UBERPUTER ATM PROTOCOL SPECIFICATION DRAFT REV.1

Mikael Forsberg <miforsb@kth.se>
Robin Gunning <rgunning@kth.se>

Programmeringsparadigm DD1361
Kungliga Tekniska Högskolan HT15

December 4, 1985

N.B: Radiometric dating indicates that this document is actually from 2015.

TABLE OF CONTENTS

1 INTRODUCTION

2 PROTOCOL

2.1 OVERVIEW

The protocol consists of three major components: a language dataset, a set of message packets, and rules for the flow and ordering of messages. The language dataset contains all strings needed for a client user interface. The set of message packets contains all messages required to be supported by servers and clients. Finally, compliant servers and clients are required to follow all rules for flow and message ordering.

The language dataset is not constant, and clients are not supposed to define or implement a copy of it themselves. Instead, clients should retrieve the dataset from the server, and should also periodically check for updates to the dataset. Both of these actions are performed by sending a message of type UPDATE REQUEST (section 2.7.1 on page 3). Clients should, however, keep a cached copy of the dataset in order to avoid superfluous network traffic.

The message packets and flow rules are centered on the basic principle of clients receiving one or more MENU ITEMs containing instructions for sending back ACTIONs. These exchanges cover nearly all of the functionality of a session. The only predefined messages a client should implement are UPDATE REQUEST, MENU REQUEST and ACTION. All other expected functionality such as authentication will be determined at runtime by MENU ITEMs sent from a server.

An ACTION can contain instructions for a client to send and / or receive a single 32-bit integer value, and can also contain an instruction for a client to receive an immediate follow-up action that a client should execute directly. This principle of follow-up actions is used to chain actions together, facilitating exchanges of several values in sequence.

2.2 LANGUAGE DATASET: FORMAT

The language dataset is supplied in a YAML-compatible format defined by the following grammar, with starting nonterminal <Data>:

```
<Data> ::= <Pair> | <List> | <Pair> CR <Data> | <List> CR <Data>
<Pair> ::= <Value> : <Value>
<Value> ::= [A-Za-z0-9_]+ | ' [A-Za-z0-9_]+ ' | " [A-Za-z0-9_]+ "
<List> ::= <Value> : CR <ListValues>
<ListValues> ::= <Indent> <Pair> | <Indent> <Pair> CR <ListValues>
<Indent> ::= SPACE SPACE SPACE
```

2.3 LANGUAGE DATASET: EXAMPLE

The following is an example language dataset.

```
_Version: 2015120408
_Useful:
   change_language : 16
   press_any_key_to_continue : 17
English:
   0: ""
   1: "Welcome to UBERBANK"
   2: Balance
   3: Deposit
   4: Withdraw
   16: Change language
   17: Press any key to continue
Svenska:
   0: ""
   1: "Välkommen till UBERBANK"
   2: Saldo
   3: Insättning
   4: Uttag
   16: Byt språk
   17: Tryck valfri tangent för att fortsätta
```

2.4 LANGUAGE DATASET: TRANSMISSION, UPDATING

The language dataset is transmitted from server to client using the message packet UPDATE (section 2.8.1 on page 4). This packet is sent in response to a client request using the UPDATE REQUEST (section 2.7.1 on page 3). As part of the UPDATE REQUEST, a client should include the value of the special field "_Version" contained in the language dataset (see example above), which is used by a server to decide whether or not to send an update. If there is no update available a server will not send the UPDATE message, but will instead simply reply with an OK (section 2.8.5 on page 6).

2.5 SYMBOLIC CONSTANTS

SYMBOL	VALUE
MSG_MENU_REQUEST	0
MSG_MENU_NUM_ITEMS	1
MSG_ACTION	2
MSG_UPDATE_REQUEST	3
MSG_MENU_ITEM	4
MSG_OK	5
MSG_FAIL	6
MSG_UPDATE	7
MSG_RESPONSE	8

SYMBOL	VALUE
TYPE_RECV_FOLLOWUP	32
TYPE_RECV_UINT32	64
TYPE_SND_UINT32	128

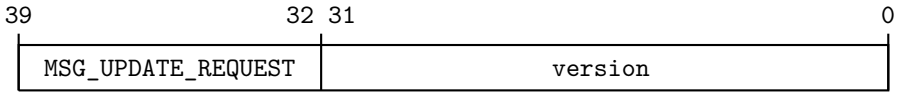
2.6 DATA FORMATS

With the exception of the data field in the server-to-client UPDATE message packet, all fields of all message packets contain unsigned integers in network byte order (big-endian) of size specified by the width of the field.

2.7 MESSAGE PACKETS: CLIENT TO SERVER

2.7.1 UPDATE REQUEST

To request an update from a server, a client should send the following 40-bit (5-byte) packet:



The "version" field should contain the unsigned integer value contained in the special field "_Version" of the language dataset (see section 2.3 on page 2). If a client does not have any such value to send (e.g. before retrieving the language dataset for the first time), the field should be set to all zeroes (32-bit integer zero).

A server will reply in one of two ways. If there is an update available, the reply will be an UPDATE (section 2.8.1 on page 4). Otherwise, the reply will simply be an OK (section 2.8.5 on page 6).

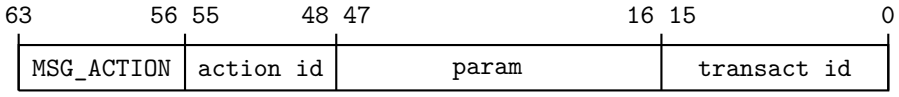
2.7.2 MENU REQUEST

To request a list of available menu options, a client should send the following 8-bit (1-byte) packet:

A server will reply with a single instance of MENU NUM ITEMS (section 2.8.2 on page 5) containing the number N of menu items to follow, followed by N instances of MENU ITEM (section 2.8.3 on page 5).

2.7.3 ACTION

To perform an action, a client should send the following 64-bit (8-byte) packet:



The "action id" field should contain an action id received in a MENU ITEM (section 2.8.3 on page 5).

The "param" field: if the "type" field of the MENU ITEM that gave rise to the action included the TYPE_SND_UINT32 bit flag, the "param" field should contain a 32-bit unsigned integer. Otherwise, the value of the param field should be set to zero.

The "transact id" field can be set to any value, however it is suggested to use a counter value that is incremented for each ACTION. The value is suitable for use in session logs, for both servers and clients.

A server will reply in one of three ways.

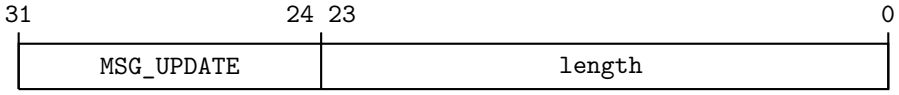
If the "type" field of the MENU ITEM that gave rise to the action included the TYPE_RECV_UINT32 bit flag, the server will reply with a RESPONSE (section 2.8.4 on page 6). Otherwise, the server will reply with either OK (section 2.8.5 on page 6) or FAIL (section 2.8.6 on page 6) depending on the outcome of the action.

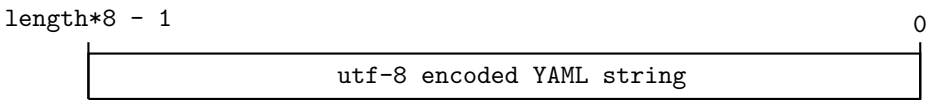
Finally, in addition to receiving one of RESPONSE, OK or FAIL as described above, if the "type" field of the MENU ITEM that gave rise to the action included the TYPE_RECV_FOLLOWUP bit flag, the server will finish the exchange by sending a single MENU ITEM containing instructions for a new action that should be executed immediately by the client.

2.8 MESSAGE PACKETS: SERVER TO CLIENT

2.8.1 UPDATE

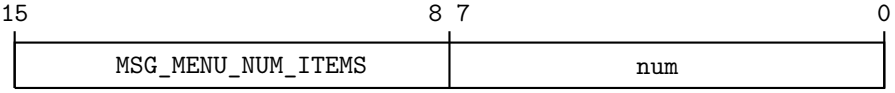
When there is an available update, a server will reply to an UPDATE REQUEST (section 2.7.1 on page 3) with the following variable-length packet, consisting of a 32-bit (4-byte) header followed by LENGTH bytes of YAML data encoded in UTF-8:





2.8.2 MENU NUM ITEMS

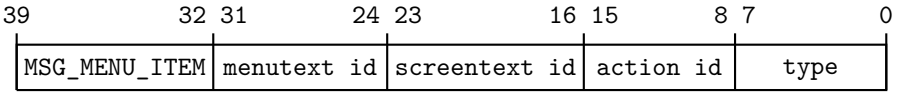
In response to a MENU REQUEST (section 2.7.2 on page 3), a server should send a single instance of the following 16-bit (2-byte) packet:



The "num" field should contain the number of MENU ITEMs (section 2.8.3 on page 5) that will follow. After sending this packet a server should immediately send "num" number of MENU ITEMs.

2.8.3 MENU ITEM

To define an action that a client is allowed to perform at the current time (immediately after receiving the message), a server should send the following 40-bit (5-byte) packet:



The "menutext id" field should contain the numeric key of the string contained in the language dataset that a client should display in the user interface where the user is supposed to select the action (before the action has been selected).

The "screentext id" field should contain the numeric key of the string contained in the language dataset that a client should display to the user when the user has selected to perform the action (after the action has been selected).

The "action id" field should contain an ID that the server wishes to receive as part of the ACTION (section 2.7.3 on page 4) sent by a client executing the action.

The "type" field is used for bitwise flags as specified by the symbolic constants prefixed with TYPE_ (section 2.5 on page 3).

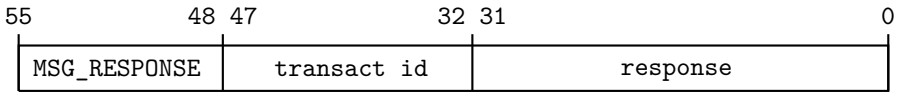
Including the flag of TYPE_SND_UINT32 instructs clients to read input from the user and to include the value of said input in the "param" field of the ACTION.

Including the flag of TYPE_RECV_UINT32 instructs clients to receive a RESPONSE (section 2.8.4 on page 6) from the server directly after having sent an ACTION. If this flag is not included, clients are instead instructed to receive an OK (section 2.8.5 on page 6)or a FAIL (section 2.8.6 on page 6) directly after sending the ACTION. The server must send the appropriate response (RESPONSE, OK or FAIL) directly after having received the ACTION.

Including the flag of TYPE_RECV_FOLLOWUP instructs clients to receive a MENU ITEM (section 2.8.3 on page 5) after having sent the ACTION and having read the response (RESPONSE, OK or FAIL) described in the previous paragraph. This MENU ITEM instructs a client to perform another ACTION immediately following the previous one. Any number of these exchanges can be chained together. A server that sends a MENU ITEM including this flag must send the follow-up MENU ITEM directly after having sent the initial response to the ACTION (RESPONSE, OK or FAIL) as described in the previous paragraph.

2.8.4 RESPONSE

To respond to an ACTION (section 2.7.3 on page 4) with type TYPE_RECV_UINT32, a server should send the following 56-bit (7-byte) packet:

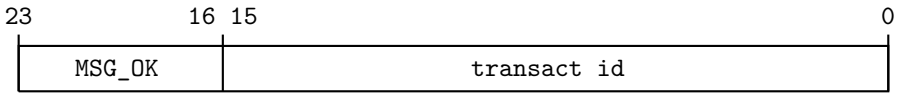


The "transact id" field should contain the exact value received in the "transact id" field of the ACTION that is being responded to.

The "response" field should contain a numeric response to the ACTION, suitable for a client to display for a user.

2.8.5 OK

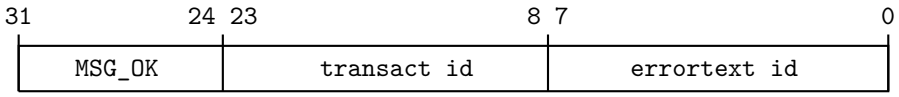
To respond to a successfully performed ACTION (section 2.7.3 on page 4) that was not flagged with type TYPE_RECV_UINT32, a server should send the following 24-bit (3-byte) packet:



The "transact id" field should contain the exact value received in the "transact id" field of the ACTION that is being responded to.

2.8.6 FAIL

To respond to a failed ACTION (section 2.7.3 on page 4) that was not flagged with type TYPE_RECV_UINT32, a server should send the following 32-bit (4-byte) packet:



The "transact id" field should contain the exact value received in the "transact id" field of the ACTION that is being responded to.

The "errortext id" field should contain the numeric key of a string contained in the language dataset that explains the nature of the failure suitable for a client to display to a user.

3 EXAMPLE SESSIONS

The following example session is illustrating a session where everything is going great.

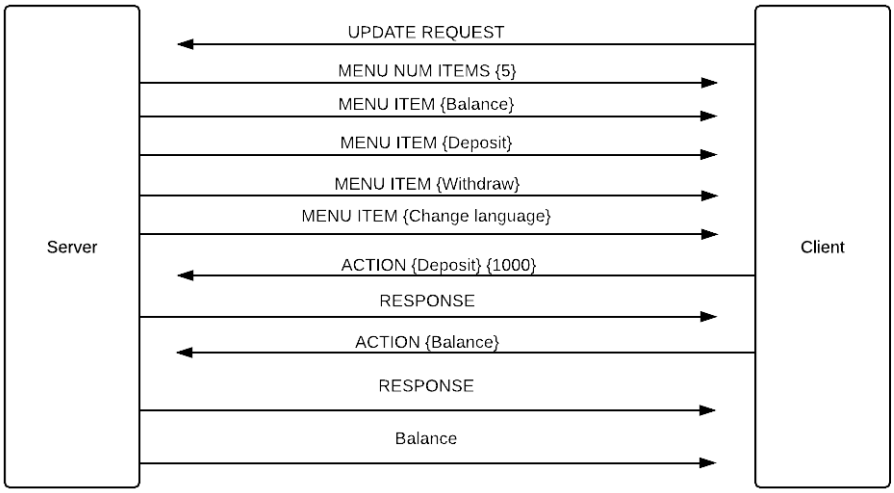


Figure 1: Example session 1

This example session is illustrating a session where an error occurs.

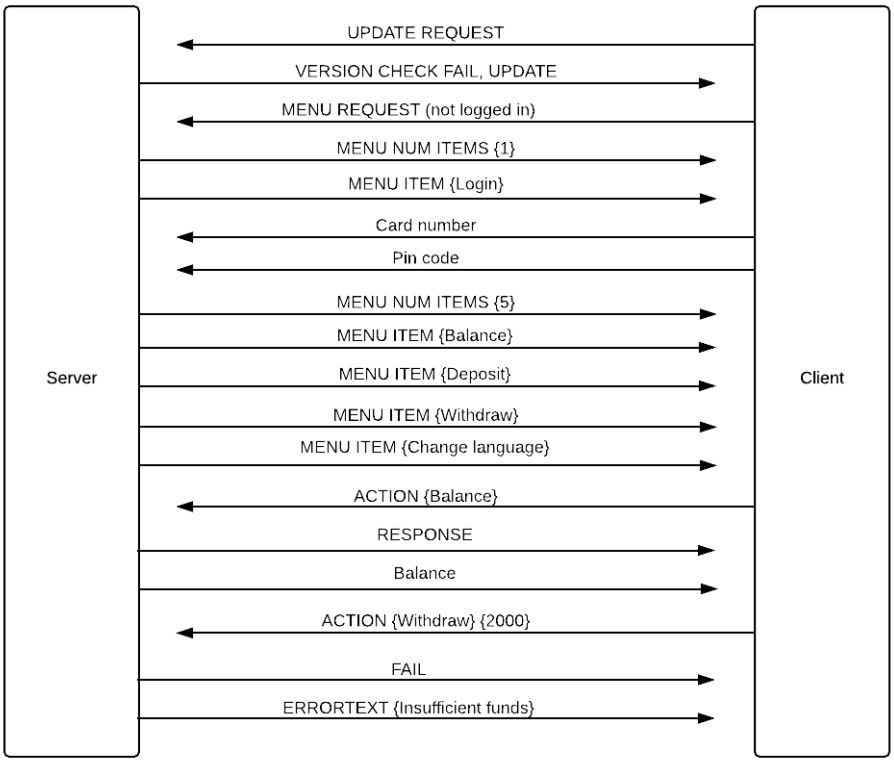


Figure 2: Example session 2