O_2 SMS Connector

Technical description of Web services interface

Content

1 Introduction

1.1 Purpose and Scope

This document is for use by O2 Business Partners (BP) wishing to send and receive text and multimedia messages to mobile subscribers through the O2 Messaging Platform using Web Services. This document provides a description of the Web Services interface for the SMS Connector service running on the O2 messaging platform and is intended for use by business partners formal part of the specification, but provides additional helpful information for the Web Services developers.

1.2 Definitions, Acronyms and Abbreviations

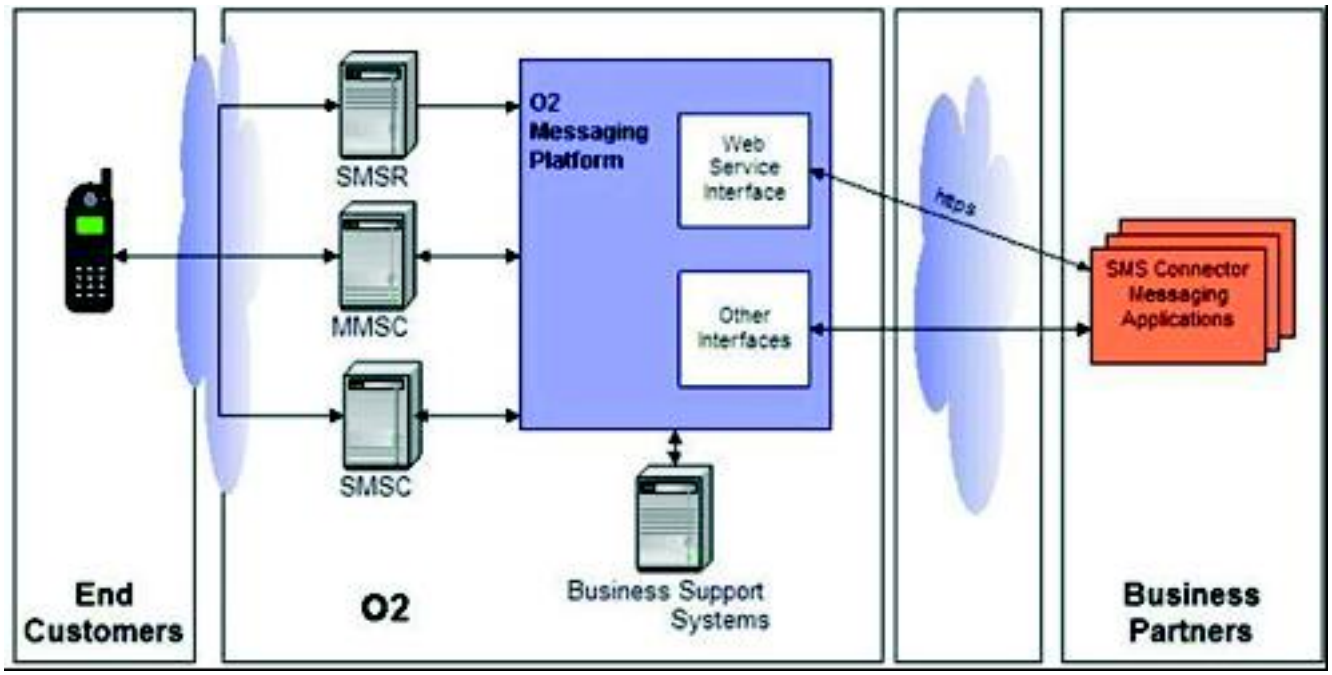
Abbreviation	Definition
AO	Application-Originated - is a term used to determine the identification of the source direction or originator of a message.
AI	Application-Terminated - is a term used to determine the identification of the target direction or destination of a message.
API	Application Program Interface
BA Profile	This is data for a business application that comes from the provisioning system. This includes all information about business application required by the Messaging Platform such as capacity, certificates, etc.
BA	Business Application
CA	Certification Authority
CDR	Call Detail Record
ET BA.wsdl	WSDL file for description of the business partners Web Service for receiving in push mode (PushPPGwPort.wsdl). See chapter 2.1
ET MP.wsdl	WSDL file for description of the O2 messages platform services (MessagingGatewayPort.wsdl)
MSISDN	The mobile user's telephone number.
MO	Mobile-Originated - is a term used to determine the identification of the source direction or originator of a message.
MT	Mobile-Terminated - is a term used to determine the identification of the target direction or destination of a message.
MP	Messaging Platform - A logical system made up of the Web Services Gateway, HTTP Server, and other internal O2 systems.
MMS	Multimedia Message Service - is a service for sending multimedia messages to mobile phones that use Global System for Mobile (GSM) communication. Multimedia

	message content examples include images, audio, text, video and combinations of these.
MMSC	MMS Center – The O2 network element that the MP sends MMS messages to and receives MMS messages from.
SMS	Short Message Service - is a service for sending text or binary messages to mobile phones that use Global System for Mobile (GSM) communication.
SMSC	SMS Center – The O2 network element that the MP sends SMS messages to and receives SMS messages from. Although ET has multiple SMSCs, from the perspective of this document, there is no difference and hence they are all grouped into this term.
SMS Connector	SMS Connector Message Service – This is the product that business partners will use to send and receive SMS messages.
RPC	Remote Procedure Call
SOAP	Simple Object Access Protocol – a W3C recommendation / standard that is used by Web Services.
Web Services	standards-based, interoperable, self-contained, modular applications that can be described, published, located, and invoked over the internet
UDDI	Universal Description, Discovery and Integration registry – used to store Web Service Definitions that can be used by applications to find information about how to connect to a Web Service.
SPK	Service Packages.
WSGW	Web Services Gateway
WSDL	Web Services Definition Language.
WSIF	Web Services Invocation Framework
XML	Extensible Markup Language
XSD	XML Schema Definition
IDE	Integrated Development Environment – An application that facilitates developing and testing applications.
SMSR	SMS-Redirector – O_2 network feature enabling BA to receive MO SMS from other operator networks

2 System Context and usage Scenarios

The O2 Messaging Platform provides business applications that are operated by business partners, with a highly scalable and reliable two-way messaging channel. The platform is provided to support SMS Connector business application – and is integrated into O2's existing back-end systems such as pre- and post-paid billing, customer care, customer management, etc.

In the diagram above the main entities are:



- O2: offers the messaging Web Services and is responsible for Messaging Delivery, Billing and Customer Care. O2 publishes the Web Service through a WSDL description.
- Business Partners: organizations that send and receive short or multimedia mobile messages by using the O2 Messaging Platform. The business partners implement business applications for SMS Connector.
- End Customer: an individual who subscribes to and receives/sends short or multimedia content from/to Business Application of a O2 Business Partner. Receiving/sending SMS from/to End Customer using fix line is not supported.
- The Web Service acts as a proxy between the inner O2 messaging platform and the business application. It provides a secure method of sending and receiving messages by using HTTPS and client side authentication (see also chapter Security). The description of the services is done via a Web Services Description Language (WSDL) file and thereby allows platform- independent access to the messaging services.
- Chapter 3 provides a short introduction in Web Services architectures and key building blocks. The following paragraphs give a more detailed view of the system components.

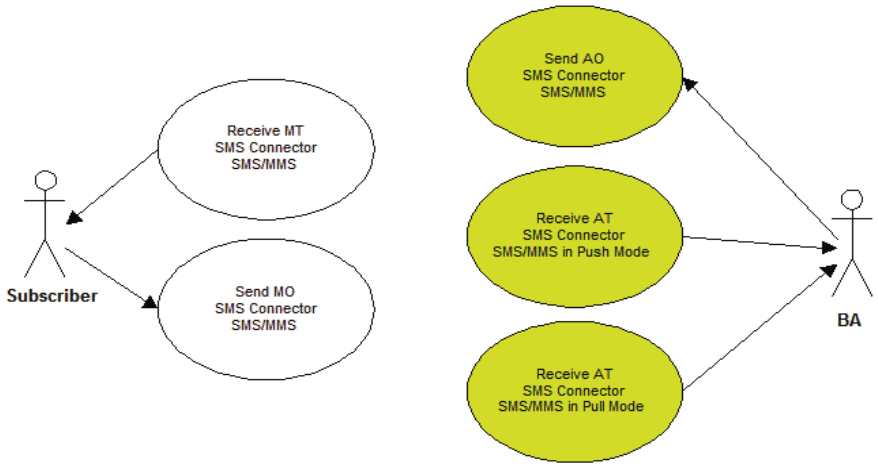
2.1 Key Usage Scenarios and Service Capabilities

2.1.1 SMS Connector Service

This product is for use by O2 Business Partners (BP) wishing to send and receive SMS messages to mobile subscribers. The platform supports text and binary SMS messages. The goal of this product is to deliver a platform which will connect the BP's business application to O2's messaging platform through a set of standardized interfaces including Web Services and HTTP Get/Post (covered in a separate specification). Although the service is designed to allow business applications to communicate with the platform using either of the two interfaces (HTTP Get/Post and Web Services), business applications must perform the receive-confirm sequence (See section 4.2) using the same interface. In addition, O2 is responsible for automated provisioning and billing.

2.1.2 Services Overview

From the perspective of the business partner the following capabilities/services are available (in yellow):



Service	Description	Comments
Send AO to SMS Connector SMS	Used by BA to send a text message/multimedia message to a subscriber	
Receive AT SMS Connector SMS in Pull Mode	Allows the BA to retrieve saved messages from the Messaging Platform. The BA connects to the messaging platform and retrieves the messages one after the other.	
Receive AT SMS Connector SMS in Push Mode	The MP actively connects to the business application and pushes the available messages.	If a business application is unavailable, the Messaging Platform attempts to resend the message a number of times, if still unsuccessful the message is stored in a message database and is available for pull mode.

2.2 Detailed Interface Architecture

The O2 Messaging Platform's Web Services are available via the Simple Object Access (SOAP) communication protocol. For security purposes SSL encryption is used for all connections (HTTPS). authentication of the client is done via client-side certificates, which are validated by comparison the certificates from the registration/provisioning process. For more information on security see chapter 6.

The platform offers two primary services or modes to business applications:
Send – simple, multi-part, multimedia or binary content to a target mobile device (AO message)
Receive - equivalent variants of contents from mobile devices (MO messages).

The platform however, supports two variants of Receive - Receive in Pull Mode and Receive in Push Mode. For both of the receive modes the First In First Out (FIFO) principle applies: all messages are processed by the messaging platform in the order of their arrival. The order of delivery to the BA, however, is not guaranteed as a BA may use parallel threads to receive messages from the MP.

2.2.1 Receive in Pull Mode

In this mode the business application calls the service methods (such as receive0 or confirm0) and receives return values. The O2 Messaging Platform acts in a service provider role, whereas the business application acts as a service consumer. In the majority of cases the business partner will choose to connect to the O2 Messaging Platform by using the “Receive in Pull Mode”. **Chyba! Nenalezen zdroj odkazů.** below shows the detailed architecture of this Web Service interface. The messaging Web Service description is done in a WSDL file which will be called “ET MP.wsdl” throughout the document.

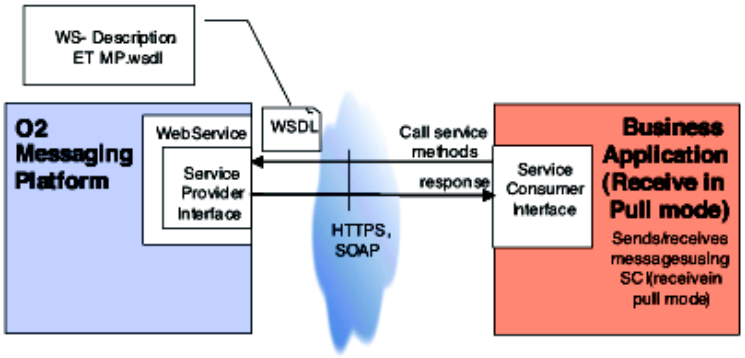


Figure 1: Web Services Architecture for “Receive in Pull Mode”

2.2.2 Receive in Push Mode

In case the customer chooses to receive the messages in Push mode (which means that the platform actively pushes the messages to the business application), the architecture is more complex. See Figure 2 for reference:

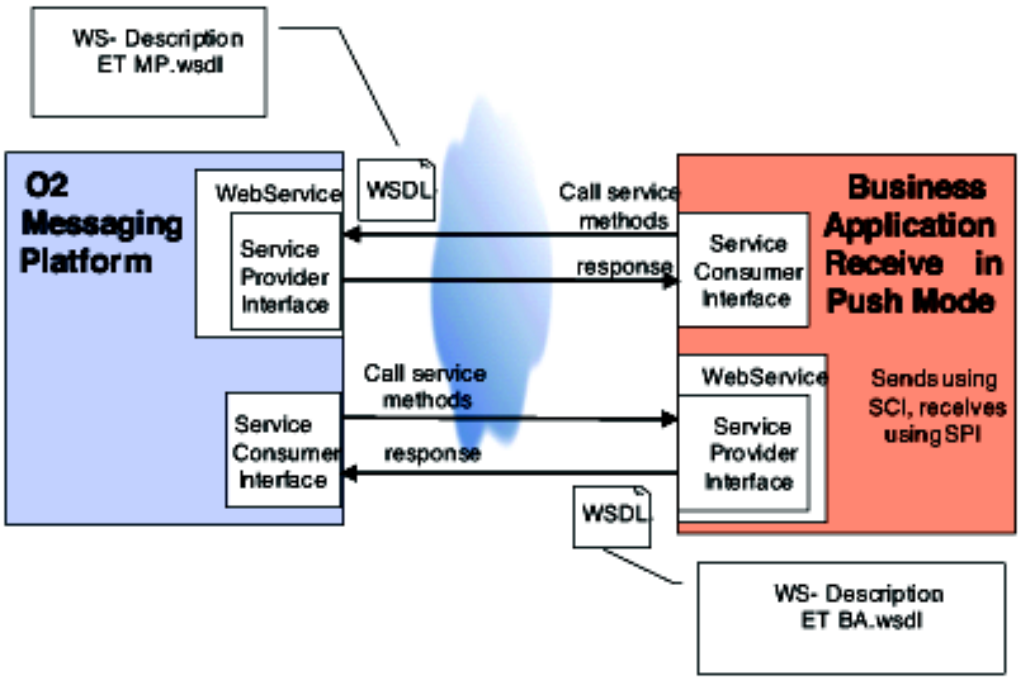


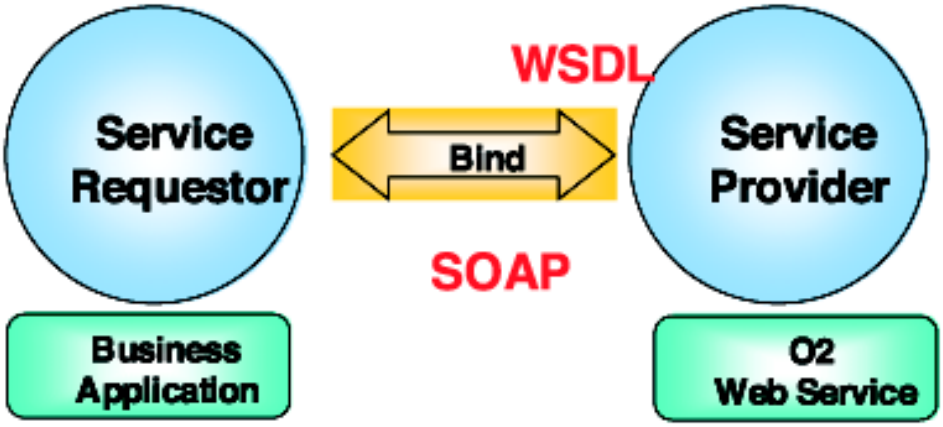
Figure 2: Web Services Architecture for “Receive in Push Mode”

For the sending of messages the service provider interface of the messaging platform Web Services are still used (see above). For the receiving of messages the business application acts as a Web Service provider and the messaging platform as a service consumer. This means that the business application will have to implement a Web Services server. It provides a simple interface for the messaging platform
to transmit the messages to the business application. This server-side Web Service is described via the ET BA.wsdl.
Push applications must implement the „pull mode“ Web Services to retrieve messages that were not delivered using the push mechanism.
This WSDL uses a similar schema (data types, messages) as the ET MP.wsdl but it offers a subset of its functionality. For details concerning the WSDL files see chapter 9.

3 Web Services Architectural Introduction

Web Services, in basic terms, are services offered by one application to other applications via the internet. They are standards-based, interoperable, self-contained, modular applications that can be described, published, located, and invoked over the internet.
The following diagram gives a conceptual overview of the general Web Services architecture. The service requester (or client), which in our case is the business application that wants to send/receive messages, connects to the service provider (O2 Web Service) to make use of its messaging functionality.
The service is described by the Web Services Description Language (WSDL) description, which the service provider offers. Today’s standard for the messaging protocol between service requester and service provider is the Simple Object Access Protocol (SOAP).

In the following chapters the relevant standards are explained briefly to provide a basic understanding of the solution.



3.1 Applicable Standards

The following standards and versions are used:
HTTPS transport protocol based on HTTP1.1 using client authentication (x.509v3 client certificates)
SOAP 1.1 as messaging standard
WSDL1.1 for the service description
XML schema 1.0 for schema definitions
The Web Service implementation will conform to WS-I Basic Profile 1.0, which defines a common interoperability standard for Web Service implementations. It is based on the XML Schema 1.0, SOAP 1.1, and WSDL 1.1 standards.

3.2 Web Service Description Language (WSDL)

Web Services Definition Language (WSDL) is the standard for Web Service descriptions; it is an XML format for describing Web Services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The WSDL files are stored in a web application and are available to business partners for download. The URLs are deployment parameters named pullWsdlUrl for Pull mode services and pushWsdlUrl for Push mode services.
The O2 project has adopted WSDL Version 1.1 (http://www.w3.org/TR/2001/NOTE-wsdl-20010315).

A WSDL document uses the following elements in the definition of network services:
Types – data type definitions using some type system (e.g., XSD).
Message – an abstract, typed definition of the data being communicated.
Operation – an abstract description of an action supported by the service.
Port Type – an abstract set of operations supported by one or more endpoints.
Binding – a concrete protocol and data format specification for a particular port type.
Port – a single endpoint defined as a combination of a binding and a network address.
Service – a collection of related endpoints

3.3 SOAP

The current industry standard for XML messaging is the Simple Object Access Protocol (SOAP). O2 will use SOAP version 1.1 as defined at the reference: http://www.w3.org/TR/2000/NOTE-SOAP-20000508.
HTTP will be used as the underlying transport protocol. SOAP is a simple and lightweight XML-based mechanism for exchanging

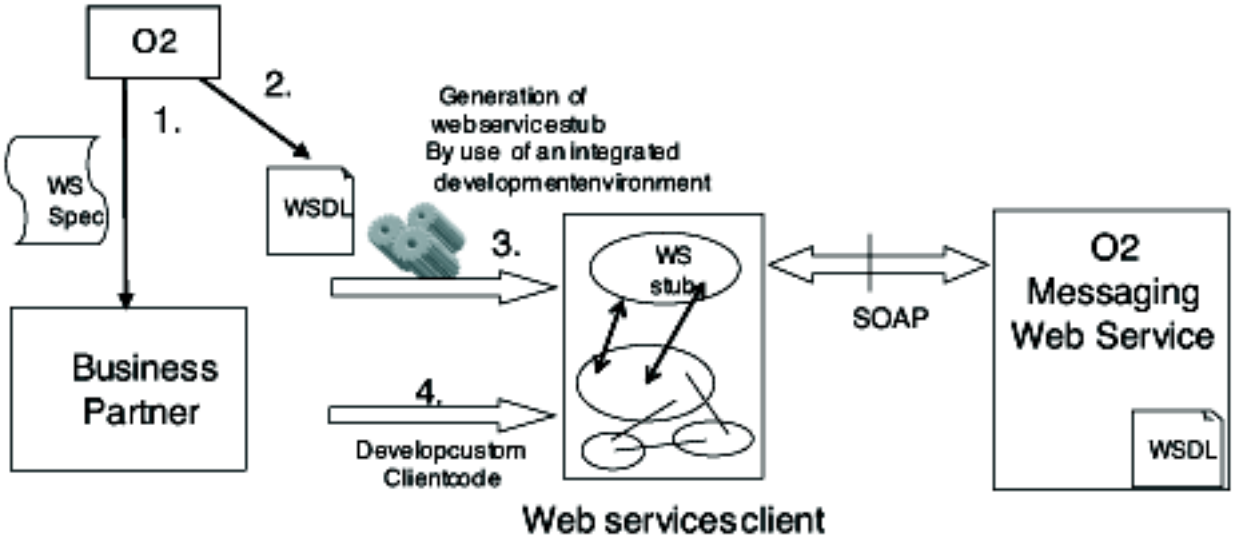
structured data between network applications.
A SOAP message consists of three parts:
The SOAP envelope – defines the overall framework for expressing what is in a message; who should deal with it, and which of its sub-components are optional or mandatory.
The SOAP encoding rules – defines a serialization mechanism that can be used to exchange instances of application-defined data types.
The SOAP RPC representation – defines a convention that can be used to represent remote procedure calls and responses.
Due to compatibility concerns the “Document – literal” encoding style was selected over the RPC style.

3.4 XML Schema Definition – XSD

XML Schemas provide a means for defining the structure, content and semantics of XML documents.
The business partner interfaces solution makes use of XSD to define the structure of the input (request) and output (response) messages. For more information about the XSD standard please refer to: http://www.w3.org/TR/xmlschema-0/

3.4.1 Creating Business Applications

For most partners, the implementation consists in implementing and deploying the Web Services client implementation, except in the Push mode where the roles are inverted, i.e. the business application becomes the Web Service server. This chapter describes the more common process of generating and developing the Web Services client.
The typical workflow for the development of a client application for a Web Service is visualized in the following diagram:



1. O2 provides the partner that has decided to implement an application based on the O2 messaging Web Services, with the detailed documentation of the Web Services (this document). The document contains the business description of the Web Services as well as the technical description (architecture, message flows, functionalities and restrictions of the interface, etc.).

2. In parallel, the business partner accesses the WSDL file, which is a technical means for describing the Web Service interface. The WSDL can be used by a commonly available integrated development environment (e.g. IBM WebSphere Studio Application

Developer or Microsoft .NET) to generate the Web Services stub components. For a deeper explanation of the examples see sections 12 and 13.

3. Most IDEs support the automatic generation of a Web Services test client. Since the O2 messaging Web Services are based on widely adopted standards, the business partner is free to choose the tool and programming language best suited to its current infrastructure, e.g. Java, J2EE or .NET.

4. The client code can now be developed on top of the generated stub without needing to have any knowledge of the underlying Web Services technology. Samples of this IDE functionality are provided later in this document (see sections 12 and 13).

3.4.2 Platforms for BA Development

Since Web Services have become a widely adopted standard for the interoperation of business applications, there are several development tools available, which support the development of Web Services and Web Services clients. The O2 MP Web Services comply with the Web Services standards as described in chapter 3.1.
The client business application must implement a Web Services client runtime platform supporting SSL client-side authentication.
The following platforms and development tools have been tested by O2:
Microsoft .NET 1.1 and Visual Studio.Net 2003
BEA WebLogic 7.0 and BEA Workshop 7.0
IBM WebSphere 5.0.2 and WebSphere Studio Application Developer 5.1
AXIS 1.1
Systinet Wasp 4.5, 4.6
SUN Web Services Developer Pack 1.2
It is also possible to use different platforms conforming to the standards described but with the risk of interoperability problems.

3.4.3 Business Application Development in Push mode

As discussed previously, the development process is very similar to the pull mode one, except that the partner needs to create the Web Service server implementation. As with the client part, a tool may be used to generate all the necessary code for the implementation as a Web Service, this approach is called the Top Down path.

1. The WSDL is provided by O2, this is what is referred to as the ET BA.wsdl in the rest of this document.
2. In the top down path, usually the developer generates the skeleton with the methods and parameters that you have to fill in to implement the Web Service.
3. Using the skeleton, the partner then completes all the methods with the appropriate logic. 15
4. Finally, the partner needs to test the new Web Service to check that it is compliant to the specification with a client implementation that could also be generated automatically by the IDE or by writing a java client.

The application is required to comply with the security policies of O2's Messaging Platform. In Push mode the application can authenticate itself using any of the certificates provisioned in O2's Messaging Platform. See sections 6 and 7 for details.
The service definition for the push mode (ET BA.wsdl) is a subset of the ET MP.wsdl.
Note that in the case of the push mode, O2 will develop a Web Services client that will use the specific location of the Business Application to send the data to the partner.
The supported platforms and recommended tools are the same as for the pull mode.

3.4.4 Access to O_2 Messaging Platform

The O_2 Messaging Platform can be accessed through the following URLs:
https://smsconnector.cz.o2.com/smsconnector/services/PPGwPort for generic messaging operations and ping; and
https://smsconnector.cz.o2.com/smsconnector/services/SimpleSmsGwPort for Simple SMS sending method.
The WSDL file describing both services (same file for both urls) is accessible via url:
https://smsconnector.cz.o2.com/smsconnector/services/PPGwPort?WSDL, or
https://smsconnector.cz.o2.com/smsconnector/services/SimpleSmsGwPort?WSDL.
For limited time period older URLs will be also supported, but business partners are adviced and kindly

asked to use new O_2 URLs.
Older URLs are:
https://smsconnector.eurotel.cz/smsconnector/services/PPGwPort
https://smsconnector.eurotel.cz/smsconnector/services/SimpleSmsGwPort
https://smsconnector.eurotel.cz/smsconnector/services/PPGwPort?WSDL
https://smsconnector.eurotel.cz/smsconnector/services/SimpleSmsGwPort?WSDL.
For detailed description please refer to section 5.1.

4 Interactional Behaviour of the Interface

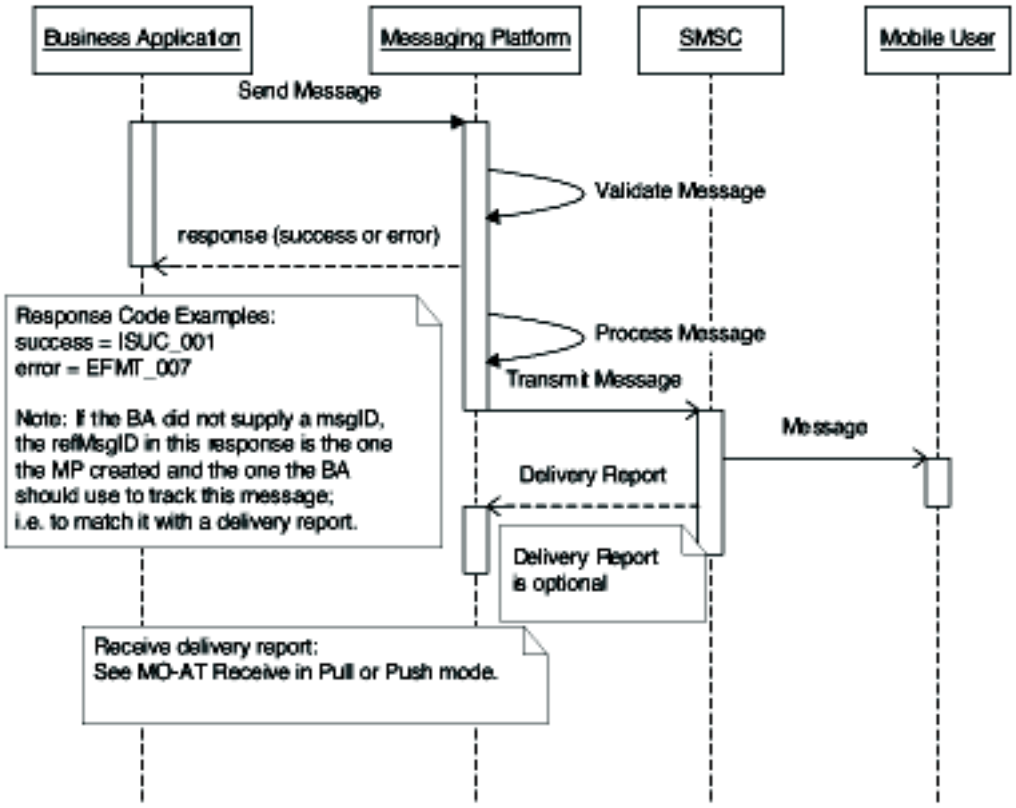
The messaging platform provides support for sending and receiving messages. The MP interface uses a generic messaging protocol offering a means of transporting messages and responses asynchronously. The „response“ of the MP to certain actions may arrive synchronously (instantly as a return value of the called method) or asynchronously (later as a separate message delivered similar to a MO message). The following types of messages can be received from the MP asynchronously:
MO messages (TextSMS, BinarySMS, MMS Objects)
Asynchronous responses (messages of type Response, typically errors related to AO messages)
Special kind of asynchronous responses: status reports in case these were requested by BA for AO messages
The last 2 messages are of Response type, see data specification 5.2
The interface allows parallel processing of messages: e.g. the BA may use several threads for retrieving messages from the MP. Each message is uniquely identified in the interface, so parallel processing is possible. The MP supports two connectivity modes for the business applications. See chapter 4.2 for more information on the pull mode. For reaching higher throughputs the use of the push mechanism is recommended, see chapters 4.3 and 4.5.
The following is common for all communication processes originated from the business application side: The MP performs the authentication of the BA (validity of the certificate and valid reference to the BA ID). This is not explicitly shown in all flow diagrams for better readability. Please refer to the security chapter 6 for more details.

4.1 Sending AO-MT Messages

The BA sends a message to the MP using the send method. The MP performs only the basic checks synchronously and returns a Response object with success (if message was accepted) or an error (the message is not accepted and will not be processed). Additional responses may be produced during the processing of the message and these are sent to the BA asynchronously. See chapter 14 for more detailed information about response codes.

If the BA requests a status report, the MP sends a status report message to the BA according to the selected report level. See section 5.2 for valid report levels. The business application may incur additional charges if the report level is set to "All". Status reports and asynchronous responses are retrieved by the BA using either the pull or push mode. If the BA requests a delivery report for a multipart SMS, the delivery report will be successful only if all parts arrive at the mobile. If one or more parts do not arrive, the delivery report will return false. There is a maximum AO message (SMS and MMS) throughput for each BA and the Messaging Platform has a throttling mechanism to ensure this limit is not exceeded. If this limit is exceeded calls to send() will slow down

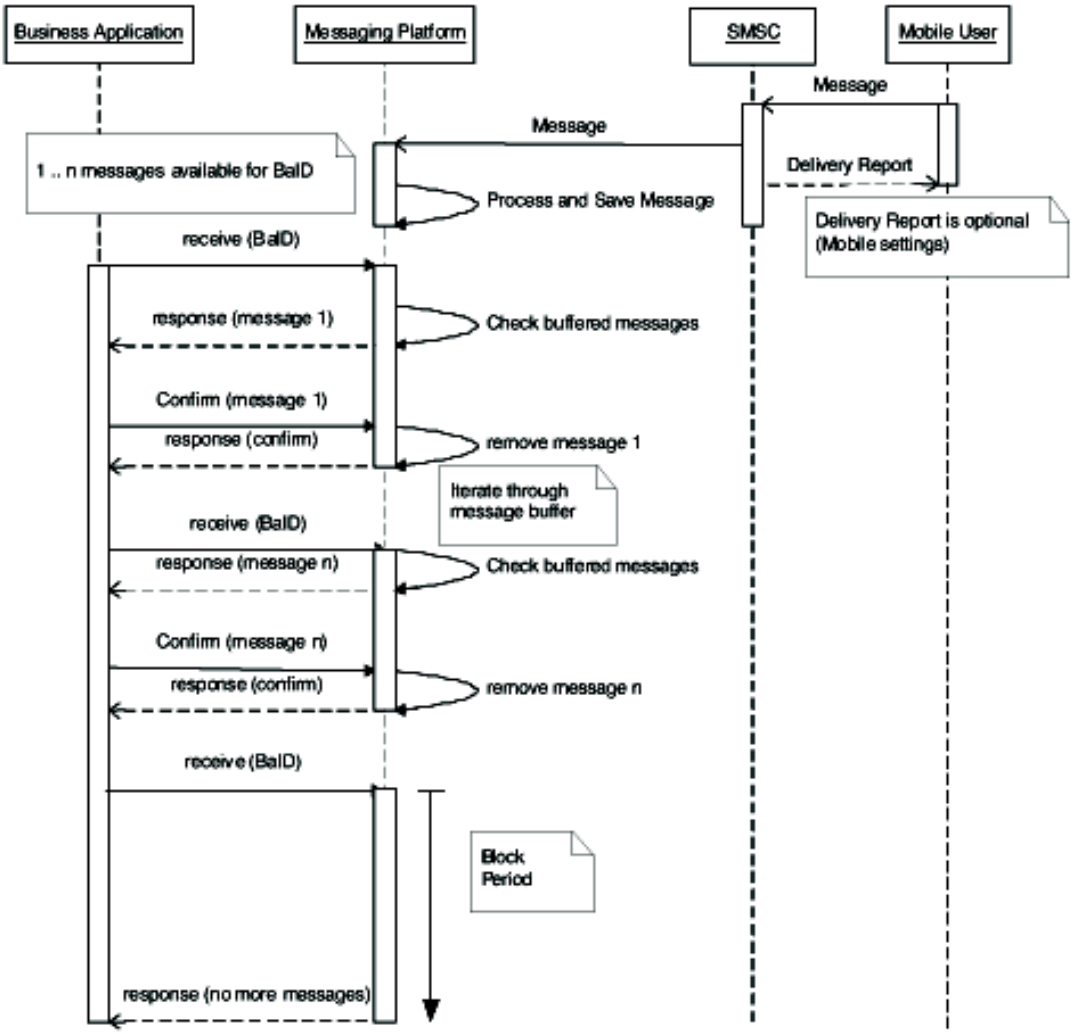
The following diagram shows the typical message flow:



4.2 Receiving MO-AT Messages in Active Pull Mode

For receiving messages the BA connects to the MP by invoking the receive method. If a message is available the MP responds by sending the first message from the delivery queue. If the platform has no messages in the BA queue, the call will be blocked for a timeout period defined by O2 (deployment specific parameter receiveBlockPeriod; see chapter 18 for more detail) or until a message is available. If a new message arrives in the BA queue during the block period, the message will be returned to the BA immediately. If no message is available and the timeout is exceeded, the BA will receive a nil as a return value. The BA confirms the receipt of a message by invoking the confirm method of the messaging platform. After confirmation the messaging platform is able to delete the message from its delivery queue. If there are several messages in the delivery queue, the business application repeats the receiveconfirm sequence until the messaging platform returns nil from receive, meaning that there are no more messages. As described above, this is done according to the First In First Out (FIFO) principle. If the MP queue is already empty (nil is received during receive action), application must not confirm such MP response.
If one or more messages are not retrieved by the BA these will be deleted after a certain time period preset by O2 (deployment specific parameter messageReceptionTimeout; see chapter 18 for more detail)
If a message was received but not confirmed within a certain time (deployment specific parameter confirmationTimeout; see chapter 18 for more detail) it is put back on the front of the queue available for polling. Please note that the overall message expiration timeout (deployment specific parameter messageReceptionTimeout) is completely independent of the receive-confirm process.
This means that the number of times a BA receives a message but does not confirm it, has no impact on when the message will expire and be deleted from the MP. If BA confirms message that has not been received or has not been confirmed after confirmationTimeout, error EAPP_025 will be returned. The MP supports a limited number of unconfirmed messages and this is defined in deployment specific parameter maxUnconfirmedMessages. If this limit is exceeded the receive() method of the MP returns a

synchronous Response, with code EAPP_037, and the BA will not be able to receive messages until the number of unconfirmed messages is reduced below the limit. This can be achieved when the BA issues an explicit confirm or the confirmation timeout is exceeded. BA should not confirm synchronous responses returned by the receive() method. These are typically technical errors encountered during receive action, for example response with EAPP_037 code mentioned in the above paragraph. Synchronous responses can easily be distinguished from the asynchronous ones, as the field refMsgID has null value for synchronous responses. The following diagram depicts the typical message flow:



4.3 Receiving MO-AT Messages in Push Mode

In push mode the messaging platform actively connects to the business application. It calls the send() method of the interface to submit the message. The response of the business application contains the status of the delivery, such as success or an error message.
If one or more messages are not retrieved by the BA these will be deleted after a certain time period preset by O_2 (deployment specific parameter messageReceptionTimeout; see chapter 22 for more detail)
In certain cases, the BA will not be able to receive the message "pushed" to them from O_2.
The following cases may occur:
Case A: Unable to reach partner Web Service because of
1. Network related problems;
2. Web Service Server is not running in the partner environment);
3. SSL connection problems;
4. The location URL of the Web Services server contains an error in the web application path part.

Case B: BA receives call but generates an Error Response for a number of reasons including:
1. The Web Services server implementation has an error/exception for example in the Web Services server implementation call other systems that are down;
2. Authentication error – mismatch of the client certificate used in the connection.

In both cases the message is not lost and the O2 Messaging Platform will try to re-deliver the message every T1 seconds and continue to do so until the message expires or is successfully delivered. The message can also be retrieved using the pull mechanism. T1 is reflected by deployment specific parameters (T1 = sendRetryInterval; see chapter 22 for more detail) defined by O2 and are the same for every BA.

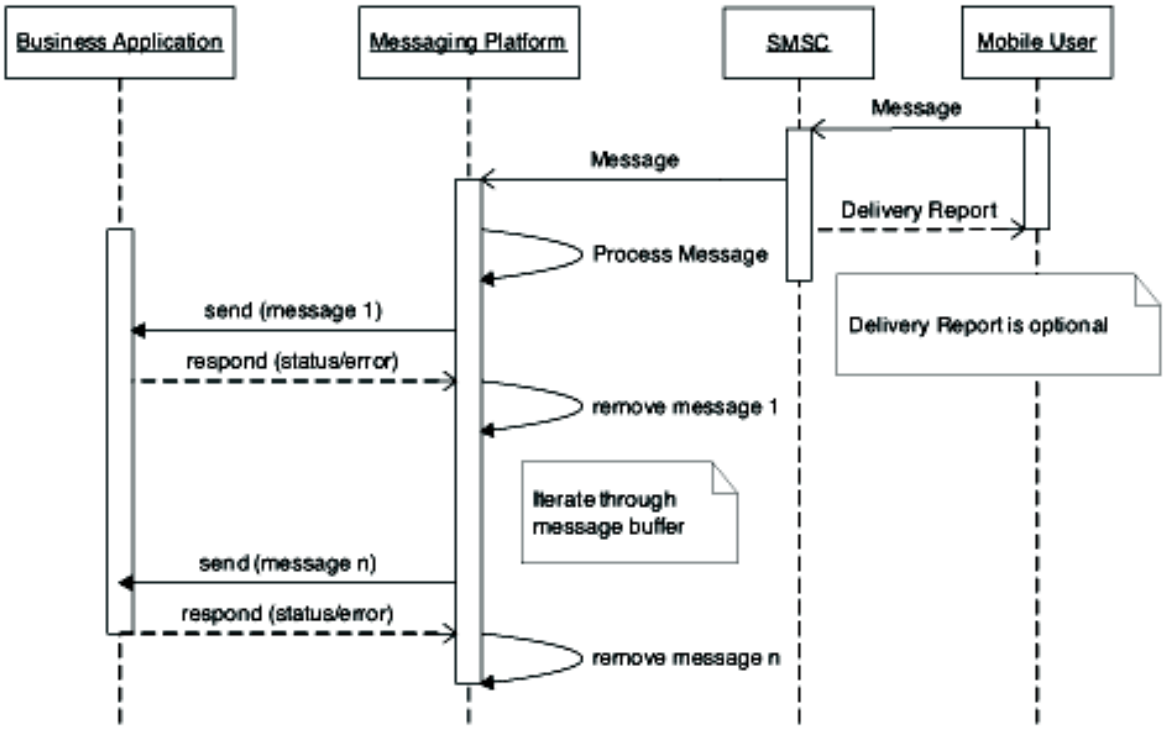
The MP uses parallelism and it is therefore possible for the MP to retry sending several messages concurrently. There is a maximum number of parallel threads that the MP will use when sending messages to a BA in push mode (see the deployment specific parameter maxPushThreads).

In case B the error message will be recorded by the MP for further investigation.

If the message cannot be "pushed" after the N attempts (deployment specific parameter sendRetryCount; see chapter 22 for more detail) and no new messages arrive in the queue, it is available to the BA via pull mode. Once the message is undeliverable in push mode, it can only be retrieved in pull mode. It will no longer be available in push mode.

There is no confirm action in Push Mode.

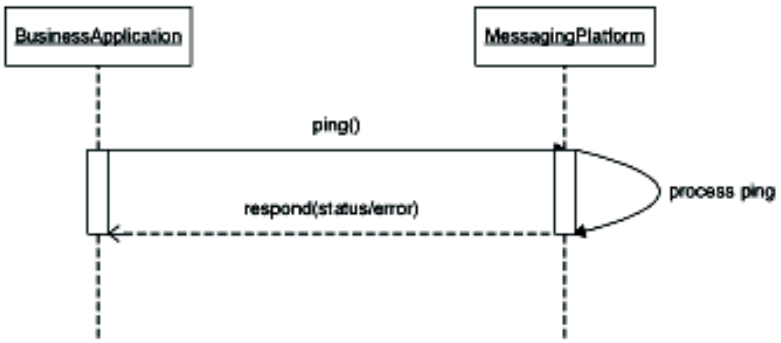
The following diagram depicts the typical message flow:



Note that if in the meantime the BA retrieves messages through the pull mode, it will be able to retrieve all the new messages including any that were not sent via the push mode and are currently not in a retry cycle.

4.4 Ping

A simple ping method is available to the Web Service to provide a means of checking the connectivity as the following sequence shows:



This ping() method can be used to test connectivity to the O2 Messaging Platform. If the BA gets a response with a success code ISUC_000, it implies that messaging is available for that interface. An error code implies that the interface is there but there is a special problem that can be identified through the response code. If no response is received it means that the Web Services interface is not available, for example through network problems.

4.5 Multi-Threading

The MP allows BAs to process messages in multiple threads. However there is a limit to the number of threads each BA is allowed for each method. If the limit is exceeded the action is rejected and a synchronous Response with an error code, such as EAPP_053, is returned to the BA. The actual thread limits are deployment specific parameters: sendThreadLimit, receiveThreadLimit and confirmThreadLimit. Thread limits for all other methods are set to 1.

5 Functional Description of the Interface

This chapter describes the O2 messaging Web Service interface methods and data from a functional perspective. The exact technical definition is provided in the chapter Appendix A: MessagingGatewayPort.wsdl (uncommented) by use of the WSDL description.

5.1 Methods

The following gives an overview of the available methods. The interface will be subdivided into four ports:
Generic messaging operations and ping (PPGw)
Simple SMS sending method (SimpleSmsGw)
PUSH –mode methods (PushPPGw) {to be implemented by business application}
The following subchapters depict these parts of the messaging interface.

5.1.1 Generic messaging operations and ping (PPGw)

The O2 messaging platform Web Service interface provides a generic messaging protocol as described in chapter 4.
This interface offers three main methods, through which the described services are used for SMS Connector:
send
receive
confirm
In addition the interface offers the following method to test connectivity
ping

MessageContainer
Due to the lack of support for polymorphism in the implementation of Web Services, the MessageContainer has been introduced as a workaround. This allows passing various object types as method arguments or return values. The MessageContainer serves as a wrapper for the different message types. The selector attribute functions as a key which tells both the BA and the MP which one, and only one, other MessageContainer attribute (textSms, binarySms, mms or response) is required to be filled.
The MessageContainer may contain only one message type (text SMS, binary SMS, MMS or Response).
The possible value pairs for the MessageContainer are:

selector value	assigned attribute
„TextSms“	textSms not null -> all other null
„BinarySms“	binarySms not null -> all other null
„Mms“	mms not null -> all other null
„Response“	response not null -> all other null
„TextSmsWithTime“	textSmsWithTime not null -> all other null
„BinarySmsWithTime“	textSmsWithTime not null -> all other null

Aside from the selector, only one other attributes can be set. The others must be null. The table above shows the attributes which have to be set according to given selector value. The parameters and the result of the methods depend on the context of the operation. E.g. the send operation may send a text SMS (MessageContainer contains a TextSms object) as well as the acceptance of a message (MessageContainer contains a Response object with an accept code).

send (MessageContainer) -> Response

The method send(...) is used by the BA in the following cases:
Sending of a SMS or MMS (objects: Sms, BinarySms, TextSms, Mms) from the BA to the MP The method send(...) is also used by the MP to:
Transmit messages to the BA in Push mode (see 4.2).

confirm (String baID, String refBaID, String refMsgID) -> Response
The method confirm() confirms the reception of an AT message. The referenced business application ID and the referenced message ID are copied from the message being confirmed.

receive (String baID) -> MessageContainer
The method receive is used to pull messages for the business application ID from the messaging platform. It returns the first message in the message queue. In case no message is available, a nil value is returned.

ping (String baID) -> Response
The ping method is used to test connectivity and the authentication mechanisms. It checks the validity of the certificate and whether the certificate and the BA ID transmitted match the ones registered with O_2. It returns a Response object with status or error information.

5.1.2 Simple SMS sending method (SimpleSmsGw)

This method is provided for business partners that do not wish to implement the above described functionality due to its complexity. It is available on a dedicated Web Service port.

sendSms(String baID, String text, String toNumber, String fromNumber, long validityPeriod, String priority, boolean intruder) -> String
This method takes the mandatory attributes of a text SMS as input parameters (except msgID) and sends the SMS in the same message flow as described above for the send method. For the other attributes of the SMS, defaults will be applied. The return parameter is a response code.

5.1.3 PushPPGw methods

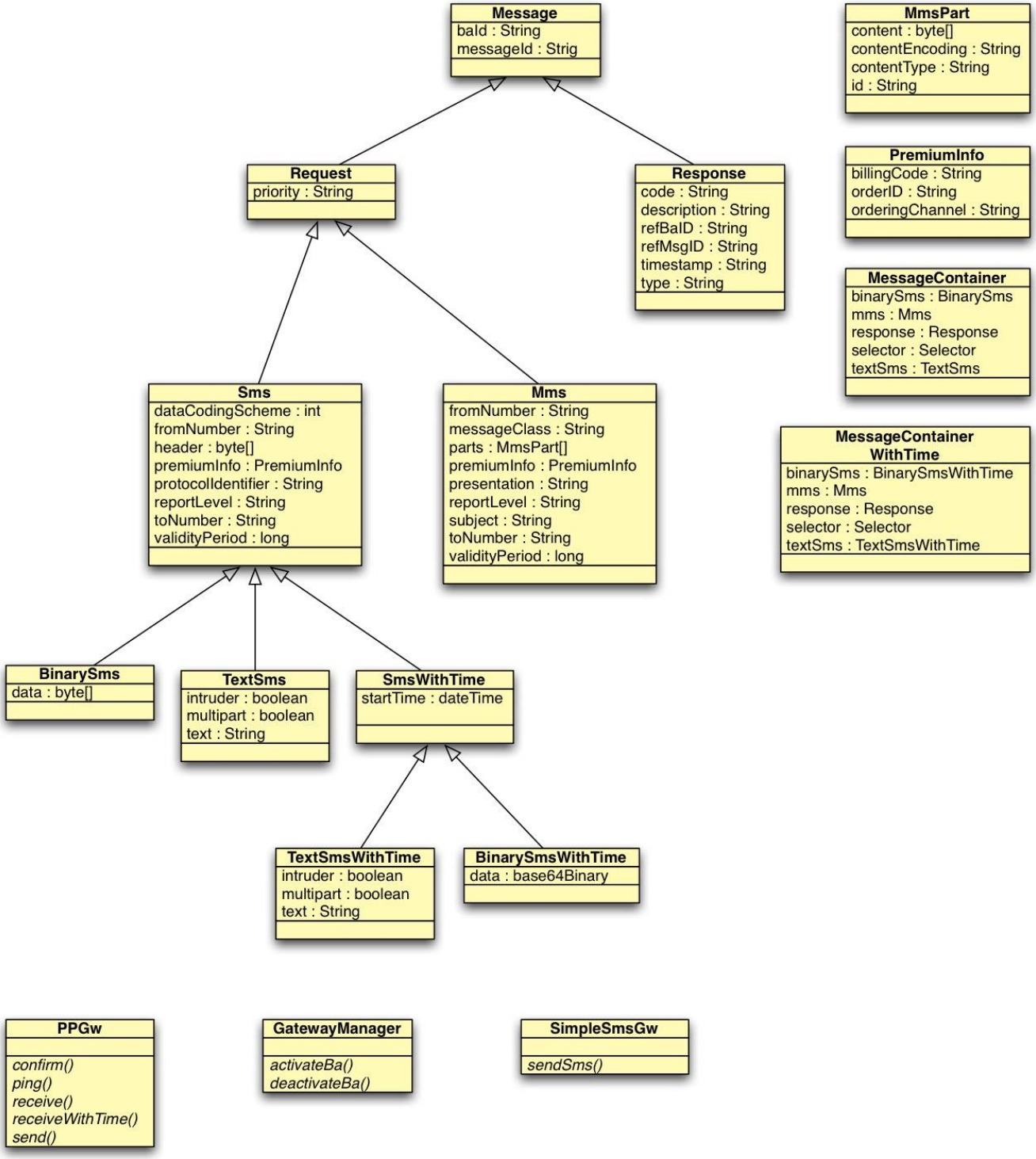
As mentioned before the push mode enables the MP and BAs to implement a high-capacity way for communication. Therefore the business applications can implement the PushPPGw Web Services port that acts as a Web Services endpoint on the BA side.

send (MessageContainer) -> Response

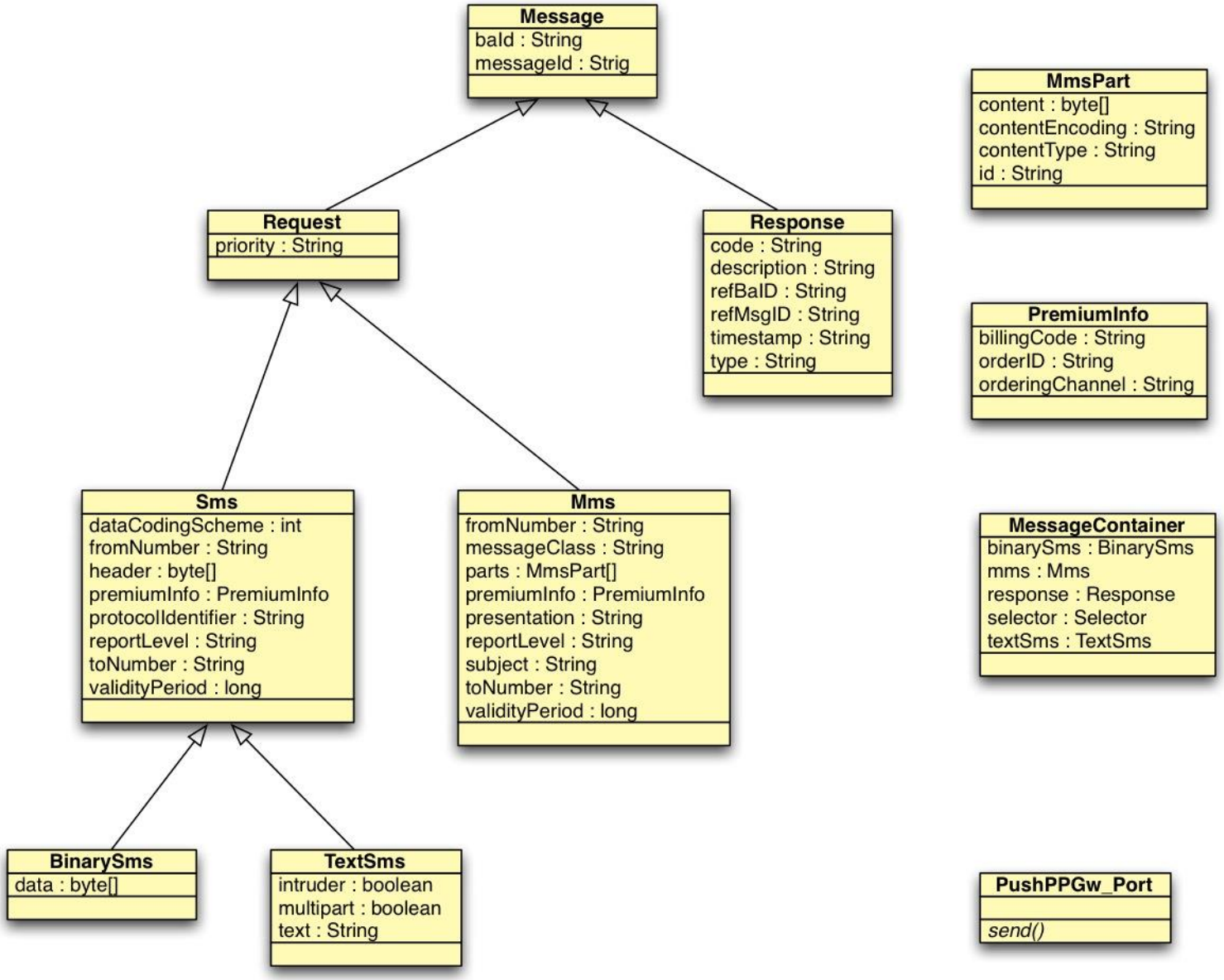
The method send() is used by the MP to send messages to the BA. The message is considered confirmed if the BA returns the success response; otherwise the MP will retry sending the message.

5.2 Data

This chapter explains the fields used in the interface for the particular cases. The data are referenced by using data objects and their respective fields. The following diagram depicts the object hierarchy:


Message
baId : String
messageId : Strig
MmsPart
content : byte[]
contentEncoding : String
contentType : String
id : String
Request
priority : String
Response
code : String
description : String
refBaID : String
refMsgID : String
timestamp : String
type : String
PremiumInfo
billingCode : String
orderID : String
orderingChannel : String
MessageContainer
binarySms : BinarySms
mms : Mms
response : Response
selector : Selector
textSms : TextSms
Sms
dataCodingScheme : int
fromNumber : String
header : byte[]
premiumInfo : PremiumInfo
protocolIdentifier : String
reportLevel : String
toNumber : String
validityPeriod : long
Mms
fromNumber : String
messageClass : String
parts : MmsPart[]
premiumInfo : PremiumInfo
presentation : String
reportLevel : String
subject : String
toNumber : String
validityPeriod : long
MessageContainer WithTime
binarySms : BinarySmsWithTime
mms : Mms
response : Response
selector : Selector
textSms : TextSmsWithTime
BinarySms
data : byte[]
TextSms
intruder : boolean
multipart : boolean
text : String
SmsWithTime
startTime : dateTime
TextSmsWithTime
intruder : boolean
multipart : boolean
text : String
BinarySmsWithTime
data : base64Binary
PPGw
confirm()
ping()
receive()
receiveWithTime()
send()
GatewayManager
activateBa()
deactivateBa()
SimpleSmsGw
sendSms()

The following diagram depicts the object hierarchy for PUSH:



Each message must be of type Request or Response. Requests have one additional attribute, priority, that defines the message priority. The Response message provides more information by having the attributes:
- refBaID = the baID of the request to which this response belongs to
- refMsgID = the msgID of the request to which this response belongs to. It contains null value for synchronous responses
- type = the type of the response: e.g. SUCCESS, FORMAT_ERROR
- code = identifies the special case: e.g. ISUC_001 of the response
- description = the textual description to the response code
- timestamp = the timestamp when the response was created

A Request may be one of three different message types. These are Mms, TextSms and BinarySms. There is a generic Sms type that holds all the data that is common to TextSms and BinarySms.

The following table shows the data transmitted at the Web Services interface. It depicts, which data are sent for which application and which type of message.

The MP makes a distinction between a null value and an empty string or value. In XML and Web Services, some elements may be made null (or “nil” in XML). This is accomplished by setting the xsi attribute of the XML element. The following examples illustrate this:

null - <abc xsi:nil=”true” /> (note: that the element must defined as nillable in the WSDL file, such as <element name="abc" nillable="true" type="string"/>)

empty - <abc /> or <abc> </abc>

If the BA is using code automatically generated by the WSDL, this should all be transparent to the BA.

Columns of the table below:
Class: object according to the figure above
Field name: object attribute according to the figure above
Mand.: marks that this field is mandatory
Def.: default if any. This may be a reference to a deployment specific parameter and is denoted by * the parameter name.
AO: marks that this attribute applies to Application-originated messages
AT: marks that this attribute applies to Application-terminated messages
Note: Some items in the table are marked as default and mandatory. This at first glance may seem unnecessary, but was required for certain cases; such as: a mandatory field is defined as an integer. The default value is 0, not null. This makes it a mandatory field with a default value.

Class	Field Name	Mand.	Def.	AO	AT	SMS Connector Description
Message	baID : String	X		X	X	7 digit identifier of the Business Application from O2's 2-Way SMS numbering range, MSISDN assigned to the application starting with 199. .g. 1991123. may be 6 digits in case of application migrated from the BMG service (starts with 99).
Message	msgID : String	X		X	X	Unique message identifier. For AO messages it must be provided by theBA; in case the field is not filled, the message will be rejected by the MP. A recommendation is to include timestamp in the message ID. messages, the MP will provide a unique message identifier and sends this to the BA. Any errors or status reports generated during processing of an AO message will reference the msgId by use of refMsgId. To guarantee proper processing, the maximum length of the msgID should not exceed 128 characters and msgID should not contain semicolon (";") character.
Request	priority : String		defaultP	X		Requested priority of the AO message. Permitted values are: 1 – VIP message priority1

1 Please see the Deployment Specific Parameters for the value.

Class	Field Name	Mand.	Def.	AO	AT	SMS Connector Description
						2 – High 3 – Normal The default is a platform parameter and may be changed. The value is described in global parameter defaultPriority (see chapter 18 for more details). The MP considers the priority of the message against the priorities of based on the algorithm defined by O2. Not used for MO messages.
Sms	fromNumber : String		BAID	X	X	Originating number of the SMS. Please see section 5.2 for more details. In the AO direction, this is either the BA ID plus the optional suffix (up to 9 digits long, which could be an order number for example) or MSISDN assigned to BA in international format with a leading “+” sign, country code and national number. An example is +420720001001. In the AT direction, this is the MSISDN of the mobile user who send the message in +420123456789 format.
Sms	toNumber : String	X		X	X	Destination number of the SMS. Please see section 5.2 for more number format details. In the AT direction, this is either the BA ID plus the optional suffix (up to 9 digits long, which could be an order number for example) or MSISDN assigned to BA in international format with a leading “+” sign, country code and national number. An example is +420720001001.

Class	Field Name	Mand.	Def.	AO	AT	SMS Connector Description
						In the AO direction, this is the MSISDN of the mobile user who should receive the message in one of the formats described in section 5.2.
Sms	reportLevel : String		none	X		This is the level of reporting that should be supplied by the MP for this message. This includes delivery report requests. Acceptable values are: (no status feedback even in case of errors) failures (only failures are reported) terminal (delivery reports are reported to BA): recommended value init (additional report that the message was accepted by the platform) all (all statuses are reported) For setting correct setting of delivery reports see also description of field validityPeriod and also Appendix Solution Constraints.
Sms	validityPeriod : long	X	0	X		For AO messages, it is validity period of the message measured in seconds from the time MP received the message from BA. MP calculates absolute time when message should expire by adding validity period to the system time of message receipt. This absolute expiration time is then managed by SMSC. If message cannot be delivered to the mobile user by the given expiration time, SMSC will discard the message. If validity period is set to “ 1”, the message is sent immediately and only once. If value is ”0” (the default when no value I specified) the validity period is managed by MP.

Class	Field Name	Mand.	Def.	AO	AT	SMS Connector Description
						The SMSC controls the final expiration time and it may shorten, lengthen or round the expiration time. This is because SMSCs only support a range of possible values. A hypothetical example is that a validity period of 7 minutes and 5 seconds would be rounded up to 10 minutes. This does not apply to MO messages. For correc setting of validityPeriod value see also Appendix Solution Constraints.
Sms	protocolIdentifier: integer		0	X	X	PID for the SMS protocol. It can be any integer value from 0…255. See the relevant GSM standards for more information.
Sms	dataCodingScheme: int	X	0	X	X	This is the SMS coding standard for this message. It can be any integer value from 0…255. See the relevant GSM standards for more information..
Sms	header : byte[]		null	X	X	Header part of the message. This typically contains UDH (User Data Header) elements carrying various attributes of the message. Using UDH decreases the maximum size of data/text in the SMS message. The first byte always contains the length of the header (UDHL – User Data Header Length).
Sms	premiumInfo		X	X		N/A
PremiumInfo	billingCode : String		X			N/A
PremiumInfo	orderingChannel : String	X (if billing Code is used)	X			N/A

Class	Field Name	Mand.	Def.	AO	AT	SMS Connector Description
PremiumInfo	orderID: String	X (if billing Code is used)		X		N/A
Binary	Sms data	X		X	X	Data body of the binary SMS.
TextSms	intruder : boolean	X	false	X		When set to true, it indicates the message should be shown immediately on the user's mobile phone display. If set to false, the message will be sent as a standard text SMS.
TextSms	multiPart : boolean	X	false	X		Identifies a multipart SMS. For AO SMS true indicates that the maximum text/data size can be exceeded, and the platform will divide the message into the correct number of parts. O2 will charge for each part. Majority of mobile phones will then assemble all parts together and present them to user as a one message
TextSms	text : String	X		X	X	Max 160 chars of user text. If multiPart field is set, user text can be max. 900 characters long. As the maximum length for multipart text SMS is handset-dependent, but the platform limits the maximum length to 900 characters. The BAs are responsible for compatibility with mobile subscribers' handsets. For MO messages the limit also depends on the mobile phone. In AO direction, all characters must have their counterparts in 7-bit default GSM encoding (defined in GSM 03.38),

Class	Field Name	Mand.	Def.	AO	AT	SMS Connector Description
						which also means that characters with Czech diacritics are not supported. The MP forwards messages as they come without changing the content. The MP does not forward AO messages containing invalid characters.
Mms	premiumInfo			X		N/A
Mms	fromNumber : String		BAID	X	X	N/A
Mms	toNumber : String	X		X	X	N/A
Mms	subject : String			X	X	N/A
Mms	reportLevel : String		none	X		N/A
Mms	validityPeriod : long	X	0	X		N/A
Mms	presentation : String			X	X	N/A
Mms	messageClass : String		info	X	X	N/A
Mms	parts : MmsPart[]		X	X	X	N/A
Mms	premiumInfo			X		N/A
Mms	Part content : byte [] (array)		X	X	X	N/A
Mms	Part contentEncoding : String		null	X	X	N/A
Mms	Part contentType : String		X	X	X	N/A
Mms	Part id : String		X	X	X	N/A
Response	refBaID : String		X	X	X	The baID from the request message that generated this response.
Response	refMsgID : String		X	X	X	The msgID from the request message that generated this response.
Response	type : String		X	X	X	Type of the response. See section 5.3 for more details. SUCCESS – positive result

Class	Field Name	Mand.	Def.	AO	AT	SMS Connector Description
						AUTHORIZATION_ERROR – authorization/authentication related error FORMAT_ERROR – one or more message parameters have the wrong format APPL_ERROR – other application specific errors INTERNAL_ERROR – serious internal error on the MP
Response	code : String		X	X	X	Code from an agreed list of response codes. See chapter 5.3.
Response	description : String			X	X	Text description that is human readable.
Response	timestamp : DateTime		X		X	Time stamp of the event , date & time of the generation of the Response. This is local time.
MessageContai	selector : String	X		X		Defines the type of message in the container. Selector can have the values: TextSms BinarySms Response

Class	Field Name	Mand.		Def.	AO	AT	SMS Connector Description
SmsWithTime	startTime : dateTime						Sms start time
TextSmsWithTime	intruder : boolean	X	false	X			When set to true, it indicates the message should be shown immediately on the user's mobile phone display. If set to false, the message will be sent as a standard text SMS.
TextSmsWithTime	multiPart : boolean	X	false	X			Identifies a multipart SMS. For AO SMS true indicates that the maximum text/data size can be exceeded, and the platform will divide the message into the correct number of parts. O2 will charge for each part. Majority of mobile phones will then assemble all parts together and present them to user as a one message
TextSmsTime	text : String	X		X	X		Max 160 chars of user text. If multiPart field is set, user text can be max. 900 characters long. As the maximum length for multipart text SMS is handset-dependent, but the platform limits the maximum length to 900 characters. The BAs are responsible for compatibility with mobile subscribers' handsets. For MO messages the limit also depends on the mobile phone. In AO direction, all characters must have their counterparts in 7-bit default GSM encoding (defined in GSM 03.38),
BinarySmsWithTime	Sms data	X		X	X		Data body of the binary SMS.

5.2.1 Mobile Number Formats

The formats differ for MSISDNs and for applications as described below:
The MSISDN in international format with a leading “+”, country code, and a national number. An example for a Czech number is +420602001234. For Czech numbers, the MP will check that the number is exactly 12 digits long, including country code, so +420123456789.
The application number is 5-16 digits, always starting with the partner prefix. An example of a SMS CONNECTOR number is 1991123.

5.3 Response Codes

The MP returns the status as a Response object in a synchronous or asynchronous fashion.
Synchronous means the response/ return value of the called method (e.g. response for send method).
Asynchronous means that the MP sends a message to the BA’s message queue after the originating call s finished. An example of this would be a status report. Asynchronous responses are received in the same way as any other message, such as a text SMS. The type of the Response will always be a response object type (see above).
BAs in push mode will send back Response objects to the MP.
There are five types of Responses that the messaging platform will return. They are described in the tables below.

Response Type	Value	Synchronous /asynchronous	Definition
Format Error	FORMAT_ERROR	Synchronous	One or more message parameters are in the wrong format
the wrong format ion Error	Authenticat RROR	AUTHORIZATION_E	Synchronous An error related to authentication
Application Error	APPL_ERROR	Asynchronous	A negative result of the message A negative result of the message processing generated by the Business Application or MP either related to the content of the message or to the functionality.
Internal Error	INTERNAL_ERROR	Synchronous + Asynchronous	A serious error in the O2 Messaging Platform or in the BA in push mode.
Success	SUCCESS	Synchronous + Asynchronous	Synchronous responses for success and asynchronous deliveryReports.

The following table presents example contexts for when these response types might happen (while sending of AO messages or while receiving AT) and how the BA may react on these response types.

Respons e Type	Context/operati on	Action for BA
Format Error	Send	BA should correct the fault of the field (e.g. enter missing field) before retrying the action.
Authentication Error	All operations	Check whether BA ID is right and is provisioned for this certificate. Check whether certificate has expired or is not registered at O2
Application Error	Send, receive	BA should correct the fault, e.g. insert only allowed pricing levels, before retrying the action. Note: undeliverable messages are reported as type Success.
Internal Error	All operations	Retry action or escalate to user.
Success	Send, Receive	Message accepted by MP, message status reports

We recommend that the BA does a plausibility check for invalid formats or missing fields on its side before sending a message. This will improve the performance, because these basic errors we will be caught immediately instead of waiting for the MP to send back an error response. In chapter 15 there is a list of all response codes categorized according to the described types.

6 Security

The security of the O2 messaging Web Service builds on the use of SSL with client-side certificates. For the BA acting as Web Service client (see 2.2) the BA should use its certificate to connect to the platform, which is done through the use of the standard mechanisms of the chosen platform. The certificate in the request is then validated at 2 levels:
The MP checks the validity of the certificate (validity of Certification Authority and whether the certificate has expired)
The MP then checks the authenticity: whether the certificate matches the given BA ID and whether it fits with the one supplied in the provisioning process for this BA (-> authentication) For the BA using the Push mode (acting as Service provider, see 2.2) the BA has to check the validity and (recommended also) the authenticity of the ET certificate. Usually the first checking mechanism will be provided by the web server infrastructure, the second by the application server runtime components. For Push mode the BA is authenticated using the certificate provisioned in the O2 Messaging Platform. For performance reason the recommended number of certificates should not be higher than 5 certificates per BA. For more detailed information about O2's security policy for using certificates please refer to the document "Policy for Using Certificates – B2B communication between O2 and its Partners", available from O2. For the registration of the BA's certificate (provisioning) at O2 please see chapter 7 or the respective documentation provided by O2. This document assumes that the client is already registered in the O2's user registry.

6.1 Cookies

We recommend that BA accept cookies as we have seen that this may increase performance. BAs may connect to the MP without accepting cookies but the performance may be slower.

7 Registering a Business Application

Each application that wants to send and receive SMS using SMS Connector service needs to be registered by O2. Following steps are needed for application registration:

1. Creating certificate – O2 accepts two types of certificates – certificates from public certification authorities (1.CA, Verisign etc.) or certificate generated by O2. If customer wants to use O2 generate certificate, he needs to download from O2 web pages special application and run it on a local computer. Several company details needs to be specified and application will generate your certificate (public key with extension *.PEM and private key with extension *.P12). Additionally, application will print service activation form.
2. Service activation form – O2 contact person in the company needs to fill in service activation form and contact O2 customer care. Additionally, public certificate key needs to be sent by e-mail to customer care. When activating service, customer needs to specify several important details:
   Voice number to which SMS Connector service will be connected. In case this voice number will be deactivated or suspended, SMS Connector will be deactivated or suspended as well. Used tariff Required capacity limit in number of SMS per second. Standard capacity offered to customer is 1 SMS per second. 5 and 20 SMS per second can be offered for additional fee. Whether PUSH or PULL mode will be used for fetching incoming messages. In case PUSH mode will be used, URL for customer web server needs to be specified Contact person for SMS Connector service – customer contact person, which will be answering inquiries from SMS message recipients Whether customer want to choose its application number (BAID)
3. Activation of the service – Based on information received from customer O2 will activate the service and will inform customer contact person, which application number has been assigned and what are daily and monthly limits of SMS messages. From this moment customer application will be able to send and receive SMS messages.

8 Appendix A: MessagingGatewayPort.wsdl (uncommented)

Remark: for the electronic version of the WSDL please check the WSDL published by the Web Service at the specified URL.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="MessagingGateway" targetNamespace="http://b2b.eurotel.cz" xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://b2b.eurotel.cz"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <types>
        <schema elementFormDefault="unqualified" targetNamespace="http://b2b.eurotel.cz"
xmlns="http://www.w3.org/2001/XMLSchema" xmlns:soap11-enc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                <complexType name="confirm">
                        <sequence>
                                <element name="baID" nillable="true" type="string"/>
                                <element name="refBaID" nillable="true" type="string"/>
                                <element name="refMsgID" nillable="true" type="string"/>
                        </sequence>
                </complexType>
                <complexType name="confirmResponse">
                        <sequence>
                                <element name="result" nillable="true" type="tns:Response"/>
                        </sequence>
                </complexType>
                <complexType name="Response">
                        <complexContent>
                                <extension base="tns:Message">
                                        <sequence>
```

```
                                        <element name="code" nillable="true" type="string"/>
                                        <element name="description" nillable="true" type="string"/>
                                        <element name="refBaID" nillable="true" type="string"/>
                                        <element name="refMsgID" nillable="true" type="string"/>
                                        <element name="timestamp" nillable="true" type="dateTime"/>
                                        <element name="type" nillable="true" type="string"/>
                                </sequence>
                        </extension>
                </complexContent>
        </complexType>
        <complexType name="Message">
                <sequence>
                        <element name="baID" nillable="true" type="string"/>
                        <element name="msgID" nillable="true" type="string"/>
                </sequence>
        </complexType>
        <complexType name="Request">
                <complexContent>
                        <extension base="tns:Message">
                                <sequence>
                                        <element name="priority" nillable="true" type="string"/>
                                </sequence>
                        </extension>
                </complexContent>
        </complexType>
        <complexType name="Mms">
                <complexContent>
                        <extension base="tns:Request">
                                <sequence>
                                        <element name="fromNumber" nillable="true" type="string"/>
                                        <element name="messageClass" nillable="true" type="string"/>
                                        <element maxOccurs="unbounded" minOccurs="0" name="parts"
nillable="true" type="tns:MmsPart"/>
                                        <element name="premiumInfo" nillable="true" type="tns:PremiumInfo"/>
                                        <element name="presentation" nillable="true" type="string"/>
                                        <element name="reportLevel" nillable="true" type="string"/>
                                        <element name="subject" nillable="true" type="string"/>
                                        <element name="toNumber" nillable="true" type="string"/>
                                        <element name="validityPeriod" type="long"/>
                                </sequence>
                        </extension>
                </complexContent>
        </complexType>
        <complexType name="MmsPart">
                <sequence>
                        <element name="content" nillable="true" type="base64Binary"/>
                        <element name="contentEncoding" nillable="true" type="string"/>
                        <element name="contentType" nillable="true" type="string"/>
                        <element name="id" nillable="true" type="string"/>
                </sequence>
        </complexType>
        <complexType name="PremiumInfo">
                <sequence>
                        <element name="billingCode" nillable="true" type="string"/>
```

```
                <element name="orderID" nillable="true" type="string"/>
                <element name="orderingChannel" nillable="true" type="string"/>
            </sequence>
        </complexType>
        <complexType name="Sms">
            <complexContent>
                <extension base="tns:Request">
                    <sequence>
                        <element name="dataCodingScheme" type="int"/>
                        <element name="fromNumber" nillable="true" type="string"/>
                        <element name="header" nillable="true" type="base64Binary"/>
                        <element name="premiumInfo" nillable="true" type="tns:PremiumInfo"/>
                        <element name="protocolIdentifier" nillable="true" type="string"/>
                        <element name="reportLevel" nillable="true" type="string"/>
                        <element name="toNumber" nillable="true" type="string"/>
                        <element name="validityPeriod" type="long"/>
                    </sequence>
                </extension>
            </complexContent>
        </complexType>
        <complexType name="SmsWithTime">
            <complexContent>
                <extension base="tns:Sms">
                    <sequence>
                        <element name="startTime" type="dateTime"/>
                    </sequence>
                </extension>
            </complexContent>
        </complexType>
        <complexType name="TextSms">
            <complexContent>
                <extension base="tns:Sms">
                    <sequence>
                        <element name="intruder" type="boolean"/>
                        <element name="multiPart" type="boolean"/>
                        <element name="text" nillable="true" type="string"/>
                    </sequence>
                </extension>
            </complexContent>
        </complexType>
        <complexType name="BinarySms">
            <complexContent>
                <extension base="tns:Sms">
                    <sequence>
                        <element name="data" nillable="true" type="base64Binary"/>
                    </sequence>
                </extension>
            </complexContent>
        </complexType>
        <complexType name="TextSmsWithTime">
            <complexContent>
                <extension base="tns:SmsWithTime">
                    <sequence>
                        <element name="intruder" type="boolean"/>
```

```
                                        <element name="multiPart" type="boolean"/>
                                        <element name="text" nillable="true" type="string"/>
                                </sequence>
                        </extension>
                </complexContent>
        </complexType>
        <complexType name="BinarySmsWithTime">
                <complexContent>
                        <extension base="tns:SmsWithTime">
                                <sequence>
                                        <element name="data" nillable="true" type="base64Binary"/>
                                </sequence>
                        </extension>
                </complexContent>
        </complexType>
        <complexType name="ping">
                <sequence>
                        <element name="baID" nillable="true" type="string"/>
                </sequence>
        </complexType>
        <complexType name="pingResponse">
                <sequence>
                        <element name="result" nillable="true" type="tns:Response"/>
                </sequence>
        </complexType>
        <complexType name="receive">
                <sequence>
                        <element name="baID" nillable="true" type="string"/>
                </sequence>
        </complexType>
        <complexType name="receiveWithTime">
                <sequence>
                        <element name="baID" nillable="true" type="string"/>
                </sequence>
        </complexType>
        <complexType name="receiveResponse">
                <sequence>
                        <element name="result" nillable="true" type="tns:MessageContainer"/>
                </sequence>
        </complexType>
        <complexType name="MessageContainer">
                <sequence>
                        <element name="binarySms" nillable="true" type="tns:BinarySms"/>
                        <element name="mms" nillable="true" type="tns:Mms"/>
                        <element name="response" nillable="true" type="tns:Response"/>
                        <element name="selector" nillable="true" type="string"/>
                        <element name="textSms" nillable="true" type="tns:TextSms"/>
                </sequence>
        </complexType>
        <complexType name="receiveResponseWithTime">
                <sequence>
                        <element name="result" nillable="true" type="tns:MessageContainerWithTime"/>
                </sequence>
        </complexType>
```

```
<complexType name="MessageContainerWithTime">
	<sequence>
		<element name="binarySms" nillable="true" type="tns:BinarySmsWithTime"/>
		<element name="mms" nillable="true" type="tns:Mms"/>
		<element name="response" nillable="true" type="tns:Response"/>
		<element name="selector" nillable="true" type="string"/>
		<element name="textSms" nillable="true" type="tns:TextSmsWithTime"/>
	</sequence>
</complexType>
<complexType name="send">
	<sequence>
		<element name="mc" nillable="true" type="tns:MessageContainer"/>
	</sequence>
</complexType>
<complexType name="sendResponse">
	<sequence>
		<element name="result" nillable="true" type="tns:Response"/>
	</sequence>
</complexType>
<complexType name="activateBA">
	<sequence>
		<element name="baID" nillable="true" type="string"/>
	</sequence>
</complexType>
<complexType name="activateBAResponse">
	<sequence>
		<element name="result" type="boolean"/>
	</sequence>
</complexType>
<complexType name="deactivateBA">
	<sequence>
		<element name="baID" nillable="true" type="string"/>
		<element name="deleteOutstandingMessages" type="boolean"/>
	</sequence>
</complexType>
<complexType name="deactivateBAResponse">
	<sequence>
		<element name="result" type="boolean"/>
	</sequence>
</complexType>
<complexType name="sendSms">
	<sequence>
		<element name="baID" nillable="true" type="string"/>
		<element name="text" nillable="true" type="string"/>
		<element name="toNumber" nillable="true" type="string"/>
		<element name="fromNumber" nillable="true" type="string"/>
		<element name="validityPeriod" type="long"/>
		<element name="priority" nillable="true" type="string"/>
		<element name="intruder" type="boolean"/>
	</sequence>
</complexType>
<complexType name="sendSmsResponse">
	<sequence>
		<element name="result" nillable="true" type="string"/>
```

```
                    </sequence>
                </complexType>
                <element name="confirm" type="tns:confirm"/>
                <element name="confirmResponse" type="tns:confirmResponse"/>
                <element name="ping" type="tns:ping"/>
                <element name="pingResponse" type="tns:pingResponse"/>
                <element name="receive" type="tns:receive"/>
                <element name="receiveWithTime" type="tns:receiveWithTime"/>
                <element name="receiveResponse" type="tns:receiveResponse"/>
                <element name="receiveResponseWithTime" type="tns:receiveResponseWithTime"/>
                <element name="send" type="tns:send"/>
                <element name="sendResponse" type="tns:sendResponse"/>
                <element name="activateBA" type="tns:activateBA"/>
                <element name="activateBAResponse" type="tns:activateBAResponse"/>
                <element name="deactivateBA" type="tns:deactivateBA"/>
                <element name="deactivateBAResponse" type="tns:deactivateBAResponse"/>
                <element name="sendSms" type="tns:sendSms"/>
                <element name="sendSmsResponse" type="tns:sendSmsResponse"/>
        </schema>
</types>
<message name="PPGw_receive">
        <part element="tns:receive" name="parameters"/>
</message>
<message name="PPGw_receiveWithTime">
        <part element="tns:receiveWithTime" name="parameters"/>
</message>
<message name="GatewayManager_activateBAResponse">
        <part element="tns:activateBAResponse" name="parameters"/>
</message>
<message name="PPGw_sendResponse">
        <part element="tns:sendResponse" name="parameters"/>
</message>
<message name="GatewayManager_deactivateBAResponse">
        <part element="tns:deactivateBAResponse" name="parameters"/>
</message>
<message name="SimpleSmsGw_sendSmsResponse">
        <part element="tns:sendSmsResponse" name="parameters"/>
</message>
<message name="SimpleSmsGw_sendSms">
        <part element="tns:sendSms" name="parameters"/>
</message>
<message name="PPGw_send">
        <part element="tns:send" name="parameters"/>
</message>
<message name="PPGw_confirmResponse">
        <part element="tns:confirmResponse" name="parameters"/>
</message>
<message name="GatewayManager_deactivateBA">
        <part element="tns:deactivateBA" name="parameters"/>
</message>
<message name="PPGw_ping">
        <part element="tns:ping" name="parameters"/>
</message>
<message name="PPGw_pingResponse">
```

```
        <part element="tns:pingResponse" name="parameters"/>
</message>
<message name="PPGw_receiveResponse">
        <part element="tns:receiveResponse" name="parameters"/>
</message>
<message name="PPGw_receiveResponseWithTime">
        <part element="tns:receiveResponseWithTime" name="parameters"/>
</message>
<message name="GatewayManager_activateBA">
        <part element="tns:activateBA" name="parameters"/>
</message>
<message name="PPGw_confirm">
        <part element="tns:confirm" name="parameters"/>
</message>
<portType name="SimpleSmsGw">
        <operation name="sendSms">
                <input message="tns:SimpleSmsGw_sendSms"/>
                <output message="tns:SimpleSmsGw_sendSmsResponse"/>
        </operation>
</portType>
<portType name="PPGw">
        <operation name="confirm">
                <input message="tns:PPGw_confirm"/>
                <output message="tns:PPGw_confirmResponse"/>
        </operation>
        <operation name="ping">
                <input message="tns:PPGw_ping"/>
                <output message="tns:PPGw_pingResponse"/>
        </operation>
        <operation name="receive">
                <input message="tns:PPGw_receive"/>
                <output message="tns:PPGw_receiveResponse"/>
        </operation>
        <operation name="receiveWithTime">
                <input message="tns:PPGw_receiveWithTime"/>
                <output message="tns:PPGw_receiveResponseWithTime"/>
        </operation>
        <operation name="send">
                <input message="tns:PPGw_send"/>
                <output message="tns:PPGw_sendResponse"/>
        </operation>
</portType>
<portType name="GatewayManager">
        <operation name="activateBA">
                <input message="tns:GatewayManager_activateBA"/>
                <output message="tns:GatewayManager_activateBAResponse"/>
        </operation>
        <operation name="deactivateBA">
                <input message="tns:GatewayManager_deactivateBA"/>
                <output message="tns:GatewayManager_deactivateBAResponse"/>
        </operation>
</portType>
<binding name="SimpleSmsGwBinding" type="tns:SimpleSmsGw">
        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
```

```
        <operation name="sendSms">
            <soap:operation soapAction="http://b2b.eurotel.cz/SimpleSmsGw/sendSms"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
        </operation>
</binding>
<binding name="GatewayManagerBinding" type="tns:GatewayManager">
        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="activateBA">
            <soap:operation soapAction="http://b2b.eurotel.cz/GatewayManager/activateBA"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
        </operation>
        <operation name="deactivateBA">
            <soap:operation soapAction="http://b2b.eurotel.cz/GatewayManager/deactivateBA"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
        </operation>
</binding>
<binding name="PPGwBinding" type="tns:PPGw">
        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="confirm">
            <soap:operation soapAction="http://b2b.eurotel.cz/PPGw/confirm"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
        </operation>
        <operation name="ping">
            <soap:operation soapAction="http://b2b.eurotel.cz/PPGw/ping"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
        </operation>
        <operation name="receive">
            <soap:operation soapAction="http://b2b.eurotel.cz/PPGw/receive"/>
            <input>
```

```
                        <soap:body use="literal"/>
                    </input>
                    <output>
                        <soap:body use="literal"/>
                    </output>
            </operation>
            <operation name="receiveWithTime">
                    <soap:operation soapAction="http://b2b.eurotel.cz/PPGw/receiveWithTime"/>
                    <input>
                        <soap:body use="literal"/>
                    </input>
                    <output>
                        <soap:body use="literal"/>
                    </output>
            </operation>
            <operation name="send">
                    <soap:operation soapAction="http://b2b.eurotel.cz/PPGw/send"/>
                    <input>
                        <soap:body use="literal"/>
                    </input>
                    <output>
                        <soap:body use="literal"/>
                    </output>
            </operation>
    </binding>
    <service name="MessagingGateway">
            <port binding="tns:PPGwBinding" name="PPGwPort">
                    <soap:address location="http://localhost/smsconnector/services/PPGwPort" />
            </port>
            <port binding="tns:SimpleSmsGwBinding" name="SimpleSmsGwPort">
                    <soap:address location="http://localhost/smsconnector/services/SimpleSmsGwPort" />
            </port>
    </service>
</definitions>
```

9 Appendix B: PushPPGwPort.wsdl (uncommented)

Remark: for the electronic version of the WSDL please check the WSDL published by the Web Service at the specified URL.

```
<?xml version="1.0" encoding="utf-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="http://b2b.eurotel.cz/PushPPGw"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
targetNamespace="http://b2b.eurotel.cz/PushPPGw" name="PushPPGw">
  <types>
            <schema xmlns="http://www.w3.org/2001/XMLSchema" xmlns:soap11-enc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://b2b.eurotel.cz/PushPPGw">
                    <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
                    <complexType name="send">
                        <sequence>
                            <element name="mc" type="tns:MessageContainer" nillable="true"/>
                        </sequence>
                    </complexType>
```

```
<complexType name="MessageContainer">
	<sequence>
		<element name="binarySms" type="tns:BinarySms" nillable="true"/>
		<element name="mms"  type="tns:Mms" nillable="true"/>
		<element name="response"  type="tns:Response" nillable="true"/>
		<element name="selector" type="string" nillable="true"/>
		<element name="textSms"  type="tns:TextSms" nillable="true"/>
	</sequence>
</complexType>
<complexType name="BinarySms">
	<complexContent>
		<extension base="tns:Sms">
			<sequence>
				<element name="data" type="base64Binary" nillable="true"/>
			</sequence>
		</extension>
	</complexContent>
</complexType>
<complexType name="PremiumInfo">
	<sequence>
		<element name="billingCode" type="string" nillable="true"/>
		<element name="orderID" type="string" nillable="true"/>
		<element name="orderingChannel" type="string" nillable="true"/>
	</sequence>
</complexType>
<complexType name="Sms">
	<complexContent>
		<extension base="tns:Request">
			<sequence>
				<element name="dataCodingScheme" type="int"/>
				<element name="fromNumber" type="string" nillable="true"/>
				<element name="header" type="base64Binary" nillable="true"/>
				<element name="premiumInfo" type="tns:PremiumInfo" nillable="true"/>
				<element name="protocolIdentifier" type="string" nillable="true"/>
				<element name="reportLevel" type="string" nillable="true"/>
				<element name="toNumber" type="string" nillable="true"/>
				<element name="validityPeriod" type="long"/>
			</sequence>
		</extension>
	</complexContent>
</complexType>
<complexType name="Request">
	<complexContent>
		<extension base="tns:Message">
			<sequence>
				<element name="priority" type="string" nillable="true"/>
			</sequence>
		</extension>
	</complexContent>
</complexType>
<complexType name="Message">
	<sequence>
		<element name="baID" type="string" nillable="true"/>
		<element name="msgID" type="string" nillable="true"/>
```

```
                </sequence>
            </complexType>
            <complexType name="Response">
                <complexContent>
                    <extension base="tns:Message">
                        <sequence>
                            <elementname="code" type="string" nillable="true"/>
                            <element name="description" type="string" nillable="true"/>
                            <element name="refBaID" type="string" nillable="true"/>
                            <element name="refMsgID" type="string" nillable="true"/>
                            <element name="timestamp" type="dateTime" nillable="true"/>
                            <element name="type" type="string" nillable="true"/>
                        </sequence>
                    </extension>
                </complexContent>
            </complexType>
            <complexType name="Mms">
                <complexContent>
                    <extension base="tns:Request">
                        <sequence>
                            <element name="fromNumber" type="string" nillable="true"/>
                            <element name="messageClass" type="string" nillable="true"/>
                            <element name="parts" type="tns:MmsPart" nillable="true" minOccurs="0"
maxOccurs="unbounded"/>
                            <element name="premiumInfo" type="tns:PremiumInfo" nillable="true"/>
                            <element name="presentation" type="string" nillable="true"/>
                            <element name="reportLevel" type="string" nillable="true"/>
                            <element name="subject" type="string" nillable="true"/>
                            <element name="toNumber" type="string" nillable="true"/>
                        </sequence>
                    </extension>
                </complexContent>
            </complexType>
            <complexType name="MmsPart">
                <sequence>
                    <element name="content" type="base64Binary" nillable="true"/>
                    <element name="contentEncoding" type="string" nillable="true"/>
                    <element name="contentType" type="string"nillable="true"/>
                    <element name="id" type="string" nillable="true"/>
                </sequence>
            </complexType>
            <complexType name="TextSms">
                <complexContent>
                    <extension base="tns:Sms">
                        <sequence>
                            <element name="intruder" type="boolean"/>
                            <element name="multiPart" type="boolean"/>
                            <element name="text" type="string" nillable="true"/>
                        </sequence>
                    </extension>
                </complexContent>
            </complexType>
            <complexType name="sendResponse">
                <sequence>
```

```
                                        <element name="result" type="tns:Response" nillable="true"/>
                                </sequence>
                        </complexType>
                        <element name="send" type="tns:send"/>
                        <element name="sendResponse" type="tns:sendResponse"/>
                </schema>
        </types>
        <message name="PushPPGw_send">
                <part name="parameters" element="tns:send"/>
        </message>
        <message name="PushPPGw_sendResponse">
                <part name="parameters" element="tns:sendResponse"/>
        </message>
        <portType name="PushPPGw">
                <operation name="send">
                        <input message="tns:PushPPGw_send"/>
                        <output message="tns:PushPPGw_sendResponse"/>
                </operation>
        </portType>
        <binding name="PushPPGwBinding" type="tns:PushPPGw">
                <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
                <operation name="send">
                        <soap:operation soapAction="http://b2b.eurotel.cz/PushPPGw/send"/>
                        <input>
                                <soap:body use="literal"/>
                        </input>
                        <output>
                                <soap:body use="literal"/>
                        </output>
                </operation>
        </binding>
        <service name="PushPPGw">
                <port name="PushPPGwPort" binding="tns:PushPPGwBinding">
                        <soap:address location="http://localhost/B2BGTWFE/services/PushPPGwPort"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
                </port>
        </service>
</definitions>
```

10 Appendix C: Encoding of CZ characters

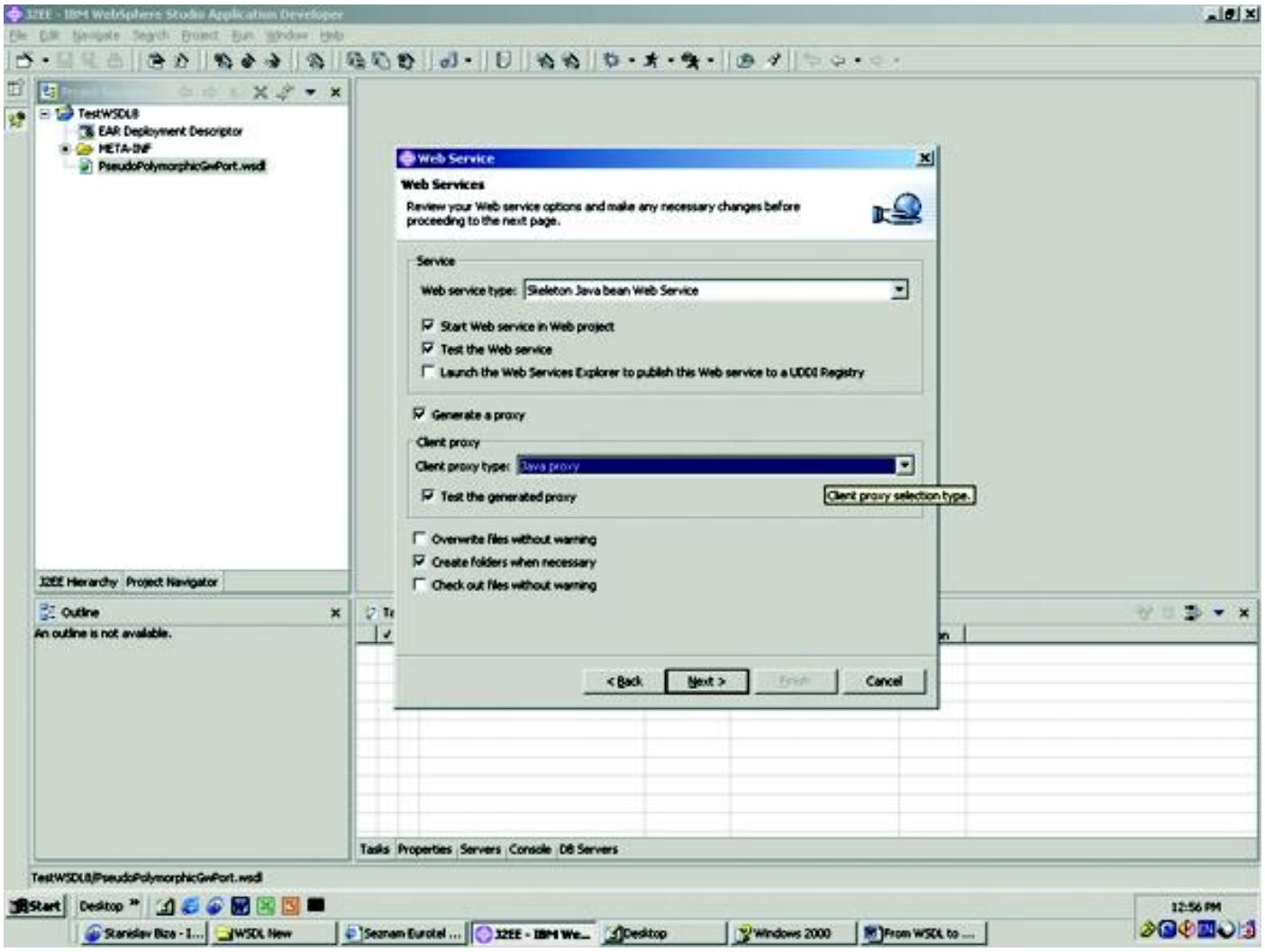
In general, O2 does not recommend to use Czech characters in AO or MO messages, as the support for Czech characters is dependant on the particular handset model. If the BA wants to send Czech characters in AO messages, they must conform to the standards GSM 03.38 and GSM 03.40. The BA is then responsible for properly encoding the message (in Unicode UCS- 2 encoding) and setting correct values for parameters header, dataCodingScheme and data or text. Knowledge of the capabilities of the receiving handset model is recommended in this case. For mobile subscribers, it is possible to send SMS messages containing Czech characters depending on their handset model and its settings. BAs must be aware that not all Czech characters might be represented in the character sets of mobile subscribers' handsets and O2's SMSCs. This could lead to changes in the message where special characters are replaced by similar characters contained in the character set or by a question mark if no similar character can be found. Moreover, depending on the handset settings, some handset models might be sending messages with the binary content even though message does not contain any diacritics.

11 Appendix D: Messages with binary content

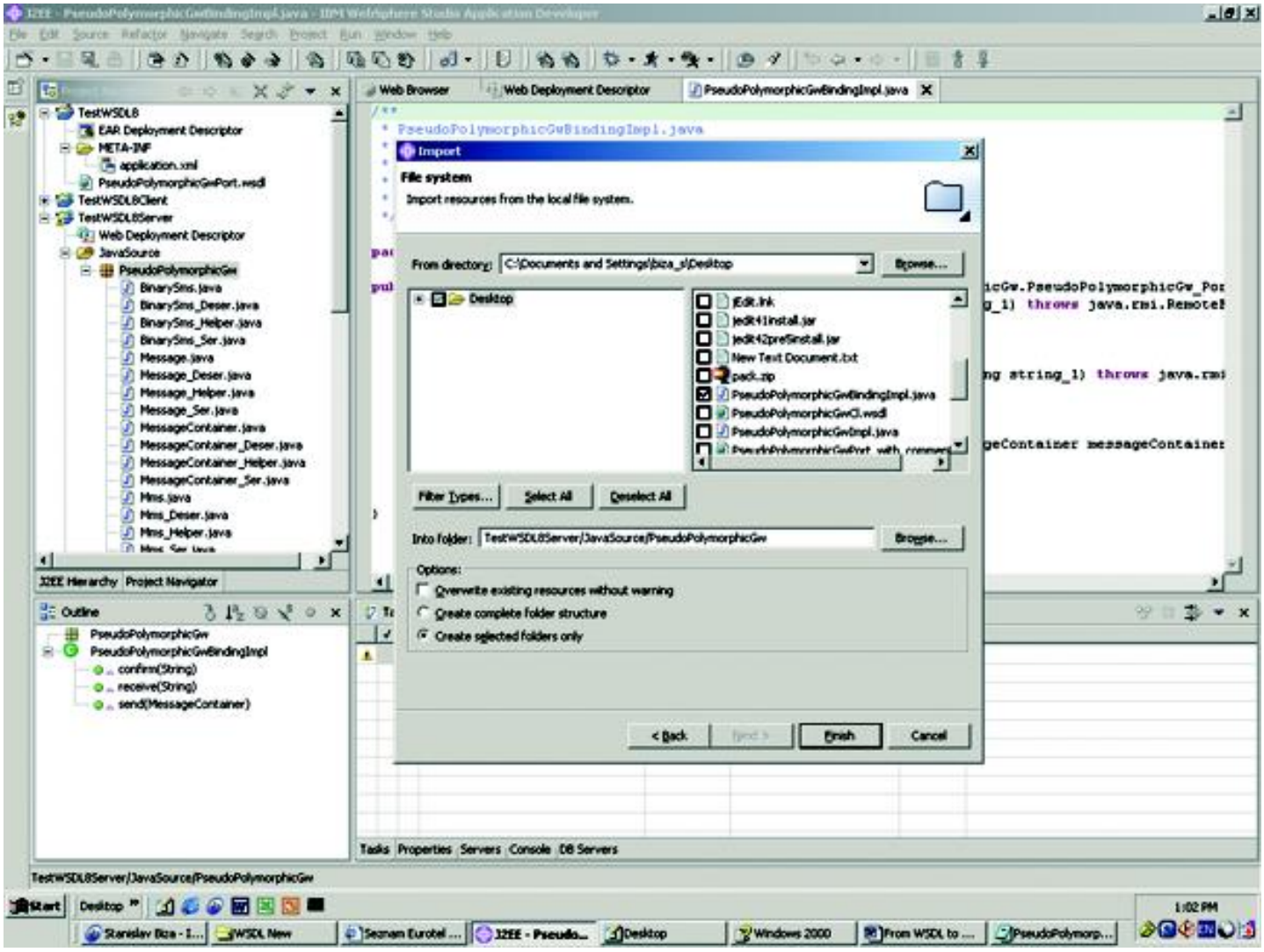
Various handset models are sending and receiving messages with binary content differently. Currently, there are two main standards supported:
Smart Messaging - supported by Nokia
Enhanced Messaging Service (EMS) – supported by Alcatel, Motorola, Siemens AG, Sony Ericsson and others
Whereas messages with binary content using Smart Messaging standard are sent with selector having value „BinarySms“, EMS phones sent such messages with selector having value „TextSms“.

12 Appendix E: Web Services Development Example Eclipse / IBM WSAD 5.1

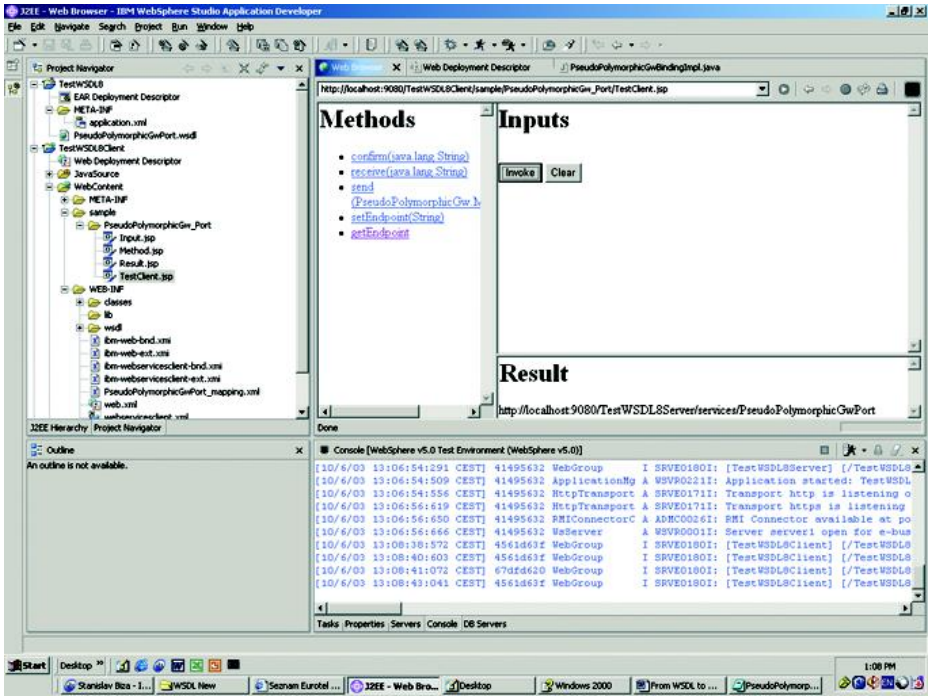
The following shows the process of generation of the Web Services stub classes. For an explanation of the general process see chapter 3.4. WebSphere Studio Application Developer 5.1 has been used as an example application; the underlying programming language is Java. For more detailed instructions see tutorials and/or online help in the WebSphere Studio Application Developer or the IBM developer webpage http://www.developer.ibm.com.
It is assumed, that the WSDL description is available either as a file or on a web address. The Web Service wizard is started via File - New - Other - Web Services - Web Service, see diagram below. It is chosen to generate a client proxy (may be also a test client)



WebSphere Studio Application Developer will generate all the required classes. As you see in the picture below, the data classes (from the WSDL types, such as Messages), the Client Stub classes(Web Service methods) and additional classes have been generated.



In addition a simple test client may be generated to test the responses of the available methods, see picture below.



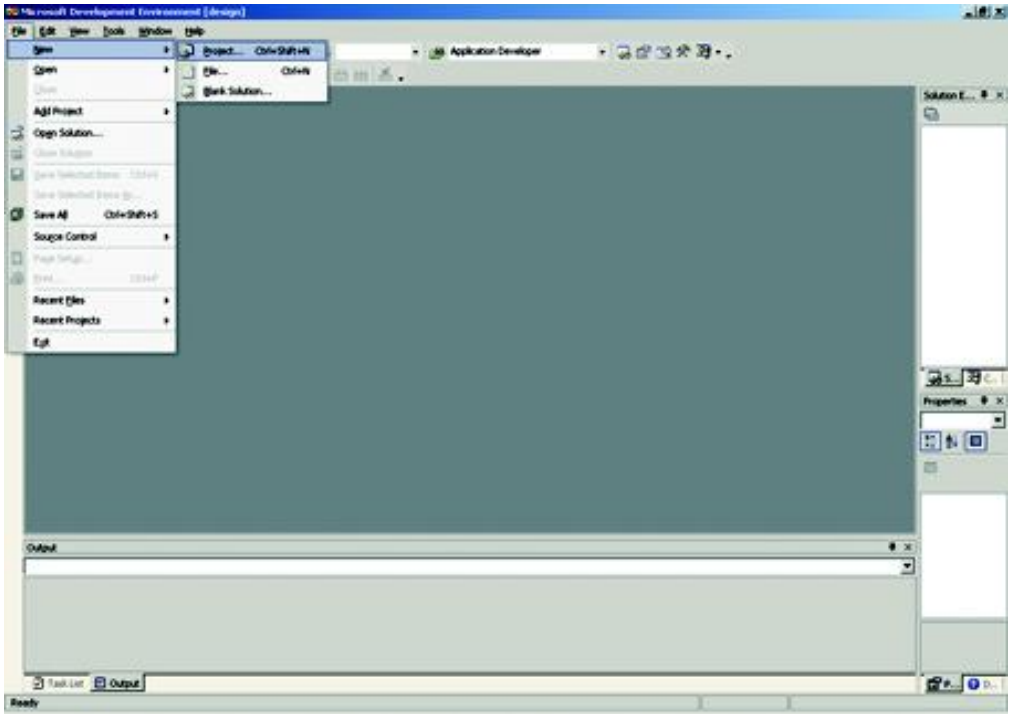
The following is a sample operation which calls the ping method. It uses classes automatically generated from the WSDL. The logmanager can be whatever the BA uses for logging, such as Log4J.

```
public OperationResult ping(String baId) throws MalformedURLException {

		logManager.logEntry(this, "ping", logManager.info, "SoapGw: ping: ---------- Pinging. BaId: " + baId);

		OperationResult or;

		try {
			PPGwBindingStub pPGw = new PPGwBindingStub(url, null);
			try {
				Response response = pPGw.ping(baId);
				if (response.getCode().equalsIgnoreCase("ISUC_000")) {
					or = OperationResult.setSuccess("Ping processed OK. StatusCode: " + response.getCode() + "; 
description: " + response.getDescription());
				}
				else {
					or = OperationResult.setError("Ping processed with error. Unexpected statusCode: " + 
response.getCode() + "; description: " + response.getDescription());
					return or;
				}
			}
			catch (RemoteException e) {
				e.printStackTrace();
				or = OperationResult.setError("Exception by ping occured." + e.toString());
				return or;
			}
```

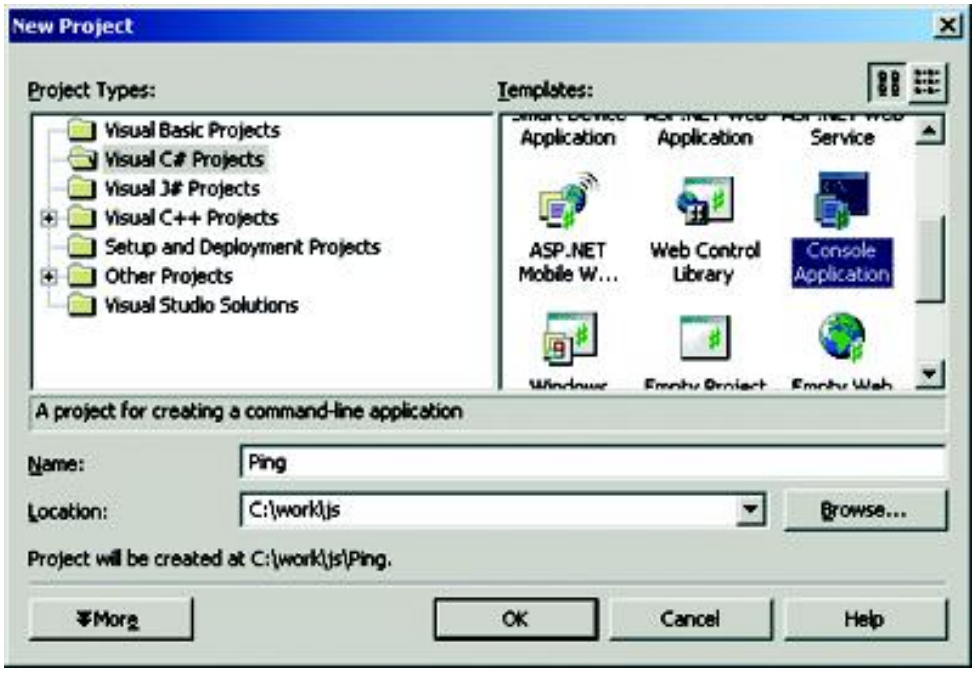
```
        }
        catch (WebServicesFault e) {
                e.printStackTrace();
                or = OperationResult.setError("Exception by ping occured." + e.toString());
                return or;
        }

        logManager.logEntry(this, "ping", logManager.debug, "Result Success=" + or.getSuccess());
        logManager.logEntry(this, "ping", logManager.debug, "Result Text=" + or.getText());

        if (or.getSuccess()) {
                logManager.logEntry(this, "ping", logManager.info, "Ping processed OK.");
        }

        return or;
}
```

13 Appendix F: Web Services Development Example .Net

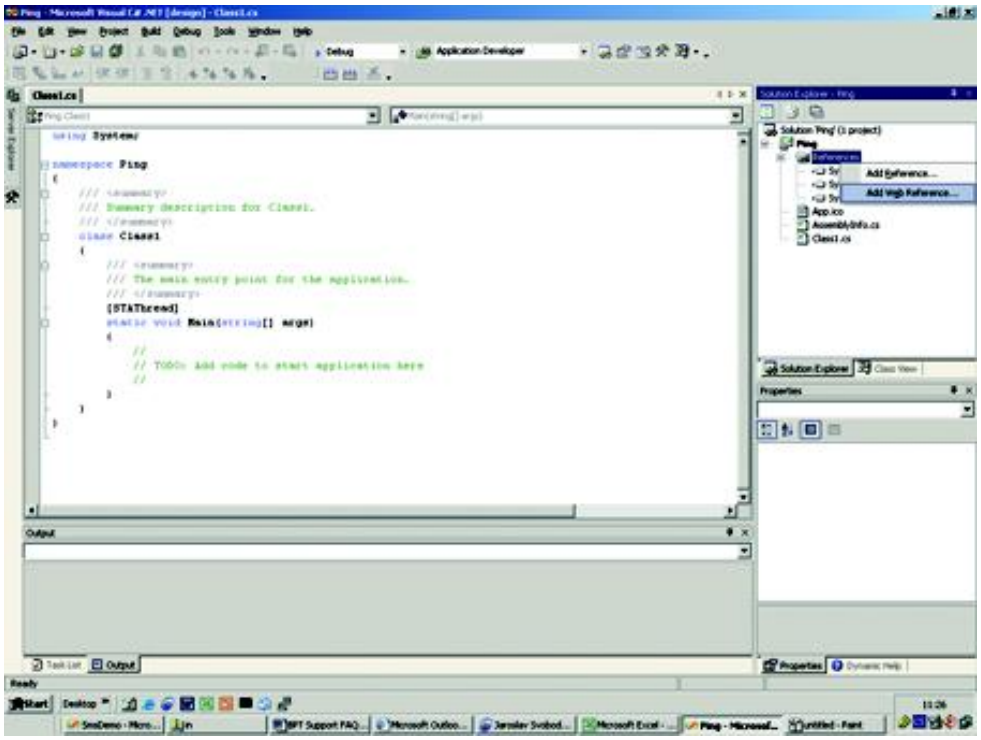
The following example describes how to create a simple C# console application in Microsoft Visual Studio .NET 2003 (VS) that calls the Web Services ping method. Before starting, the BA must obtain a client certificate and Business Application ID (baID) from O2. The certificate must be imported into Windows Certificate store and exported into the cer file (certFilePath) in der format.
From the VS menu select File -> New -> Project



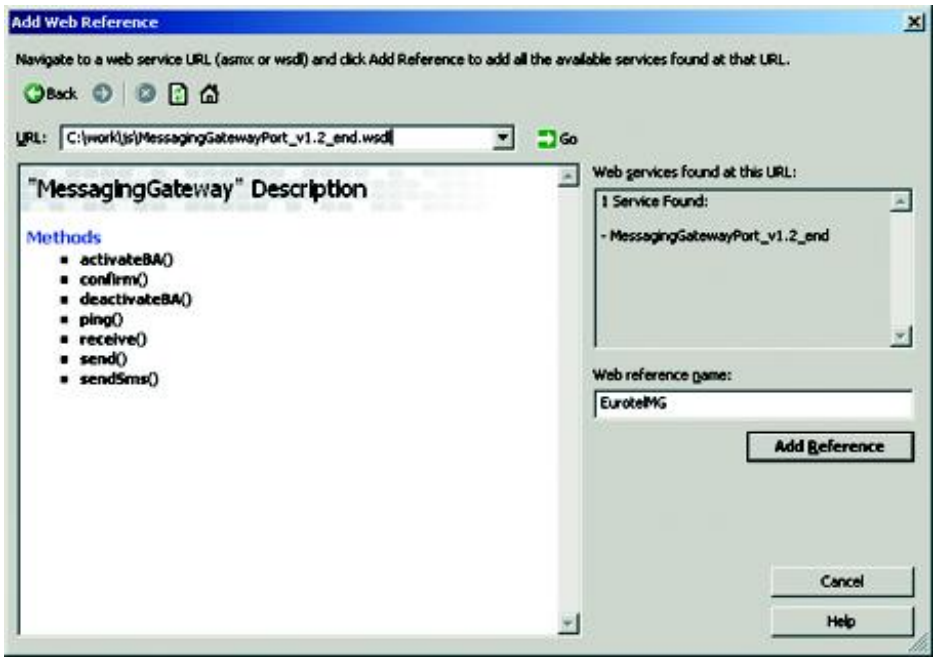
In Project Types select “Visual C# Project” and in Templates “Console Application”. In Name field type Ping and press OK.



In Solution Explorer press right mouse button on References, then select Add Web Reference …



In URL field type path to saved WSDL file (because there is problem with opening wsdl via HTTPS) and click Go. Into Web reference name field type EurotelMG, then click Add Reference.

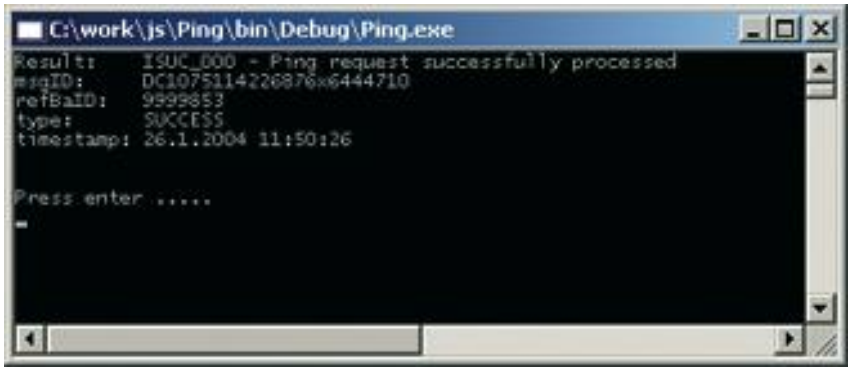


Place this code into Class1.cs file and retype real values for baID and certFilePath:

```
using System;
using System.Net;
using System.Security;
using System.Security.Cryptography.X509Certificates;
namespace Ping
{
/// <summary>
/// Summary description for Class1.
/// </summary>
class Class1
{
/// <summary>
/// The main entry point for the application.
/// </summary>
[STAThread]
static void Main(string[] args)
{
string baID = "9999853";
string certFilePath = @"C:\work\js\SmsDemo\MISIBM9999853.cer";
EurotelMG.PPGwBinding ws = new EurotelMG.PPGwBinding();
EurotelMG.Response response = new EurotelMG.Response();
System.Net.ServicePointManager.CertificatePolicy = new trustedCertificatePolicy();
X509Certificate ClientCert = X509Certificate.CreateFromCertFile(certFilePath);
ws.ClientCertificates.Add(ClientCert);
response = ws.ping(baID);
System.Console.WriteLine("Result: " + response.code + " - " + response.description);
```

```
System.Console.WriteLine("msgID: " + response.msgID);
System.Console.WriteLine("refBaID: " + response.refBaID);
System.Console.WriteLine("type: " + response.type);
System.Console.WriteLine("timestamp: " + response.timestamp);
System.Console.WriteLine("\n\nPress enter .....");
System.Console.ReadLine();
}
}
public class trustedCertificatePolicy : System.Net.ICertificatePolicy
{
public trustedCertificatePolicy() {}
public bool CheckValidationResult
(
System.Net.ServicePoint sp,
X509Certificate certificate,
System.Net.WebRequest request,
int problem)
{
return true;
}
}
}
```

Now to test this very simple application:



14 Appendix G: Request and Response Examples

This section provides reference examples of what the Web Services requests and responses should look like for different function calls. These are provided as a reference and there could be slight variations based upon each BA's requirements. These examples are platform independent, so they should look the same in .NET, Java, etc.

14.1 Ping

14.1.1 Request

```
POST /smsconnector/services/PPGwPort HTTP/1.1
```

```
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client Protocol 1.1.4322.573)
Content-Type: text/xml; charset=utf-8
SOAPAction: "http://b2b.eurotel.cz/PPGw/ping"
Content-Length: 836
Expect: 100-continue
Connection: Keep-Alive
Host: 127.0.0.1

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Header>
        <wsrp:path soap:actor="http://schemas.xmlsoap.org/soap/actor/next" xmlns:wsrp="http://schemas.xmlsoap.org/rp">
              <wsrp:action>http://b2b.eurotel.cz/PPGw/ping</wsrp:action>
              <wsrp:to>https://smsconnector.cz.o2.com/smsconnector/services/PPGwPort</wsrp:to>
              <wsrp:id>uuid:b3c8da5f-63e0-4756-baac-721014d6019a</wsrp:id>
        </wsrp:path>
        <wsu:Timestamp xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/07/utility">
              <wsu:Created>2004-03-24T12:27:22Z</wsu:Created>
              <wsu:Expires>2004-03-24T12:32:22Z</wsu:Expires>
        </wsu:Timestamp>
  </soap:Header>
  <soap:Body>
        <ping xmlns="http://b2b.eurotel.cz">
              <baID xmlns="">1991001</baID>
        </ping>
  </soap:Body>
</soap:Envelope>
```

14.1.2 Response

```
HTTP/1.1 100 Continue

HTTP/1.1 200 OK
Server: WebSphere Application Server/5.0
Content-Type: text/xml; charset=utf-8
Content-Language: en-US
Transfer-Encoding: chunked

29f
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
        <pingResponse xmlns="http://b2b.eurotel.cz">
              <result xmlns="">
                    <baID>1991001</baID>
                    <msgID>phasdi15z-20040324111732-69</msgID>
                    <code>ISUC_000</code>
                    <description>Ping request performed successfully</description>
                    <refBaID xsi:nil="true"/>
```

```
                        <refMsgID xsi:nil="true"/>
                        <timestamp>2004-03-24T12:27:15.197Z</timestamp>
                        <type>SUCCESS_INFO</type>
                    </result>
            </pingResponse>
  </soapenv:Body>
</soapenv:Envelope>
0
```

14.2 Receive

14.2.1 Request

```
POST /smsconnector/services/PPGwPort HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client Protocol 1.1.4322.573)
Content-Type: text/xml; charset=utf-8
SOAPAction: "http://b2b.eurotel.cz/PPGw/receive"
Content-Length: 845
Expect: 100-continue
Connection: Keep-Alive
Host: 127.0.0.1

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Header>
        <wsrp:path soap:actor="http://schemas.xmlsoap.org/soap/actor/next" xmlns:wsrp="http://schemas.xmlsoap.org/rp">
            <wsrp:action>http://b2b.eurotel.cz/PPGw/receive</wsrp:action>
            <wsrp:to>https://smsconnector.cz.o2.com/smsconnector/services/PPGwPort</wsrp:to>
            <wsrp:id>uuid:299574e8-1522-4866-8526-fad427378bbe</wsrp:id>
        </wsrp:path>
        <wsu:Timestamp xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/07/utility">
            <wsu:Created>2004-03-24T12:25:35Z</wsu:Created>
            <wsu:Expires>2004-03-24T12:30:35Z</wsu:Expires>
        </wsu:Timestamp>
  </soap:Header>
  <soap:Body>
        <receive xmlns="http://b2b.eurotel.cz">
            <baID xmlns="">1991001</baID>
        </receive>
  </soap:Body>
</soap:Envelope>
```

14.2.2 Response

```
HTTP/1.1 100 Continue
HTTP/1.1 200 OK
Server: WebSphere Application Server/5.0
Content-Type: text/xml; charset=utf-8
```

```
Content-Language: en-US
Transfer-Encoding: chunked
41c
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
        <receiveResponse xmlns="http://b2b.eurotel.cz">
              <result xmlns="">
                    <binarySms xsi:nil="true"/>
                    <mms xsi:nil="true"/>
                    <response xsi:nil="true"/>
                    <selector>TextSms</selector>
                    <textSms>
                          <baID>1991001</baID>
                          <msgID>ID:10801311420004206025456875401400000000000000000000</msgID>
                          <priority xsi:nil="true"/>
                          <dataCodingScheme>0</dataCodingScheme>
                          <fromNumber>+420602545687</fromNumber>
                          <header xsi:nil="true"/>
                          <premiumInfo>
                                <billingCode xsi:nil="true"/>
                                <orderID xsi:nil="true"/>
                                <orderingChannel xsi:nil="true"/>
                          </premiumInfo>
                          <protocolIdentifier>0</protocolIdentifier>
                          <reportLevel xsi:nil="true"/>
                          <toNumber>1991001</toNumber>
                          <validityPeriod>0</validityPeriod>
                          <intruder>0</intruder>
                          <multiPart>0</multiPart>
                          <text>Test</text>
                    </textSms>
              </result>
        </receiveResponse>
  </soapenv:Body>
</soapenv:Envelope>
0
```

14.3 Confirm

14.3.1 Request

```
POST /smsconnector/services/PPGwPort HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client Protocol 1.1.4322.573)
Content-Type: text/xml; charset=utf-8
SOAPAction: "http://b2b.eurotel.cz/PPGw/confirm"
Content-Length: 959
Expect: 100-continue
Host: 10.32.121.87:9999
```

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Header>
        <wsrp:path soap:actor="http://schemas.xmlsoap.org/soap/actor/next" xmlns:wsrp="http://schemas.xmlsoap.org/rp">
              <wsrp:action>http://b2b.eurotel.cz/PPGw/confirm</wsrp:action>
              <wsrp:to>https://smsconnector.cz.o2.com/smsconnector/services/PPGwPort</wsrp:to>
              <wsrp:id>uuid:aa861e55-3d39-426c-82a0-9fd1d9234ee0</wsrp:id>
        </wsrp:path>
        <wsu:Timestamp xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/07/utility">
              <wsu:Created>2004-03-24T12:25:47Z</wsu:Created>
              <wsu:Expires>2004-03-24T12:30:47Z</wsu:Expires>
        </wsu:Timestamp>
  </soap:Header>
  <soap:Body>
        <confirm xmlns="http://b2b.eurotel.cz">
              <baID xmlns="">1991001</baID>
              <refBaID xmlns="">1991001</refBaID>
              <refMsgID xmlns="">ID:108013114200042060254568754014000000000000000000</refMsgID>
        </confirm>
  </soap:Body>
</soap:Envelope>
```

14.3.2 Response

```
HTTP/1.1 100 Continue

HTTP/1.1 200 OK
Server: WebSphere Application Server/5.0
Content-Type: text/xml; charset=utf-8
Content-Language: en-US
Transfer-Encoding: chunked

2d5
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
        <confirmResponse xmlns="http://b2b.eurotel.cz">
              <result xmlns="">
                    <baID>1991001</baID>
                    <msgID>phasdi15z-20040324111732-66</msgID>
                    <code>ISUC_002</code>
                    <description>Confirm request successfully processed</description>
                    <refBaID>1991001</refBaID>
                    <refMsgID>ID:108013114200042060254568754014000000000000000000</refMsgID>
                    <timestamp>2004-03-24T12:25:40.259Z</timestamp>
                    <type>SUCCESS_INFO</type>
              </result>
        </confirmResponse>
  </soapenv:Body>
</soapenv:Envelope>
0
```

14.4 Send

14.4.1 Request

```
POST /smsconnector/services/PPGwPort HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client Protocol 1.1.4322.573)
Content-Type: text/xml; charset=utf-8
SOAPAction: "http://b2b.eurotel.cz/PPGw/send"
Content-Length: 1259
Expect: 100-continue
Host: 10.32.121.87:9999

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Header>
        <wsrp:path soap:actor="http://schemas.xmlsoap.org/soap/actor/next" xmlns:wsrp="http://schemas.xmlsoap.org/rp">
              <wsrp:action>http://b2b.eurotel.cz/PPGw/send</wsrp:action>
              <wsrp:to>https://smsconnector.cz.o2.com/smsconnector/services/PPGwPort</wsrp:to>
              <wsrp:id>uuid:118ef735-5314-41f2-9f32-eba2dcfb4fae</wsrp:id>
        </wsrp:path>
        <wsu:Timestamp xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/07/utility">
              <wsu:Created>2004-03-24T12:22:34Z</wsu:Created>
              <wsu:Expires>2004-03-24T12:27:34Z</wsu:Expires>
        </wsu:Timestamp>
  </soap:Header>
  <soap:Body>
        <send xmlns="http://b2b.eurotel.cz">
              <mc xmlns="">
                    <binarySms xsi:nil="true" />
                    <mms xsi:nil="true" />
                    <response xsi:nil="true" />
                    <selector>TextSms</selector>
                    <textSms>
                          <baID>1991001</baID>
                          <msgID>c0465d63-38f0-4f31-93b3-0dc80a063baf</msgID>
                          <priority>1</priority>
                          <dataCodingScheme>0</dataCodingScheme>
                          <fromNumber>1991001</fromNumber>
                          <header xsi:nil="true" />
                          <premiumInfo xsi:nil="true" />
                          <protocolIdentifier>64</protocolIdentifier>
                          <reportLevel>terminal</reportLevel>
                          <toNumber>+420602123456</toNumber>
                          <validityPeriod>10000</validityPeriod>
                          <intruder>false</intruder>
                          <multiPart>false</multiPart>
                          <text>Test zprava :-)</text>
                    </textSms>
              </mc>
        </send>
  </soap:Body>
</soap:Envelope>
```

14.4.2 Response

```
HTTP/1.1 100 Continue

HTTP/1.1 200 OK
Server: WebSphere Application Server/5.0
Content-Type: text/xml; charset=utf-8
Content-Language: en-US
Transfer-Encoding: chunked

2bf
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
        <sendResponse xmlns="http://b2b.eurotel.cz">
              <result xmlns="">
                    <baID>1991001</baID>
                    <msgID>cspfe113-20040517125212-233908</msgID>
                    <code>ISUC_001</code>
                    <description>Pozadavek Send uspesne zpracovan - ISUC_001 - Send request successfully
processed</description>
                    <refBaID>1991001</refBaID>
                    <refMsgID>dcdd7fba-0529-47af-b3c4-31e041d99239</refMsgID>
                    <timestamp>2004-05-26T15:26:41.040Z</timestamp>
                    <type>SUCCESS</type>
              </result>
        </sendResponse>
  </soapenv:Body>
</soapenv:Envelope>
0
```

14.5 SendBinary

NOTE: Binary SMS must have header and data elements Base64 encoded.

14.5.1 Request

```
POST /smsconnector/services/PPGwPort HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client Protocol 1.1.4322.573)
Content-Type: text/xml; charset=utf-8
SOAPAction: "http://b2b.eurotel.cz/PPGw/send"
Content-Length: 1259
Expect: 100-continue
Host: 10.32.121.87:9999

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Header>
        <wsrp:path soap:actor="http://schemas.xmlsoap.org/soap/actor/next" xmlns:wsrp="http://schemas.xmlsoap.org/rp">
              <wsrp:action>http://b2b.eurotel.cz/PPGw/send</wsrp:action>
              <wsrp:to>https://smsconnector.cz.o2.com/smsconnector/services/PPGwPort</wsrp:to>
```

```
                <wsrp:id>uuid:118ef735-5314-41f2-9f32-eba2dcfb4fae</wsrp:id>
        </wsrp:path>
        <wsu:Timestamp xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/07/utility">
                <wsu:Created>2004-03-24T12:22:34Z</wsu:Created>
                <wsu:Expires>2004-03-24T12:27:34Z</wsu:Expires>
        </wsu:Timestamp>
  </soap:Header>
  <soap:Body>
        <send xmlns="http://b2b.eurotel.cz">
                <mc xmlns="">
                        <binarySms>
                                <baID>1991001</baID>
                                <msgID>1</msgID>
                                <priority>1</priority>
                                <fromNumber>1991001</fromNumber>
                                <toNumber>+420602545687</toNumber>
                                <intruder>false</intruder>
                                <validityPeriod>10000</validityPeriod>
                                <data>VGVzdCB6cHJhdmEgOik=</data>
                        </binarySms>
                        <mms xsi:nil="true" />
                        <response xsi:nil="true" />
                        <selector>BinarySms</selector>
                        <textSms xsi:nil="true" />
                </mc>
        </send>
  </soap:Body>
</soap:Envelope>
```

14.5.2 Response

```
HTTP/1.1 200 OK
Server: WebSphere Application Server/5.0
Content-Type: text/xml; charset=utf-8
Content-Language: en-US
Transfer-Encoding: chunked

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
        <ns2:sendResponse xmlns:ns2="http://b2b.eurotel.cz">
                <result>
                        <baID>29913471</baID>
                        <msgID>ID-42740-1300711928869-1-10</msgID>
                        <code>ISUC_001</code>
                        <description>Pozadavek Send uspesne zpracovan - ISUC_001 - Send request successfully
processed</description>
                        <refBaID>29913471</refBaID>
                        <refMsgID>1</refMsgID>
                        <timestamp>2011-03-21T13:48:12.636Z</timestamp>
                        <type>SUCCESS</type>
                </result>
        </ns2:sendResponse>
  </soap:Body>
```

```
</soap:Envelope>
```

14.6 SimpleSend

14.6.1 Request

```
POST /smsconnector/services/SimpleSmsGwPort HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client Protocol 1.1.4322.573)
Content-Type: text/xml; charset=utf-8
SOAPAction: "http://b2b.eurotel.cz/SimpleSmsGw/sendSms"
Content-Length: 1093
Expect: 100-continue
Connection: Keep-Alive
Host: 127.0.0.1

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Header>
        <wsrp:path soap:actor="http://schemas.xmlsoap.org/soap/actor/next" xmlns:wsrp="http://schemas.xmlsoap.org/rp">
              <wsrp:action>http://b2b.eurotel.cz/SimpleSmsGw/sendSms</wsrp:action>
              <wsrp:to>https://smsconnector.cz.o2.com/smsconnector/services/SimpleSmsGwPort</wsrp:to>
              <wsrp:id>uuid:f576931b-ef71-4867-ae07-5bd04d3faa97</wsrp:id>
        </wsrp:path>
        <wsu:Timestamp xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/07/utility">
              <wsu:Created>2004-03-24T12:28:46Z</wsu:Created>
              <wsu:Expires>2004-03-24T12:33:46Z</wsu:Expires>
        </wsu:Timestamp>
  </soap:Header>
  <soap:Body>
        <sendSms xmlns="http://b2b.eurotel.cz">
              <baID xmlns="">1991001</baID>
              <text xmlns="">Test zprava :-)</text>
              <toNumber xmlns="">+420602545687</toNumber>
              <fromNumber xmlns="">1991001</fromNumber>
              <validityPeriod xmlns="">10000</validityPeriod>
              <priority xmlns="">1</priority>
              <intruder xmlns="">false</intruder>
        </sendSms>
  </soap:Body>
</soap:Envelope>
```

14.6.2 Response

```
HTTP/1.1 100 Continue

HTTP/1.1 200 OK
Server: WebSphere Application Server/5.0
Content-Type: text/xml; charset=utf-8
Content-Language: en-US
Transfer-Encoding: chunked
```

```
1a3
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
        <sendSmsResponse xmlns="http://b2b.eurotel.cz">
              <result xmlns="">ISUC_001</result>
        </sendSmsResponse>
  </soapenv:Body>
</soapenv:Envelope>

0
```

15 Appendix H: Response Codes

The list of possible error codes with code, description and its mapping to method calls is provided in the table below. O2 reserves the right to change response message texts without notification.

Type	Code	Response message text	Response Send timing	Receive	Confirm	Ping	simpleSMS	Send (PUSH)
APPL_ERROR	EAPP_003	BaID neni aktivni - EAPP_003 - BaID is not active	Synchronous	X	X	X	X	X
APPL_ERROR	EAPP_019	Zprava zaslana po terminu platnosti zkusebniho rezimu - EAPP_019 - Trial period has expired	Asynchronous		X			
APPL_ERROR	EAPP_020	Prekrocen maximalni pocet zprav, ktere je mozne zaslat v jednom mesici - EAPP_023 - Maximum quota of messages per month exceeded	Asynchronous		X			
APPL_ERROR	EAPP_024	Prekrocen maximaln pocet zprav, ktere je mozne zaslat behem jednoho dne - EAPP_024 - Maximum quota of messages per day exceeded	Asynchronous		X			

Type	Code	Response message text	Response timing	Send	Receive	Confirm	Ping	simpleSMS	Send (PUSH)
APPL_ERROR	EAPP_025	Zprava potvrzovana pres pozadavek Confirm neni k dispozici - EAPP_025 - No message found for Confirm request	Synchronous			X			
APPL_ERROR	EEAPP_037	Limit zprav cekajicich na potvrzeni prijeti prekrocen - EAPP_037 - Too many messages waiting for confirmation	Synchronous		X				
APPL_ERROR	EEAPP_050	Maximalni limit soucasnych pozadavku prekrocen - EAPP_050 - Maximum number of concurrent requests exceeded	Synchronous	X	X	X	X	X	
APPL_ERROR	EEAPP_051	Nadlimitni pocet soucasnych pozadavku na potvrzeni prijeti zpravy - EAPP_051 - Too many concurrent confirm requests	Synchronous			X			

Type	Code	Response message text	Response timing	Send	Receive	Confirm	Ping	simpleSMS	Send (PUSH)
APPL_ERROR	EEAPP_052	Nadlimitnich pocet soucasnych pozadavku na prijeti zpravy - EAPP_052 - Too many concurrent receive requests	Synchronous		X				
APPL_ERROR	EEAPP_053	Nadlimitni pocet soucasnych pozadavku na zaslani zpravy - EAPP_053 - Too many concurrent send requests	Synchronous	X				X	
APPL_ERROR	EEAPP_060	Aplikace neni opravnena pouzivat cislo MSISDN v poli fromNumber - EAPP_060 - Application is not authorised to use MSISDN number in fromNumber field	Asynchronous		X				
APPL_ERROR	EEAPP_101	Zprava musi byt potvrzena pres stejne rozhrani, pres jake byla prijata - EAPP_101 - Message must be confirmed via same interface through which it was received	Synchronous			X			

Type	Code	Response message text	Response timing	Send	Receive	Confirm	Ping	simpleSMS	Send (PUSH)
APPL_ERROR	EAPP_107	Aplikace SMS Connector nemuze posilat zpravy MMS - EAPP_107 - SMS Connector application is not allowed to send MMSs	Synchronous	X					
APPL_ERROR	EAPP_109	Aplikace SMS Connector nemuze posilat premiove zpravy - EAPP_109 - SMS Connector application is not allowed to send premium messages	Synchronous	X					
AUTHORIZATION ERROR	EAUT_001	Neplatny certifikat - EAUT_001 - Invalid certificate2	Synchronous	X	X	X	X	X	
AUTHORIZATION ERROR	EAUT_002	BA ID neni platne nebo neni autorizovano pro certifikat - EAUT_002 - BA ID not valid or not authorised for certificate	Synchronous	X	X	X	X	X	
AUTHORIZATION ERROR	EAUT_003	Ref BA ID neni autorizovano pro certifikat - EAUT_003 - Ref BA ID not authorised for certificate	Synchronous		X				
FORMAT_ERROR	EFMT_004	Neplatny identifikator zpravy - EFMT_004 - Invalid message ID	Synchronous	X					

2 Business application attempts to establish the https connection with invalid certificate, i.e. with the certificate not provisioned or if the certificate has expired or is suspended.

Type	Code	Response message text	Response timing	Send	Receive	Confirm	Ping	simpleSMS	Send (PUSH)
FORMAT_ERROR	EFMT_005	Neplatna priorita zpravy - EFMT_005 - Invalid message priority	Synchronous	X				X	
FORMAT_ERROR	EFMT_006	Nesoulad cisla fromNumber s BA ID - EFMT_006 - Invalid fromNumber (mismatch with BA ID)	Synchronous	X				X	
FORMAT_ERROR	EFMT_007	Neplatny format MSISDN cisla prijemce - EFMT_007 - Invalid recipient MSISDN number format	Synchronous	X				X	
APPL_ERROR	EFMT_008	Neznama hodnota pole Selector - EFMT_008 - Unknown Selector value		X					
FORMAT_ERROR	EFMT_009	Prilis dlouhy sufix zpravy - EFMT_009 - Message suffix too long	Synchronous	X				X	
FORMAT_ERROR	EFMT_010	Neplatny format MSISDN cisla fromNumber - EFMT_010 - Invalid MSISDN number format of fromNumber	Synchronous	X				X	

Type	Code	Response message text	Response timing	Send	Receive	Confirm	Ping	simpleSMS	Send (PUSH)
FORMAT_ERROR	EFMT_011	Sufix zpravy musi byt ciselna hodnota - EFMT_011 - Message suffix must be numeric	Synchronous	X				X	
FORMAT_ERROR	EFMT_013	Neplatna hodnota pole Report Level / Delivery Report - EFMT_013 - Invalid Report Level / Delivery Report	Synchronous	X					
FORMAT_ERROR	EFMT_016	Neplatny format pole Validity period - EFMT-016 – Incorrect message validity format	Synchronous	X				X	
FORMAT_ERROR	EFMT_023	Binarni SMS neobsahuje zadna data - EFMT_023 - No data in binary SMS	Synchronous	X					
FORMAT_ERROR	EFMT_024	Binarni zprava je delsi nez 140 znaku - EFMT_024 - Binary message longer than 140 characters	Synchronous	X					
FORMAT_ERROR	EFMT_025	SMS nemuze byt soucasne poslana jako multipart a intruder - EFMT_025 - SMS cannot be sent as multipart and intruder at the same time.	Synchronous	X					

Type	Code	Response message text	Response timing	Send	Receive	Confirm	Ping	simpleSMS	Send (PUSH)
FORMAT_ERROR	EFMT_026	Nepripustna delka zpravy - EFMT_026 - Message too long	Synchronous	X				X	
FORMAT_ERROR	EFMT_032	Pole BA ID je prazdne - EFMT_032 - BA ID empty	Synchronous	X	X	X	X	X	
FORMAT_ERROR	EFMT_034	Zprava neobsahuje zadny text - EFMT_034 - Text message empty	Synchronous	X				X	
FORMAT_ERROR	EFMT_035	Neplatny pozadavek - EFMT_035 - Bad request	Synchronous	X	X	X			
FORMAT_ERROR	EFMT_102	Neplatna hodnota pole multipart - EFMT_102 - Invalid multipart format	Synchronous	X					
FORMAT_ERROR	EFMT_105	Text zpravy obsahuje nepodporovane znaky - EFMT_105 - SMS text contains unsupported characters	Synchronous	X				X	
FORMAT_ERROR	EFMT_106	Neplatna hodnota pole data coding scheme - EFMT_106 - Invalid data coding scheme	Synchronous	X					
FORMAT_ERROR	EFMT_121	Neplatna hodnota pole protocolIdentifier - EFMT_121 - Invalid protocolIdentifier value	Synchronous	X					

Type	Code	Response message text	Response timing	Send	Receive	Confirm	Ping	simpleSMS	Send (PUSH)
INTERNAL_ERROR	EINT_005	Zprava zamitnuta SMSC/MMSC - EINT_005 - Messagerejected by SMSC/MMSC3	Asynchronous		X				
INTERNAL_ERROR	EINT_012	O2: Chyba - Zprava nedorucena z technickych duvodu - EINT_012 - Message not delivered for technical reasons	Synchronous	X	X	X	X	X	
INTERNAL_ERROR	EINT_102	Zprava nedorucena z technickych duvodu - EINT_102 - Message not delivered for technical reasons	Asynchronous		X				
SUCCESS	ISUC_000	Pozadavek Ping byl uspesny - ISUC_000 - Ping request performed successfully	Synchronous				X		
SUCCESS	ISUC_001	Pozadavek Send uspesne zpracovan - ISUC_001 - Send request successfully processed	Synchronous	X				X	X
SUCCESS	ISUC_002	Pozadavek Confirm uspesne zpracovan - ISUC_002 - Confirm request successfully processed	Synchronous			X			

3 Most probable reason of EINT_005 is incorrect international MSISDN prefix or incorrect MSISDN length.

Type	Code	Response message text	Response timing	Send	Receive	Confirm	Ping	simpleSMS	Send (PUSH)
SUCCESS	ISUC_005	Zprava byla dorucena - ISUC_005 - Message delivered	Asynchronous		X				
SUCCESS	ISUC_006	Zprava nebyla dorucena - ISUC_006 - Message delivery failed	Asynchronous		X				
SUCCESS	ISUC_010	Zprava byla dorucena na SMSC/MMSC - ISUC_010 - Message forwarded to SMSC/MMSC	Asynchronous		X				

16 Appendix I: response codes returned to end-user's mobile phone

The following table describes the list of response codes, which are generated as a result of MO SMS sent from end-user's mobile phone, and which are returned by the MP in case of error in the form of informative SMS to end-user's mobile (please see the note below).

Type	Code	Response message text
APPL_ERROR	EAPP_033	O2: Chyba - Prilis dlouhe cislo sluzby - EAPP_033 - Service number too long
APPL_ERROR	EAPP_034	O2: Chyba - Cislo sluzby neexistuje – EAPP_034 - Service number does not exist
APPL_ERROR	EAPP 047	O2: Chyba - Zprava nedorucena - casovy limit prekrocen - EAPP_047 - Message not delivered - timeout expired
FORMAT_ERROR	EAPP_113	O2: Chyba - Zprava nedorucena z technickych duvodu - EAPP_113 - Message not delivered for technical reasons
INTERNAL_ERROR	EINT_012	O2: Chyba - Zprava nedorucena z technickych duvodu - EINT_012 - Message not delivered for technical reasons
INTERNAL_ERROR	EINT_107	O2: Chyba - Zprava nedorucena z technickych duvodu - EINT_107 - Message not delivered for technical reasons
INTERNAL_ERROR	EINT_108	O2: Chyba - Zprava nedorucena z technickych duvodu - EINT_108 - Message not delivered for technical reasons
INTERNAL_ERROR	EINT_109	O2: Chyba - Zprava nedorucena z technickych duvodu - EINT_109 - Message not delivered for technical reasons

Note: Although MP generates an error EAPP_047, this is not actually returned to end-user's mobile phone.

17 Appendix L: General Requirements for Servers

In the push mode, the BA servers must listen on TCP ports 443 or 8443.

18 Appendix M: Deployment Specific Parameters

Please see the document "SMS Connector Deployment Specific Parameters".

19 Appendix N: Solution Constraints

The following lists known MP constraints:
Maximum value of validityPeriod used for SMS AO message will be 388800 seconds (4,5 days), even if BA sets higher value of validityPeriod field. Maximum time, during which the platform sends Delivery Reports (positive or negative - ISUC_005 and ISUC_006 Response codes) as responses to AO SMS messages is 46800 seconds (13 hours). If BA asks for Delivery reports, sets validityPeriod SMS field value higher and mobile terminal is not reached within 13 hours, neither positive nor negative Delivery Report will be sent back to BA. In exceptional curcumstances the BA can receive an HTTP error 403 request, even if there is no problem with authorization. The BA should repeat the same request after 5 seconds. Maximum number of retries should be limited to 5.