



CALYPSO Specifications Product Remote Loading

Draft

In use

Obsolete

Author



Editor



Revision list

Version	Date	Author	Modifications
1.0	090403	Florent CETIER	First Version.
1.1	090424	Florent CETIER	Minor rewritings and typos corrections.

Table of contents

1	INTRODUCTION	6
1.1	Objectives.....	6
1.2	Scope	6
1.3	Target Audience	6
1.4	Reference.....	6
2	CONTEXT	7
2.1	Client	7
2.2	Remote Facility (Server).....	7
3	FUNCTIONALITIES OVERVIEW	8
3.1	Protocol Management.....	8
3.1.1	Open and Close Communication	9
3.1.2	Error management	9
3.1.3	Available Commands	9
3.2	Display and Input Management	10
3.2.1	Available Commands:	10
3.3	Redirection to Another Service	11
3.3.1	Available Commands:	11
3.4	Portable Object Transaction Management.....	12
3.4.1	Security.....	12
3.4.2	Available Commands:	12
4	INTERFACE DESCRIPTION	13
4.1	DATA FORMAT	13
4.2	XML Document Creation (composition)	13
4.3	List of Commands.....	14
4.4	Description of the Commands.....	15
4.4.1	Client and Portable Object Description (id = 200).....	15
4.4.2	ISO 7816 Command Sending (id = 301)	16
4.4.3	Response to an ISO 7816 Command (id = 201).....	17
4.4.4	GUI command (id = 302).....	18
4.4.5	GUI Command Response (id = 202)	20
4.4.6	Redirect Request (id = 305)	21
4.4.7	Redirect Response (id = 205).....	21
4.4.8	Communication Acknowledgement (id = 203 or 303)	22
4.4.9	Transaction Cancel (id = 204)	22
4.4.10	Communication End (id = 900).....	23
4.4.11	Error Message (id = 999)	23
5	ALLOWED RESPONSES TO A COMMAND	25
5.1	Command Sent by Client.....	25
5.2	Command sent by server.....	25
6	SEQUENCE DIAGRAM EXAMPLE	26
6.1	Transaction Example	26
6.2	Error Sequence	27
7	PUBLICATION FUNCTIONALITIES	28
7.1	Page – Template to Display.....	28
7.1.1	Overview.....	28
7.1.2	GUI Command Details	28
7.1.3	Results.....	36
8	APPENDIX	39
8.1	AID Selection Method.....	39
8.2	CNA Reloading Ticket XML Transformation for (U)SIM Card Applet.....	40

8.2.1 Overview40
8.2.2 Protocol Optimization41
8.2.3 Command Interface Definition43
8.2.4 Appendix A: Tag Value51

FOREWORD

The document was developed through Calypso Networks Association workgroup 1 – workpackage 4. The document is the result of a joint undertaking by the members of workgroup 1 – workpackage 4, listed below in alphabetical order:

- Gemalto
- Mercur
- Oberthur
- Octal
- Parkeon
- RATP
- SNCF
- Spirtech
- Transdev
- Veolia

It is an executive summary drawing on the convergence between the different Technical Specifications of the 'Product Remote Loading Protocol' provided by the following stakeholders: ORANGE, TRANSDEV, and RATP.

Pending the different stakeholders' agreement, the document will be published.

1 INTRODUCTION

1.1 Objectives

The document details standard interface specifications between:

- **a client** communicating with a user and with a Calypso application in a portable object and
- **a remote facility** (such as a server) performing security mechanisms

The interface based on XML file exchange enables product-related services and/or functions within the Calypso application.

The specifications are generic, and should be used for any remote secure services without any applicative context restrictions.

1.2 Scope

The interface is independent of network technology (internet, GSM, and so on) and equipment when managing user exchange (screen on dedicated machine, Internet browser, PDA or mobile display, etc.). It is independent of form factor, support, and application domain.

Remark: The specific implementation (optimization) due to the form factor (SIM, for example) is not within the scope of this document, but explanations can be found in the **Appendices**.

The interface is for encapsulating Calypso commands to manage user interface, errors and recovery during communication, and redirect of the Client towards another entity (for payment purposes, for instance).

1.3 Target Audience

The document is intended to help any organization working on the implementation of Product Remote Loading protocols.

1.4 Reference

Title	Version	Reference
Merchant-Manager – M-ticketing Mobile Application Interface ORANGE	1.8	FT/R&D/MAPS/ECB/MET/312.06
Protocole de rechargement à distance TRANSDEV PARKEON	1.0	RBODOY-DOC151007-1A
Rechargement à distance – Spécifications du protocole RATP	1.4	SIT-ISV/SIT0005158/04-39
CALYPSO SPECIFICATION FOR TICKETING – Revision 2 Card Application	2.4	010209-MU-CalypsoCardSpec
CALYPSO SPECIFICATION Portable Object Application - Revision 3	3.0	060708-CalypsoAppli – 25/12/2006

2 CONTEXT

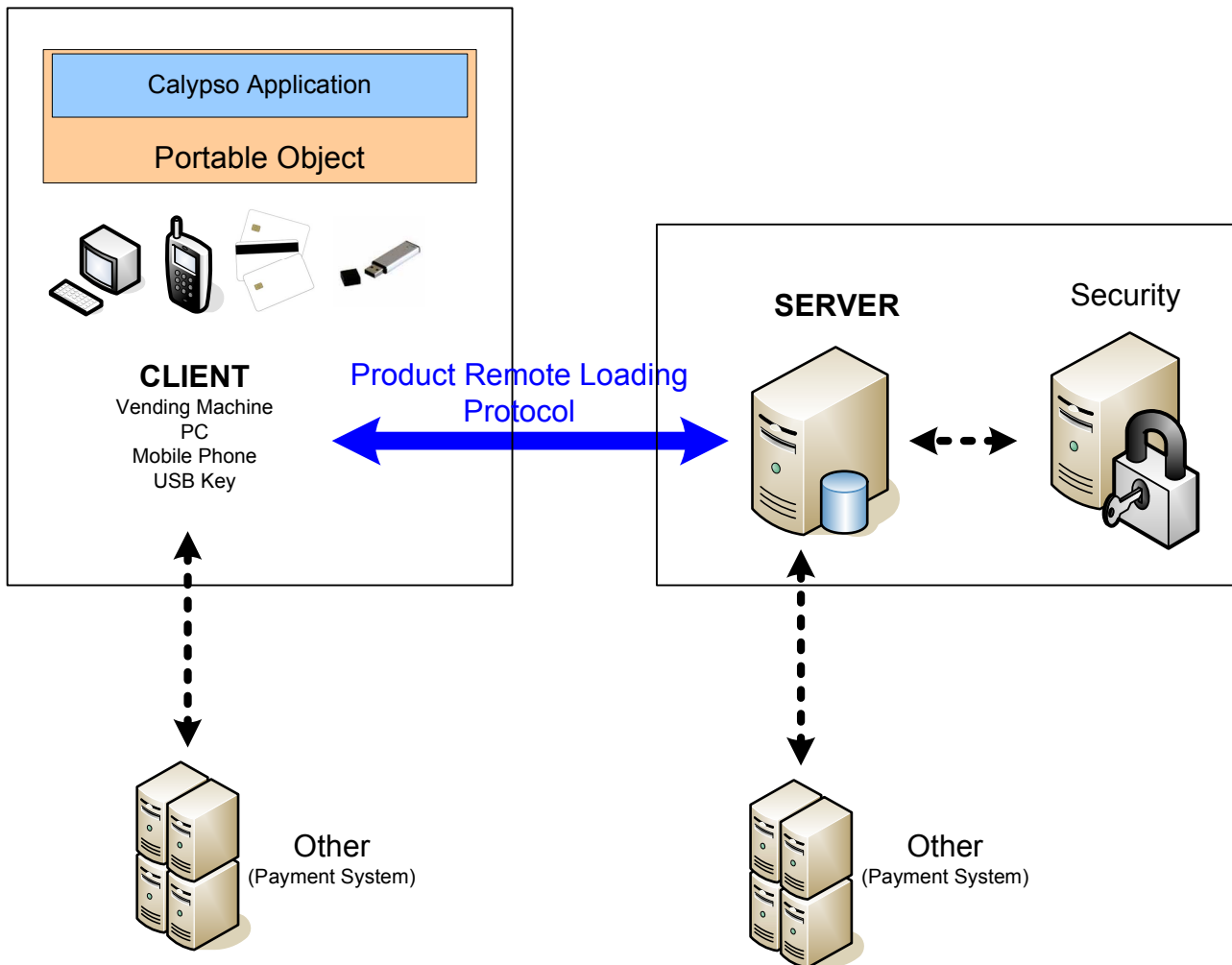


Figure 1 : ecosystem

2.1 Client

The Client is an entity communicating with a Calypso application located in a portable object; it can be a vending machine, a PC connected to the Internet, personalization equipment, a mobile phone, a SIM card application, etc.

The portable object complies with Calypso specifications (Revision 3) or Calypso specifications V2.4, meaning that APDU commands are in accordance with one of the specifications.

2.2 Remote Facility (Server)

The server provides services to a client, it can exchange information that is or is not secured between the 2 entities.

The server includes Calypso security management.

3 FUNCTIONALITIES OVERVIEW

The protocol is for transporting data, information, and APDU commands between the Client and the Server. The document determines how to proceed:

3.1 Protocol Management

Client-Server communication can contain one or more transactions.

The XML document can contain 1 to n commands.

Command processing must be sequential (this mean in XML document order, and next command after completion of the previous one).

The Client will reply to the Server commands (as responses) and sort according to process commands (requests from Server) order.

When there are several commands in the same XML document that require responses, all the responses come in the same XML document.

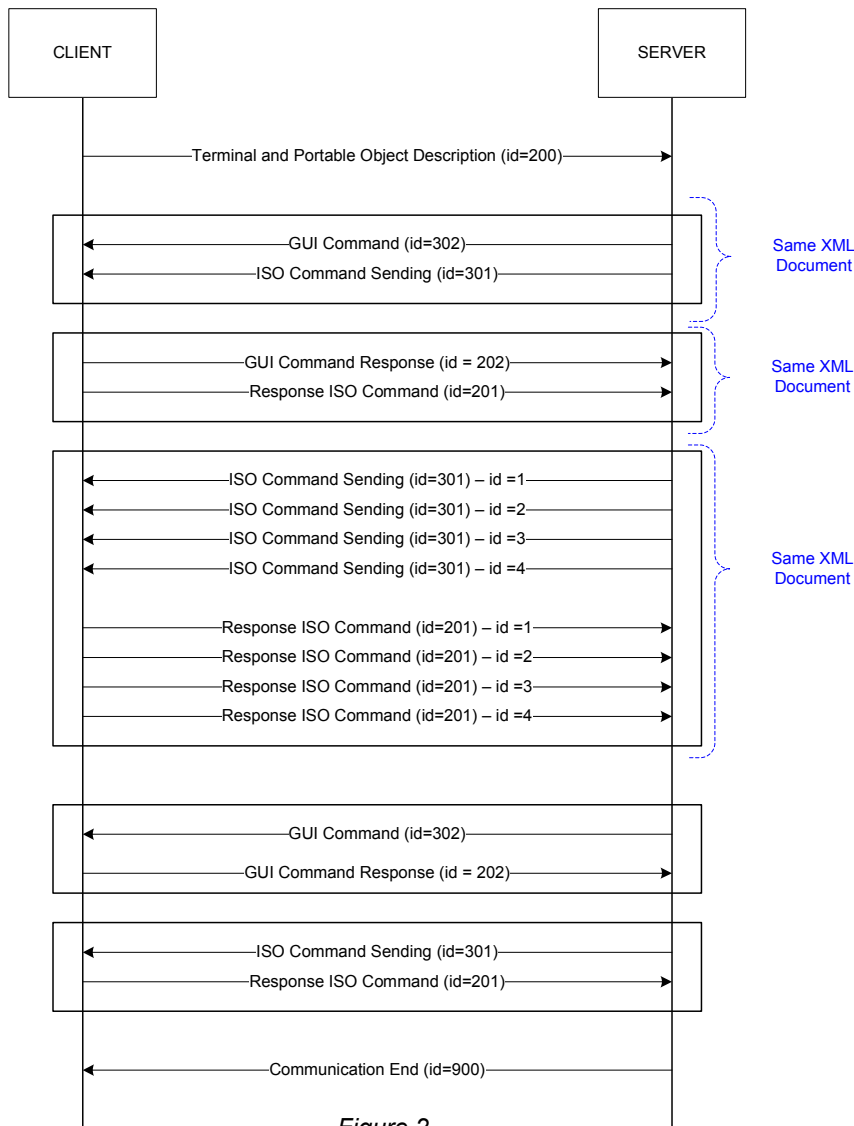


Figure 2

3.1.1 Open and Close Communication

The Client always opens the communication (pull mode), the 'Client and Portable Object Application Description' command is the only one that can start the communication.

The command that closes the communication must be either 'Communication End', or 'Transaction Cancel' or 'Error Message'.

Transaction Cancel

Transaction Cancel is sent to the Server to indicate transaction cancellation (for example when the user stops reloading). It contains a cancellation code that details the reason for cancellation.

Communication End

At the end of the transaction, the Server sends a Communication End message indicating that communication is over.

3.1.2 Error management

Error management checks that data is not invalid due to interruption during the transaction, or that received or sent data is not corrupted.

An 'Error Message' stops the communication if the command flags so indicate.

3.1.3 Available Commands

Client and Portable Object Application Description (Client→Server)

The Client sends this message at the beginning of each transaction in order to give the Server information about its display capabilities, its version, etc.

Communication Acknowledgement (Client ↔ Server)

At the end of a transaction (a reload), the Client sends the Server an acknowledgement containing a transaction code (transaction OK, KO, and so on).

At the end of a transaction (a rehabilitate, invalidate or contract priority modification), the Server sends the Client an acknowledgement containing a transaction code (transaction OK, KO, and so on).

Transaction Cancel (Client → Server)

The command is sent to the Server to indicate a transaction cancellation. It contains a cancellation code to detail the reason for cancellation.

Error Message (Server ↔ Client)

The message sends an error detected by the Server or by the Client.

3.2 Display and Input Management

During processing, some information (text, data read from Calypso application) must be displayed to the customer, and the Server can analyse customer input.

However, the protocol must be able to perform Client interoperability regardless of display capabilities (text only vs. graphics, small screens vs. large screens, etc.), regardless of operative technology (Web based thin UI, heavy application UI, Middlet UI, etc.), and regardless of the Calypso application business usage.

Consequently, GUI management is based on a 'Template' design: a Template is a page that contains all the static information required for the data display/input through the Client display media. In other words, a Template can be viewed as a kind of 'background' page content with 'blanks' that have to be filled by dynamic information: data send by the Server to the Client through the GUI Command message are used to fill these blanks before the display to end-user.

Once the display comes up, the end-user may enter input data. The data will then be routed from the Client to the Server in the GUI Command Response message.

A Client will probably have to host several Template pages to manage the overall reloading sequence. Practically speaking, a set of Template pages has to be defined for each type of Client (with the relevant logos; adapted to screen size, with the correct look and feel, etc.).

Each set of Templates has to be installed at the Client installation time (Vending Machine installation time, PC application installation time, Middlet installation time, cardlet installation time, etc.).

The current document only details how dynamic data is sent between Server and Client.

Language used to define the template, template provisioning process, and template storage solutions are not within the scope of this document, and are Client implementation specific.

3.2.1 Available Commands:

GUI Command (Server → Client)

Command sent by the Server to display text or to ask the user for input information.

GUI Command Response (Client→Server)

Command sent by the Client if asked by the Server to provide the information input by user.

3.3 Redirection to Another Service

During Client-Server exchange, connecting to another service (Data Base consultation, payment service or access to another entity for example) may be necessary.

In the case of redirection to a payment service, the security aspects associated with remote payment constraints are not supported by the protocol.

3.3.1 Available Commands:

Redirect Request (Server → Client)

Sent to the Client so the Client can reach another service.

Redirect Response (Client→Server)

Command sent by the Client to give the Server the information from another service.

3.4 Portable Object Transaction Management

The Client communicates with a Calypso application located in a portable object; a Calypso portable object may have different uses. For a user, typical uses for a transit network would be:

- Personalization: Activating a new Calypso application.
- Reloading: New products (rights, contracts) are loaded onto the Calypso application.
- Reading: the Calypso application content is checked.

CNA recommends that all modifications be performed in a single secure session to ensure the integrity of the data in the portable object.

If a transaction contains more than one secure session (e.g., if multiple products must be loaded), it is the responsibility of the Server to ensure that the card data is coherent at the end of each session (since the portable object may be removed and the transaction aborted at any time).

Example:

During new contract loading in a Transport Application, updating the contract file and the contract list is often required. Both files must be modified during the same session so that if the transaction with the card is aborted (connection cut-off), card data remains coherent. Similarly, each session must update the contract list when the multiple contracts are loaded.

3.4.1 Security

The data exchanged between the Server and the Calypso portable object are secured by the Calypso security mechanisms (session); Security at XML exchange level is not within the scope of this document.

3.4.2 Available Commands:

ISO 7816 Command Sending (Server → Client)

The Server sends commands gathering one or several ISO command(s), which are transmitted to the Client. The commands are used to read file content or to reload an offer on a card, for example.

Response to an ISO 7816 Command (Client → Server)

The Client responds to one or several ISO command(s) previously sent by the Server to the Client.

4 INTERFACE DESCRIPTION

4.1 DATA FORMAT

XML document encoding: UTF-8

[Sequence]: composed data

[Date]: YYYY-MM-DD HR:MIN:SEC (2007-12-25 23:05:35 for 25 December 2007)

[String]: alphanumerical. ASCII.

[Integer]: decimal value (for example an offer composed of 3 offers: NbOffers = '3')

'ISO command' means that commands are ISO 7816-4 compliant, and all Calypso commands are ISO 7816-4 compliant.

<tag></tag> or <tag/> means empty

4.2 XML Document Creation (composition)

The messages (commands and responses) are exchanged between the Client and the Server and encoded in XML files, as described in this document.

Each XML file has the following structure:

Level	Tag	Mnd Opt	Attributes [attribute type]	Parameters [parameter type]	Description
0	<XML>	Mnd	/	[sequence]	Defines the root node of the document
<XML>	<TransactionId>	Mnd	/	[string]	Unique ID for the transaction, number initially chosen by the Client, or any other unique ID.
<XML>	<SelectAnswer>	Opt	/	[string]	Answer to the Select Application, initially automatically sent by the client, if a Select Application was automatically sent. Data out – SW1SW2
<XML>	<AID>	Opt	/	[string]	AID used by the client in the initial automatic Select Application. Be careful that this may be different from the full AID returned in response to the Select Application command.
<XML>	<Command>	Mnd			

The XML document can contain 1 to n commands.

When there are several commands in the same XML document that require responses, all the responses come in the same XML document.

The terminal establishes the communication with a portable object. It then selects the Calypso application by sending the Select Application command with the Calypso application name command to the portable object, and gets back the Calypso serial number along with other information.

An example of AID selection method is illustrated in APPENDIX 8.1

Note - Important reminder: with Calypso Rev 2 application, the serial number obtained through the ATR is the same as the one obtained by the specific 'Select Application' command. In the meantime, for Calypso Rev 3 devices, the serial number obtained through the ATR is different from the one obtained by the specific 'Select Application' command. In addition, with Calypso Rev 3 specification, the list of AID can be obtained by sending a specific command or by 'parsing' the different DF.

4.3 List of Commands

Command Id	Direction	Description
200	Client -> Server	Client and PO description
201	Client -> Server	response to an ISO command
202	Client -> Server	GUI response
203	Client -> Server	Communication acknowledgement
204	Client -> Server	Transaction cancellation
205	Client -> Server	Redirection Answer
301	Server -> Client	ISO command sending
302	Server -> Client	GUI command
303	Server -> Client	Communication acknowledgement
305	Server -> Client	Request for redirection
900	Server -> Client	Communication End
999	Server <-> Client	Error message

The commands are sent to the Client or Server by using the above-described functions (in the request or response). To do so, the XML file associated with the command is transmitted.

4.4 Description of the Commands

4.4.1 Client and Portable Object Description (id = 200)

Level	Tag	Mnd/ Opt	Attributes [attribute type]	Parameters [parameter type]	Description
<XML>	<Command>	Mnd	<i>Id = '200'</i> : command number - [integer]	[Sequence]	Defines the command
<Command>	<language>	Opt	/	[string]	Default language must be English 'eng'
<Command>	<xxx>	Opt/ Mnd	/	[string]/[Sequence]/[integer]	

In this command, the developer can add XML tags according to need.
 <language> tag uses the ISO 639-3 coding: 'eng' for English or 'fra' for French.

Example:

Display parameters for the peripheral device
 Protocol version
 Type of Device

XML Example

```
<Command id="200">
  <Language>eng</Language>
  <DeviceType>1</DeviceType>
  <Display>
    <NbLines>10</NbLines>
    <NbColumns>15</NbColumns>
  </Display>
  <Version mta="1.0" sta="1.0" />
  <Locked>0</Locked>
  <SecurityProperty>XXXXXXXXXX</SecurityProperty>
</Command>
```

4.4.2 ISO 7816 Command Sending (id = 301)

Level	Tag	Mnd Opt	Attributes [attribute type]	Parameters [parameter type]	Description
<XML>	<Command>	Mnd	<i>id</i> = '301': command number [integer]	[Sequence]	Defines the command with at least one APDU
<Command>	<Channel>	Opt	<i>action</i> = 'open'; 'close'		Defines if the channel must be opened or closed.
<Command>	<C-APDU>	Mnd	<i>id</i> : APDU command id [integer]	[String]	Value of the command to launch.

<C-APDU> tag uses the ISO 7816-4 APDU format: CLA-INS-P1-P2-[Lc]-[Data]-[Le]
 The 'APDU command id' is arbitrary and does not necessary begins with '1'.
 The 'APDU command id' is unique in the command, in the XML document.

Note: APDU format description is referenced in the document, [ISO/IEC 7816-3:2006, section 12.1.3](#).

<Channel> tag is used by the Server to retain full management of the cinematic. In a double tap cinematic or other cinematic, the server can send all the necessary APDU between an open and a close command so that users only presents their PO when needed.

The open channel command is not mandatory. Indeed, if there is no double tap in the cinematic of the transaction, the channel will probably already be opened on the client's initiative before it first connects to the server.

The close channel command will always have to be sent (even if there is no double tap in the cinematic of the transaction), at least after the last APDU command.

The channel is automatically open if a command is received and the channel is closed.

The open action is ignored if the channel is already opened.

A close action followed by an open action will reset the portable object.

XML Examples

```
<Command id="301">
  <Channel action="open"></Channel>
  <C-APDU id="21">94DC014C021122</C-APDU>
  <C-APDU id="33">94B2014C1D</C-APDU>
</Command>
```


4.4.4 GUI command (id = 302)

Command used by the Server to display text on the screen, or to ask the user for input.

Level	Tag	Mnd Opt	Attributes [attribute type]	Parameters [parameter type]	Description
<XML>	<Command>	Mnd	<i>id</i> : command number [integer]	[Sequence]	Defines the command for the GUI
<Command>	<Language>	Opt		[String]	Default language must be English 'eng'
<Command> <PageSet>	<Page>	Opt	<i>id</i> : page id [Integer] <i>templateRef</i> : reference of the template to use [String]	[Sequence]	Specifies a page. Id of the page must be unique over the XML file
<Page> <True> <False> <Item>	<Substitute>	Opt	<i>name</i> : name of the parameter [String] <i>type</i> : type of parameter [String]	[String]	Specifies a parameter to be replaced in the template. Type is optional. Allowed types are 'String', 'Integer', and 'Date'
<Page> <True> <False> <Item>	<Condition>	Opt	<i>name</i> : name of the condition [String]	<True> or <False>	Specifies a condition to be evaluated in the template
<Condition>	<True>	Opt	/	[Sequence]	Specifies that the condition is 'true'
<Condition>	<False>	Opt	/	[Sequence]	Specifies that the condition is 'false'
<Page> <True> <False> <Item>	<Loop>	Opt	<i>name</i> : name of the loop [String]	[<Item> Sequence]	Specifies a loop to be evaluated in the template
<Loop>	<Item>	Mnd	/	[Sequence]	Item of the loop
<Command>	<PageSet>	Opt	<i>id</i> : page id [Integer] <i>firstPageId</i> : id of the first page to be displayed [Integer]	[<Page> Sequence]	Specifies a page set. Id of the page set must be unique over the XML file

A GUI Command basically contains a single 'page' to be displayed. As several GUI Commands in the same XML are possible, each page is uniquely identified within the XML file.

A page refers to a page template, and specifies the data to be replaced, the values of the conditional fields, and the list of items of the collections. Data is typed so the Client may adapt the display if required.

In order to optimize Server-Client exchanges, several pages can be sent to the Client at the same time. It is up to Client implementation to display all the pages on a single screen, or to navigate between the pages (according to Client display capabilities, and to template definition).

Consequently, a GUI Command can contain a single 'page set'.

A page set is composed of one to several pages (that are themselves uniquely identified over the XML file). One of the pages of the page set is considered as the one that will be displayed first by the Client.

<language> tag uses the ISO 639-3: 'eng' for English or 'fra' for French.

XML Examples

<Command id="302">

```
<Language>eng<Language/>
<Page id="1" templateRef="SimpleMessageTemplate">
  <Substitute name="message">Reloading in process</Substitute>
</Page/>
</Command>

<Command id="302">
  <Language>eng<Language/>
  <Page id="1" templateRef="OfferDisplayTemplate">
    <Substitute name="nbrOffers">2</Substitute>
    <Loop name="offer list">
      <Item>
        <Substitute name="period" type="String">Month</Substitute>
        <Substitute name="price" type="Integer">23</Substitute>
        <Substitute name="currency" type="String">€</Substitute>
      </Item>
      <Item>
        <Substitute name="period" type="String">Week</Substitute>
        <Substitute name="price" type="Integer">10</Substitute>
        <Substitute name="currency" type="String">€</Substitute>
      </Item>
    </Loop>
    <Condition name="special offer">
      <True>
        <Substitute name="discount" type="String">10% immediate discount</Substitute>
      </True>
    </Condition>
    <Condition name="first class equivalent offer">
      <False>
      </False>
    </Condition>
  </Page/>
</Command>

<Command id="302">
  <PageSet id="12">
    <Page id="1" templateRef="SimpleMessageTemplate">
      <Substitute name="message">Click here to display offers</Substitute>
    </Page/>

    <Page id="2" templateRef="OfferDisplayTemplate">
      [<...>]
    </Page/>
  </PageResponse/>
</Command>
```

4.4.5 GUI Command Response (id = 202)

This message is sent by the Client, if so asked by the Server.

Level	Tag	Mnd Opt	Attributes [attribute type]	Parameters [parameter type]	Description
<XML>	<Command>	Mnd	<i>id: command number</i> [integer]	[Sequence]	Defines the command for the GUI
<Command> <PageSetResponse>	<PageResponse>	Opt	<i>id: id of the page it refers to</i> [Integer]	[<Input> Sequence]	Specifies a page response
<PageResponse>	<Input>	Mnd	<i>name: name of the parameter</i> [String]	[String]	Specifies a data (name and value) that has been entered by the end user
<Command>	<PageSetResponse>	Opt	<i>id: id of the page set it refers to</i> [Integer]	[<PageResponse> Sequence]	Specifies a page set response

A GUI Response answers the previous GUI Command.

A GUI Response contains either a single 'page response' to the previously received 'page' or a single 'page response set' to the previously received 'page set'.

Each page response mentions the id of the page it refers to.

Each page response set mentions the id of the page set it refers to.

As it is mandatory for a GUI Response to answer a GUI Command, a page response may be empty (if no data is to be returned). A page set response may also contain fewer page responses than the number of pages sent.

XML Examples

```
<Command id="202">
  <PageResponse id="1">
    <Input name="from">Paris</Input>
    <Input name="to">Marseille</Input>
  </PageResponse>
</Command>

<Command id="202">
  <PageSetResponse id="12">
    <PageResponse id="1">
      <Input name="from">Paris</Input>
      <Input name="to">Marseille</Input>
    </PageResponse>
    <PageResponse id="2">
      <Input name="period">Month</Input>
    </PageResponse>
  </PageSetResponse>
</Command>
```

4.4.6 Redirect Request (id = 305)

Level	Tag	Mnd/ Opt	Attributes [attribute type]	Parameters [parameter type]	Description
<XML>	<Command>	Mnd	id: command number [integer]	[Sequence]	Defines the command
<Command>	<Request_Data>	Mnd	/	[Sequence]	
<Request_Data>	<xxx>	Mnd	/	/	Content of the URL of redirection, for example

XML Examples

```
<XML>
    <Command id="305">
    <Request_Data href="https://paymentService.com?param1=value1?param2=value2" .</
Request_Data >
    </Command>
</XML>
```

4.4.7 Redirect Response (id = 205)

Level	Tag	Mnd/ Opt	Attributes [attribute type]	Parameters [parameter type]	Description
<XML>	<Command>	Mnd	id: command number [integer]	[Sequence]	Defines the command
<Command>	<Response>	Mnd	/	[String] or [Integer]	Describes the response

XML Examples *Payment form response*

```
<Command id="205">
<Response>OK - 123456</Response>
</Command>
```

4.4.8 Communication Acknowledgement (id = 203 or 303)

Level	Tag	Mnd Opt	Attributes [attribute type]	Parameters [parameter type]	Description
<XML>	<Command>	Mnd	id: command number [integer]	[Sequence]	Defines the command
<Command>	<ACKCode>	Mnd	/	[Integer]	Communication acknowledgement result

XML Example

```
<Command id="203">
<ACKCode>0</ACKCode>
</Command>
```

Values	Definitions
0	Negative acknowledgement
1	Positive acknowledgement

4.4.9 Transaction Cancel (id = 204)

Level	Tag	Mnd Opt	Attributes [attribute type]	Parameters [parameter type]	Description
<XML>	<Command>	Mnd	id: command number [integer]	[Sequence]	Defines the command
<Command>	<CancelCode>	Mnd	/	[Integer]	Reason for transaction cancellation

Remark: the customer will try to send a Transaction cancellation message with CancelCode set to 0 when an internal error occurs.

XML examples

```
<Command id="204">
    <CancelCode>1</CancelCode>
</Command>
```

Values	Definitions
0	Internal cancellation request
1	User cancellation request

4.4.10 Communication End (id = 900)

Level	Tag	Mnd Opt	Attributes [attribute type]	Parameters [parameter type]	Description
<XML>	<Command>	Mnd	<i>id</i> : command number [integer]	[Sequence]	Defines the command
<Command>	<End>	Mnd	/	[Integer]	Communication end result

XML Example

```
<Command id="900">
<End>0</End>
</Command>
```

4.4.11 Error Message (id = 999)

Level	Tag	Mnd Opt	Attributes [attribute type]	Parameters [parameter type]	Description
<XML>	<Command>	Mnd	<i>id</i> : command number [integer]	[Sequence]	Defines the command
<Command>	<ErrorCode>	Mnd	Ack[Integer]	<i>error identifier</i> [Integer]	The optional ack attribute indicates if the Server is waiting for a Communication Acknowledgement from the Client (0: no, 1: yes). Default is 0.
<Command>	<Text>	Opt	/	[String]	To indicate an error message

XML Example

```
<Command id="999">
  <ErrorCode>300</ErrorCode>
  <Text>Service unavailable</Text>
</Command>
```

Here is a listing of the error codes sent by the Server or the Client:

Sent by the server

Values	Definitions
300	Server unavailable, no more SAM available for reloading
301	Incorrect XML document structure: the received document is not the expected one
302	Incorrect XML document structure: tags not understood
303	Incorrect Tag value in XML document
304	Session time out (server received a document for an expired session id)
305	UserSessionId unregistered: the user is not allowed to access the service
306	Internal error (DB, SAM ...)
307	One or more tags are lacking

Sent by the Client

Values	Definitions
200	Incorrect XML document structure: the received document is not the awaited one
201	Incorrect XML document structure: tags not understood
202	Incorrect Tag value in XML document
203	Internal error: APDU execution failed ...
204	Session time-out

5 ALLOWED RESPONSES TO A COMMAND

5.1 Command Sent by Client

Command (entry)	Command(s) (response)
Client and Portable Object Description	GUI command ISO 7816 Command Error Message Communication Ack Communication End
Response to an ISO 7816 Command	ISO 7816 Command GUI command Error Message Communication Ack Communication End
Transaction Ack	None Communication End
Transaction Cancel	None
GUI command Response	ISO 7816 Command GUI command Error Message CommunicationAck Communication End
Redirect response	ISO 7816 Command GUI command Error Message Communication Ack Communication End
Error Message	Communication Ack

5.2 Command sent by server

Command (entry)	Command(s) (response)
ISO 7816 Command Sending	Response to an ISO 7816 Command Error Message Transaction Cancel
GUI command	GUI command Response Error Message Transaction Cancel Communication Ack
Redirect request	Redirect response Error Message Transaction Cancel Communication Ack
Transaction Ack	None
Communication End	None
Error Message	Communication Ack

6 SEQUENCE DIAGRAM EXAMPLE

6.1 Transaction Example

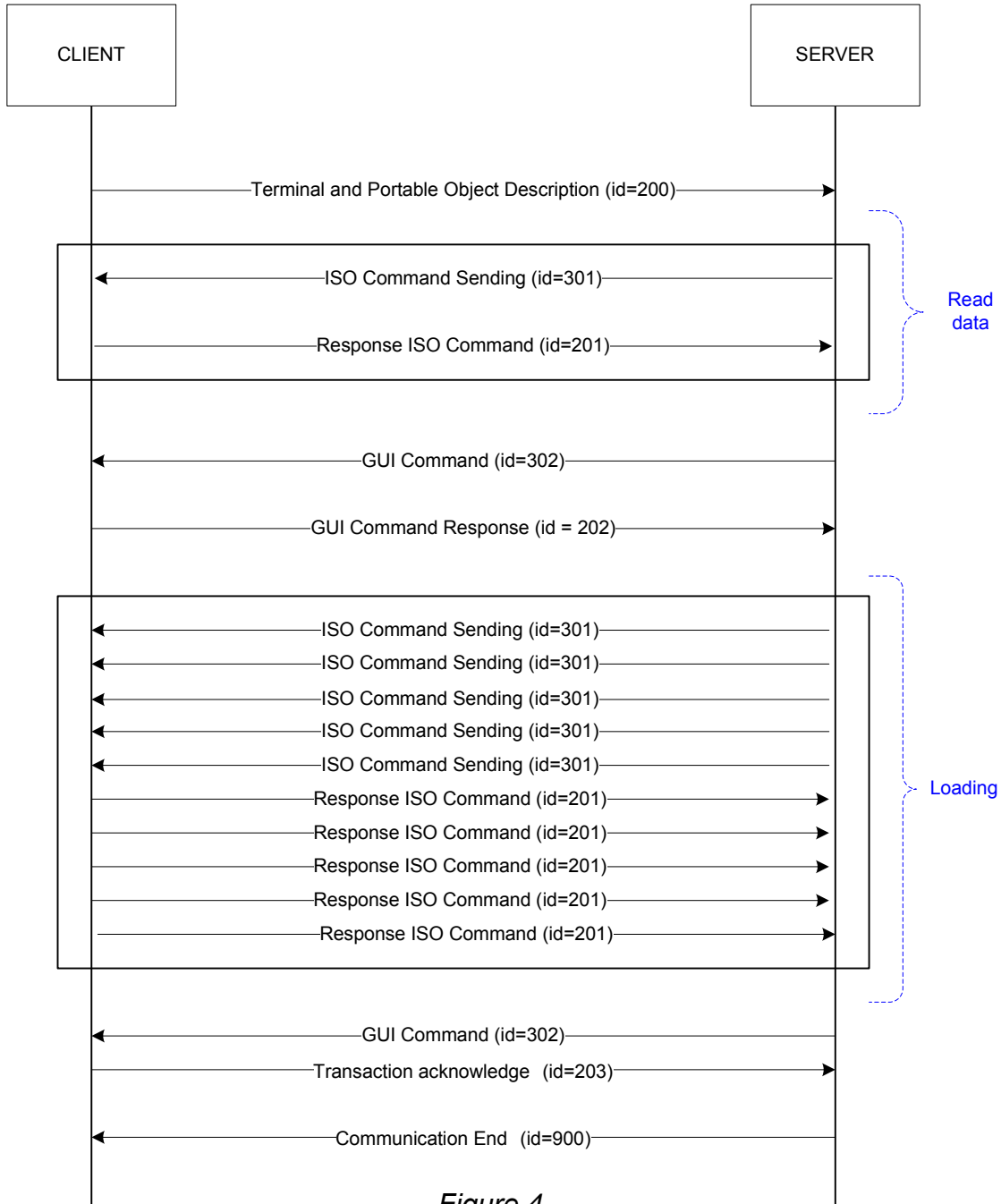


Figure 4

6.2 Error Sequence

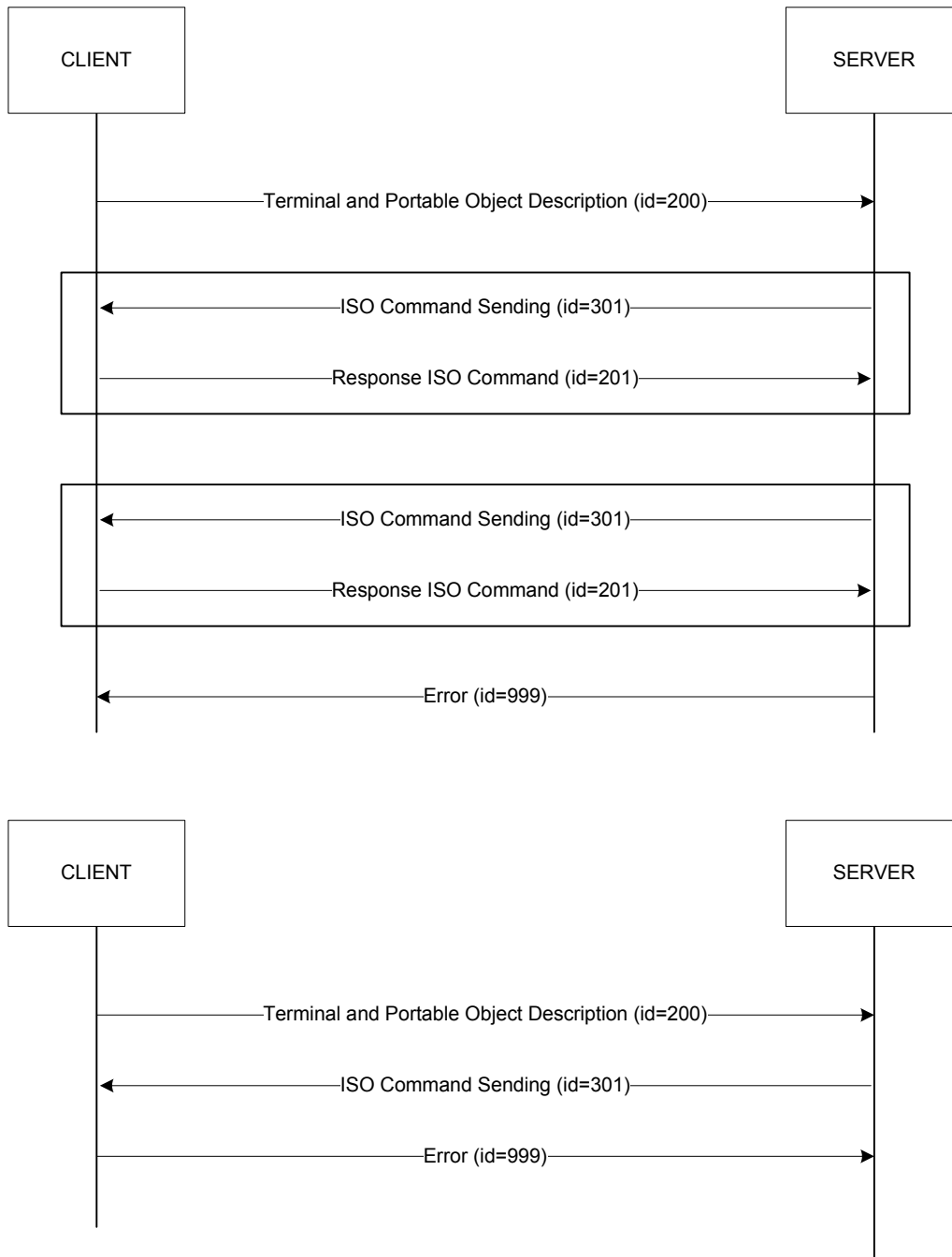


Figure 5

7 PUBLICATION FUNCTIONALITIES

7.1 Page – Template to Display

7.1.1 Overview

A Template is a page that contains all the static information required for the display/input of data through the Client display media. In other words, a Template can be viewed as a kind of 'background' page content with 'blanks' that have to be filled with dynamic information: data sent by the Server to the Client through the GUI Command message is used to fill these blanks, before the display to the end user. Once displayed, the end user may enter input data. The data will then be retransmitted from the Client to the Server in the GUI Command Response message.

A Client will probably need to host several Template pages in order to manage the overall reloading sequence. Concretely, a set of Template pages has to be defined for each type of Client (with the relevant logos; adapted to screen size, with the correct look and feel, etc.). Each set of Templates has to be installed at the Client installation time (Vending Machine installation time, PC application installation time, Middlet installation time, cardlet installation time, etc.).

The advantages of this solution are:

- Using this solution makes the GUI command independent of client type (dynamic content is the same for all client types).
- The template static content will naturally adjust to client capability (screen size, GUI format, etc.), but at the personalization stage and not the reloading stage.
- The solution forestalls problem and makes client development easier.
- The size of the data will be reduced (useful for OTA communications, among others).

Note that the Template is a kind of contract between the Client and the Server as:

- The Client is supplied with templates that correspond to expected Server loading behaviour.
- The Server must know what information must be sent to fill the template.
- The Server must know the list of data that is entered by the end user and transmitted back to the Server.

Note: to have a powerful template mechanism, concepts of conditional behaviour, loop, and page set are also defined.

7.1.2 GUI Command Details

When the Server wants the Client to display a particular page, it will send a GUI Command XML message to specify the page to display.

A 'page' includes a reference to a template plus all the information enabling the Client to build the screen to display.

Consequently, the GUI Command contains a <Page> element. The <Page> element contains the following information:

- The (id) attribute that gives the identifier of the displayed page: This id must be unique in the XML document (whatever the number of GUI Commands) to allow linking GUI Command and GUI Response.

- The (templateRef) attribute that contains the reference of the template that the Client should use

7.1.2.1 Blanks fill-in

The <Page> element contains several <Substitute> elements. The <Substitute> element contains the following information:

- The (name) attribute that gives the name of the blank to fill
- Optionally, the (type) attribute that gives the type of the data that fills the blank. The information can be used by the client to adapt the display to its capability. For example, on a small handset screen a date can be displayed in the '07/09/08' format, while on a PC screen the date will be displayed as 'Monday 9 June 2008'.
- The data that should be used to fill the blank

```
<Command id="309"> <!-- GUI Command -->
  <Page id="xxx" templateRef="xxx">
    <Substitute name="xxx" [type="String|Integer|Date"]>My text1</Substitute>
    <Substitute name="yyy" [type="String|Integer|Date"]>My text2</Substitute>
    ...
  </Page>
</Command>
```

For example, take a simple page displayed during the reloading phase. On the page, there should only be a simple text displayed in the middle of the page: 'Reloading in process'.

Here is an example of what the template (called 'SimpleMessageTemplate') may be for the page:



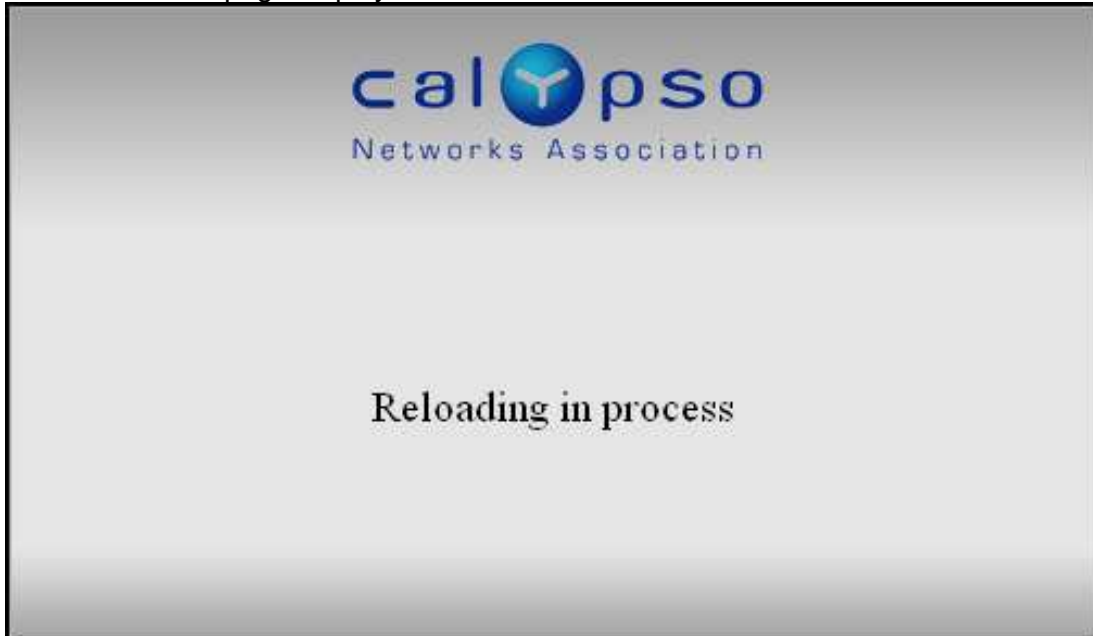
In this template, there is one blank called 'message'.

Here is an example of the GUI Command message sent by the Server to request page display:

```
<Command id="309">
  <Page id="1" templateRef="SimpleMessageTemplate">
    <Substitute name="message" type="String">
      Reloading in process
    </Substitute>
  </Page>
</Command>
```

</Page>
</Command>

Here is the final on-screen page display:



Conditional behaviour

To avoid managing too many templates, the GUI Command can contain information to manage conditional behaviour in the template. For instance, it can be used to hide/show a part of the page that is optional, such as specials offers for example.

In the <Page> element of the GUI Command, <Condition> elements may be added to manage the conditional behaviour. The <Condition> element contains a <True> element or a <False> element. The template has to check the condition value, and display the matching elements.

For example:

```
<Command id="309"><!-- GUI Command -->
  <Page id="xxx" templateRef="xxx">
    <Substitute name="xxx">My text1</Substitute>
    ...
    <Condition name="yyy">
      <True>
        <Substitute name="zzz">
          My text2
        </Substitute>
      </True>
    </Condition>
    ...
  </Page>
</Command>
```

or

```
<Command id="309"><!-- GUI Command -->
  <Substitute name="xxx">My text1</Substitute>
  ...
  <Condition name="yyy">
    <False>
      <Substitute name="bbb">My text3</substitute>
      ...
    </False>
  </Condition>
  ...
</Page>
</Command>
```

In the next example, the Special Offer and First Class Special Offer are two areas that are conditionally displayed depending on the GUI Command content.

The screenshot shows a GUI interface for the Calypso Networks Association. At the top, there is a logo for 'calypso Networks Association'. Below the logo, there is a list item: '• Ticket for 1 [Substitute "period"] : [Substitute "price"] [Substitute "currency"] ○'. Below this, there are two conditional offer boxes. The first box is titled '[IF (specialoffer)]' and contains a red starburst icon followed by 'Special Offer' and '[Substitute "discount"] immediate discount'. The second box is titled '[IF (first class equivalent offer)]' and contains a red starburst icon followed by 'First Class Special Offer' and 'Travel in first class for only [Substitute "first class price"] [Substitute "currency"] more. □'. At the bottom of the interface, there is a 'Send' button.

The following GUI Command can be sent by the Server. In this example, the Special Offer will be displayed ('specialoffer' condition set to true), and the First Class Special Offer will not be displayed ('first class equivalent offer' set to false):

```
<Command id="309">
  <Page id="1" templateRef="OfferDisplayTemplate2">
    <Substitute name="period">Month</Substitute>
    <Substitute name="price">23</Substitute>
    <Substitute name="currency">€</Substitute>
    <Condition name="special offer">
      <True>
        <Substitute name="discount">10% immediate discount</Substitute>
      </True>
    </Condition>
    <Condition name="first class equivalent offer">
      <False>
      </False>
    </Condition>
  </Page>
</Command>
```

The result is:



Loop management

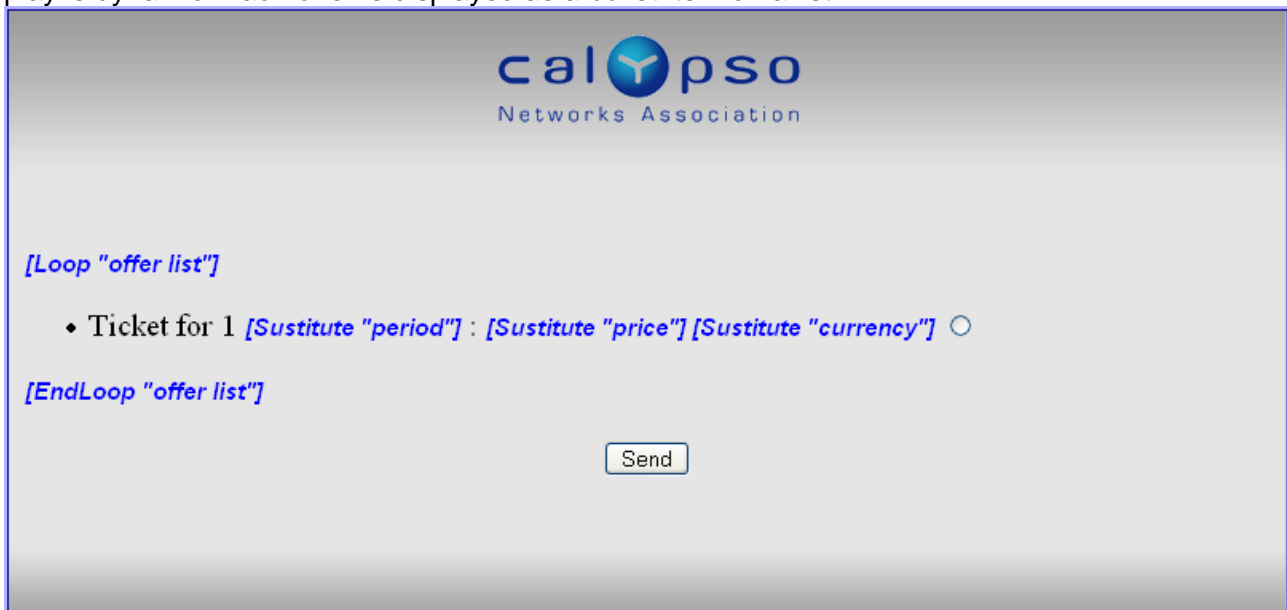
Static template mechanism doesn't match the pages containing elements that can be repeated several times. To manage this case, loop management must be added to the GUI Command.

In the <Page> element of the GUI command, <Loop> elements may be added to manage loops. The <Loop> element contains several <Item> elements.

For example:

```
<Command id="309"><!-- GUI Command -->
  <Page id="xxx" templateRef="xxx">
    <Substitute name="xxx">My text1</Substitute>
    ...
    <Loop name="yyy">
      <Item>
        <Substitute name="ZZZ">My Text 2</Substitute>
        ...
      </Item>
      <Item>
        <Substitute name="ZZZ">My Text 3</Substitute>
        ...
      </Item>
    </Loop>
    ...
  </Page>
</Command>
```

One example of loop usage is the multi-offer display. In the following template, the number of offers to display is dynamic. Each offer is displayed as a bullet item on a list.

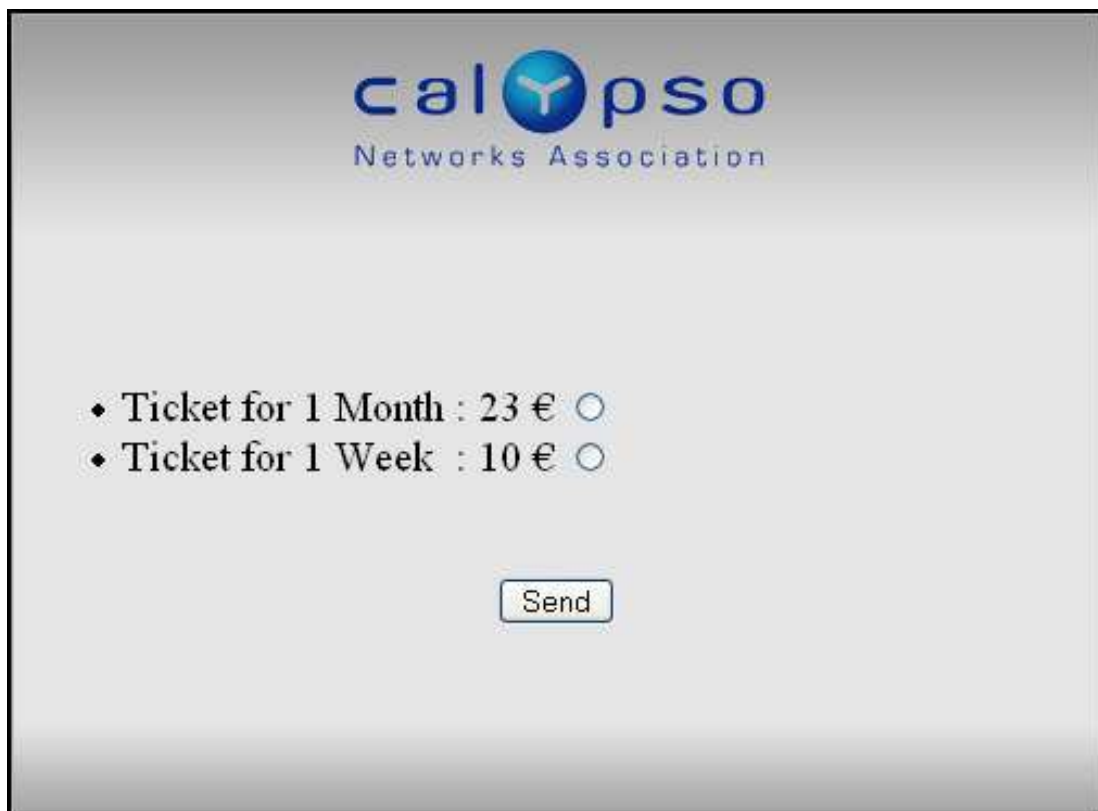


The following GUI Command can be sent by the Server:

```
<Command id="309">
  <Page id="1" templateRef="OfferDisplayTemplate3">
    <Loop name="offer list">
      <Item>
        <Substitute name="period">Month</Substitute>
        <Substitute name="price">23</Substitute>
        <Substitute name="currency">€</Substitute>
      </Item>
    </Loop>
  </Page>
</Command>
```

```
</Item>  
<Item>  
  <Substitute name="period">Week</Substitute>  
  <Substitute name="price">10</Substitute>  
  <Substitute name="currency">€</Substitute>  
</Item>  
</Loop>  
</Page>  
</Command>
```

The result is:



A second example of loop usage is selection list management. In the following template there are two lists managing the trip start and destination (for clarity purposes, the selection list syntax used below is HTML):



```
<select name="liste" >  
  [Loop "offer list"]  
From  <option value="[Substitute "from"]">[Substitute "from"]  
  [EndLoop "offer list"]  
</select>
```

```
<select name="liste" >  
  [Loop "offer list"]  
To    <option value="[Substitute "to"]">[Substitute "to"]  
  [EndLoop "offer list"]  
</select>
```

Send

The following GUI Command can be sent by the Server:

```
<Command id="309"><!-- GUI Command -->  
<Page id="1" templateRef="OfferDisplay4">  
  <Loop name="from">  
    <Item>  
      <Substitute name="from">Paris</Substitute>  
    </Item>  
    <Item>  
      <Substitute name="from">Lyon</Substitute>  
    </Item>  
    <Item>  
      <Substitute name="from">Marseille</Substitute>  
    </Item>  
  </Loop>  
  <Loop name="to">  
    <Item>  
      <Substitute name="to">Paris</Substitute>  
    </Item>  
    <Item>  
      <substitute name="to">Lyon</Substitute>  
    </Item>  
    <Item>  
      <Substitute name="to">Marseille</Substitute>  
    </Item>  
  </Loop>
```

</Page>
</Command>

The result is:



The screenshot shows the Calypso Networks Association logo at the top. Below it, there is a form with two dropdown menus. The 'From' dropdown is set to 'Paris'. The 'To' dropdown is also set to 'Paris', and a list of cities is displayed below it: Paris, Lyon (highlighted in blue), and Marseille. To the right of the dropdowns is a 'Send' button.

7.1.3 Results

For the GUI Command, there is a sort of Contract between the Server and the Client. For a particular Template, the Server knows the information to send to the Client.

It is the same for the GUI Response. For a particular Template, the Server must know the information that it will get back in the GUI Response. If we take the previous template where there are two combo boxes for choosing the origin and the destination of the trip, in the GUI Response, the Server will expect two items of information: 'From' town and 'To' town.

The format of the GUI Response is:

```
<Command id="209"> <!-- Gui Response -->
  <PageResponse id="X">
    <Input name="Field1">Response1</Input>
    <Input name="Field2">Response2</Input>
    ...
  </PageResponse>
</Command>
```

The GUI Response Command contains one <PageResponse> element.

The <PageResponse> element contains the (id) attribute. The identifier must be the same as the one given in the GUI Command for the correct mapping between the GUI Response and the GUI Command.

The <PageResponse> element contains several <Input> elements.

The <Input> element contains the (name) attribute that gives the name of the parameter that has been entered by the end user. Data of the Input element is the information selected by the End User in the GUI.

Going back to the previous example where the user has chosen a trip from Paris to Lyon, the GUI Response will be:

```
<Command id="209"> <!-- Gui Response -->
  <PageResponse id="1">
    <Input name="From">Paris</Input>
    <Input name="To">Lyon</Input>
  </PageResponse>
</Command>
```

Page sets

In the previous example, at least two exchanges between the Server and the Client had to be done. The Client returns information for the selected trip origin and destination, and the Server sends back another GUI Command with detailed information on the selected trip.

The Server may wish to send in a single GUI Command the first page where the origin and destination must be chosen, and the usual detailed trip information pages. Navigation between all these pages is not within the scope of this document/protocol and is fully Client-implementation dependant. However, the Server must indicate the first page that must be displayed.

Sending several pages will optimize exchanges between the Client and Server.

So as to manage several pages in the same GUI command, the <PageSet> element can be put in the GUI command. The <PageSet> element contains the (id) attribute that identifies the Page Set in the XML Document, and the (firstPageId) attribute to specify which page must be displayed first. The <PageSet> elements then contain several <Page> elements.

```
<Command id="309"><!-- GUI Command -->
  <PageSet id="XX" firstPageId="ZZ">
    <Page id="YY" templateRef="template1">
      ...
    </Page>

    <Page id="ZZ" templateRef="template1">
      ...
    </Page>
    ...
  </PageSet>
</Command>
```

Consequently, the GUI Command XML can only contain either a single <Page> element, or a single <PageSet> element (as <PageSet> element is able to contain several <Page> elements).

When the page set is used, several pages can be displayed for the end user. It means that the response can include several Pages Responses. When a <PageSet> is sent in GUI Command, the relevant Command Response must contain a <PageSetResponse> element.

The <PageSetResponse> element contains the (id) attribute, this identifier must be the same as the one given in the GUI Command.

The <PageSetResponse> element contains several <PageResponse> elements.

```
<Command id="209"> <!-- Gui Response -->
```

```
<PageSetResponse id="XX">
  <PageResponse id="YY">
    <Input name="Field1">Response1</Input>
    <Input name="Field2">Response2</Input>
    ...
  </PageResponse>
  ...
</PageSetResponse>
</Command>
```

As a consequence, the GUI Response XML can only contain either one unique <PageResponse> element, or a single <PageSetResponse> element (as <PageSetResponse> element is able to contain several <PageResponse> elements).

8 APPENDIX

8.1 AID Selection Method

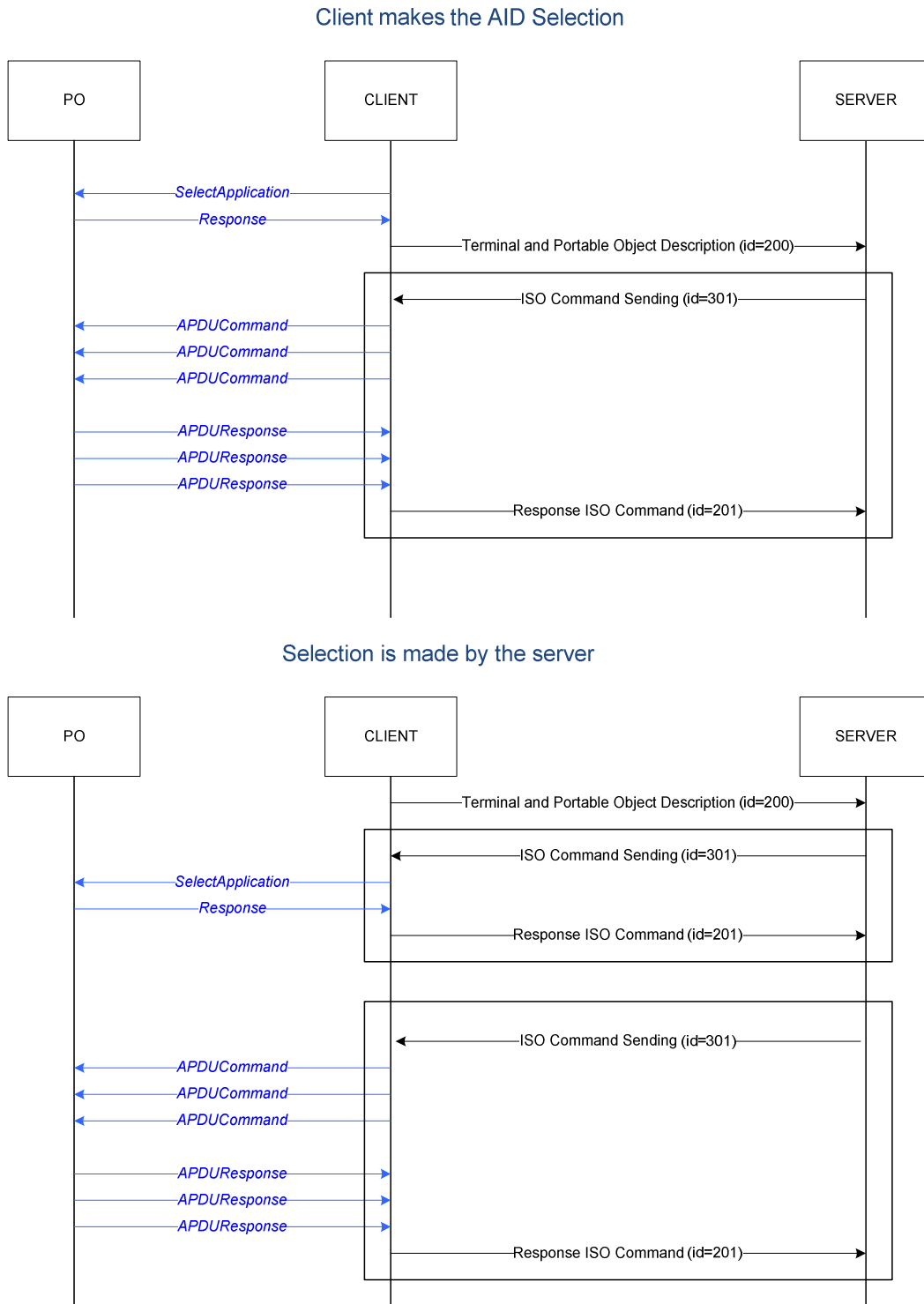


Figure 3

8.2 CNA Reloading Ticket XML Transformation for (U)SIM Card Applet

The normative appendix provides the guidelines for the implementation of the CNA WG1-WP4 'Product Remote loading' in case the SE (Secure Element) targeted Over-The-Air (OTA) has a UICC form factor, and the Calypso application and the Client proxy are hosted on the UICC (i.e., the appendix does not concern the case where the Client proxy is a middlet on the mobile phone).

Besides, a direct communication in XML format with the UICC is not forbidden, if the UICC implements a Client proxy able to parse the XML format directly.

8.2.1 Overview

The XML 'Remote (re)load protocol' presents the following drawbacks in case of an (U)SIM form factor:

- Dialog oriented protocol: OTA communication must deal with reliability problems of the mobile network. Thus, the more messages are sent, the more potential problems may occur. In case of problems, complex and expensive solutions must be set up (for session management, retry mechanisms, validity period, etc.). That's why most of OTA use cases are 'one shot'-oriented.
- XML-based protocol: facing current (U)SIM card processing capabilities, no true implementation of XML parser exists on a (U)SIM. This difficulty to perform XML parsing is especially due to performance problems and to RAM availability problems. Only very simple XML is currently parsable (predictive tag order, no deep structure, etc.). Moreover, OTA performance is also concerned: it is often advisable not to send XML messages directly OTA, but rather to use binary and shorter messages (for example, OMA has defined WBXML for all OTA exchanges, to avoid too wide XML messages).
- Dynamic UI: UI dynamic generation based on XML input is very CPU and RAM consuming.

So as to offset these drawbacks, it is advisable to introduce a new server, called the Proxy Server. The job of the Proxy Server is to translate the XML 'remote (re)loading protocol' into a more appropriate format for the (U)SIM card.

The Proxy Server can be viewed as a part of the Client, as shown on the following diagram:

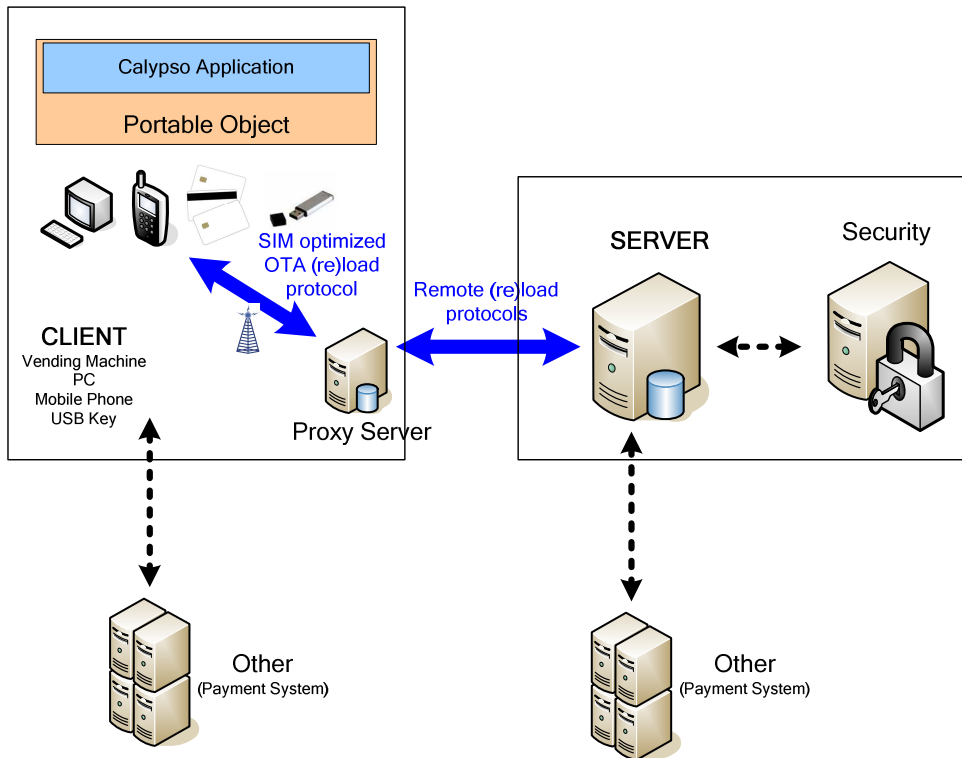


Figure 2 : SIM optimized protocol

The Proxy Server is a server that dialogs with the Server as if it was a standard Client, i.e., using the XML remote (re)-loading protocol defined in CNA WP1-WG4 specifications. On the other hand, it transforms the incoming data so it is optimized for OTA transmission to the SIM.

8.2.2 Protocol Optimization

In the (U)SIM card industry, OTA transmitted data is usually formatted in BER-TLV format. This format makes it possible to lower data footprint (for compliancy with low bandwidth networks) and facilitate the data processing by applets (for compliancy with reduced card processing capabilities).

The current document then defines BER-TLV structures corresponding to CNA WG1 – WP4 XML.

For instance, the 'GUI command' described by the following XML:

```
<Command id="302">
  <Language>eng</language>
  <Page id="1" templateRef="SimpleMessageTemplate">
    <Substitute name="message">Reloading in process</Substitute>
  </Page>
</Command>
```

is translated into:

```
[Tag Command] [Length of the Command Data]
  [Tag Command Id] [Length] [value = 302]
  [Tag Language] [Length] [value = eng]
  [Tag Page] [Length of the Page Data]
```

```
[Tag Id] [Length] [value = 1]
[Tag templateRef] [Length] [value = SimpleMessageTemplate]
[Tag Substitute] [Length of the Substitute Data]
  [Tag name] [Length] [value = message]
  [Tag substituteValue] [Length] [value = Reloading in process]
```

In order to enhance the BER-TLV parsing capability, it may be advisable to prevent the (U)SIM Client from looping on the overall message flow in order to find the correct BER-TLV. The rule is particularly useful for the 'GUI Command' message.

A practical example: if several Substitute tags are on a Page tag of a message, it would be advisable for the Client and Proxy Server to agree on a BER-TLV parsing order. When transforming the XML message into a BER-TLV message, it then becomes the role of the Proxy Server to re-order the Substitute tags (within a particular Page tag) to comply with the agreed order.

On (U)SIM side, optimization then consists in ensuring a linear parsing of the BER-TLV message. The (U)SIM Client will expect the tags to be in a particular order. So, if a tag is not in the right place, it will be considered as missing.

Note that ordering is merely an agreement between the (U)SIM client implementation and the Proxy Server implementation. Such agreement must define the scope of the order (one particular order must be followed within the scope of the Page tag; another particular order must be followed within the scope of the Item tag, etc.).

It may be used:

- To order XML elements between themselves, still using the 'name' or 'id' attribute to differentiate each of them
- To order XML attributes between themselves: request the 'id' to always be the first attribute, then 'name' the second one (etc.)
- To manage multiple occurrences of the same XML tag: as a TLV order is strictly followed by the Proxy Server, just repeat the tag several times (one for each occurrence of the sequence), without specifying a 'sequence number' or a 'item number'.

If no particular ordering rule is defined on the Proxy Server, then the BER-TLV tags order will be the XML order.

For instance, let us consider that the (U)SIM Client expects the following elements to be placed in the following order (within a particular Page): Message, Offer List, Special Offer.

Moreover, the elements of the Page tag must be in the order: id, templateRef.

This means that the following XML file received by the Proxy Server will be reordered:

```
<Command id="302">
  <Language>eng</language/>
  <Page templateRef="SimpleMessageTemplate" id="1">
    <Loop name="offer list">
      <Item>
        ...
      </Item>
    </Loop>

    <Condition name="special offer">
      ...
    </Condition>
  </Page/>

  <Substitute name="message">Reloading in process</Substitute>
```

</Command>

Once reordered, the BER-TLV message will present the Message, Offer List, and Special Offer elements in this particular order, and the id, templateRef attributes in this particular order, as if the received XML file was as follows:

```
<Command id="302">
  <Language>eng</language>
  <Page id="1" templateRef="SimpleMessageTemplate">
    <Substitute name="message">Reloading in process</Substitute>

    <Loop name="offer list">
      <Item>
        ...
      </Item>
    </Loop>

    <Condition name="special offer">
      ...
    </Condition>
  </Page>
</Command>
```

8.2.3 Command Interface Definition

8.2.3.1 Rules for Optional Fields

When optional information is not present in the XML, the corresponding TLV will not be sent.

8.2.3.2 TLV Message

All BER-TLV messages start with at least the TransactionId TLV and contain one to several Command TLVs.

<u>Presence</u>	<u>Length</u>	<u>Description</u>	<u>Value (Hexa)</u>
M	1	Transaction Id Tag	See appendix A
		Transaction Id Length	X
	X	Transaction Id details	
O	1	Select Answer Tag	See appendix A
		Select Answer Length	X
	X	Select Answer details	
O	1	AID Tag	See appendix A
		AID Length	X
	X	AID details	
M	1	<Command> Tag	See each individual Command chapter
	1	<Command> Length	2

	2	<Command detail>	See each individual Command chapter
--	---	------------------	-------------------------------------

The <Command> Tag is described in each individual Command chapter.

8.2.3.2.1 Portable Object Description (id=200) Command Details

The <Command> Tag of the 'Portable Object Description' command is 60h.

<u>Presence</u>	<u>Length</u>	<u>Description</u>	<u>Value (Hexa)</u>
O	1	Language Tag	See appendix A
		Language Length	3
	3	Language	See ISO 639-3

The 'Portable Object Description' command may include other proprietary elements depending on the Client. Each element should have a corresponding BER-TLV (not defined in the current document).

8.2.3.3 ISO Command Sending (id=301) Command Details

The <Command> Tag of the 'ISO Command Sending' command is 71h.

<u>Presence</u>	<u>Length</u>	<u>Description</u>	<u>Value (Hexa)</u>
O	1	Channel Tag	See appendix A
	1	Channel Length	1
	1	Channel	
M*	1	APDU ID Tag	See appendix A
	1	APDU ID Length	2
	2	APDU ID	
	1	C-APDU Tag	See appendix A
		C-APDU Length	X
X		C-APDU	

* Several APDU ID and C-APDU TLVs can be present. Each APDU ID TLV must be immediately followed by a C-APDU TLV.

Channel value is:

'action' parameter value	b7	b6	b5	b4	b3	b2	b1	b0
'open'	0	0	0	0	0	0	-	1
'close'	0	0	0	0	0	0	1	-

8.2.3.4 Response to an ISO 7816 Command (id=201) Command Details

The <Command> Tag of the 'Response to an ISO 7816' command is 61h.

<u>Presence</u>	<u>Length</u>	<u>Description</u>	<u>Value (Hexa)</u>
M*	1	APDU ID Tag	See appendix A
	1	APDU ID Length	2
	2	APDU ID	
	1	R-APDU Tag	See appendix A

		R-APDU Length	X
	X	R-APDU	APDU data response follow by 2 bytes for APDU status
* Several APDU ID and R-APDU TLVs can be present. Each APDU ID TLV must be immediately followed by a R-APDU TLV.			

8.2.3.5 GUI Command (id=302) Command Details

The <Command> Tag of the 'GUI command' command is 72h.

<u>Presence</u>	<u>Length</u>	<u>Description</u>	<u>Value (Hexa)</u>
O	1	Language Tag	See appendix A
		Language Length	3
	3	Language	See ISO 639-3
O	1	Page Tag	See appendix A
		Page Length	X
	X	Page	See Page details chapter
O	1	PageSet Tag	See appendix A
		PageSet Length	X
	X	PageSet	See PageSet details chapter

The GUI Command can only contain only one Page or only one PageSet.

8.2.3.5.1 Page details

<u>Presence</u>	<u>Length</u>	<u>Description</u>	<u>Value (Hexa)</u>
M	1	Page Id Tag	See appendix A
		Page Id Length	2
	2	Page Id	
M	1	Template Ref Tag	See appendix A
		Template Ref Length	X
	X	Template Ref	
O*	1	Substitute Tag	See appendix A
		Substitute Length	X
	X	Substitute	See Substitute details chapter
O*	1	Condition Tag	See appendix A
		Condition Length	X
	X	Condition	See Condition details chapter
O*	1	Loop Tag	See appendix A
		Loop Length	X
	X	Loop	See Loop details chapter
* Several Substitute, or Condition, or Loop TLVs can be present.			

- Substitute details

<u>Presence</u>	<u>Length</u>	<u>Description</u>	<u>Value (Hexa)</u>
M	1	Name Tag	See appendix A
		Name Length	X
	X	Name	
	1	Type Tag	See appendix A
		Type Length	1
	1	Type	Allowed types are: 01 for 'String' 02 for 'Integer' 03 for 'Date' Others are proprietary
	1	Substitute Value Tag	See appendix A
		Substitute Value Length	X
	X	Substitute Value	

- Condition details

<u>Presence</u>	<u>Length</u>	<u>Description</u>	<u>Value (Hexa)</u>
M	1	Name Tag	See appendix A
		Name Length	X
	X	Name	
	1	Condition Value Tag	See appendix A
		Condition Value Length	1
	1	Condition Value	Allowed values are: 00 for false 01 for true

- Loop details

<u>Presence</u>	<u>Length</u>	<u>Description</u>	<u>Value (Hexa)</u>
M	1	Name Tag	See appendix A
		Name Length	X
	X	Name	
M*	1	Item Tag	See appendix A
		Item Length	X
	X	Item	See Item details chapter

* Several Item TLVs can be present.

- Item details

<u>Presence</u>	<u>Length</u>	<u>Description</u>	<u>Value (Hexa)</u>
O*	1	Substitute Tag	See appendix A
		Substitute Length	X
	X	Substitute	See Substitute details chapter
O*	1	Condition Tag	See appendix A
		Condition Length	X
	X	Condition	See Condition details chapter
O*	1	Loop Tag	See appendix A
		Loop Length	X
	X	Loop	See Loop details chapter
* Several Substitute, or Condition, or Loop TLVs can be present.			

8.2.3.5.2 PageSet Details

<u>Presence</u>	<u>Length</u>	<u>Description</u>	<u>Value (Hexa)</u>
M	1	PageSet Id Tag	See appendix A
		PageSet Id Length	2
	2	PageSet Id	
M	1	First Page Id Tag	See appendix A
		First Page Id Length	2
	2	First Page Id	
M*	1	Page Tag	See appendix A
		Page Length	X
	X	Page	See Page details chapter
* Several Page TLVs can be present.			

8.2.3.6 GUI Response (id=202) Command Details

The <Command> Tag of the 'GUI response' command is 62.

<u>Presence</u>	<u>Length</u>	<u>Description</u>	<u>Value (Hexa)</u>
O	1	Page Response Tag	See appendix A
		Page Response Length	X
	X	Page Response	See Page Response details chapter
O	1	PageSet Response Tag	See appendix A
		PageSet Response Length	X
	X	PageSet Response	See PageSet Response details chapter

The GUI response can only contain only one Page Response or only one PageSet Response.

8.2.3.6.1 Page Response Details

<u>Presence</u>	<u>Length</u>	<u>Description</u>	<u>Value (Hexa)</u>
M	1	Page Id Tag	See appendix A
		Page Id Length	2
	2	Page Id	
O*	1	Input Tag	See appendix A
		Input Length	X
	X	Input	See Input details chapter
* Several Input TLVs can be present.			

- **Input details**

<u>Presence</u>	<u>Length</u>	<u>Description</u>	<u>Value (Hexa)</u>
M*	1	Name Tag	See appendix A
		Name Length	X
	X	Name	
	1	Input Value Tag	See appendix A
		Input Value Length	X
	X	Input Value	
* Several Name and Input TLVs can be present.			

8.2.3.6.2 PageSet Response details

<u>Presence</u>	<u>Length</u>	<u>Description</u>	<u>Value (Hexa)</u>
M	1	PageSet Response Id Tag	See appendix A
		PageSet Response Id Length	2
	2	PageSet Response Id	
M*	1	Page Response Tag	See appendix A
		Page Response Length	X
	X	Page Response	See Page Response details chapter
* Several Page TLVs can be present.			

8.2.3.7 Redirect Request (id=305) Command Details

The <Command> Tag of the 'redirect request' command is 75.

<u>Presence</u>	<u>Length</u>	<u>Description</u>	<u>Value (Hexa)</u>
M	1	Request Data Tag	See appendix A
		Request Data Length	X
	X	Request Data	

8.2.3.8 Redirect Response (id=205) Command Details

The <Command> Tag of the 'Redirect response' command is 65.

<u>Presence</u>	<u>Length</u>	<u>Description</u>	<u>Value (Hexa)</u>
M	1	Response Data Tag	See appendix A
		Response Data Length	X
	X	Response Data	

8.2.3.9 Communication Acknowledgement (id=203 or 303) Command Details

The <Command> Tag of the 'Communication Acknowledgement' command is 63.

<u>Presence</u>	<u>Length</u>	<u>Description</u>	<u>Value (Hexa)</u>
M	1	Ack Code Tag	See appendix A
		Ack Code Length	1
	1	Ack Code	Allowed values are: 00 for KO 01 for OK

8.2.3.10 Communication Cancel (id=204) Command Details

The <Command> Tag of the 'Communication cancel' command is 64.

<u>Presence</u>	<u>Length</u>	<u>Description</u>	<u>Value (Hexa)</u>
M	1	Cancel Code Tag	See appendix A
		Cancel Code Length	1
	1	Cancel Code	Allowed values are: 00 for internal cancel 01 for user cancel

8.2.3.11 Communication End (id=900) Command Details

The <Command> Tag of the 'Communication end' command is 68.

<u>Presence</u>	<u>Length</u>	<u>Description</u>	<u>Value (Hexa)</u>
M	1	End Result Tag	See appendix A
		End Result Length	2
	2	End Result	

8.2.3.12 Error Message (id=999) Details

The <Command> Tag of the 'Error message' command is 69.

<u>Presence</u>	<u>Length</u>	<u>Description</u>	<u>Value (Hexa)</u>
-----------------	---------------	--------------------	---------------------

M	1	Error Code Tag	See appendix A
		Error Code Length	2
	2	Error Code	
O	1	Communication Ack Tag	See appendix A
		Communication Ack Length	1
	1	Communication Ack	Allowed values are: 00 for no 01 for yes
O	1	Error Text Tag	See appendix A
		Error Text Length	X
	X	Error Text	

8.2.4 Appendix A: Tag Value

Commands tags	
Portable Object Description	60
ISO Command Sending	71
Response to an ISO 7816 Command	61
GUI command	72
GUI response	62
Redirect request	75
Redirect response	65
Communication acknowledge	63
Communication cancel	64
Communication end	68
Error message	69
Tags in command details	
Language	41
Channel	42
APDU ID	43
C-APDU	22
R-APDU	23
Page	77
PageSet	78
Page Id	44
Template Ref	45
Substitute	79
Condition	7A
Loop	7B
Name	46
Type	47
Substitute Value	48
Condition Value	49
Item	7C
PageSet Id	4A
First Page Id	4B
Page Response	7D
PageSet Response	7E
Input	7F
Input Value	4C
PageSet Response Id	4D
Request Data	4E
Response Data	4F
Ack Code	50
Cancel Code	51
End Result	52
Error Code	53
Communication Ack	54
Error Text	55