

# **MX-ONE - Proprietary IP Client**

INTERWORKING DESCRIPTION



### **Copyright**

© Copyright Aastra Technologies Limited, 2017. All rights reserved.

### **Disclaimer**

No part of this document may be reproduced in any form without the written permission of the copyright owner.

The contents of this document are subject to revision without notice due to continued progress in methodology, design and manufacturing. Aastra shall have no liability for any error or damage of any kind resulting from the use of this document.

# 1 General

This document describes the communications protocol between MX-ONE and proprietary IP Clients. As of today, the only existing type of proprietary IP Client is the **Office IP-Phone**, so the interworking refers specifically to that client. When new clients are developed, the protocol detailed here can be adapted to the new clients or used as it is.

This interworking is based in a subset of the WAP standard as specified in Version 1.2, 4-Nov-1999. In particular, the interaction is based on the WSP (version 5-Nov-1999) and WML (version 4-November-1999) specifications.

## 2 General protocol use

The interaction between the proprietary IP-Client and the MX-ONE will be of the Client-Server type with several communication layers. On the highest layer (application layer), the communication will be made via WML documents containing the information to be transferred between entities. Each WML document contained in any given message is called a **deck**, and is composed by a number of WML **cards**.

On a lower level, a connectionless session layer will support the communication, and this will be itself on top of an unreliable transport layer.

The next sections will describe these different layers in a more detailed way.

## 3 Transport layer use

In order to decrease the overhead caused by TCP connections on the Network Interface elements of a system, it has been decided to use the more lightweight UDP as the underlying transport protocol for WSP.

There will be a reserved UDP port per IPLU/MGU board for WSP in each MX-ONE's Gatekeeper. If opened, then the Gatekeeper must listen on this address for incoming WSP messages from any registered endpoint. In the same way, on the client side, each endpoint will use a UDP port for WSP. The port number to be used in the server belongs to the IANA-registered port numbers dominion and it's the particular one

assigned to the WSP connectionless service. In principle, the client may choose any port it wishes to carry out the communication, but it's recommended to use the same port number as the server does. The IANA-registered port number is:

**Port number for WAP Connectionless Session Service = 9200**

## 4 Session layer use

The connectionless session service of WSP will be used. In this service there are no reliability mechanisms, so the reliability will have to be added at the Application layer.

In the connectionless session service of WSP, only three primitives are available to the Application layer, being different for client applications and for server applications. These primitives are:

- **S-Unit-MethodInvoke(.req/.ind)** : invoke a method in the server in a non-confirmed manner.

This service primitive has the following parameters and lengths:

- *Server Address* : UDP address (6 octets). Mandatory.
- *Client Address* : UDP address (6 octets). Mandatory.
- *Transaction Id* : unsigned integer (1 octet). Mandatory.
- *Method* : unsigned integer (1 octet). Mandatory.
- *Request URI* : null-terminated string (max. 256 octets including terminating null). Mandatory.
- *Request Headers* : set of attribute information (any number of octets). Optional.
- *Request Body* : data associated to the request (any number of octets). Not included if *Method* does not allow an entity-body, otherwise it is optional.

- **S-Unit-MethodResult (.req/.ind)** : return the result of a method invocation from the server in a non-confirmed manner.

This service primitive has the following parameters and lengths:

- *Server Address* : UDP address (6 octets). Mandatory.
- *Client Address* : UDP address (6 octets). Mandatory.
- *Transaction Id* : unsigned integer (1 octet). Mandatory.
- *Status* : unsigned integer (1 octet). Mandatory.

- *Response Headers* : set of attribute information (any number of octets). Optional.
- *Response Body* : data associated to the response (any number of octets). Should be included if *Status* indicates an error, otherwise it is optional.
- **S-Unit-Push (.req/.ind)** : send unsolicited information from the server to the client in a non-confirmed manner.

This service primitive has the following parameters and lengths:

- *Server Address* : UDP address (6 octets). Mandatory.
- *Client Address* : UDP address (6 octets). Mandatory.
- *Push Id* : unsigned integer (1 octet). Optional.
- *Push Headers* : set of attribute information (any number of octets). **Content-Location** header should be included if location of the pushed entity needs to be indicated, otherwise it is optional. The rest of headers are optional.
- *Push Body* : data associated to the push (any number of octets). Optional.

IP clients will always play the client role of the session, whilst the MX-ONE will always play the server role.

IP clients can invoke only the primitive S-Unit-MethodInvoke.req, and can receive invocations of the primitives S-Unit-MethodResult.ind and S-Unit-Push.ind.

IPLP can invoke both the primitives S-Unit-MethodResult.req and S-Unit-Push.req, and can receive invocations of the primitive S-Unit-MethodRequest.ind only.

Following is a figure which shows the interaction model between clients and servers:

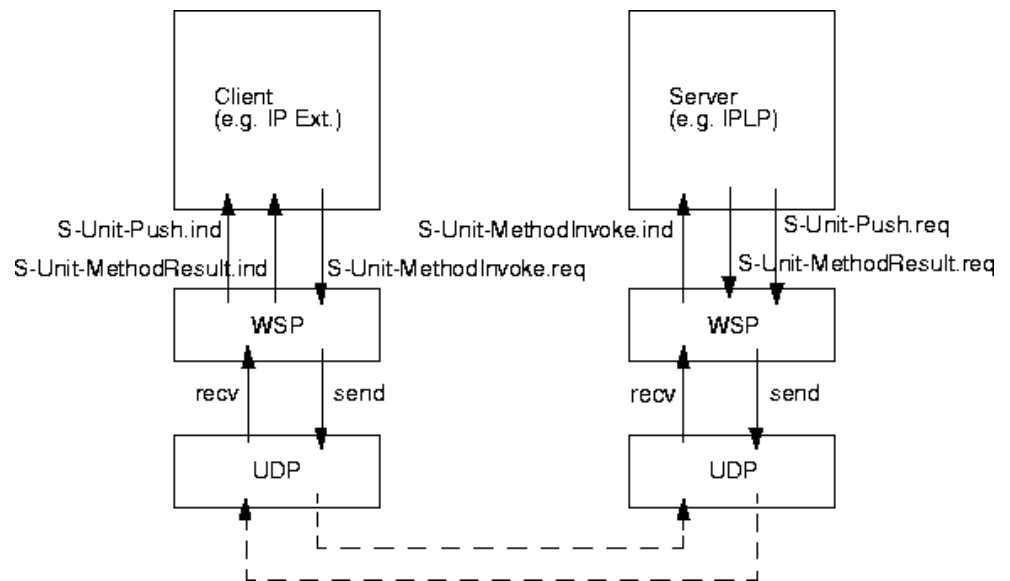


Figure 1: Client-Server interaction model using WSP primitives

## 5 Application layer use

Only a subset of the WML specification will be used for specifying the interactions between IPLP and the IP Client.

The following WML elements might be used for specifying the interaction. However, only those elements written with **boldface** in the list below must be supported; the rest of elements are for future use.

- **variable reference**
- **prev**
- **setvar**
- refresh
- **postfield**
- **go**
- do
- noop
- **a**
- **anchor**
- **onevent**

- **wml**
- template
- **card**
- select
- option
- timer
- table
- tr
- td
- **em**
- **p**
- br
- **normal ASCII text**
- WML Script used for CSTA

What follows is a description of the syntax of every WML element, together with its associated binary encoding and client behaviour.

The binary encoding has been obtained from [3] and section 14.3 in [1] and is included here only for convenience. In case of doubt, the standard must be followed. Please bear in mind that the binary encoding of the tags is not unique, i.e. according to the rules specified in [3], for any given tag, the following is valid:

- Each tag is coded in binary format in 1 byte
- The actual tag value is contained into the 6 lowest bits (B5..B0). The two most significant bits (B7 and B6) are flags, with the indicating if the tag has attributes (B7 = 1) and if there are contents inside the tag (B6 = 1).

So, in the subsections that follow, only one form (maximum two) of tag encoding is shown per each tag. Additionally, text written using *italic characters* represents text that must be replaced by text supplied by every particular WML description. Text enclosed within [square brackets] represents optional parts of an element, which may be omitted in a particular WML description. Text within (parentheses) represent different options, which are found inside the brackets and separated by commas.

## 5.1 Variable reference

### 5.1.1 Description

This element is used to reference a variable in the WML text.

Whenever a variable reference exists in a WML card, that reference must be replaced by its corresponding value at the moment of evaluating the reference.

### 5.1.2 Syntax

*\$ name*

### 5.1.3 Binary encoding

*42 name 00*

## 5.2 prev

### 5.2.1 Description

This element is used to order the client to go to the page that was being displayed before the one in which the prev element is found.

This implies that the client must have a store where the card that is being displayed just before jumping to another card is put. In other words, every time a go element is processed or a new card is received from the server, the current card must be put into this store. According to the WAP standard, a minimum of 10 WML cards must be kept stored in this space.

When a prev element is processed, the last card stored must be fetched and displayed from that space. Before displaying the mentioned card, the setvar elements inside the prev element must be processed.

### 5.2.2 Syntax

*<prev> set of setvar elements </prev>*

### 5.2.3 Binary encoding

*72 set of setvar elements 01*



## 5.3 **setvar**

### 5.3.1 **Description**

This element is used to assign value to a variable.

The attribute name is the name of the variable to set, and the attribute value is the value to assign to the variable.

### 5.3.2 **Syntax**

```
<setvar name=" name " value=" value "/>
```

### 5.3.3 **Binary encoding**

```
BE 21 03 name 00 4D 03 value 00 01
```

## 5.4 **refresh**

### 5.4.1 **Description**

This element is used to request the client to redisplay the current card's contents. It is useful when a card which contains some setvar elements referring to variables that are being displayed is processed, and the designer of the card wants the new values to be shown.

### 5.4.2 **Syntax**

```
<refresh/>
```

### 5.4.3 **Binary encoding**

```
36
```

## 5.5 **onevent**

### 5.5.1 **Description**

This element is used to specify actions the client must carry out when certain events take place.

There are two types of events: intrinsic events and application-defined events.

Intrinsic events are those defined by the standard and that must be supported by any standard-compliant client. Application-defined events, on the other hand, are events defined for every particular application and that do not have to be supported by all client implementations.

In this particular implementation, the intrinsic events used are `ontimer` and `onenterforward`. No application defined events are used in our present implementation. What follows is an explanation of the meaning of every event:

- `ontimer`: this event is triggered when a timer specified by the deck template or any card expires. Note that an `ontimer` event included in a card overrides an `ontimer` event included in the template.
- `onenterforward`: this event is triggered just before the card in which it is included is displayed.

### 5.5.2 **Syntax**

```
<onevent type="(ontimer,onenterforward)"> ( go element , prev element ) </onevent>
```

### 5.5.3 **Binary encoding**

```
F3 (3F,3E) 01 ( go element , prev element ) 01
```

## 5.6 **wml**

### 5.6.1 **Description**

This element indicates the start of a WML deck. A WML deck is similar to an HTML page, with the difference that a WML deck is splitted in several parts, called cards, in order to facilitate displaying of the deck in terminals with small displays.

## 5.6.2 Syntax

`<wml> deck contents </wml>`

## 5.6.3 Binary encoding

`7F deck contents 01`

# 5.7 template

## 5.7.1 Description

This element is used to define behaviours common to each and every card of the deck. **In this particular implementation, it is not used at all.**

## 5.7.2 Syntax

`<template> contents </template>`

## 5.7.3 Binary encoding

`7B contents 01`

# 5.8 card

## 5.8.1 Description

This element is used to define a WML card. A WML deck is divided in cards. A card's contents must be displayed as a whole on the terminal's display. However, a small display terminal is free to add scrolling functions to its user interface in order to make possible browsing the whole card.

## 5.8.2 Syntax

`<card id=" identifier "> contents </card>`

### 5.8.3 Bynary encoding

E7 55 03 *identifier* 00 01 *contents* 01

## 5.9 go

### 5.9.1 Description

This element instructs the client to get the URI associated to the go element. If the URI has no URL part, it is assumed that a local resource is being identified.

Cards inside a given deck can be identified by means of a '#' character, followed by the card identifier. For instance, <go href="http://myServer.se/myDeck#first"> instructs the client to issue a Get message for obtaining the card which identifier is "first" into the deck resource called "myDeck" in machine "myServer.se".

The attribute can be "post" or "get", being "get" the default value. That is, if the attribute is not present, a "get" method must be issued. **Note that "post" is not used at all in the present implementation**

Additional attributes for the message may be specified in the optional *set of postfield* elements. The only difference is that while in the "post" case, those attributes will be included in the Content field of the message as WML data, in the "get" case, they will be included in the URI.

### 5.9.2 Syntax

The syntax is:

```
<go href=" URI "> [ set of setvar elements ] [ set of postfield elements ]
</go>
```

### 5.9.3 Binary encoding

The encoding is:

```
[AB,EB] 4A 03 URI 00 01 [ set of setvar elements 01] [ set of postfield
elements 01]
```

The second variant is encoded as:

If the URI refers to a resource obtainable via http (i.e. "http://..."), substitute 4A with 4B. The AB value is used when no optional setvar or post-field elements are present.

## 5.10 **postfield**

### 5.10.1 **Description**

This element is used together with `go` elements to specify the additional attributes that a message, generated by a `<go href="URI" method="...">` element, will send to the server.

### 5.10.2 **Syntax**

```
<postfield name=" name " value=" value "/>
```

### 5.10.3 **Binary encoding**

```
A1 21 03 name 00 4D 03 value 00 01
```

## 5.11 **do**

### 5.11.1 **Description**

This element instructs the client to present a user interface element to the user which, when selected by the user, causes the execution of the action defined by the enclosed `go` element, i.e. sends a "get" message to the server.

The `label` attribute specifies a text which may be displayed by the user interface to help the user to identify the function of the user interface element associated to the `do` element.

The `name` attribute specifies the name of the user interface element. It is used for internal purposes only.

Please note that this tag is not used in our present implementation.

### 5.11.2 **Syntax**

```
<do type="accept" [label=" label "] name=" name "> go element </do>
```

### 5.11.3 **Binary encoding**

```
E8 38 [18 03 label 00] 21 03 name 00 01 go element 01
```

## 5.12 **noop**

### 5.12.1 **Description**

This element is used for "shadowing" tasks defined in the deck template. When a task is "shadowed", it is disabled for the card containing the noop element. Tasks are "shadowed" based on the name attribute for go elements, or on the type attribute for onevent elements. In other words, a noop task shadows a template task if both the template element and the card element have the same name value, in case of do elements, or the same type value, in case of onevent elements.

### 5.12.2 **Syntax**

```
<do type="accept" name=" name "> noop </do>
<onevent type="(ontimer,onenterforward)"> noop </onevent>
```

### 5.12.3 **Binary encoding**

```
E8 38 21 03 name 00 01 31 01
F3 (3F,3E) 01 31 01
```

## 5.13 **anchor**

### 5.13.1 **Description**

This element is used to define a link associated to a particular element (text in our implementation). These links have an associated task that specifies the desired behaviour when the anchor is selected. In our particular implementation for the OIP-Phone, an anchor is associated to a certain soft-keys and is selected upon pressing that soft-key.

### 5.13.2 **Syntax**

```
<anchor>label
<go href=" URI "> [ set of setvar elements ] [ set of postfield elements ]
</go>
</anchor>
```

### 5.13.3 Binary encoding

62 03 label 00 [AB,EB] 4A 03 *URI* 00 01 [ *set of setvar elements* 01] [ *set of postfield elements* 01] 01 01

## 5.14 a

### 5.14.1 Description

This element is a simplified form of the <anchor> element and is always bound to a go task without variables.

### 5.14.2 Syntax

<a href=" *destination URI* "> *text* </a>

### 5.14.3 Binary encoding

DC 4A 03 *destination URI* 00 01 03 *text* 00 01

## 5.15 select

### 5.15.1 Description

**This element is not used in our present implementation .**

This element is used to describe selection lists which could be presented in the user interface of the client.

The selection lists are single selection lists, i.e. only one element of the list can be selected at a time.

Index numbers of options range from 1 for the first option to n for the last option, being n the number of options in the select list.

### 5.15.2 Syntax

<select iname="i"> *set of option elements* </select>

### 5.15.3 Binary encoding

F7 16 03 69 00 01 *set of option elements* 01

## 5.16 option

### 5.16.1 Description

**This element is not used in our present implementation .**

These elements are used for building up selection lists. They are always enclosed in select elements.

The title attribute represents the text that should be shown on the user interface of the client when rendering the option element.

The onpick attribute specifies the URI which will be included in a Get message, which will be sent by the client when the user interface element that represents the option is actuated.

### 5.16.2 Syntax

<option title=" *title* " onpick=" *URI* "/>

### 5.16.3 Binary encoding

B5 36 03 *title* 00 24 03 *URI* 00 01

## 5.17 input

### 5.17.1 Description

**This element is not used in our present implementation .**

This element is used for collecting text strings from the user via a user interface element.

The title attribute specifies the text string that should be used as prompt in the user interface element associated to the input element.

The size attribute specifies the number of characters the user interface element associated to the input element should show at a time.

The maxlength attribute specifies the maximum number of characters that can be entered through the user interface element associated to the



input element. If `maxlength > size`, the user interface element might carry out some sort of text scrolling in order to show the last entered characters at any moment.

The `name` attribute specifies the name of the variable that will receive the collected string.

The `value` attribute specifies the default value for that variable. The user interface element associated with the input element should show this value by default as if the user had entered it.

The `type` attribute specifies whether the characters typed by the user should be made visible or, on the other hand, they should be shown as generic characters (e.g. asterisks '\*'). If this attribute is present, the user interface should hide the actual characters typed by the user, showing a generic character in replace.

The `format` attribute specifies the allowed characters in the text string. It is a string of characters, with every character representing one character type. The possible values are:

- A: uppercase alphabetic or punctuation character
- a: lowercase alphabetic or punctuation character
- N: a numeric character
- X: any uppercase character, alphabetic or numeric or punctuation
- x: any lowercase character, alphabetic or numeric or punctuation

## 5.17.2 Syntax

```
<input [title=" title "], size= size , maxlength= maxlength , name=" name " , [value=" value " ,] [type="password",] format="(A,a,N,X,x)"/>
```

## 5.17.3 Binary encoding

```
AF [36 03 title 00] 31 03 size 00 1A 03 maxlength 00 21 03 name 00
[4D 03 value 00] [3B] 12 03 (41,61,4E,58,78) 00 01
```

## 5.18 timer

### 5.18.1 Description

This element is used to specify a timer which must be started by the client when rendering the card in which the timer element is located.

The timer is initialized to the value specified by the value attribute, in units of 1/10th of second. Note that value is a string which must be converted to an integer number by the client.

When the timer expires, the event associated to the onevent element present in the card (in case there's one) is triggered. If there's not ontimer element in the card, nothing happens.

## 5.18.2 Syntax

```
<timer value=" value "/>
```

## 5.18.3 Binary encoding

```
BC 4D 03 value 00 01
```

# 5.19 em

## 5.19.1 Description

This element has the function of marking page elements for display with special attributes.

For this specific implementation, page elements marked with the em element will be displayed using blinking characters.

## 5.19.2 Syntax

```
<em> text </em>
```

## 5.19.3 Binary encoding

```
69 03 text 00 01
```

## 5.20 **p**

### 5.20.1 **Description**

This element is used to specify the grouping and alignment of the enclosed elements. If no alignment is specified, left alignment is assumed.

### 5.20.2 **Syntax**

`<p> set of elements </p>` or  
`<p align="(left,center,right)"> set of elements </p>`

### 5.20.3 **Binary encoding**

60 *elements* 01 or  
 E0 (08,07,0A) *elements* 01

## 5.21 **br**

### 5.21.1 **Description**

This element is used to specify the start of a new line. Text before and after the br element should be displayed in different, consecutive lines on the user interface.

### 5.21.2 **Syntax**

`<br/>`

### 5.21.3 **Binary encoding**

26

## 5.22 Normal ASCII text

### 5.22.1 Description

This element is not such. It just represents the normal text which should be shown on the user interface.

**Only ASCII characters with codes the range 32 - 127 are allowed.**

### 5.22.2 Syntax

*string of characters*

### 5.22.3 Binary encoding

*03 ASCII codes for string of characters 00*

## 5.23 table

### 5.23.1 Description

This element is used to define a table. Tables are not used in this particular implementation yet, although they could be in further versions of the document.

### 5.23.2 Syntax

*<table columns=" number "> set of table row elements </table>*

### 5.23.3 Binary encoding

*DF 53 03 number(in text form) 00 01 set of table row elements 01*

## 5.24 **tr**

### 5.24.1 **Description**

This element is used to define a row of a table. It's used just as a container for the cells in the row.

### 5.24.2 **Syntax**

`<tr> set of cell elements </tr>`

### 5.24.3 **Binary encoding**

`5E set of cell elements 01`

## 5.25 **td**

### 5.25.1 **Description**

This element is used to define a cell inside a row of a table. Note that a cell may be empty.

### 5.25.2 **Syntax**

`<td> text </td>` or `<td/>` (empty cell)

### 5.25.3 **Binary encoding**

`5D text 01` or `1D` (empty cell)

# 6 **Generic Message format**

In this section clauses that are of application to all the messages between MX-ONE and the IP client are written.

Messages between MX-ONE and the IP Client will be embedded in WSP PDUs and further carried by UDP packets.

IPLU/MGU must be transparent for the WSP PDUs. IPLU/MGU will provide a signalling interface for:

- Opening virtual UDP incoming connections (i.e. listening sockets)
- Sending and receiving UDP packets through those connections (sockets).

In the following figure, the transit of WSP messages throughout the IPL block is shown.

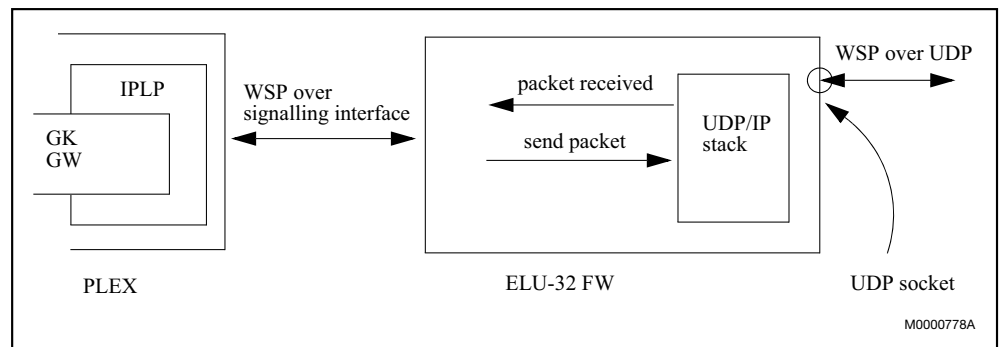


Figure 2: Flow of WSP messages through IPL block

## 6.1 Encoding of WSP PDUs

WSP PDUs are encoded according to the WSP specification.

There is a common PDU format for all the WSP messages, which is as follows:

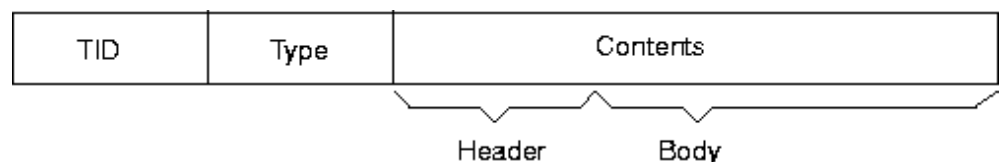


Figure 3: Common PDU format for all WSP messages

Fields in the figure are as follows:

- TID (8 bits) - an unsigned integer number that associates a request message with its corresponding response.

This field is mapped to/from parameters *Transaction Id* or *Push Id* of the session service primitive associated to this PDU (see 4 Session layer use on page 4 Session layer use ).

- Type (8 bits) - an unsigned integer number that identifies the type of this message. This field is mandatory. For the particular WSP implementation adopted in this document, it can be:
  - Push (0x06), sent using the S-Unit-Push.req primitive

- Get (0x40), sent using the S-Unit-MethodInvoke.req primitive
- Post (0x60), sent using the S-Unit-MethodInvoke.req primitive
- confirmedPush (0x07)

This field is mapped to/from parameter *Method* of the session service primitive associated to this PDU (see 4 Session layer use on page 4 Session layer use ).

- Contents (any number of octets) - message specific information

The Contents element is divided in two different parts: a message header part and a message body part.

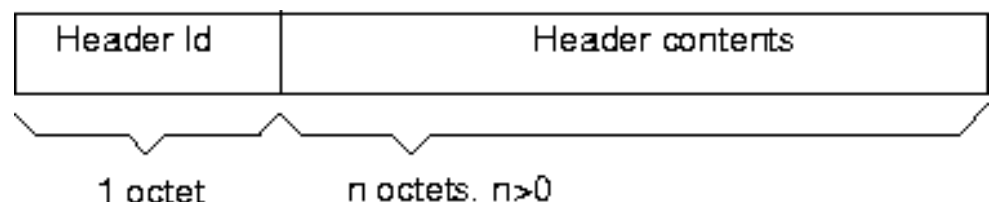
- The message header part is composed of fields mapped from/to the session service primitive parameters except *Transaction Id* , **Push Id** and *Method* , which are already mapped from/to TID and Type fields of the WSP PDU.

A message header is composed of different fields, depending on the value of the Type field and the session service primitive associated to the PDU being encoded/decoded, but the message header may in turn contain one or more headers, equivalent to HTTP/1.1 entity-headers and encoded as in on page 23

The specific message header contents and encoding for every value of the Type field and session service primitive associated to the PDU being encoded/decoded is described in 7 Message set on page 24 .

- The message body part is mapped from/to session service primitive parameters *Request Body* , *Response Body* or **Push Body** , depending on the session service primitive associated to this PDU. Encoding of the message body is application-dependant, but it is advisable to use the HTTP/1.1 format for entity-body elements for interoperability.

In section 9 Message set , the format of the field Contents for every Type value is described.



**Important:** for this version of the document, n=1

Figure 4: Encoding of headers in the Header part of field Contents

# 7 Message set

Following is the description, format and encoding of the set of messages that can be sent or received by clients and servers complying with this Interworking. In principle the scope is limited to MX-ONE and the OIP-Phone, but there's nothing particular to these entities in the interworking that limits its applicability to another entities like e.g. GSM phones, HTTP clients and servers etc.

The encoded messages according to the sections below, form what has been referred to as the Contents part of the WSP PDU in 8.1 Encoding of WSP PDUs .

## 7.1 Messages from client to server (IP Client -> IPLP)

Message	Type (hex)	Description
Get	40	Requests certain resource, identified by a Request-URI, to be delivered by the server to the client.

### 7.1.1 Get

This message is issued by a client when it wants to have some resource, identified by a Request-URI, delivered by the server.

The Get message is a transaction-oriented message. This means that it must have a Transaction Id associated, and that a response message is expected by the client that has issued the Get message.

The server, on reception of this message, should locate the requested resource and act take actions depending on that resource. Then, a response message must be generated, indicating the result of the request contained in the Get message.

The fields that compose the Get message are as follows:

- URILen (uintvar) - Length in octets of the URI field
- URI (URILen octets) - Identifier of the resource that the client wants to obtain. It must not be a null-terminated string, but a string with length determined by the URILen field.

This field is mapped to/from the *Request URI* parameter of primitive **S-Unit-MethodInvoke** .



- Headers (multiple octets) - set of Headers associated with the request. The length of this field comes determined by the length of the Contents field of the WSP PDU minus the length of the URILen and URI fields.

No headers, equivalent to HTTP/1.1 entity-headers, are used in our present implementation.

This field is mapped to/from the *Request Headers* parameter of primitive **S-Unit-MethodInvoke**.

Note that the Get message does not include any field which maps to the *Request Body* parameter of primitive **S-Unit-MethodInvoke**. This means that Get messages do not send any request content to the server.

## 7.2 Messages from server to client (IPLP -> IP Client)

Message	Message Type (hex)	Description
Push	06	Sends unsolicited information to the client, identifying a resource to which the information is related by a Content-Location header.
Reply	04	Returns a response code indicating the global outcome of the operation requested.

### 7.2.1 Push

This message is issued by a server which wants some unsolicited information to be received by the client. This information may be associated to a given resource by means of the Content-Location header.

The Push message is not transaction-oriented. That means that the server does not expect a response to be sent by the client when receiving this message.

A client, upon reception of this message, should look for the Content-Location header and, if present, associate the received information to the resource identified by the header. In case the Content-Location header was not present, the client actions when receiving this message are implementation-dependent.

If the Content-Location header is present and it identifies a resource inside the client, the client must store the received message body under that resource. This is mainly used by the server to update the instrument

(i.e. sending traffic case, tones etc.). In this case the Content-Location header field identifies the resource and the message body carries the attributes.

The fields that compose the Push message are as follows:

- **HeadersLen** (uintvar) - Length in octets of the **ContentType** and **Headers** fields combined
- **ContentType** (multiple octets) - Identifier of the type of contents of the **Data** field.

This field is mapped to/from the *Content-Type* header of the *Push Headers* parameter of primitive **S-Unit-Push**.

- **Headers** (HeadersLen minus length of **ContentType** octets) - The headers, equivalent to HTTP/1.1 entity-headers, that must be possible to send by servers and receive by clients for Push messages are defined in 1 Headers allowed in Push messages on page 26

This field is mapped to/from the *Push Headers* parameter of primitive **S-Unit-Push**.

- **Data** (multiple octets) - Information, optionally associated to the resource identified by a Content-Location header.

The length of this field is determined from the length of the **Contents** field of the WSP PDU minus the combined length of the rest of fields that compose the Push message.

This field is mapped to/from the *Push Body* parameter of primitive **S-Unit-Push**.

*Table 1 Headers allowed in Push messages*

Header	Id (hex)	Contents
Content-Location	0E	URL, identifying the resource to which the contents of the <b>Data</b> field of the message is associated.
Content-Type	11	"application/vnd.wap.wbxml" (encoded as 29 hex), identifying the type of contents of the <b>Data</b> field of the message.

**Note:** Please observe that, as presented above, the Content-Type header maps to/from the Content Type field of the WSP PDU. Therefore the Content-Type header won't be included within the Headers field of the PDU, in order to avoid having duplicate information.

## 7.2.2

### Message Reply

This message is a generic response message issued by a server as response to a Get or Post message received from a client.

The response message must have the same *TID* value as that of its associated request.

The fields that compose the Reply message are as follows:

- Status (one octet) - result code of the attempt to understand and satisfy the request.

The possible result codes are equivalent to HTTP/1.1 result codes and are shown in 2 Status codes and their meanings on page 28.

This field is mapped to/from parameter *Status* of primitive **S-Unit-MethodResult**.

- HeadersLen (uintvar) - Length in octets of the ContentType and Headers fields combined
- ContentType (multiple octets) - Identifier of the type of contents of the Data field.

This field is mapped to/from the Content-Type header of the *Response Headers* parameter of primitive **S-Unit-MethodResult**.

- Headers (HeadersLen minus length of ContentType octets) - The headers, equivalent to HTTP/1.1 entity-headers, that must be possible to send by servers and receive by clients for Reply messages are defined in 3 Headers allowed in Reply messages on page 28.

This field is mapped to/from the *Response Headers* parameter of primitive **S-Unit-MethodResult**.

- Data (multiple octets) - Information casted by the response.

**In our implementation, contents of the Data field is a human-readable string which explains the cause of the error. These characters must NOT be shown to the user by the client when receiving the reply.**

The length of this field is determined from the length of the Contents field of the WSP PDU minus the combined length of the rest of fields that compose the Push message.

This field is mapped to/from the *Response Body* parameter of primitive **S-Unit-MethodResult**.

*Table 2 Status codes and their meanings*

HTTP Status Code	Id (hex)	Meaning
200	20	OK, success
400	40	Bad request - server could not understand request
403	43	Forbidden - operation is understood but refused
404	44	Not found
405	45	Method not allowed
406	46	Not acceptable
409	49	Conflict
500	50	Internal server error
501	61	Not implemented
504	64	Gateway timeout

*Table 3 Headers allowed in Reply messages*

Header	Id (hex)	Contents
Content-Location	0E	URL, identifying the resource to which the contents of the Data field of the message is associated.
Content-Type	11	"text/plain" (encoded as 03 hex).
Date	12	Current Date & Time according to the WAP standard. Long integer value (4 bytes).

**Note:** As explained in a later section, the Date Header will only be present when the Get message contains a "Status Information Request" service.

**Note:** Please observe that, as presented above, the Content-Type header maps to/from the Content Type field of the WSP PDU. Therefore the Content-Type header won't be included within the Headers field of that PDU, in order to avoid having duplicate information.

# 8 Tag identifiers

## 8.1 Identifiers for global resources

Global resources are those which can be found both in clients and servers.

The global resources used in the implementation can be denoted by a Uniform Resource Identifier (URI) of the form:

"<URL>/<path>/<resource name>"

, being <URL> a valid URL for the server or client machine according to RFC2068, <path> a string of names separated by slashes ('/') indicating the internal location of the resource inside the server or client, and <resource name> the assigned name into the storage space of the client or the server.

In order to allow servers to modify resources previously transferred to clients, a given client receiving a resource from a server must store that resource keeping the same path and name that the resource has in the server.

For instance, the resource identified by "wsp://ASB501.ericsson.se/DirNo/U1", where "ASB501.ericsson.se" is the URL for the server, must be stored in a client which receives it as "PCClient1.ericsson.se/DirNo/U1", being "PCClient1.ericsson.se" the URL for the client.

### 8.1.1 Proprietary IP Client Resource Identifier

A particular type of proprietary IP Client can be represented by a **unique** logical model which includes all the specific features of that kind of client. These features are:

- Display information: that is, the particular set of WML-defined screens optimized for that type of client.
- Status information: that is, ring type, tone type, traffic case, traffic subcase and visual indicators for several services.

This resource is of the "application/vnd.wap.wbxml" type, and it's used to allow MX-ONE to update the client state via *push* messages, identifying the resource and its corresponding attributes in the Content-Location header and the Body part of the WSP PDU respectively. Additionally, all the requests made by the client to the server will be also referred to that particular resource.

The resource identifier is:

**Proprietary IP Client logical representation URI = "<URL>/DirNo/Ux"**

being <URL> a valid URL for the client or server machines, DirNo the client directory number and Ux the particular type of user interface the client presents. For the OIP-Phone, the user interface is designed to be as similar as possible as that of a DBC203 DTS, and is denoted by U1, so

**OIP-Phone logical representation URI = "/DirNo/U1"**

Please note that when sending the URI (Content-Location) in Get/Reply or Push messages, the local resource form must be used. This form allows the resource to be referenced as "/DirNo/U1", being the <URL> part implicitly stated by the message receiver's address.

## 8.2 Identifiers for local resources

Local resources are those that can be found only in clients or in server, but not in both.

The meaning and contents of local resources are known to the resource holder only. For instance, the resource named "wsp://PCClient.ericsson.se/H323" represents the H.323 stack in the client with URL "wsp://PCClient.ericsson.se". A server may send a push message with information to be stored under this resource, but will never receive the resource in a post message since it is local.

Please note that local resources are implementation dependant, and that it cannot be expected that a generic WAP terminal identifies these resources.

### 8.2.1 Resource identifier for the H.323 stack

*This resource is for future use. It does not have to be implemented for the products compliant to the present version of this document .*

The H.323 stack in a client or a server can be identified by:

**H.323 stack resource URI = "<URL>/DirNo/H323"**

being <URL> the URL of the client which holds the resource.

This local resource can be used once third-party control of the client is implemented. For instance, for requesting a client with DirNo=3000 to start a new call to the number 2000, a **push** message like the following might be sent from the server:

```
push(Content-Location="wsp://PCClient.ericsson.se/3000/H323,
Body="ac=sc&cn=2997")
```

## 8.3 Identifiers for resource attributes

In this section, the attribute identifiers associated to the resources described in the previous sections are listed.

Note that these identifiers are composed by two ASCII characters in order to reduce the number of octets sent.

### 8.3.1 Attribute Identifiers for the OIP-Phone logical representation

*Table 4 Attribute Identifiers for the OIP-Phone logical representation: server side*

ID	Length (bytes)	Attribute	Type of contents
'ac'	2	Action	actionType
'cN'	4	4	4 ASCII characters representing a 16-bit Call Signalling Call Reference Value in hexadecimal form (N = 1, 2, ...)
'kl'	M	Key List	String containing the key list
'st'	1	Status code used to response to PBX's action	statusType

These attributes are associated to the logical representation of the OIP-Phone on the MX-ONE side, i.e. they define properties of that representation in the server and are used only in the client to server communication. Please note that these attributes will be included in the URI header as a text string, so only pure ASCII characters are valid (i.e. no "strange" binary values outside the ASCII range).

*Table 5 Attribute Identifiers for the OIP-Phone logical representation: client side*

ID	Length (bytes)	Attribute	Type of contents
'ln'	1	Language	language
'rt'	2	Ring	ringType
'tt'	2	Tone	toneType
'tc'	2	Traffic Case	trafficCase
'ts'	4	Traffic Sub-case	trafficSubcase

ID	Length (bytes)	Attribute	Type of contents
'vi'	8	Visual Indicators	visIndType
'cr'	M	Calling Number	String containing the calling number
'ce'	M	Calling Name	String containing the calling name
'mn'	M	MMW Number	String containing the mmw number
'mw'	M	MW Number	String containing the mw number
'tn'	M	Translated Number	String containing the translated number
'kl'	M	Key List	String containing the key list
'info'	1	Command or information	infoType
'led'	M	The status of a certain LED	ledType
'nml'	M	NaMe List of the monitored extensions	String containing the name list

These attributes are associated to the logical representation of the OIP-Phone on the phone side, i.e. they define properties of that representation in the client itself and are used in the server to client communication so that the client can be updated.

Please note that, as those attributes will be sent from the server in text strings within WML cards, all of them will be pure ASCII characters.

Attributes in shadowed cells are not used in the current implementation.

## 9 Data types

In the following sub-sections the data types referenced in the "Type of contents" column of Attribute Identifiers for the OIP-Phone logical representation: server side and Attribute identifiers for the OIP-Phone logical representation: client side are defined.



## 9.1

### actionType

The attributes of this type are used to give a certain order to the associated resource.

In our present implementation, this type will state the particular service requested from the MX-ONE by the client, via a get message.

When the service requested concerns one or more calls, it is necessary that the service request carries a reference to each one of the calls involved. The parameter chosen to achieve this is the *Call Signalling Call Reference Value* (CRV from now on), as defined in the ITU-T H.225.0 recommendation. If more than a CRV is included in the request, then they must be ordered, being the CRV of the active call the first one in the parameter list.

Table 6 actionType

actionType	Value (hex)	Explanation
Account code	'ac'	An account code is entered. A parameter, defining the code, must be included in the request.
Authorization code	'au'	An authorization code is entered. A parameter, defining the code, must be included in the request.
Call-back	'cb'	Call-back is requested. A parameter specifying the CRV of the call against which the callback is requested must be included in the request.
Call Waiting	'cw'	Call waiting is requested. A parameter specifying the CRV of the call against which the call waiting is requested must be included in the request.
Call Intrusion	'ci'	Call intrusion is requested. A parameter specifying the CRV of the call to be intruded upon must be included in the request.
Conference	'cf'	Conference is requested. Two parameters specifying the CRVs of the calls to be put in conference must be included in the request. ( <i>In the future, a list of more than 2 CRVs could be included in the request</i> )
Individual Call Pick-up	'cp'	Individual Call Pick-up is requested. A parameter specifying the CRV of the call against which the call pickup is requested must be included in the request.

<b>actionType</b>	<b>Value (hex)</b>	<b>Explanation</b>
Message Diversion	'md'	Message diversion is requested. A parameter specifying the CRV of the call to be diverted to the message diversion position must be included in the request.
Call Transfer	'tr'	Call transfer is requested. Two parameters specifying the CRVs of the transferred and transferred-to calls must be included in the request.
Call Parking	'pk'	Call parking (hold) is requested. A parameter specifying the CRV of the call to park must be included in the request.
Call Retrieve	're'	Call retrieve is requested. A parameter specifying the CRV of the call to retrieve must be included in the request.
Status Information Request	'si'	Status information is requested. No additional parameters are needed.
Report Key List	'rkl'	
Acknowledgment	'ack'	

Attributes in shadowed cells are not used in the current implementation.

## 9.2 language

Attributes of this type notify the Proprietary IP Client the language in which the current display information is sent. For the use of this attribute in the OIP-Phone particular case, please 10.2 Requirements for the OIP-Phone on page 61

*Table 7 language*

<b>language</b>	<b>Value (hex)</b>	<b>Explanation</b>
English	30	ASCII code associated to '0'
French	31	ASCII code associated to '1'
German	32	ASCII code associated to '2'
Spanish	33	ASCII code associated to '3'
Italian	34	ASCII code associated to '4'

language	Value (hex)	Explanation
Other	35-39	Spare languages. Modifiable by MX-ONE command

## 9.3 ringType

Attributes of this type notify the Proprietary IP Client a certain ring must be generated. The phone is allowed to generate any sound on its own, i.e. in an OIP-Phone the tone might be a strong ring played through the loudspeaker, but in a PC-based client it might be a sound file in .WAV format.

Please note that the client will receive a text string with two ASCII characters, e.g. if the value is 88, the associated WBXML string will be 03 38 38 00.

*Table 8 ringType*

ringType	Value (hex)	Explanation
Internal call	88	Ring used when called by an internal party
1st ring tone	89	Ring used when receiving a call while in another call
External call	8A	Ring used when called by an external party
Callback	8C	Ring used when called, in callback execution
Stop ring	8E	Stop playing any ring
Voice Mail	B2	Ring used when called by the Voice Mail system

**Note:** Values for ringType attributes correspond to the ring tone numbers defined in FS 379/155 17 -APD 101 02 Rev. G1.

## 9.4 toneType

The attributes of this type tell the proprietary IP Client the tone it must play through the handset speaker or loudspeaker. Again, any given proprietary IP Client is free to generate the particular sound it's desired.

Please note that the client will receive a text string with two ASCII characters, e.g. if the value is 03, the associated WBXML string will be 03 30 33 00.

**Note:** The following list has been obtained from FS 83/155 17 -APD 101 02 Rev. H, and is repeated here for convenience. All the values used here are decimal, not hex.

<b>No.</b>	<b>Name</b>
<b>00</b>	Silence (pause)
<b>01</b>	Dial tone
<b>02</b>	Special dial tone
<b>03</b>	Busy tone
<b>04</b>	Congestion tone
<b>05</b>	NU (Number Unobtainable) tone
<b>06</b>	Warning tone for intrusion
<b>07</b>	Switching tone
<b>08</b>	Faultmans ringback tone
<b>09</b>	Inverse ring tone
<b>10</b>	Test tone
<b>11</b>	Call waiting tone
<b>12</b>	Interception tone
<b>13</b>	Verification tone
<b>14</b>	Warning tone for conference
<b>15</b>	Ring tone
<b>16</b>	Call diversion
<b>17</b>	1st ring tone
<b>18</b>	Queue tone
<b>19</b>	Call tone for loudspeaker paging
<b>20</b>	- (not used)
<b>21</b>	Tone used for routine check
<b>22</b>	Music on hold1
<b>23</b>	Music on hold2
<b>24</b>	Music on hold3
<b>25</b>	Special ring tone
<b>26</b>	External dial tone

27	Warning tone for intrusion from operator
28	Recall dial tone
29	Call waiting tone, external
30	Offhook queuing tone
31	Expensive route warning tone
32	Message waiting dial tone
33	Message waiting special dial tone
34	Disconnect tone
35	Tone for recall to operator at no answer, CAS - ETN
36	Tone for recall to operator after camp-on, CAS - ETN
37	Tone for confirmation of parking/Recall to operator after parking, CAS - ETN
38	Tone for internal call to operator/Transfer to operator, CAS - ETN
39	Confirmation tone
41	Global dial/reorder tone
42	Global ring tone
43	Global busy tone
44	Global interception tone
45	External busy tone
46	External ring tone
47	External generated tone

The tone with number 0 (Silence) will be used by the server to instruct the IP Client to stop playing any tone.

For a description of every type of tone, refer to the FS.

## 9.5 trafficCase

Attributes of this type report the traffic case that is taking place. It is used for the IP Client to know what's going on in order to be able to react in a specific, non-standard way to certain traffic cases. *For products compliant with this version of the document, attributes of this type are not used.*

The following paragraphs have been obtained from FS 83/155 17 -APD 101 02 Rev. H., and are repeated here for convenience.

Table 9 *trafficCase*

<b>Traffic case</b>	<b>Value (hex)</b>	<b>Explanation</b>
Miscellaneous	00	Traffic case for several non-related sub-traffic cases
Free	01	Terminal is on-hook
Register	02	Terminal has gone off-hook or has requested a service which takes it into dialling a new number
Busy	03	Call has found a busy party
No progress	04	Call can not progress for some reason
Call originator	05	Called terminal is alerting
Call terminator	06	Terminal is receiving a call from another party
Speech	07	Speech path is established between parties
External line conn.	08	Terminal is connected to external line
Start call	09	Terminal receives notification of that a new call can be started or is about to be received
Parked	0A	Terminal has been parked by other party or has parked another party
Reserved	0B	Terminal has been reserved by MX-ONE
Div. on no reply	0C	Called party has diversion on no reply feature active
New call	29	
Call waiting request	30	Terminal is receiving a call waiting notification
Message waiting req.	40	Terminal is receiving a message waiting notification

## 9.6 **trafficSubcase**

Attributes of this type further define the traffic case that is taking place. *One attribute of this type cannot be present in a message when no attribute of type trafficCase is present.*

The following sections have been obtained from FS 83, and are repeated here for convenience.

*In this version of the document this attribute is of no use .*

### 9.6.1

#### **trafficCase = Register**

The states where the calling party is sending digits for addressing of another party (register signalling) are called Register States.

During an internal call, the calling extension enters Register State and receives dial tone after having lifted the handset. Register state is left when number analysis is finished.

##### REGISTER 1.

- New call from extension without direct diversion, follow-me, message diversion, Individual Do Not Disturb or Group Do Not Disturb activated.
- New incoming call from both-way external line or incoming external line.
- Cancellation of direct diversion, follow-me, message diversion or External Follow Me successful.
- Cancellation of diversion on busy or diversion on no answer successful.
- Extension without direct diversion, follow-me or message diversion activated has attempted to cancel an individual or all call-backs and this has been successful.
- Extension without direct diversion, follow-me or message diversion activated has requested call-back to a route or external line and this is permitted. The extension can now dial external digits and end characters.
- Programming of individual abbreviated number successful.
- Ordering of temporary diverttee position for night service successful.
- Cancellation of temporary diverttee position for night service successful.
- General cancellation has succeeded.
- Cancellation of Individual or Group Do Not Disturb successful.

##### REGISTER 2.

- New call from extension with direct diversion, follow-me, message diversion, Individual Do Not Disturb or Group Do Not Disturb activated.
- Ordering of direct diversion, follow-me, message diversion or External Call Forwarding successful.
- Ordering of diversion on busy or diversion on no answer successful.

- Ordering of Individual or Group Do Not Disturb successful.
- Extension with direct diversion, follow-me or message diversion activated has attempted to cancel an individual or all call-backs and this has been successful.
- Extension with direct diversion, follow-me or message diversion activated has requested call-back to a route or external line and this is permitted. The extension can now dial external digits and end characters.

REGISTER 4.

- Inquiry call from extension without direct diversion, follow-me or message diversion activated.

REGISTER 5.

- Call from extension, valid Account code has been dialled prior to dialling the called number. Tone visual message sent from system indicates that the entered Account code is valid and additional digits can be dialled.

REGISTER 6.

- Inquiry call from extension with direct diversion, follow-me or message diversion activated.

REGISTER 7.

- Call from extension, invalid Account code has been dialled prior to dialling the called number. Tone/visual message sent from system indicates that the entered Account code is invalid and additional digits can be dialled.

REGISTER 10.

- Call from extension or external line, valid Authorization code has been dialled prior to dialling the called number. Tone/visual message sent from system indicates that the entered Authorization code is valid and additional digits can be entered.

REGISTER 11.

- Call from external line, a valid DISA-number has been dialled. DISA message is activated. Send relevant B-answer information to cooperating exchange (The normal line signalling procedure which can also include EOS-message).

REGISTER 14.

- New call from extension which has "manual Message Waiting" activated.

REGISTER 15.



- New call from extension which has Message Waiting activated (Message Waiting from an information system).

REGISTER 16.

- New call from extension which has both Message Waiting (Message Waiting from an information system) and diversion or follow-me activated.

REGISTER 17.

- New call from extension which has both "Manual Message Waiting" and diversion or follow-me activated.

REGISTER 18.

- Extension has dialled the first part of procedure for External Call Forwarding and the system is waiting for the external number to be dialled.

REGISTER 19.

- Call to an External Follow me diverted extension from the moment when the "ECF" is detected until an outgoing trunk line is seized.

REGISTER 23.

- Procedure for interrogation of divertee position of dialled position.

REGISTER 255.

- No register message shall be sent due to hot line or direct call without digit reception.

## 9.6.2

### **trafficCase = Call Originator**

The states where the calling party has addressed the called party and receives FREE message are called Call Originator States.

During an internal call, the calling extension enters Call Originator State and receives ring tone when number analysis is finished and B-party is called. Call Originator State is left when B-party answers the call.

CALORG 1.

- Call to free extension, present individual or common operator (free or busy).
- Automatically (directly or at extending) initiated call- waiting call.

CALORG 2.

- Ordering of call-back at no answer has failed.

CALORG 4.

- Call-waiting towards a digital extension is executed.

- Call to extension with diversion on no answer and 2 s before diversion is executed. Calling party receives a diversion message ("connection click").

CALORG 5.

- Ordering of paging has succeeded, waiting for reply. Waiting for meet-me (the paged person will answer by dialling a procedure on a telephone set) after conclusion of speech on radio.

CALORG 6.

- The call is queued to common-bell.
- Call to internal group number when all members in the sought group are busy, but the call has been placed in a queue to the group. The calling party is an operator or an external line.

CALORG 7.

- The call is queued to a busy night common bell.

CALORG 8.

- Paging is ordered and the call is in queue for paging equipment.

CALORG 9.

- Call to external line, send ring tone.

CALORG 10.

- Call to external line, do not send ring tone.

CALORG 11.

- Operator in central PABX has an extension in satellite PABX parked via a Release Line Trunk (RLT) and calls a busy extension in the satellite PABX, via the RLT. Busy Camp-On acknowledge tone is received by operator via the RLT (CAS - ETN).

CALORG 12.

- Operator in central PABX has an extension in satellite PABX parked via a Release Line Trunk (RLT) and calls a special PBX group (used for parking purpose) in the satellite PABX, via the RLT. Parking acknowledge tone is received by operator via the RLT (CAS - ETN).

CALORG 13.

- Operator in central PABX tries to park an extension in satellite PABX via a Release Line Trunk (RLT). Parking rejection tone is received by operator via the RLT (CAS - ETN).

CALORG 14.

- Called party is free after Camp-On Busy in network (terminating PBX).

CALORG 15.

- Ordered Call Message is executed in network (terminating PBX).

CALORG 16.

- Ordered Call Offer is executed in network (terminating PBX).

CALORG 17.

- Intruded party has cleared during intrusion in network (terminating PBX).

CALORG 18.

- Ordered Call Back in network is executed. "Call Back Completed" message will be sent to A-party in originating PBX, indicating that ringing on supervised party has started (terminating PBX).

CALORG 19.

- Intrusion in network is executing, with or without prior validation (originating or terminating PBX).

CALORG 20.

- The call is queued to PBX/ACD group. Music from channel 1 connected.

CALORG 21.

- The call is queued to PBX/ACD group. Music from channel 2 connected.

CALORG 22.

- A call to busy extension, call waiting is permitted at extending. A queue record is seized.(terminating PBX).

CALORG 23.

- The call is queued to PBX/ACD group, do not send any tone.

CALORG 25.

- A call to busy fax extension, call is queued to a busy fax extension. (TLP27 application)

CALORG 26.

- A call to operator, call is queued to operator. (TLP27 application)

CALORG 27.

- A call to night answering position, call is queued to night answering position.(TLP27 application)

CALORG 28.

- Call-waiting has been executed do not send ring tone.

CALORG 29.

- Ordered call-offer is executed. Do not send ring tone. (Transit PBX in network).

### 9.6.3

#### **trafficCase = Speech**

The states where a call is answered and through-connection between calling party and called party is made are called Speech States. (Data Phase State for data extension).

During an internal call the calling party (and the called party) enters Speech State and gets speech connection when the called party answers the call. Speech state is left when any of the parties disconnect.

SPEECH 1.

- Speech position with internal party.

SPEECH 2.

- Speech position with external party.

SPEECH 3.

- Connected to conference as conference leader.
- Connected to alarm centre with more than two participants.

SPEECH 4.

- Connected to conference as conference participator.

SPEECH 5.

- Successful initiation of intrusion.Connected to an intrusion Call as intruding party.

SPEECH 6.

- Speech position with open line (both-way speech on operators instrument).

SPEECH 9.

- The call has been rerouted to a Vacant Number Answering position. B-answer shall not be sent.

SPEECH 10.

- Successful connection of a DTMF tone digit sender when End-To-End DTMF is used.

## SPEECH 11.

- Speech after recall on no answer to central operator via Release Line Trunk (CAS - ETN). Used in satellite PBX.

## SPEECH 12.

- Speech after direct or enquiry call to central operator via Release Line Trunk (CAS - ETN). Used in satellite PBX.

## SPEECH 13.

- Speech after recall from Camp-On busy to central operator via Release Line Trunk (CAS - ETN). Used in satellite PBX.

## SPEECH 14.

- Speech after recall from parked call to central operator via Release Line Trunk (CAS - ETN). Used in satellite PBX.

## SPEECH 15.

- Speech with external "Release Line Trunk"-categorized party (CAS - ETN). Used in central PBX.

## SPEECH 16.

- Third party has cleared during intrusion in network (terminating PBX).

## SPEECH 17.

- Forced release of third party during intrusion in network (terminating PBX).

## SPEECH 18.

- Two party speech after forced release of third party or third party has cleared during intrusion in network (originating PBX).

## SPEECH 19.

- Two party speech after clear from intruding party during intrusion in network.

## SPEECH 20.

- Voice mail DTMF. Successful connection of a DTMF tone digit sender when end-to-end DTMF is used.

## SPEECH 21.

- Speech after recall in transit exchange. A new answer is necessary to send to gateway exchange. (Only valid for CCS trunks)

## SPEECH 22.

- Speech after the announced party has answered. (Only valid for OL).

#### SPEECH 29.

- The call has been deflected from an ACD queue, with maintained queue position.

### 9.6.4

#### **trafficCase = No Progress**

The states where a call connection is cleared are called No Progress States.

No Progress States are entered after time-out in the different Call Progress States:

- Register States
- Call Originator States
- Busy Message States
- External line connected States

No Progress States are also entered in situations like:

- Congestion cases
- The other party in speech connection has replaced
- Dialling of vacant number or faulty format of service code
- Dialling of number or service code when not having the right class of service for this

During an internal call, when in speech connection, the party that remains off-hook will enter No Progress States and receive a No Progress Message when the other party goes on-hook. The remaining party leaves No Progress State when going on-hook.

#### NOPROGRESS 1.

- Time out when waiting for internal digit in register state.
- Time out when waiting for external digit in External Line Connected State (exllincon) and the dialled external number is for certain incomplete.

#### NOPROGRESS 2.

- Switch congestion or lack of common resources.

#### NOPROGRESS 3.

- Time out in Call Originator State (calorg).

#### NOPROGRESS 4.

- Time out in Busy Message State (bsymes).

## NOPROGRESS 5.

- Time out when waiting in External Line Connected State (exllincon) for proceed-to-send signal from cooperating exchange.

## NOPROGRESS 6.

- Call to route when all lines are busy or call to individual busy external line when ordering of call-back to route/external line is not permitted. Alternative routing may be carried out.

## NOPROGRESS 7.

- Calling party is barred for connection to called party due to categories or called party is blocked.

## NOPROGRESS 8.

- Called party is in line-locked-out state or in local mode (to EL, OL, TL or KL).
- Time out during timing sequence when incoming and outgoing external traffic (to TL).

## NOPROGRESS 9.

- Dialed number or service code is vacant.
- Attempt to use facility for last number redial on external calls without any number being stored.
- Attempt to use vacant individual abbreviated number.
- Unauthorized party attempting to use individual abbreviated number.
- Attempt by operator to initiate charging call via a route which is not a charging route.

## NOPROGRESS 10.

- Number to off duty marked individual operator or number to common operator in a night-switched exchange.

## NOPROGRESS 11.

- The other party in a speech connection has disconnected the call.

## NOPROGRESS 12.

- Ordering or cancellation of internal or external call-back has been successful.
- Programming of individual abbreviated number has been successful.

- Ordering or cancellation of temporary diverttee position for night service has been successful.
- General cancellation has succeeded.
- Activation/Deactivation of Do Not Disturb has been successful.
- The IRD activation/deactivation procedure has been completed successfully.
- The Selection of a new language for an extension has been successful.

NOPROGRESS 13.

- Ordering/cancelling of diversion or message diversion has failed due to class of service barring, diverttee position does not exist or ordering party is not authorized.
- Ordering of follow-me has failed due to vacant number or ordering party is not authorized.
- Diversion by-pass has failed due to class of service barring.
- Programming of individual abbreviated number has failed due to vacant number or class of service barring.
- Unsuccessful cancellation of call-back.
- Ordering/cancelling of temporary diverttee position for night service has failed due to the fact that the dialled line number is vacant, already busy or not intended to be provided with a temporary diverttee position.
- General cancellation has failed.
- Activation/Deactivation of Individual or Group Do Not Disturb has failed due to class of service barring or ordering party is not authorized.
- Ordering/cancelling of External Call Forwarding has failed due to Class Of Service barring or ordering party is not authorized.
- Activation/Deactivation of a list (IRD service) has failed.

NOPROGRESS 14.

- Dialled individual abbreviated number is not programmed.

NOPROGRESS 15.

- Call to a PBX-group/data group without available member.

NOPROGRESS 16.

- Attempt to answer common-bell call or group call pick-up call without being member of the group.



## NOPROGRESS 17.

- Queue-congestion at call to common-bell.

## NOPROGRESS 18.

- Attempt to answer common-bell call, group call pick-up call or paging call without any waiting call.

## NOPROGRESS 19.

- Ordering of paging call to receiver marked in unattended mode (absent marked).

## NOPROGRESS 20.

- Ordering of paging call has failed due to: Class of service barring Temporary congestion in call queue to pager Display message to receiver without display Speech call to receiver without speech circuit Sought number does not have receiver Congestion on speech path Other search in progress for this number
- Unsuccessful ordering of paging with after-dialling procedure when ringing to extension.

## NOPROGRESS 21.

- Dialed internal or external number is barred due to class of service or trunk line discrimination.
- Attempt to use a common abbreviated number which is not permitted due to class of service.

## NOPROGRESS 22.

- Alarm centre already has the maximal number of parties connected.

## NOPROGRESS 23.

- Answer on call-back to originator (Faultman"s ringback)

## NOPROGRESS 24.

- Connected operator has cleared.

## NOPROGRESS 25.

- Dialed procedure has wrong format.
- The capacity of the register"s digit store has been exceeded.

## NOPROGRESS 26.

- Party in outgoing call position to operator, when individual operator marks himself off-duty or last operator in group marks himself off-duty, whereupon night-service applies. The call can not be rerouted.

NOPROGRESS 27.

- Not used.

NOPROGRESS 28.

- Facility ordered with a service code can not be executed.

NOPROGRESS 29.

- Exchange is in emergency state and originator is not allowed to originate call.

NOPROGRESS 30.

- Terminator is in local mode (data extension).

NOPROGRESS 31.

- Originator is barred for connection to terminator due to terminal interfaces are not compatible.

NOPROGRESS 32.

- Time-out during handshaking between the modems when incoming or outgoing external traffic (data extension).

NOPROGRESS 34.

- Call to route and all external lines are busy and call-back to route is not permitted. Alternative routing may not be used (external route with terminal exchange route category).

NOPROGRESS 35.

- Sought party in other exchange busy. After dialling facilities (use of suffix digits) not permitted.

NOPROGRESS 36.

- Time-out when no answer at paging, at extended call before answer. Valid for operator only.

NOPROGRESS 37.

- A data extension has initiated "connect when free" on itself to check if it has been properly initiated.

NOPROGRESS 38.

- The calling party is blocked, in other words it is not permitted to initiate calls.

NOPROGRESS 39.

- Operator who is not permitted programming attempts to enter programming mode.

NOPROGRESS 40.

- Clear from external party in speech state or call terminator state. Clear from ISDN terminal in speech state or call terminator state.

NOPROGRESS 41.

- Unknown virtual call request (terminating or gateway PBX).

NOPROGRESS 42.

- Unknown service request (terminating or gateway PBX).

NOPROGRESS 43.

- Subscriber incompatible, SIC (Service Indicator Code) or BC (Bearer Capability) not allowed for termination on selected data or telephony extension (terminating PBX).

NOPROGRESS 44.

- Net service not allowed.
- Net service function restricted.
- Route optimization not allowed.

NOPROGRESS 45.

- Called party is message diverted (ICS diverted). Originator = digital extension.

NOPROGRESS 46.

- Invalid Account code has been dialled when Class Of Service is Forced Account code.

NOPROGRESS 47.

- Invalid Authorization code has been dialled and the user must hang up the phone and try again. Call is rejected and tone/visual message sent from system indicates that the entered Authorization code is invalid.

NOPROGRESS 48.

- Called data extension is in test mode.

NOPROGRESS 49.

- Access to individual trunk line has failed due to terminator is blocked.

NOPROGRESS 50.

- Call from operator or external line to group number without free member, but with at least one member busy and available, where the call can not be placed in queue to the group.

NOPROGRESS 51.

- Vacant number at call to voice mail.

NOPROGRESS 52.

- Extension user has successfully locked his/her own extension. Tone/visual message sent from system indicates that the extension has been locked successfully.
- Operator has successfully locked a user's extension. Tone/visual message sent from system indicates that the extension has been locked successfully.

NOPROGRESS 53.

- Extension user has successfully unlocked his/her own extension. Tone/visual message sent from system indicates that the extension has been unlocked successfully.
- Operator has successfully unlocked a user's extension. Tone/visual message sent from system indicates that the extension has been unlocked successfully.

NOPROGRESS 54.

- Extension user has failed to lock his/her own extension. Tone/visual message sent from system indicates that the attempt to lock the extension has failed.
- Operator has failed to lock a user's extension. Tone/visual message sent from system indicates that the attempt to lock the extension has failed.

NOPROGRESS 55.

- Extension user has failed to unlock his/her own extension. Tone/visual message sent from system indicates that the attempt to unlock the extension has failed.
- Operator has failed to unlock a user's extension. Tone/visual message sent from system indicates that the attempt to unlock the extension has failed.

NOPROGRESS 56.

- Internal call to an Individual or Group Do Not Disturb marked extension.
- Attempt to by-pass Individual Do Not Disturb from an extension.
- Attempt to by-pass Group Do Not Disturb from an extension which is not a Master extension.

NOPROGRESS 57.

- Originator's Facilities Restriction Level is not accepted for routing the call forward.

## NOPROGRESS 58.

- Call to an External Call Forwarding diverted extension when no outgoing trunk is available.

## NOPROGRESS 59.

- Call to an External Call Forwarding diverted extension has failed due to both A- and B-party lacks clearing signal.

## NOPROGRESS 60.

- Call to an External Call Forwarding diverted extension which is busy in an ECF call.

## NOPROGRESS 61.

- Ordering of External Call Forwarding from operator has been successful.

## NOPROGRESS 62.

- Cancellation of External Call Forwarding from operator has been successful.

## NOPROGRESS 63.

- Analogue extension in automatic answer mode after restart.

## NOPROGRESS 64.

- Max number of transit PBX's at routing in network exceeded (transit PBX).

## NOPROGRESS 65.

- Requested Malicious Call Tracing has been initiated. Send one way acknowledgement tone.

## NOPROGRESS 66.

- Requested Malicious Call Tracing can not be initiated. Send one way rejection tone.

## NOPROGRESS 67.

- Terminating PBX in network. Service request is not allowed when B-party, in terminating PBX, is paging equipment, modem or trunk line.

## NOPROGRESS 68.

- Clear of virtual call or clear of call connection for service request through network.

NOPROGRESS 69.

- ISDN-extension line busy, no free B-channel.

NOPROGRESS 70.

- No user responding on ISDN-extension line.

NOPROGRESS 71.

- Terminating PBX in network. Message to third party that forced release of third party has occurred.

NOPROGRESS 72.

- Originator's "Call Service Information" is not accepted for routing the call forward.

NOPROGRESS 73.

- Requested BC (Bearer Capability) not allowed for selected route (gateway, transit or originating PBX).

NOPROGRESS 74.

- Signalling systems incompatible. Requested supplementary service is not supported by the signalling system of the selected route or the service is not supported in gateway.

NOPROGRESS 75.

- Service Unavailable. The requested supplementary service is supported by the PBX, but not by the called party (terminating PBX in network).

NOPROGRESS 76.

- Service Temporarily Unavailable. The requested supplementary service is available on the PBX, but cannot be provided at the moment (terminating PBX in network).

NOPROGRESS 77.

- Unsuccessful attempt to unlock an extension that already is unlocked.

NOPROGRESS 78

- Direct call to an unobtainable ACD agent member.

NOPROGRESS 80

- "Hot line to" diversion procedure executed successfully.

NOPROGRESS 82

- Incompatible call, e.g. ISDN voice to data extension.

NOPROGRESS 83.

- Terminator is busy in local conversation, suffix usage is not permitted. Originator is a trunk line. Used in the China application.

NOPROGRESS 93.

- Called party is out of radio range or switched off.
- The call is disconnected when the last position in the active list reaches end of time supervision (not available) due to the execution of the IRD service.

NOPROGRESS 94.

- Terminator is busy in terminating PBX. No queue record seized. No net services available. A-party is always an operator.

### 9.6.5

#### **trafficCase = Parked**

An extension or operator that is in Speech State and connected to an internal or external party, can park the connected party. The parked party will remain in Parked State until:

- The extension or operator picks up the parked party
- Recall to operator after time-out is made (when parked by operator)
- Supervised extension becomes free (when parked with Camp-On-Busy by operator)
- The parked party disconnects

PARKED 1.

- Parked at Inquiry or Call Waiting or Call-Back (A has initiated Call-Back towards B. B has become free and has gone OFF-Hook while A is calling.

PARKED 2.

- Parked with Camp-On-Busy after operator extending.

PARKED 3.

- Parked by an operator.

PARKED 4.

- Inquiry parked by an extension, which is allowed to initiate Malicious Call Tracing. The parked party, which is external, shall not receive Music-On-Hold.

PARKED 5.

- Parked when calling a PBX group. Tone message Music-On-Hold1 will be received by parked party.

**PARKED 6.**

- Parked when calling a PBX group. Tone message Music-On-Hold2 will be received by parked party.

**9.6.6****trafficCase = Busy**

These states are entered when calling a busy party. Busy means that the called party is not available at the moment (e.g. in speech connection with another party when calling an extension).

The calling party stays in Busy Message State until time-out occurs and No Progress State is entered. An operator is not time supervised in Busy Message State and will not reach time-out.

An extension being in Busy Message State can use a post-dialling procedure, e.g. use of suffix digit to initiate call-back.

**BSYMES 1.**

- Called party is busy, in other words not free, not blocked, not in Line Lock-Out State. If originator is operator calling an extension, extension is busy with call camped-on.
- Call from extension to PBX-group/data group without free members but with at least one member busy and available.
- Call to Common Bell group with full queue.

**BSYMES 2.**

- All external lines are busy when making a call attempt to external route, but call-back to route can be ordered.

**BSYMES 3.**

- Ordering of call-back at busy to internal party has failed.

**BSYMES 4.**

- Initiation of call-waiting with post dialling procedure. (use of suffix digit) has failed.
- Initiation of connect-when-free towards a data extension has failed.

**BSYMES 5.**

- Initiation of intrusion has failed.

**BSYMES 6.**

- External route is busy at least cost routing. Another route selection can be executed by dialling a suffix digit.

**BSYMES 7.**



- Called party is busy, in other words not free, not blocked, not in Line Lock-Out State, with this call camped-on. (originator always = operator).

## BSYMES 8.

- Called party is busy, in other words not free, not blocked, not in Line Lock-Out State, with this call camped-on. Permitted to initiate call-waiting in connection with operator call extending (originator always = operator).

## BSYMES 9.

- Ordering of call-back at busy external route/trunk line has failed due to lack of common resources.

## BSYMES 10.

- Called party in cooperating exchange is busy. Possible to initiate facilities. (originator always = operator).

## BSYMES 11.

- Intrusion has failed due to lack of common resources.

## BSYMES 12.

- Call to a busy PBX group has been put in queue. Originator always = operator.

## BSYMES 14.

- Access to individual trunk line has failed due to terminator is busy.

## BSYMES 15.

- A Diversion Immediate has been executed to a busy party in own PBX (terminating PBX in network).

## BSYMES 16.

- Terminator is busy and intrusion is permitted. Originator is a trunk line. Used in the China application.

## BSYMES 17.

- All external lines are busy when making a call attempt to external route. Automatic ordering of call-back is allowed and no more digits are required to complete the external number.

## BSYMES 18.

- All external lines are busy when making a call attempt to external route. Automatic ordering of call-back is allowed and more digits are required to complete the external number.

## BSYMES 19.

- Ordering of call-back at busy external route/trunk line has failed due to that a call-back mission already exists for the ordering extension/operator.

BSYMES 20.

- Called party is out of radio range or switched off.
- The call is disconnected when the last position in the active list reaches end of time supervision (not available) due to the execution of the IRD service.

### 9.6.7

#### **trafficCase = External Line Connection**

The states, where the calling party is connected to an external line and is addressing the other party, are called External Line Connected States.

External Line Connected State is entered when external line is connected and is left when addressing is finished and the other party is called.

During a call from an extension to a subscriber in the public network, the extension enters External Line Connected State after having dialled access code to public trunk line whereupon external dial tone is received. The state is left when last digit in subscriber number is dialled and the subscriber is called.

EXLLINCON 1.

- No proceed-to-send signal (dial tone) to calling party.

EXLLINCON 2.

- Listening path throughout entire repetition of digits.

EXLLINCON 3.

- Listening path until first digit after proceed-to-send signal has been received.

EXLLINCON 4.

- Internal generation of proceed-to-send signal.i

EXLLINCON 5.

- Do not send ring tone or connect listening path until proceed-to-send arrives.

EXLLINCON 6.

- Call to an External follow me diverted extension from the moment when an outgoing trunk line is seized until Call Originator state is entered.

**9.6.8****trafficCase = Miscellaneous****LINE LOCKOUT 1.**

- Extension or data extension has reached time-out in No Progress State. Call has to be terminated to reach Idle State.

**OTGEXLCON 1.**

- For data calls only. External line is connected. Ready signal from trunk received. Awaiting ready signal from modem before entering Data Phase.

**INEXLCON 1.**

- For data calls only. "Incoming" external line is connected. Awaiting ready signal from modem before entering Data Phase.

**SPEEQUCON 1.**

- A call from operator to a data extension is answered (data extension state).

**CONWHNFRE 1.**

- For data calls only. A data extension is waiting for a busy data extension to become free ("on-hook" queuing).

**CALTER.**

- Call to terminator. The B-party is called but has not yet answered the call.

**9.7****visIndType**

The attributes of this type are used to specify the status of the different generic visual indicators an IP Client User Interface might have (e.g. in the current implementation of the OIP-Phone, the LEDs).

The actual mapping of one of these visual indicator to a physical or logical device in the IP Client is a responsibility of the client itself. For example, in a phone-like device these indicators would most probably be mapped to LED indicators.

The type is defined as an 8-byte String with every byte specifying the status (On/Off) of its associated indicator, which is dependent on a particular parameter of the instrument state. The meaning of every byte is in the table that follows:

*Table 10 visIndType*

Byte position	Explanation
0 (LSB)	Diversion
1	Callback
2	Message Waiting
3	Manual Message Waiting
4-7	<i>Not used in the present version of the implementation</i>

Please note that the client will receive a complete text string, for instance, if MW is activated, the client will receive 03 30 30 30 30 30 31 30 30 00, binary equivalent to "00000100".

## 9.8 String

*Table 11 String type*

Data Type	Length	Values
String	M	An indeterminate number of printable ASCII characters ended by a trailing byte with value 0x00.

# 10 Requirements on involved systems

This section lists all the requirements both the WAP Server and the OIP-Phone must fulfill in order to guarantee a seamless interworking for all the services and traffic cases.

## 10.1 Requirements on WAP Server

- The WAP server must not send tone information to the OIP-Phone when the call being set up is external and the Fast Connect procedure has been completed previous to the call setup completion.

**Note:** The Fast Connect procedure will not be implemented during this project.

## 10.2

## Requirements for the OIP-Phone

- The OIP-Phone is must handle the interaction between its internal H.323 stack, the WAP Interface and the User Interface, passing the adequate values and notifications amongst them as necessary.
- The OIP-Phone handles the time and date internally, and must take care of showing the current time and date to the end-user.
- The OIP-Phone has a set of internal screens and is responsible for handling them in an autonomous way. If no WAP information is available, the phone must still be able of carrying out a normal H.323 call.
- The OIP-Phone must handle the ODN keys by itself. If any of those keys is pressed, it is also responsible of following the behaviour as specified in this interworking, that is generate a get request to the server with the adequate action and parameters (see 6 actionType on page 33 ). If an ODN key with a call in speech state is pressed, a parking request must be issued. If an ODN key in which a call has been parked is pressed, a retrieve call request must be issued.
- The OIP-Phone must know the different function keys ("hard-keys") it has available and if any of those keys is pressed, it is also responsible of following the behaviour specified in this interworking, that is generate a **get** request to the server with the adequate action and parameters (see 6 actionType on page 33 ).
- The OIP-Phone must issue a "Status Information" request every-time the user acts on the CLEAR-call control or goes onhook (see 6 actionType on page 33 ).
- The OIP-Phone must take care of showing the <anchor> elements that model the soft-key menu in WML cards in such a way that the different options can be visually associated to the different soft-keys it has available. If a soft-key is pressed, then the phone is also required to follow the required behaviour as specified in the current WML card (send the request to the server with the adequate parameters as indicated in 6 actionType on page 33 ).
- The OIP-Phone is responsible for the dialling process.
- The OIP-Phone is responsible for any kind of visual indicators (like e.g. key LEDs). The OIP-Phone will have enough information from the server to be able to handle the following indicators: MW, MMW, CAB, DIV. The rest of the indicators must be handled based on

other information sources (e.g. call information received from the H.323 stack).

- The OIP-Phone is responsible for ring and tone generation, and may generate the particular sound it wishes for a certain situation. The OIP-Phone will play default rings and tones for every traffic case where a ring or tone is appropriate. However, if a ring or tone information is received from the server, the default ring or tone must be replaced by the ring or tone adequate to the information received from the server.
- The OIP-Phone will stop playing any ring or tone regardless of what might have been previously ordered from the server, when the call steps into connected state -i.e. **connect** received from peer- and when the user acts on the CLEAR-call control or goes onhook.
- The OIP-Phone must send the *call proceeding* message in all cases when receiving the *setup* from the MX-ONE. The server will not send any information to the IP Client until the *call proceeding* message has been received by MX-ONE, that is no information will be sent in the H.323 states: ReferenceExchange, LocalSeizure or CallInitiated.
- The OIP-Phone must synchronize the language used in the User Interface screens to the language received from the server in the current WML deck, to ensure that the information is always displayed in the same language regardless of which type of screen is being shown. This synchronization must only be done when the language value is 0, 1, 2, 3 or 4 ( corresponding to the standard languages supported by the MX-ONE: English, French, German, Spanish and Italian). For other languages, it cannot be guaranteed that both types of screens (User Interface and WML) are shown in the same language.

## 11 Impact on the H.323 signaling

In order to be able to start-up the WSP session between the proprietary IP Client and the MX-ONE, some new information needs to be included in certain H.323 messages.

## 11.1 Registration Request message (RRQ)

### 11.1.1 endpointVendor IE

The **endpointVendor** IE in the *RRQ* message must include information that identifies the IP Client as an proprietary client.

Specifically, the following fields must be provided by the IP Client upon registration:

- endpointVendor->vendor->t35CountryCode
- endpointVendor->vendor->t35Extension
- endpointVendor->vendor->manufacturerCode
- endpointVendor->productId
- endpointVendor->versionId

### 11.1.2 nonStandardData

Two data must be sent from a proprietary IP Client within the **nonStandardData** information element: two bytes for the desired **port number for WSP** in the client side and an additional byte to identify the requested **type of User Interface** (deck).

As of today, the only existing proprietary IP Client is the OIP-Phone, which will use the User Interface with resource name "U1" or "U3" (short for "User Interface, type 1 or type 3"). This type of interface will be coded as 1 or 3.

The purpose of this information is twofold. On one hand, to be used by the MX-ONE to know which kind of User Interface this particular terminal is using - in the future, there may be more User Interfaces stored in the MX-ONE for different types of terminals. On the other hand, to be used by MX-ONE to find the full UDP address **push** messages must be sent to. Other necessary data like Directory Number and IP Client IP-address are already sent in other standard information elements within the **RRQ** message.

The **nonStandardData** information element content must be identified by a **nonStandardData.nonStandardIdentifier.object** field. This field must be assigned the value XXXX, which belongs to the ERICSSON's own branch within ITU-assigned Object Identifiers.

For instance, an OIP-Phone which has been assigned the IP address 164.48.105.105 and Directory Number 3000, and it is using the recommended port number (2948) for WSP with User Interface "U3", will send the following data in the **nonStandardData** field of the *RRQ* message:

XXXX 0B 84 03

## 11.2 Registration Confirm message (RCF)

### 11.2.1 nonStandardData

If the registration is successful, the client is recognized as a proprietary IP Client and the WAP server within the MX-ONE is active, then the *RCF* message sent back from the MX-ONE to the proprietary IP Client will carry a **nonStandardData** information element in which two data will be contained: four bytes for the **server IP address** of the WSP protocol and two bytes for the **server port number** to be used for the protocol.

In a usual situation, the server IP address will be the IPLU/MGU IP address where the client is registered and the port number will be the IANA-registered one for connectionless WSP communication (i.e. 9200). However, the use of the mechanism explained above allows the server to redirect the communication to a different port within the same board, to a different board in the exchange or to a completely different server. This capability is not used today, but may be useful for future implementations of the protocol.

As for the *RRQ* message, the **nonStandardData** information element content in the *RCF* message must be identified by a **nonStandard-Data.nonStandardIdentifier.object** field. The value assigned to this field belongs to the ERICSSON's own branch within ITU-assigned Object Identifiers too. However, the actual value must be different from the *RCF* case, as the two contents are not the same and are associated to different events (i.e. "registration request from an proprietary IP Client" and "registration confirmation from the MX-ONE"). The assigned value for this case is YYYY.

For instance, if the selected IPLU/MGU board within the MX-ONE has an IP address of 164.48.106.133 and it is using the UDP port number 9200 for WSP, the byte string to send in the **nonStandardData** field of the *RCF* message should be:

YYYY A4 30 6A 85 23 F0

**Note:** To be able to make this implementation work, and as a temporary solution while adequate nonStandardIdentifiers are assigned for this, the values XXXX = 02 9A and YYYY = 02 9B have been chosen.



# 12 General Strategy - Use Cases

In this section, the three basic types of interaction between the OIP-Phone and the MX-ONE will be shown. These interactions may be useful in the future for other versions of proprietary IP Clients.

## 12.1 OIP-Phone registration

This step is the first one chronologically, and it's mandatory to initiate the communication between the OIP-Phone and the MX-ONE.

As explained in a previous section, the **nonStandardData** information element within the *RRQ* message sent from the OIP-Phone will contain: two bytes for the desired **port number for WSP** in the client side and an additional byte to identify the requested **type of User Interface** (U1 = 1)

If the registration is successful, the client is recognized as a proprietary IP Client and the WAP server within the MX-ONE is active, then the *RCF* message sent back from the MX-ONE to the proprietary IP Client will carry a **nonStandardData** information element in which two data will be contained: four bytes for the **server IP address** of the WSP protocol and two bytes for the **server port number** to be used for the protocol.

If no valid **nonStandardData** is received within the *RCF* message, the phone must assume neither services nor display information are available from the MX-ONE and act as a normal IP extension in a complete autonomous way, handling the H.323 basic call and displays by itself.

Once the registration process is complete, the MX-ONE will spontaneously send the OIP-Phone its current status information (initial WML deck) including the display configuration and service status. This deck will keep the general rules explained in Section 13 OIP-Phone WML deck format on page 71 .

Note that the IP Client must know in advance the number that identifies the desired user interface in the server. This information might be obtained from a configuration parameter.

The overall process is described in the following signal flow:

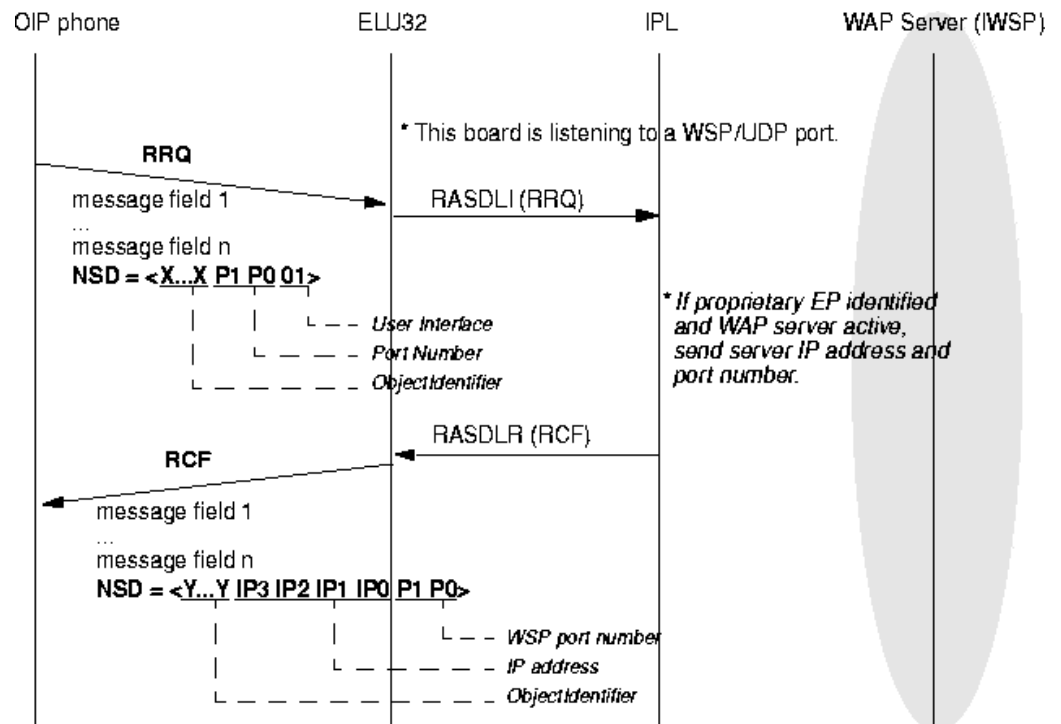


Figure 5: WSP/UDP initial communication at registration of OIP-Phone

## 12.2 OIP-Phone service invocation

Once registered, an OIP-Phone might request a number of different services ( 6 actionType on page 33 ). In the present implementation, the services are always requested via a *get* message in which the URI contains all the parameters needed to identify the service type and attributes in the server:

**URI = //ServIPAddress/DirNo/U1?ac=A1 A0[&c1=B3 B2 B1 B0[&c2=C3 C2 C1 C0]]**

That is, the service will be identified by two characters assigned to "ac" (actionType)., where those two characters are as stated in 6 actionType on page 33. Then, zero, one or two four-character CRVs (assigned to "c1" and "c2"), will define the calls involved in the service request.

Please do note that the URI form used includes the destination IP address. Although this is not strictly necessary from the MX-ONE point of view, it seems so in the OIP-Phone side. If a Domain Name Server is available in the future, then the complete URI, in the "wsp://ASB501.ericsson.se/DirNo/U1" form, could be used.

This service invocation can arise from three sources:

- A user selection of any active soft-key (menu key). The specific type of service is defined via an hyperlink in a given WML card, as the available services associated to menu keys are dependent on

the WML card being currently shown, that is, those services are constantly varying so the WML deck will specify which services exist or not in any given state.

- A user action on any function key, ODN key or CLEAR-call control (or going onhook). In this case, the specific type of service needs not be explicitly sent, because it's not associated to a WML card. Should this situation occur, the OIP-Phone **must** act as specified in this document and issue the get request with the adequate URI ( 6 ActionType on page 33 ). That is, if a parking situation occurs, then the adequate parking request must be sent, if a fixed key defined as message waiting key is pressed and a message waiting number is defined, then initiate a basic H.323 call to that number, etc.
- An automatic invocation done by the phone to synchronize its status and Date/Time information.

Once the request is processed by the MX-ONE, a *reply* message will be sent back to the OIP-Phone. This *reply* message will contain the actual result of the service execution in the header fields (as shown in 2 Status codes and their meanings on page 28 ) and a human-readable text string in the message body. This text-string must not be shown to the OIP-Phone end user, but it's sent for compliance with the RFC2068 (the text string and result might be conceivably used to generate some kind of internal log).

If the result of the service request is reported as an error, the OIP-Phone may ignore it, repeat the request or take independent actions according to the type of error reported.

The *reply* message will be immediately followed by a *push* message which will contain the new WML deck to display as a result of the service. This new deck will contain, as mentioned before, not only display information, but also all the new status information (tone, ring, visual indicators...) resulting from the service execution. Please note that, according with the general strategy described in previous sections, both messages (push and reply) will be referred to the same resource **//ClientIPAddress/DirNo/U1** , that is the OIP-Phone logical representation. Here the IP address will be included in the Content-Location for the same reasons it was included in the URI.

The process is summarized in the following figure:

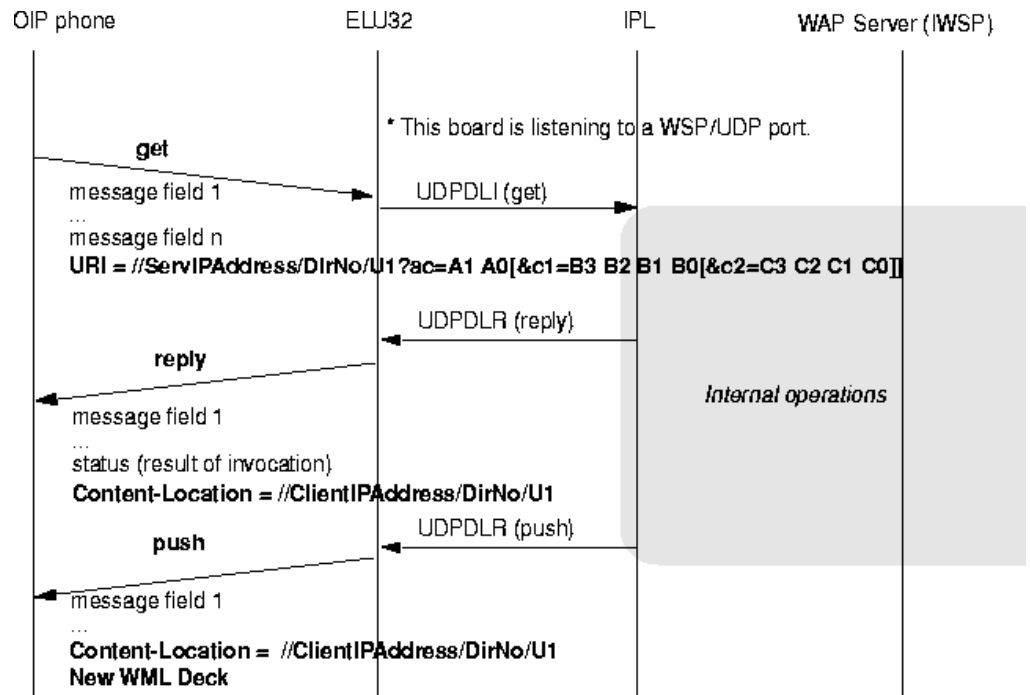


Figure 6: OIP-Phone service invocation

## 12.3 MX-ONE notification to OIP-Phone

There are many situations in which the MX-ONE needs to send unsolicited information to the OIP-Phone. This happens, for instance, when the operator activates a service towards the phone, when there's an incoming call, or when a special tone must be sent (e.g. off-hook queuing tone).

All these situations will be treated the same way, by sending a *push* message in which the information will be contained in the body part of the message, as explained before. This information will enable the phone to show the correct display information and modify ring, tone or the state of the visual indicators to reflect its new state.

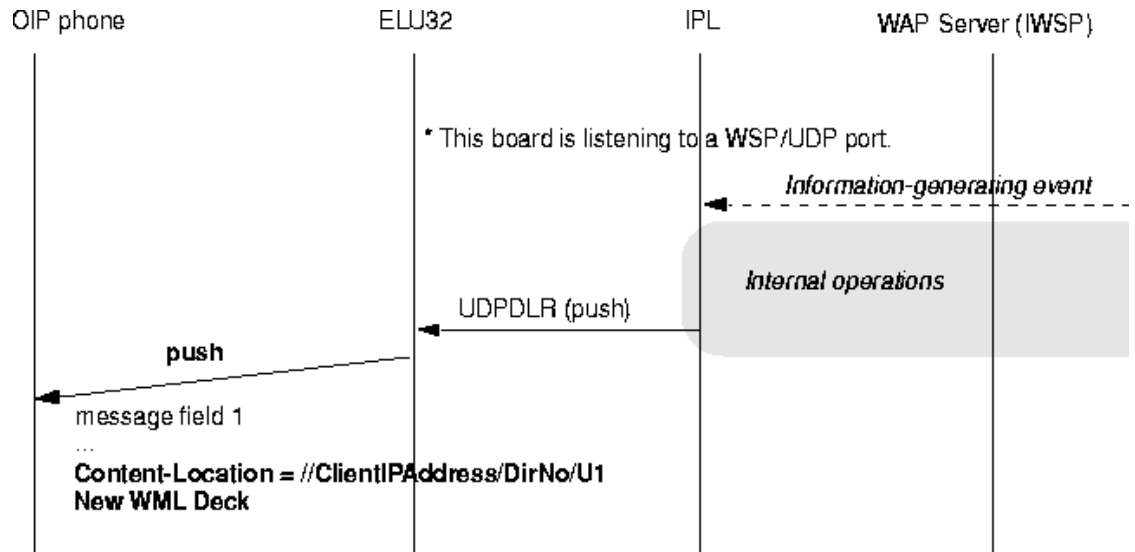


Figure 7: MX-ONE notification to OIP-Phone

## 12.4 Special Situations

The following sub-sections describe situations in which some special actions must be taken.

### 12.4.1 Status Information Request

This service is issued by the OIP-Phone in two situations:

- Everytime the OIP-Phone wishes to synchronize its internal time and date information.
- Everytime the end-user acts on the CLEAR-call control or goes onhook, to guarantee the correctness of the current information in the phone

The process for this service will be the same described in Section 12.2 OIP-Phone service invocation on page 66 with the following particularity: in this case, the *reply* message containing the service result will also carry an extra header - **Date-Header** - and also the current Date and Time information coded according to the WAP standard (a long integer obtained as the number of seconds between 1 Jan 1970, 00:00 GMT and the current Date and Time). The accompanying *push* message will contain the new WML deck, with display information, service status, etc.

## 12.4.2

### Message Waiting and Manual Message Waiting

When a MW or MMW is activated towards the OIP-Phone, the notification will be sent via a *push* message as always. In this case, the WML deck pushed will contain information about the service activation as indicated in 10 visIndType on page 60 . Additionally, a special variable indicating the MW or MMW number will be sent ( 5 Attribute Identifiers for the OIP-Phone logical representation: client side on page 31 ). The visual indicator must be activated and if the end-user acts on the indicator, a basic H.323 call must be initiated towards the MW or MMW number sent.

## 12.4.3

### Last Number Redial / Common Abbreviated Numbers

These two services are affected due to the decision of using the overlap-sending mode to send digits from the OIP-Phone. In both of them, the user dials a certain service code (e.g. \*\*\* for last number redial) which is sent - digit by digit - to the MX-ONE via H.323. Once the MX-ONE has identified the service code as belonging to these particular cases, it will send a WML deck so that the phone is able to display the translated number obtained, i.e. the last external number dialed or the common abbreviated number. In this deck, a special variable containing that number will be also included. If the number sent is a complete number, the MX-ONE is initiating the appropriate actions while the WML deck is being sent, (change register state to calorg, seizing and external line, etc.). If the number is not complete, then the OIP-Phone must remain in register state and allow the end-user to dial more digits. When the next digit is detected in the User Interface of the phone, the internal (GUI) register screen replace the WML card with the translated number, but the input field of the register screen must not be empty, but contain the translated number to which the new digits will be appended. An example is summarized in the following figure:

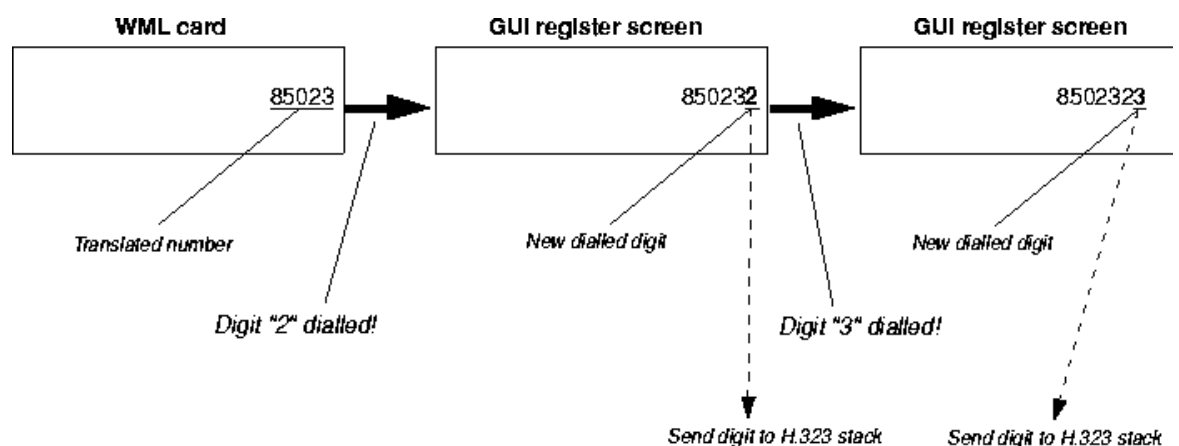


Figure 8: Translated number received from the MX-ONE

This process will continue till a complete number has been sent from the phone and identified in the MX-ONE. Then, the usual actions will be taken (change of the terminal state to Calorg, sending of *push* message with the connected number information and new state information, etc.)

**Please do note that the Translated Number is only valid for the current call.**

## 13 OIP-Phone WML deck format

This section details the format of the WML decks to be used for the OIP-Phone.

### 13.1 General Rules

All the OIP-Phone WML decks will follow these rules.

- 1) The first card in the deck will contain the status information not specifically related to the instrument display: Language of current deck, Ring, Tone, state of Visual Indicators, Calling Number, Calling Name, MW Number, MMW Number and Translated Number. Of course, not all of these variables will exist in all the deck, but the ones who do exist, will always be included in the first card.

The information will be given via `<setvar>` tags, to avoid any unpleasant side-effects in the display:

**`<setvar name="..." value="..."/>`**

Where name is as defined in Attribute Identifiers for the OIP-Phone logical representation: client side and value is defined in action-Type, language, ringType, trafficCase, visIndType or String type, depending on the specific attribute.

This card will be processed by the OIP-Phone, and it could be sent to the browser with no undesired side effects for the end-user, as no display information is defined within the card. The values assigned to the attributes will be used by the OIP-Phone to take actions on the terminal resources, like lighting a LED or setting a tone.

- 2) The second card in the deck will contain the display information, including the soft-key information for the current card. If one of the

soft-keys is pressed, the actions indicated in the associated tag will be followed (issue the adequate *get* message to the MX-ONE).

- 3) *In future implementations there might be additional cards linked to the second one to specify other screens which should be displayed on certain actions with no intervention from the MX-ONE. This situation will occur, for instance, when the Authorization or Account Code soft-keys are implemented .*

In the particular case of the OIP-Phone, with strict requirements regarding the way information is displayed (to simulate as closely as possible the "traditional" DBC203 display), only three text lines per screen are available to show all the relevant information.

The **top line will be controlled locally by the phone itself** to show the internal time and date, and possibly other internal data.

The **top line will be controlled by the MX-ONE via the WML cards** . It will be used to show the name and number information and possibly other variables like service state, active call information, second incoming call information, etc. **This line will be composed by four paragraphs sent with <p>...</p> tags** . That is the four <p>...</p> tags will be shown in the same line following these rules:

- The four paragraphs must be shown in the line in the same order in which they are received within the WML card, overwriting any previous information that might exist in the line.
- The default length of the text in a paragraph is 10 characters. That text might be left-aligned or right-aligned for each paragraph in particular. If left-alignment is chosen, the reference position (starting from the left hand-side) will be: 0, 10, 20, 30 for the 1st, 2nd, 3rd or 4th paragraph respectively. If right-alignment is chosen, the reference position will be: 9, 19, 29, 39 for the 1st, 2nd, 3rd or 4th paragraph respectively.
- If the length of the text within any of those paragraphs is more than ten characters, the 11th, 12th, etc. characters will occupy the paragraph on the left (if right-alignment has been chosen for the current one) or the paragraph on the right (if left-alignment has been chosen). If this should be the case the "occupied" positions on the left paragraph will be overwritten.
- Empty paragraphs won't cause any action on the display, and the next paragraph will be processed according to the rules.

The third line will be a paragraph <p>...</p> holding the soft key menu. **Each one of the menu options will be sent within an <anchor>...</anchor> tag** . If, for a given menu, not all the options are available, empty <anchor> elements will be sent for the not available options, so there will always be 4 <anchor> elements in any given screen.



With these considerations in mind, an example of WML deck when an OIP-Phone with extension number 23250 is idle can be:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" "http://www.wapforum.org/DTD/
wml_1.1.xml">

<wml>
  <card id="a">
    <onevent type = "onenterforward">
      <go href="#b">
        <setvar name="ln" value="0"/>      <!-- English language selected --!>
        <setvar name="vi" value="00000100"/> <!-- MW activated --!>
        <setvar name="mw" value="3000"/>    <!-- MW number --!>
      </go>
    </onevent>
  </card>

  <!------- Idle display ----->
  <card id="b">
    <p align="left">MESSAGE WAITING</p>
    <p></p>
    <p align="left">TOM J. </p>
    <p align="right">23250 </p>
    <p>
      <a href="#b"> </a>      <!-- empty anchor -->
      <a href="#b"> </a>      <!-- empty anchor -->
      <a href="#b"> </a>      <!-- empty anchor -->
      <a href="#b"> </a>      <!-- empty anchor -->
    </p>
  </card>
</wml>
```

*Figure 9: WML deck example for the OIP-Phone in idle state*

Note that no menu soft-keys are available, and thus 4 empty <anchor> elements are sent.

Another example of deck when the phone has made a call and the terminator is busy:

```

<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" "http://www.wapforum.org/DTD/
wml_1.1.xml">

<wml>
  <card id="a">
    <onevent type="onenterforward">
      <go href="#b">
        <setvar name="ln" value="0"/>    <!-- English language selected ASCII '30-->
        <setvar name="vi" value="00000000"/>    <!-- no services activated -->
        <setvar name="mw" value="3000"/>    <!-- MW number -->
      </go>
    </onevent>
  </card>

  <!------- Busy display ----->
  <card id="b">
    <p align="left">EXTENSION BUSY </p>
    <p></p>
    <p align="left">WILL S.</p>
    <p align="right">23200 </p>
    <p>
      <anchor>CAB    <!-- normal anchor -->
      <go href="/23250/U1">
        <postfield name="ac" value="cb"/>
        <postfield name="c1" value="$CRV1"/>
      </go>
    </anchor>
    <anchor>CAW    <!-- normal anchor -->
    <go href="/23250/U1">
      <postfield name="ac" value="cw"/>
      <postfield name="c1" value="$CRV1"/>
    </go>
    </anchor>
    <anchor>CUP    <!-- normal anchor -->
    <go href="/23250/U1">
      <postfield name="ac" value="cp"/>
      <postfield name="c1" value="$CRV1"/>
    </go>
    </anchor>
    <anchor>INTR    <!-- normal anchor -->
    <go href="/23250/U1">
      <postfield name="ac" value="ci"/>
      <postfield name="c1" value="$CRV1"/>
    </go>
    </anchor>
  </p>
</card>
</wml>

```

*Figure 10:WML deck example for the OIP-Phone in busy state*

# 14

## References

- WAP WML v1.1
- WAP WSP (5-Nov-1999)
- WAP WBXML v1.1
- HTTP v1.1 (RFC2068)