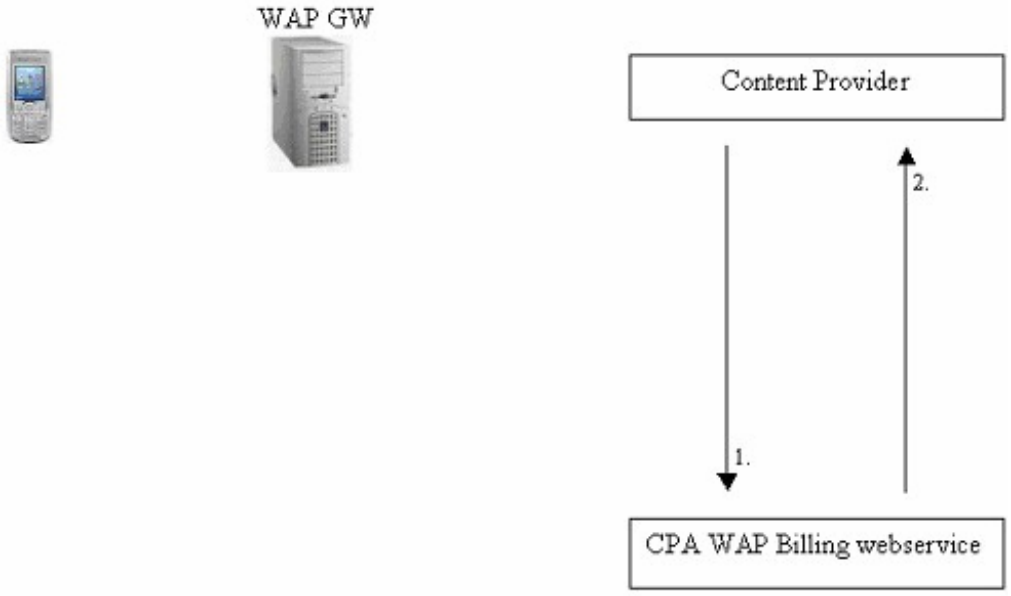
3. Content provider receives a response code indicating billing status.

   Response
   -MSISDN
   -TID
   -RID
   -Message

4. After the billing has succeeded (or failed). The end-user is redirected to the content (or error) page

4.2 Refund

If the service of the content provider was not delivered to the end-user after charge was conducted CP must refund the end-user. Refund must be completed within 24 hours after charge was initiated. Refund is done by making a “Web Service Refund Request”



1. If delivery of content failed, Content Provider must refund end user within 24 hours using the Web Service interface

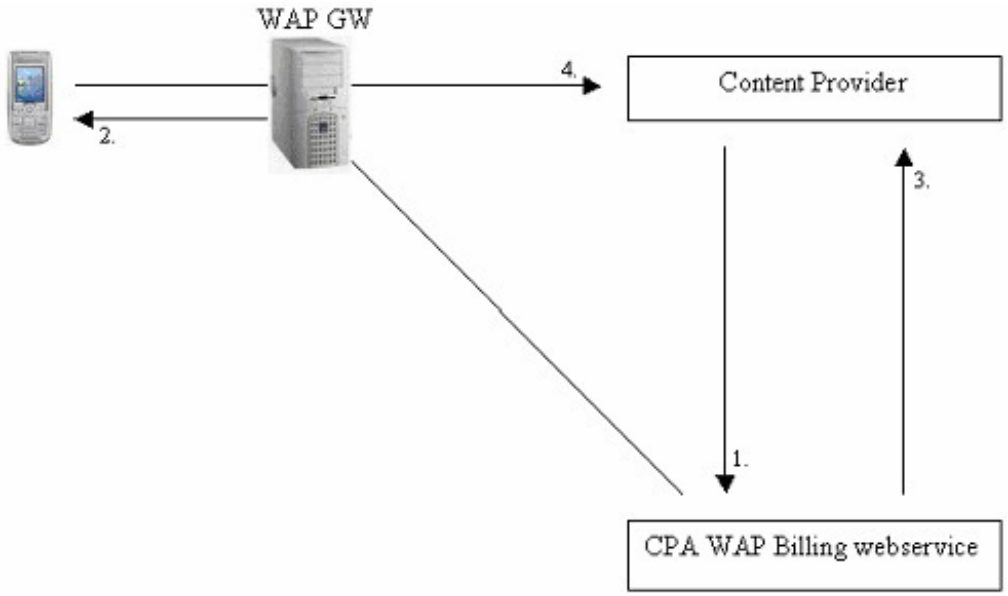
Refund
-UserID
-CPID
-TID

2. Respons on refund

Response
-MSISDN
-TID
-RID
-Message

4.3 WAP Push

Through CPA WAP, Telenor offers an API to send Service Initiate (SI) WAP Push messages to Telenor subscribers. These messages contain a CP specified text and URL. Telenor does not offer premium WAP Push messages. In order to charge an end user for premium services CP must use the Charge method described in Chapter 4.1, after the end user has received the push and is connected to CP's site.



1. The Content provider initiates a Web Service Push Request towards Telenor

WAP Push
-CPID
-TID
-PushText
-PushURL
-MSISDN
-Signature

2. CPA WAP sends the WAP Push message through the Telenor network to the end user.

3. Given the result of the WAP Push message delivery, CP receives respond of status

Response

-MSISDN
-TID
-RID
-Message

4. End user activates the WAP Push link and connects to the CP's site.

CP can then bill the end user by means of "charge" method and deliver the desired content to the end user.

5 Interface Description

5.1.1 Input Parameters

The basic parameters needed for making a request to CPA WAP are described in this section. All of the parameters must be filled and **cannot** be **null**. For **push** request, the parameters in must be put in the **right sequence** in order to achieve a correct signature.

Table 5.1 describes the type, length and meaning of these parameters which content provider must pass in the SOAP request to CPA WAP.

Name	Type / Max. length	Description / Comments	Method		
			charge	refund	push
UserID	String / 16	UserID is the identification of the end user provided by Telenor. It can be found in the HTTP header, "**X-Nokia-Alias"**, of the request from the WAP Gateway.	X	X	
IPAddress	String	IP address (in the format nnn.nnn.nnn.nnn) of the end-user's phone.	X		
CPID	String / 9	Content provider ID provided by Telenor	X	X	X
SID	String / 6	Tariff class used for billing. For WAP content, the tariff class "BASIS" must be used.	X		
TID	Digits / 18	Unique transaction ID (number) generated by the content provider.	X	X	X
Price	Digits / 4	The price of the service in 1/100 of a NOK (in Norwegian øre). CP is free to choose the price as long as obeying the rules given by Telenor.	X		

Name	Type / Max. length	Description / Comments	Method		
			charge	refund	push
InvoiceText	String	The invoice text used to indicate which content/service the user has paid for. This information will be shown on user's detailed invoice and used only for customer services and complains. It can be in a form as: **<content-type>:<content name>** e.g., a user purchases a game, "Puzzle", for the content provider "GamePortal", Then the invoice text maybe as: InvoiceText=Game:Puzzle No Norwegian characters, line feed, semicolon(;), colon(:), double quotes (""). Comma(,), tabulator, tilda(~) or equal(=) are allowed	X		
Origin	String	The original source/website from which the content provider is receiving the redirected request of a customer. The values of Origin should be: **DF**: Default CPA WAP Content Provider **PL**: Telenor Mobil's Portal **TL**: Telenor Entry **DS**: Default CPA WAP Content Provider - Streaming **PS**: Telenor Mobil's Portal - Streaming **ES**: Telenor Entry - Streaming	X		
ServiceType	String	Indicate the type of the service. The values of ServiceType should be one of following (at this moment): **WA** – default WAP contents and downloads **ST:** Time based streaming service (continual, time-interval based charging)	X		
FileSize	String	Information in KB of the content file been downloaded. The meaning of this field depends on **ServiceType**: If ServiceType = **WA**, FileSize = 0 If ServiceType = ST, it is the bit-rate of the file downloading, e.g., 22kbps, 50kbps or 75kbps.	X		

Name	Type / Max. length	Description / Comments	Method		
			charge	refund	push
PushText	String	**Used only for WAP push**. The descriptive text that is presented to the receiver when receiving the push request (before pulling the content).			X
PushURL	String	**Used only for WAP push**. The URL to CP's page where price information can be found and contents can be downloaded.			X
MSISDN	String	**Used only for WAP push**. The MSISDN of the message receiver.			X
Signature	String	**Used only for WAP push**. A signature obtained by first concatenating all of the reqired parameters (**in the right order**) plus CP's secret key at the end, and then hashing the result string, and finally encoding the string using Base64 encoding.			X

Table 5.1

5.1.2 Output Parameters

CPA WAP always returns a response to content providers for each request being received. Content provider **MUST** check the return parameters, specially RID value, before taken any further actions. If RID is a positive number, i.e. RID is 200 or 205, the transaction should be assumed as billed. Otherwise, the transaction is not billed. If the end user still wants to purchase the content and the reason for failure is not because of insufficient credit, then the content provider has to send a new "Charge" request with a **NEW** transaction ID.

Table 5.2 describes the parameters returned to the content provider.

Parameter	Type	Description / Comments	Method		
			charge	refund	push
TID	Digits	The unique transaction ID sent by the content provider for the Charge/Refund/Push request.	X	X	X
RID	String	Receipt to CP to indicate the status of the request. See Chapter 5.1.3 for more information about the possible RID values and their meanings.	X	X	X
MSISDN	String	The MSISDN number of the customer who bought the content.	X	X	X
Message	String	Return message about the result of the request	X	X	X

Table 5.2

5.1.3 Return code RID

The following table sums up the values and descriptions of the return code RID:

RID value	Description
200	OK. The request was processed and billing or refunding was done successfully.
205	Delayed billing due to internal problem. Charging **will be tried** late on, but no guaranteed billing of prepaid subscriber. Content **must be delivered** unless refund is made. The transaction should **not** be sent again!
-300	CPA WAP internal error
-310	Parameter missing in the request.
-311	CPID is missing or illegal
-312	Replay/Duplicate TID for "Charge" request. The given TID has been used before or the transaction has been registered earlier.
-313	Invalid TID for "Charge" request. TID must be a (positive) digit no large than the number of ciphers defined in this document (18 digits).
-315	Missing or invalid price. Parameter PR should be an integer between 100 to 6000 Øre (1 to 60 Norwegian krone) and should be a whole krone (increase by 100 Øre).
-319	SID is missing or illegal (the value of SID is neither BASIS nor a registered service name in CPA WAP).
-340	Missing or invalid customer IP address
-341	Missing or invalid FileSize. This parameter must be given in kbps for streaming services.
-342	Missing or invalid CURL
-343	Missing Invoice-text
-350	Missing or invalid "Origin" value
-351	Missing or invalid "ServiceType" value
-360	NOK 0,- charge not allowed. Parameter PR should be an integer between 100 to 6000 Øre (1 to 60 Norwegian krone) and should be a whole krone (increase by 100 Øre).
-401	Invalid "Refund" request. Either the required transaction does not exist in CPA WAP or it was refunded earlier.
-402	Refunding failed. The reason can be one of the following: • Customer's account is barred • Maximum time limit for refunding is expired • Problems in the backend billing system In this case, content provider must make a manual refund request through Telenor's Net-front service (customer-service agent for content providers)
-800	UserID is missing or it is not a legal identification. The user may not be a Telenor customer.
-1003	Unable to connect to the backend billing system. The transaction will be put on a queue for retry. No billing guaranty for prepaid customers. This code is used internally only. RID **205** will be sent out to CP instead.
-1004	Prepaid account has no enough credit for the service.
-1005	Non-active customer. The user is not online in a WAP or wireless

RID value	Description
	internet Session.
-1006	Invalid customer account. The user is not a valid Telenor customer.
-1009	Customer is barred. The customer account is for some reason blocked by Telenor (stolen mobile, missing payment etc).

Table 5.3 Values for RID

6 Support

All questions and problems regarding CPA WAP should be addressed to the support email box:

supportcpa@telenor.com

7 Appendix

7.1 WSDL

The updated WSDL file can be downloaded from CPA WAP's URL:

Production server: http://access.mobil.telenor.no/wbg/jsp/wsdl.jsp (available after 20 January 2005)
Test server: http://billing-pilot.mobil.telenor.no/wbg/jsp/wsdl.jsp

Following text can only be viewed as an example of the WSDL file.

```
<?xml version="1.0" encoding="UTF-8" ?>
http://billing-pilot.mobil.telenor.no/wbg/jsp/ <definitions
    xmlns:tns="http://www.telenor.com/webservices/wbg"
    xmlns:wsr="http://www.openuri.org/2002/10/soap/reliability/"
    xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
    xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
    xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:soap12enc="http://www.w3.org/2003/05/soap-encoding"
    xmlns:conv="http://www.openuri.org/2002/04/wsdl/conversation/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns="http://schemas.xmlsoap.org/wsdl/"
    targetNamespace="http://www.telenor.com/webservices/wbg">

http://billing-pilot.mobil.telenor.no/wbg/jsp/ <types xmlns:tns="http://www.telenor.com/webservices/wbg"
    xmlns:wsr="http://www.openuri.org/2002/10/soap/reliability/"
    xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
    xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
    xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:soap12enc="http://www.w3.org/2003/05/soapencoding"
    xmlns:conv="http://www.openuri.org/2002/04/wsdl/conversation/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns="http://schemas.xmlsoap.org/wsdl/">

http://billing-pilot.mobil.telenor.no/wbg/jsp/ <xsd:schema
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:stns="java:com.telenor.wbg.common"
    elementFormDefault="qualified" attributeFormDefault="qualified"
    targetNamespace="java:com.telenor.wbg.common">

http://billing-pilot.mobil.telenor.no/wbg/jsp/ <xsd:complexType name="CommonResponse">
```

Status:

```
http://billing-pilot.mobil.telenor.no/wbg/jsp/ <xsd:sequence>
  <xsd:element type="xsd:string" name="MSISDN" minOccurs="1" nillable="true" maxOccurs="1" />
  <xsd:element type="xsd:int" name="RID" minOccurs="1" maxOccurs="1" />
  <xsd:element type="xsd:long" name="TID" minOccurs="1" maxOccurs="1" />
  <xsd:element type="xsd:string" name="message" minOccurs="1" nillable="true" maxOccurs="1" />
  </xsd:sequence>
  </xsd:complexType>
  </xsd:schema>
  </types>
http://billing-pilot.mobil.telenor.no/wbg/jsp/ <message name="charge">
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema" type="partns:string" name="string" />
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema" type="partns:string" name="string0" />
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema"
     type="partns:int" name="intVal" />
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema" type="partns:string" name="string1" />
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema" type="partns:long" name="longVal" />
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema" type="partns:decimal" name="bigDecimal" />
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema" type="partns:string" name="string2" />
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema" type="partns:string" name="string3" />
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema" type="partns:string" name="string4" />
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema" type="partns:int" name="intVal0" />
  </message>
http://billing-pilot.mobil.telenor.no/wbg/jsp/ <message name="chargeResponse">
  <part xmlns:partns="java:com.telenor.wbg.common" type="partns:CommonResponse" name="result" />
  </message>
http://billing-pilot.mobil.telenor.no/wbg/jsp/ <message name="refund">
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema" type="partns:string" name="string" />
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema" type="partns:int" name="intVal" />
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema" type="partns:long" name="longVal" />
  </message>
http://billing-pilot.mobil.telenor.no/wbg/jsp/ <message name="refundResponse">
  <part xmlns:partns="java:com.telenor.wbg.common" type="partns:CommonResponse" name="result" />
  </message>
http://billing-pilot.mobil.telenor.no/wbg/jsp/ <message name="push">
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema" type="partns:int" name="intVal" />
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema" type="partns:long" name="longVal" />
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema" type="partns:string" name="string" />
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema" type="partns:string" name="string0" />
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema" type="partns:string" name="string1" />
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema" type="partns:string" name="string2" />
  </message>
http://billing-pilot.mobil.telenor.no/wbg/jsp/ <message name="pushResponse">
  <part xmlns:partns="java:com.telenor.wbg.common" type="partns:CommonResponse" name="result" />
  </message>
http://billing-pilot.mobil.telenor.no/wbg/jsp/ <portType name="ServiceManagerPort">
http://billing-pilot.mobil.telenor.no/wbg/jsp/ <operation name="charge">
  <input message="tns:charge" />
  <output message="tns:chargeResponse" />
  </operation>
http://billing-pilot.mobil.telenor.no/wbg/jsp/ <operation name="refund">
```

Status:

```
  <input message="tns:refund" />
  <output message="tns:refundResponse" />
  </operation>
http://billing-pilot.mobil.telenor.no/wbg/jsp/ <operation name="push">
  <input message="tns:push" />
  <output message="tns:pushResponse" />
  </operation>
  </portType>
http://billing-pilot.mobil.telenor.no/wbg/jsp/ <binding type="tns:ServiceManagerPort"
      name="ServiceManagerPort">
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
http://billing-pilot.mobil.telenor.no/wbg/jsp/ <operation name="charge">
  <soap:operation style="rpc" soapAction="" />
http://billing-pilot.mobil.telenor.no/wbg/jsp/ <input>
  <soap:body namespace="http://www.telenor.com/webservices/wbg"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded" />
  </input>
http://billing-pilot.mobil.telenor.no/wbg/jsp/ <output>
  <soap:body namespace="http://www.telenor.com/webservices/wbg"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded" />
  </output>
  </operation>
http://billing-pilot.mobil.telenor.no/wbg/jsp/ <operation name="refund">
  <soap:operation style="rpc" soapAction="" />
http://billing-pilot.mobil.telenor.no/wbg/jsp/ <input>
  <soap:body namespace="http://www.telenor.com/webservices/wbg"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded" />
  </input>
http://billing-pilot.mobil.telenor.no/wbg/jsp/ <output>
  <soap:body namespace="http://www.telenor.com/webservices/wbg"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded" />
  </output>
  </operation>
http://billing-pilot.mobil.telenor.no/wbg/jsp/ <operation name="push">
  <soap:operation style="rpc" soapAction="" />
http://billing-pilot.mobil.telenor.no/wbg/jsp/ <input>
  <soap:body namespace="http://www.telenor.com/webservices/wb"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded" />
  </input>
http://billing-pilot.mobil.telenor.no/wbg/jsp/ <output>
  <soap:body namespace="http://www.telenor.com/webservices/wbg"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded" />
  </output>
  </operation>
  </binding>
http://billing-pilot.mobil.telenor.no/wbg/jsp/ <service name="ServiceManager">
 http://billing-pilot.mobil.telenor.no/wbg/jsp/ <port name="ServiceManagerPort"
      binding="tns:ServiceManagerPort">
  <soap:address location="http://billing-pilot.mobil.telenor.no/service/ServiceManager" />
  </port>
  </service>
  </definitions>
```

Status:

7.2 HTTP header Example

Following text is an example of the HTTP headers sent by Telenor's WAP Gateway.

accept: text/html, application/xhtml+xml;profile=http://www.wapforum.org/xhtml,
application/vnd.wap.xhtml+xml, text/vnd.wap.wml, application/vnd.wap.wmlc,
application/vnd.wap.wbxml, application/vnd.wap.wmlscriptc, application/vnd.wap.multip art.mixed,
multipart/mixed, text/x-vcard, text/x-vcalendar, audio/*, application/*, */*

user-agent: SonyEricssonP900/R101 Profile/MIDP-2.0 Configuration/CLDC-1.0

accept-language: no, en

accept-charset: UTF-8, utf-16; q=0.6, ISO-8859-1, *

bearer-indication: 0

X-Network-info: GPRS,UizrzjxFEuHElQ==,10.8.9.245,unsecured

X-Nokia-ipaddress: 10.8.9.245

X-Nokia-alias: UizrzjxFEueHlQ==

X-Nokia-CONNECTION_MODE: CMODE

X-Nokia-BEARER: GPRS

X-Nokia-gateway-id: NWG/4.0/Build60

x-wap-profile: http://www.sonyericsson.com/UAProf/P900R101.xml

x-nokia.wia.accept.original: text/html,application/xhtml+xml;
profile=http://www.wapforum.org/xhtml,application/vnd.wap.xhtml+xml,text/vnd.wap.wml,
application/vnd.wap.wmlc,application/vnd.wap.wbxml,application/vnd.wap.wmlscriptc,
application/vnd.wap.multipart.mixed,multipart/mixed,text/x-vCard,text/x-vCalendar,audio/*,
application/*,*/*

Via: WTP/1.1 wap-gw-test.mobil.telenor.no (Nokia WAP Gateway 4.0/ECD9/4.0.60), 1.1 S1PS

7.3 Example of Client Codes

Content Provider will get a package of jar-file and client files for making call to CPA WAP and billing the end users. This chapter gives only some example codes to show how to the client program is built and how to the messages.

7.3.1 Java Clients

7.3.1.1 Charge and Refund

```
import com.telenor.wbg.webservice.ServiceManager_Impl;
import com.telenor.wbg.webservice.ServiceManagerPort;
import com.telenor.wbg.common.CommonResponse;
import java.math.BigDecimal;
import java.util.Random;

/**
 * This class illustrates how to use Telenor's CPAWAP web-service interface
 * to do billing and refunding
*/

public final class Client {

  public Client() {}

  public static void main(String[] argv)
    throws Exception
  {
    // Setup the global JAXM message factory
    System.setProperty("javax.xml.soap.MessageFactory",
      "weblogic.webservice.core.soap.MessageFactoryImpl");

    // Setup the global JAX-RPC service factory
    System.setProperty( "javax.xml.rpc.ServiceFactory",
      "weblogic.webservice.core.rpc.ServiceFactoryImpl");

    // set up URL for WSDL
    String wsdlUrl = "http://billing-pilot.mobil.telenor.no/wbg/jsp/wsdl.jsp";

      ServiceManager_Impl service = new ServiceManager_Impl( wsdlUrl );
      ServiceManagerPort port = service.getServiceManagerPort();

      // hard code test parameters. Some of these should originally come from HTTP headers
      String userID = "UizrzDxFH+DBmA=="; // 4797124725
      String ipAddress = "10.8.9.181";
      int fileSize = 64;
      int cpID = 300;
      String SID = "BASIS";

      BigDecimal price = new BigDecimal(100);
      String invoiceText = "Test client";
      String origin = "PL";
      String serviceType = "WA";
```

```
        // hardcode TID for testing. It should be a running and unique number
        long TID = 10000l;

        // do charge
            CommonResponse result = port.charge(
                    userID,
                    ipAddress,
                    cpID,
                    SID,
                    TID,
                    price,
                    invoiceText,
                    origin,
                    serviceType,
                    fileSize
                    );

            System.out.println("Get result for charge request:\n" +
                    "TID=" + result.getTID() + "\n" +
                    "RID=" + result.getRID() + "\n" +
                    "MSISDN=" + result.getMSISDN() + "\n" +
                    "message: " + result.getMessage());

           // do refund
            CommonResponse refundresult = port.refund(
                    userID,
                    cpID,
                    TID
                    );
            System.out.println("Get result for refund request:\n" +
                    "TID=" + refundresult.getTID() + "\n" +
                    "RID=" + refundresult.getRID() + "\n" +
                    "MSISDN=" + refundresult.getMSISDN() + "\n" +
                    "message: " + refundresult.getMessage());
    }
}
```

7.3.1.2 Push

```
package com.telenor.wbg.serviceclients;

import com.telenor.wbg.common.LocalSecurity;
import com.telenor.wbg.webservice.ServiceManagerPort;
import com.telenor.wbg.webservice.ServiceManager_Impl;
import com.telenor.wbg.webservice.ServiceManagerPort_Stub;
import com.telenor.wbg.common.CommonResponse;
import java.io.IOException;
import java.rmi.RemoteException;
import weblogic.webservice.binding.BindingInfo;


public class PushClient {
```

```
    private ServiceManagerPort servicePort;
    private LocalSecurity localSecurity = new LocalSecurity();
    private String secretKey = "hjtNwv1UAgsqwT1GV3UZdv6A4FIf1p3s";

    public PushClient(String wsdl) {

        // Setup the global JAXM message factory
        System.setProperty("javax.xml.soap.MessageFactory",
                "weblogic.webservice.core.soap.MessageFactoryImpl");
        // Setup the global JAX-RPC service factory
        System.setProperty("javax.xml.rpc.ServiceFactory",
                "weblogic.webservice.core.rpc.ServiceFactoryImpl");

        try {
            ServiceManager_Impl ws = new ServiceManager_Impl(wsdl);

            servicePort = ws.getServiceManagerPort();
            ServiceManagerPort_Stub serviceStub = (ServiceManagerPort_Stub) servicePort;
            BindingInfo info =
(BindingInfo)serviceStub._getProperty("weblogic.webservice.bindinginfo" );
             info.setCharset( "UTF-8" );

        } catch (IOException e) {
            e.printStackTrace();
        }

    }

    public CommonResponse push(int cpID, long tID, String pushText, String pushURL, String
msisdn) {

        // Generate signature
        String requestParameters[] = {String.valueOf(cpID),
                                      String.valueOf(tID),
                                      pushText,
                                      pushURL,
                                      msisdn

        };
        localSecurity.setSharedSecret (secretKey);
        String signature = localSecurity.signMessageWithHash(requestParameters);

        CommonResponse response = null;
        try {
            response = servicePort.push(cpID, tID, pushText, pushURL, msisdn, signature);
        } catch (RemoteException e) {
            e.printStackTrace();
        }

        return response;

    }
```

```
    public static void main(String[] argv)
            throws Exception {

        String wsdl = "http://billing-pilot.mobil.telenor.no/wbg/jsp/wsdl.jsp";

        long tID = (new Double(Math.random() * 100000000L)).longValue();
        PushClient pc = new PushClient(wsdl);
        String pushText = "Hello, push webservice is speaking æøåÆØÅ";
        CommonResponse response = pc.push(400, tID, pushText, "wap.nettavisen.no",
"4797124725");
        System.out.println("Response Code: " + response.getRID());
        System.out.println("Response Message: " + response.getMessage());
    }
}
```

7.3.2 .Net Client

```
using System;
using System.Security.Cryptography;
using System.Text;

class PushClient

{
        private string secretKey = " hjtNwv1UAgsqwT1GV3UZdv6A4FIf1p3s";
        private static int cpID = 400;

        public string GenerateSignature ( string [] parameters, string secretkey) {

                int nParams = parameters.Length;
                string paramstr = "";

                for (int i = 0; i < nParams; i ++ ) {
                        paramstr = paramstr + parameters [i];
                }

                string message = paramstr + secretkey;

                byte[] data = new UTF8Encoding().GetBytes(message);
                byte[] result;
                SHA1 sha = new SHA1CryptoServiceProvider();
                result = sha.ComputeHash(data);
                string encresult = Convert.ToBase64String(result);
                return encresult;

        }

        public CommonResponse push (int cpID, long tID, string pushText, string pushURL,
                string msisdn) {

                string  []parameterArray = {Convert.ToString(cpID),
                Convert.ToString(tID),
                pushText,
```

```
                pushURL,
                //      Convert.ToString(price),
                //      sID,
                msisdn
                //      invoiceText,
                //      version
                };

                string signature = GenerateSignature (parameterArray, secretKey);


                ServiceManager serviceManager = new ServiceManager();
                CommonResponse response = null;
                response = serviceManager.push (cpID, tID,pushText, pushURL, msisdn, signature);

                return response;

        }

        public static void Main(string[] args)
        {
                PushClient pc = new PushClient();

                Random random = new Random();
                long tID = random.Next();

                CommonResponse response = pc.push(cpID, tID,"Hello I am the .NET pusher",
"wap.nettavisen.no", "4797124725");
                Console.WriteLine("Response Code: " + response.RID);
                Console.WriteLine("Response Message: " + response.message);
        }
}
```

NOTE: The class ServiceManager is generated automatically by running the .NET WSDL generator (wsdl.exe)

7.3.3 How to sign push message

To sign the parameters before sending a push request, you should do the following:

1. Construct the parameter string in an **order** described in Chapter 5.1.1 “<input1><input2>…<inputN>”
2. **Append** your **secret key** (as issued by Telenor) to the **end** of the parameter-string.
3. Obtain the signature by computing the hash of this new text string using SHA-1 in base64 mode.

Following text is an example Java class used to make a signature. Content provider will get such a jar file from Telenor CPA WAP to sign messages. See Chapter 7.3.1.2 about how to make a push request with signed parameters.

```
/**
 * A class used to sign the messages being sent to Telenor Mobil.
 */
public class LocalSecurity {
```

```
        private String mySharedSecret;
        private String myMessageDigestID = "SHA-1";

public void setSharedSecret( String strSharedSecret ) {
                mySharedSecret = strSharedSecret;
}

          /*
           * Construct the textstring "<value1><value2>...<value n>".
          * @param astrMessageParts an array of strings - array is assumed not to be null
           */
          private String makeAString( String [] astrMessageParts ) {

          String astrMessage = astrMessageParts [0];
          for ( int i = 1; i < astrMessageParts.length; i++ ) {
                    astrMessage += astrMessageParts [i];
                }
                return astrMessage;
          }

          private byte [] generateMessageDigest( String strMessage )
                throws Exception
          {
                MessageDigest messageDigestOriginal =
                   MessageDigest.getInstance( myMessageDigestID );
                MessageDigest messageDigest =
                   ( MessageDigest ) messageDigestOriginal.clone();
                messageDigest.reset();
                messageDigest.update( strMessage.getBytes( "UTF8" ) );
                return messageDigest.digest();
        }


public String signMessageWithHash( String [] astrMessageParts )
        {
                String strMessage = makeAString( astrMessageParts ) + mySharedSecret;

                BASE64Encoder base64Encoder = new BASE64Encoder();
                return base64Encoder.encode( generateMessageDigest( strMessage ) );
          }
}
```