LetUgo v5.0.0

Remote Socket Server Protocol Description v1.0

2013. April

Table of Contents

Introduction

A LetUgo Remote Socket Server protocol (hereinafter called RSOSRV or simply Server) enables a 3rd party application to communicate with the LetUgo.
The communication is socket based (the transfer protocol is TCP/IP).

The two main directions of the communication

- The client sends a request to which the server sends a response.
- The server sends an event type message to the client without any preceding request

Brief Description of RSOSRV Communication

LetUgo.RSO_SRV is a socket server, via which one can get events and images from the LetUgo application. There is a header (9 bytes long), then an UTF-8 encoded XML segment which follows the header in a package:

```
#---------------------------------------------------------------------------------#
|   H   E   A   D   E   R  |                              XML                     |
|---------------------------------------------------------------------------------|
|00|01|02|03|04|05|06|07|08|                                                 ...nn|
|ff|ff|00|01|00|LL|LL|LL|LL|<?xml version="1.0" encoding="UTF-8"?>...             |
|---------------------------------------------------------------------------------|
|  S  |P |P |P |     P     |  Abb.: - SIGNATURE : A magic number for the protocol  |
|  I  |I |M |M |     L     |        - PID : Protocol IDentifier                    |
|  G  |D |A |I |     E     |        - PMA : Protocol MAin version number           |
|  N  |  |  |  |     N     |        - PMI : Protocol MInor version number          |
|  A  |  |  |  |     G     |        - PLENGTH : Length of payload (XML part)       |
|  T  |  |  |  |     T     |                                                       |
|  U  |  |  |  |     H     |                                                       |
|  R  |  |  |  |           |                                                       |
|  E  |  |  |  |           |                                                       |
#---------------------------------------------------------------------------------#
```

Client – Server Communication:

```
#---------------------------------------------------------------------------------------#
|                                                                                       |
|     | RSO_CLIENT               |      RSO_SERVER      | Note                          |
|     |                          |                      |                               |
| (1) | Connect             ---> |                      | RSO_CLIENT_REQ                |
|     |                          | <--- [Accept OR]     | RSO_SERVER_ANSWER OR Close Socket |
|     |                          |                      |                               |
| (2) | Ping                ---> |                      | RSO_CLIENT_REQ                |
|     |                          | <--- Pong            | RSO_SERVER_ANSWER             |
|     |                          |                      |                               |
| (3) |                          | <--- CarEvent        | RSO_SERVER_EVENT              |
|     |                          |                      |                               |
| (4) | GetCarEventImage    ---> |                      | RSO_CLIENT_REQ                |
|     |                          | <--- CarEventImage   | RSO_SERVER_ANSWER             |
|     |                          |                      |                               |
| (5) | Close               ---> |                      | RSO_CLIENT_EVENT              |
|     |                          |                      | Close Socket                  |
|                                                                                       |
#---------------------------------------------------------------------------------------#
```

NOTE!

- Close Socket: socket connection closed by server.
- RSO_CLIENT will get a 'Key' at the RSO_SERVER_CONNECT_ACCEPT. This 'Key' is necessary for all further requests. (1)
- RSO_SERVER_EVENT(s) is generated automatically by LetUgo application. (3)
- RSO_CLIENT can get the images based on 'EventFinished' and 'SpotID' values which are delivered by *CarEvent*.

Setting Up the LetUgo Application

Parameters of the RSOSRV can be set up in the configuration file of the LetUgo (Letugo_App.exe.config). This file is located in the LetUgo directory that is installed to following directory by default:

- In case of 32 bit operating systems: C:\Program Files\LetUgo.
- In case of 64 bit operating systems: C:\Program Files (x86)\LetUgo.

Relevant configuration values (settings are applied only after restarting LetUgo):

- RSOSRVOn: Enabling (True) or disabling (False) RSOSRV.
- RSOSRVListenerIPAddress: the RSOSRV listens on this IP address for client requests. If 0.0.0.0 is set, then the Server listens on every IP address of the local PC.
- RSOSRVListenerPort: the RSOSRV uses the TCP port specified here for client communication.

Structure of a Message (request or response)

A message consists of two parts: header and payload.
Messages have little-endian byte order.

The Header

```
#-----------------------------------------------------------------------------------#
|   H   E   A   D   E   R |                                XML                     |
|-----------------------------------------------------------------------------------|
|00|01|02|03|04|05|06|07|08|                                                   ...nn|
|ff|ff|00|01|00|LL|LL|LL|LL|<?xml version="1.0" encoding="UTF-8"?>...               |
|-----------------------------------------------------------------------------------|
|  S  |P |P |P |     P     |  Abb.: - SIGNATURE : A magic number for the protocol   |
|  I  |I |M |M |     L     |        - PID : Protocol IDentifier                     |
|  G  |D |A |I |     E     |        - PMA : Protocol MAin version number            |
|  N  |  |  |  |     N     |        - PMI : Protocol MInor version number           |
|  A  |  |  |  |     G     |        - PLENGTH : Length of payload (XML part)        |
|  T  |  |  |  |     T     |                                                        |
|  U  |  |  |  |     H     |                                                        |
|  R  |  |  |  |           |                                                        |
|  E  |  |  |  |           |                                                        |
#-----------------------------------------------------------------------------------#
```

The header consists of nine bytes:

- Signature: 2-byte unsigned integer (WORD). Its value: 0xFFFF.
- PID: (protocol identifier). Currently, only the RSOSRV protocol is supported, its ID is **0**.
- PMA: main version number. Main version number used at document release: **1**.
- PMI: secondary (minor) version number. Secondary version number used at document release: **0**.
- Payload length: 4-byte unsigned integer (DOUBLE WORD) that contains the length of payload (XML part of message).

Comprehending Protocol Version Numbers

Secondary version number: ensures backward compatibility within the main version.
Within a new secondary version:

- New message types may appear and content of existing messages may be extended with new, optional elements.
- Compulsory content of messages defined in previous protocol versions must not be modified. Existing message type must not expire.

E.g. the LetUgo supports the 1.3 RSOSRV protocol. The client may use the 1.1, 1.2 and 1.3 protocol versions to communicate with the server, but e.g. the 1.4, 1.5 etc. versions are not supported.

Main version number: Compatibility with the previous versions is not supported.

The Payload

The payload is always a UTF-8 encoded xml with little-endian byte order. The xml elements are case sensitive, the order of the elements does not matter (xs:all).

Communication

The communication must be initiated by the client with a handshake that is the Connect message. In case of successful identification, the server sends an Accept response. In every other case (invalid request, invalid user name / password, no privilege etc.) the server terminates the socket connection immediately.

In the following cases, the server terminates the socket connection immediately, without response (the cause of the error can be found in the LetUgo Event Log).

- The first client message was not the Connect request.
- Connect message: unsuccessful authentication or the user does not have admin privileges.
- Error in the Connect message.
- Unknown request or error in the data and there has not been successful authentication yet.
- Invalid key in the client request.

If the server receives an invalid message from the client after a successful authentication (e.g. unknown message type), then it answers a Reject response (except: invalid key or another unsuccessful Connect call, because, in these cases, the server terminates the connection immediately).

Message Types: Client Requests

Connect

As first message, it is the handshake. In order to establish connection, the client must send this message to the server. The server continues the communication only in case of successful authentication; otherwise, it terminates the connection immediately. The authentication can be successful only for LetUgo users of admin privileges. If the user name and/or password are invalid, the server terminates the socket connection immediately. The cause of the error is enlisted in the LetUgo log.
After a successful handshake, the client may recall a Connect function anytime (e.g. subscribing on other events or switching user) but, in these cases, the server terminates the socket connection on any irregularity immediately – because of security reasons.

Content:

- Origin (string): Name and address of the connecting client. Currently, it is not included in the authentication process but is enlisted in the LetUgo log. ("RSOSRV: client connected. Origin: <Origin>. Client: <client IP address:port>.").
- CryptedPassword (boolean): if the password is transferred in an encrypted or unencrypted way. Encrypted passwords are not supported in the current version.

- LoginName (string): Login name used for LetUgo login.
- Password (string): Password of the Login name used for LetUgo login.
- EventSubscribe (string): In this string, the client specifies events to which it wants to subscribe.
- Possible values:
    - NONE: no subscriptions.
    - CAREVENT: subscription on vehicle entry/exit events.

Close

The client may request the connection to be terminated. After receiving such message, the server terminates the socket connection. The client connection can be terminated without sending the Close message: by simply closing the socket connection. In these cases, the following error message appears in the LetUgo log:

"RSOSRV data receive failed.<Error description>".

Content:

- Key (string): Key received from the server during the handshake.

Ping

This is for verifying a connection. The server sends a Pong answer as soon as possible.

- Key (string): Key received from the server during the handshake.

GetCarEventImage

The client requests for downloading an image of a car event. The EventFinished and the SpotID together determine the event while the ImageIndex identifies the image of it.

Content:

- Key (string): Key received from the server during the handshake.
- EventFinished (dateTime): Ending date and time of the event (e.g. content of the CarEvent's EventFinished element.
- SpotID (int): ID of the event's access point (e.g. content of the CarEvent's SpotID element).
- TransactionID (string): Identifier generated by the client. The server returns the same value in the response; this is for parsing the request and the response easily.
- ImageIndex (int): identifies the image within the event. Possible values:
    - 0: ANPR camera image.
    - 1: Image of the first overview camera.
    - 2: Image of the second overview camera.
    - 3: Image of the third overview camera.

Possible server responses:

- CarEventImage: response to the request, it contains the relevant image.
- Reject: event not found or the value of the ImageIndex is invalid.

a. If the event is not found then the value of the RejectCode will be **4**.

b. If the ImageIndex is invalid then the value of the RejectCode will be **5**. E.g. only one overview camera image belongs to the access point but the client has sent a request with ImageIndex = 2.

NOTE: Every event has ANPR camera image.

Message Types: RSOSRV Responses

Accept

The server sends it on the Connect request, if the authentication was successful.

Content:

- Key (string): the server provides a unique key. The client has to use this key in every future message. If the key is invalid, the server terminates the socket connection immediately.

Reject

After a successful authentication, if the server receives an invalid message from the client (unknown message type), it answers a Reject response.

Content:

- Key (string): unique key specified in the Accept message.
- RejectCode (int): code of the error:
  - 1: other error. Detailed error description can be found in the Reason.
  - 2: a request that is unknown, not defined in the protocol has been received
  - 3: invalid request arrived
  - 4: event not found
  - 5: invalid parameter: ImageIndex
  - 6: reserved for future use
  - 7: reserved for future use
  - 8: reserved for future use
  - 9: reserved for future use
  - Reason (string): Textual description of the error cause.

Close

On closing the LetUgo application, the server sends a Close message to successfully logged in client(s) then terminates the connection 50ms later.

Content:

- Key (string): unique key specified in the Accept message.

Pong

It is a response to the client's ping request. The server forwards it as soon as possible.

Content:

- Key (string): unique key specified in the Accept message.

CarEvent

It is a vehicle entry/exit event. The client receives it only if it has been subscribed to this event in the Connect message.
Content:

- Key (string): unique key specified in the Accept message.
- EventType (string): type of the event. Possible values:
    - UNAUTHORIZED: unauthorized, automatic entry/exit
    - BACKINGUP: backing up event.
    - MOVE: authorized entry/exit if no camera direction is specified
    - MOVEIN: authorized drive in.
    - MOVEOUT: authorized drive out.
    - BLACKLISTEDIN: blacklisted drive in.
    - BLACKLISTEDOUT: blacklisted drive out.
    - BLACKLISTED: blacklisted drive through, if no camera direction is specified
    - HANDOPENIN: unauthorized, manual drive in
    - HANDOPENOUT: unauthorized, manual drive out
    - HANDOPEN: unauthorized, manual open, if no camera direction is specified
- EventStarted (dateTime): Starting date and time of the event when the vehicle has arrived to the entry/exit point.
- EventFinished (dateTime): Date and time of the event when the vehicle has left the entry/exit point.
- SpotID (int): Unique ID of the access point.
- SpotName (string): name of the access point.
- Operator (string): name of the operator (it is not the login name). If there is no operator logged in, it is an empty string.
- NumberPlate (string): the recognized number plate.
- CorrectedNumberPlate (string): number plate modified by the operator. If the operator has not made any modifications, it is an empty string.
- Direction (string): direction of the observed lane.
    - IN: drive in
    - OUT: drive out
    - UNDEFINED: camera direction is not defined
- BlackListed (boolean): it is true if the vehicle is on the blacklist.

- Permitted (boolean): it is true if the vehicle has permission.
- ParkingFee (long): amount of parking charge. It is defined only at drive out for vehicles having no permission. Otherwise, its value is -1.
- TimeSpentIn (long): time spent by vehicles with permission (in minutes). It is defined only at drive out (in case of backing up, it is -1)

CarEventImage

Response to the client's GetCarEventImage request.

Content:

- Key (string): Key received from the server during the handshake.
- EventFinished (dateTime): Relevant value of the client request.
- SpotID (int): Relevant value of the client request.
- TransactionID (string): Identifier generated by the client for parsing the request and the response easily.
- ImageIndex (int): Relevant value of the client request.
- Image (base64Binary): base64 encoded JPEG image of the car event.

Error Handling

Every error is enlisted in the LetUgo log. Error messages related to the RSOSRV always contain the RSOSRV word.

Appendices

Examples

Example: memory map of a Connect request:

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  FF FF 00 01 00 52 01 00 00 3C 3F 78 6D 6C 20 76  ÿÿ...R...<?xml v
00000010  65 72 73 69 6F 6E 3D 22 31 2E 30 22 20 65 6E 63  ersion="1.0" enc
00000020  6F 64 69 6E 67 3D 22 55 54 46 2D 38 22 3F 3E 0D  oding="UTF-8"?>.
00000030  0A 3C 43 6F 6E 6E 65 63 74 20 78 6D 6C 6E 73 3A  .<Connect xmlns:
00000040  78 73 69 3D 22 68 74 74 70 3A 2F 2F 77 77 77 2E  xsi="http://www.
00000050  77 33 2E 6F 72 67 2F 32 30 30 31 2F 58 4D 4C 53  w3.org/2001/XMLS
00000060  63 68 65 6D 61 2D 69 6E 73 74 61 6E 63 65 22 20  chema-instance" 
00000070  78 73 69 3A 6E 6F 4E 61 6D 65 73 70 61 63 65 53  xsi:noNamespaceS
00000080  63 68 65 6D 61 4C 6F 63 61 74 69 6F 6E 3D 22 63  chemaLocation="c
00000090  6F 6E 6E 65 63 74 2E 78 73 64 22 3E 0D 0A 09 3C  onnect.xsd">...<
000000A0  4F 72 69 67 69 6E 3E 54 65 73 74 43 6C 69 65 6E  Origin>TestClien
000000B0  74 3C 2F 4F 72 69 67 69 6E 3E 0D 0A 09 3C 43 72  t</Origin>...<Cr
000000C0  79 70 74 65 64 50 61 73 73 77 6F 72 64 3E 66 61  yptedPassword>fa
000000D0  6C 73 65 3C 2F 43 72 79 70 74 65 64 50 61 73 73  lse</CryptedPass
000000E0  77 6F 72 64 3E 0D 0A 09 3C 4C 6F 67 69 6E 4E 61  word>...<LoginNa
000000F0  6D 65 3E 74 65 73 74 31 3C 2F 4C 6F 67 69 6E 4E  me>test1</LoginN
00000100  61 6D 65 3E 0D 0A 09 3C 50 61 73 73 77 6F 72 64  ame>...<Password
00000110  3E 54 65 73 74 50 3C 2F 50 61 73 73 77 6F 72 64  >TestP</Password
00000120  3E 0D 0A 09 3C 45 76 65 6E 74 53 75 62 73 63 72  >...<EventSubscr
00000130  69 62 65 3E 43 41 52 45 56 45 4E 54 3C 2F 45 76  ibe>CAREVENT</Ev
00000140  65 6E 74 53 75 62 73 63 72 69 62 65 3E 0D 0A 3C  entSubscribe>..<
00000150  2F 43 6F 6E 6E 65 63 74 3E 0D 0A                 /Connect>..
```

(The header is marked with a red rectangle.)

- Signature: 0xFF (offset 00 – 01).
- Protocol identifier: 0x00 (offset 02).
- Protocol main version number: 0x01 (offset 03).
- Protocol secondary version number: 0x00 (offset 04).
- Length of payload: 0x0152 (offset 05 – 08).
- Payload: offset 09 – 015A

Example:
Connect request xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<Connect xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="connect.xsd">
    <Origin>TestClient</Origin>
    <CryptedPassword>false</CryptedPassword>
    <LoginName>testuser</LoginName>
    <Password> testuserpwd</Password>
    <EventSubscribe>CAREVENT</EventSubscribe>
</Connect>
```

Example: GetCarEventImage request xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<GetCarEventImage xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="getcareventimage.xsd">
    <Key>42930b66-6e30-4d8d-ba4b-9d175607a755</Key>
    <EventFinished>2012-01-01T11:48:13.0056789+02:00</EventFinished>
    <SpotID>1</SpotID>
    <ImageIndex>0</ImageIndex>
    <TransactionID>Tr0001</TransactionID>
</GetCarEventImage>
```