CSL (Client Side Logging) Project HLD

**Version**: 2.1

**Authors**: Dmitry Kudryavtsev, Evgeny Gavrilov, Alex Arkhipov, Stanislav Pavnyk

# 1. Contents

[1. Contents 2](#_Toc448150075)

[2. Revision history 4](#_Toc448150076)

[3. Architecture 5](#_Toc448150077)

[3.1 Application basic architecture 5](#_Toc448150078)

[3.2 Application diagram 6](#_Toc448150079)

[4. Hardware Requirements 6](#_Toc448150080)

[4.1 Disk 6](#_Toc448150081)

[4.2 Memory 7](#_Toc448150082)

[5. Software Requirements 7](#_Toc448150083)

[5.1 Software versions 7](#_Toc448150084)

[5.2 Ports 7](#_Toc448150085)

[6. MongoDB Implementation 7](#_Toc448150086)

[6.1 Data format 7](#_Toc448150087)

[6.1.1 Events 7](#_Toc448150088)

[6.1.2 Logs. 16](#_Toc448150089)

[6.1.3 Sessions 18](#_Toc448150090)

[6.2 Capacity 18](#_Toc448150091)

[6.3 Data lifecycle 19](#_Toc448150092)

[6.4 Security 19](#_Toc448150093)

[6.5 Additional info 19](#_Toc448150094)

[6.6 Working with duplicates / Handling connection problems 19](#_Toc448150095)

[7. API Implementation 20](#_Toc448150096)

[7.1 Authorization 20](#_Toc448150097)

[7.2 Protocol and data compressing 22](#_Toc448150098)

[7.3 API methods 22](#_Toc448150099)

[7.3.1 Session 23](#_Toc448150100)

[7.3.2 Log 23](#_Toc448150101)

[7.3.3 Event 25](#_Toc448150102)

[8. Deployment and upgrade 26](#_Toc448150103)

[8.1 MongoDB 26](#_Toc448150104)

[8.2 Node.js and npm (node.js package manager) 27](#_Toc448150105)

[8.3 Nginx 27](#_Toc448150106)

[8.4 Perforce 28](#_Toc448150107)

[8.5 CSL Application 28](#_Toc448150108)

[9. Troubleshooting 28](#_Toc448150109)

[9.1 Logs 28](#_Toc448150110)

[9.1.1 Mongo DB logs 28](#_Toc448150111)

[9.1.2 UI logs 28](#_Toc448150112)

[9.1.3 Nginx logs 29](#_Toc448150113)

[9.2 How to check mongoDB status 29](#_Toc448150114)

[9.3 How to restart UI 29](#_Toc448150115)

[9.4 How to restart MongoDB 29](#_Toc448150116)

[9.5 How to check that scripts for monitoring free space and deletion of duplicates work 29](#_Toc448150117)

[9.6 How to add a new event (server side) 29](#_Toc448150118)

[10. Mobile Application CSL Implementation 30](#_Toc448150119)

[10.1 Requirements 30](#_Toc448150120)

[10.2 Messages and Events 30](#_Toc448150121)

[10.3 API 31](#_Toc448150122)

[10.4 Local Storage 31](#_Toc448150123)

[10.5 CSLManager (Client side logging) 35](#_Toc448150124)

[10.6 CSLEvent description 35](#_Toc448150125)

[10.7 CSLMessage description 37](#_Toc448150126)

[11. Future plans 38](#_Toc448150127)

[11.1 UI implementation 38](#_Toc448150128)

[11.2 Reports implementation 38](#_Toc448150129)

[12. Testing considerations 38](#_Toc448150130)

[13. Additional info 38](#_Toc448150131)

[13.1 Perforce 38](#_Toc448150132)

[14. Server Information 38](#_Toc448150133)

[14.1 Disk 38](#_Toc448150134)

[14.2 Memory 38](#_Toc448150135)

[14.3 CPU 38](#_Toc448150136)

[14.4 Software versions 38](#_Toc448150137)

[14.5 Server addresses and ports 39](#_Toc448150138)

[15. Appendix 39](#_Toc448150139)

[15.1 Mapping Dictionary 39](#_Toc448150140)

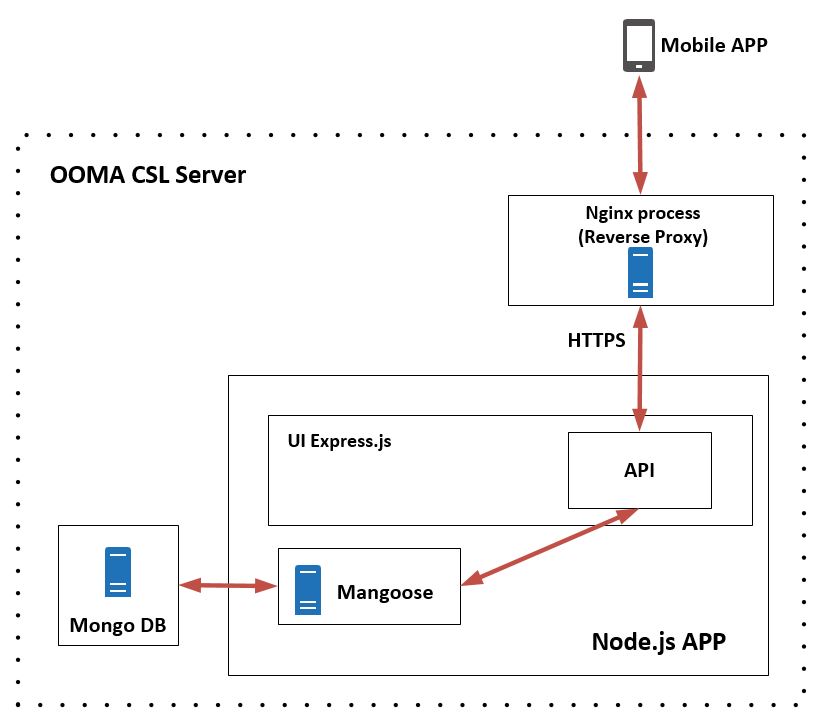
[15.2 Event Name and Event Type Mapping 40](#_Toc448150141)

# 2. Revision history

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Summary of changes** |
| 0.1 | 10/19/2015 | Dmitry Kudryavtsev | First draft |
| 0.2 | 10/26/2015 | Dmitry Kudryavtsev | Added Database part |
| 0.3 | 10/27/2015 | Dmitry Kudryavtsev, Evgeny Gavrilov | Modify after internal review. |
| 0.4 | 10/27/2015 | Alex Arkhipov | Modify after internal review |
| 0.5 | 10/28/2015 | Dmitry Kudryavtsev, Evgeny Gavrilov, Stanislav Pavnyk | Modify after internal review |
| 0.6 | 10/29/2015 | Stanislav Pavnyk | Modify according to new requirements |
| 0.7 | 11/2/2015 | Dmitry Kudryavtsev, Evgeny Gavrilov | Sections 7.1“Authorization”, 7.4”API Methods” have been updated. |
| 0.8 | 11/6/2015 | Evgeny Gavrilov | Section 6.2 “Capacity” added,  Section 6.5 “Working with duplicates” updated. |
| 0.9 | 11/11/2015 | Evgeny Gavrilov | Section 7.1 “Authorization” is updated. |
| 1.0 | 11/16/2015 | Evgeny Gavrilov | Figure 4. “Request for session\_id” is added. Section 6.4 “Security” is added. |
| 1.1 | 11/17/2015 | Evgeny Gavrilov | Log and event examples are updated. (Sections 7.3.2 and 7.3.3) |
| 1.2 | 11/26/2015 | Evgeny Gavrilov | Added Table of Events (Section 6.1.1). Section 7.1 has been updated. |
| 1.3 | 12/4/2015 | Evgeny Gavrilov | Updated Section 8.1 Mongo DB |
| 1.4 | 12/11/2015 | Evgeny Gavrilov | Added Optional data specification to RT\_CALL\_END and RT\_BATTERY\_LVL events in the Table of events (Section 6.1.1). |
| 1.5 | 1/13/2016 | Evgeny Gavrilov | Added events in 6.1.1 (Table of events) and label fields in 7.3.3 (Mapping dictionary). |
| 1.6 | 1/18/2016 | Evgeny Gavrilov | Added events in 6.1.1(Table of events).  Changed main events format. |
| 1.7 | 1/20/2016 | Evgeny Gavrilov | Added Section 10. Mobile Application SCL Implementation |
| 1.8 | 1/26/2016 | Evgeny Gavrilov | Updated Section 5.1 “Software versions”. Updated Section 5.2 “Ports”. Updated the Table of events. Removed geolocation and motion. Updated dictionaries. Added Section 8.4  “Perforce”, added Section 8.5 “CSL Application”. Updated Section 8.1 “MongoDB”. Added Section 9.6 “How to add a new event”. |
| 1.9 | 1/26/2016 | Evgeny Gavrilov | Added Section 14. “Server Information”. |
| 2.0 | 3/9/2016 | Evgeny Gavrilov | Added the name of the cellular network operator in Section 6.1.1. Events (Table of events) and 15.1 Mapping Dictionary |
| 2.1 | 4/11/2016 | Evgeny Gavrilov | Updated Section 6.3 “Data lifecycle” |

# 3. Architecture

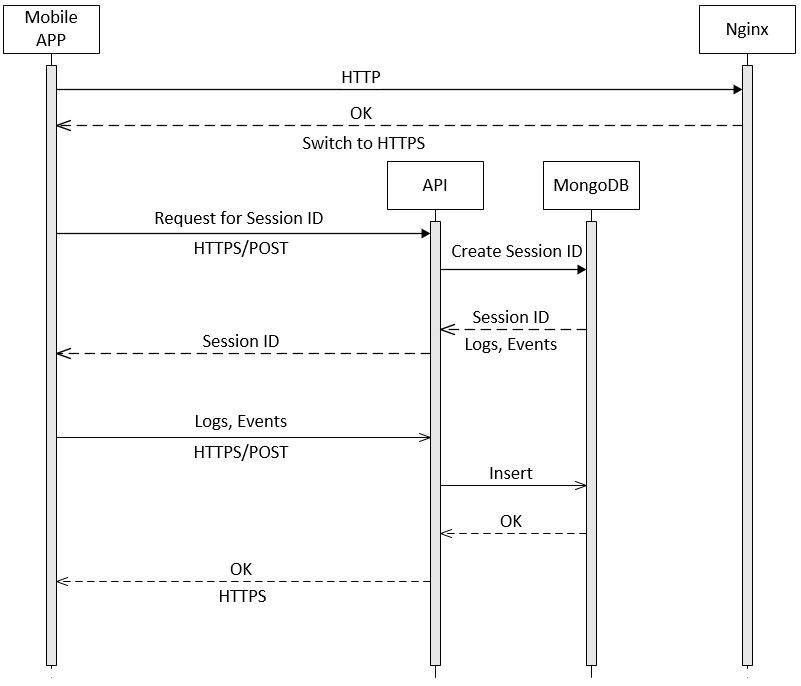
## 3.1



**Figure 1. Application Basic Architecture**

isTuses

## 3.2 Application diagram



**Figure 2. Application Diagram**

# 4. Hardware Requirements

## 4.1 Disk

With the WiredTiger storage engine, use of XFS is strongly recommended to avoid performance issues found when using EXT4 with WiredTiger.

For data storage in a first phase MongoDB will use ONE disk with capacity 10TB – after first month of launch measure of logs size will be done and total needed size of disk will be calculated.

(CSL-REQ2.2).

RAID controller will be used for storage aggregation. Only in case if it is impossible to add any more storage to a single node Shared Cluster mode of MongoDB should be used, since in a current implementation one MongoDB instance can use only one **dbpath** (CSL\_REQ2.1).

## 4.2 Memory

New Storage engine with better performance and additional features (like compressing data on disk)were added in MongoDB v3.\*. WiredTiger performance is based on RAM size. By default half of RAM size is used by it. Recommended minimum is 32 GB of RAM.

# 5. Software Requirements

## 5.1 Software versions

Nginx - v1.9.2

Node.js & npm - v0.10.33

MongoDB - 3.2

Perforce - v1319959

## 5.2 Ports

80/443 – open to the public internet – listened by Nginx.

8001,8002,8003,8004 – Reserved local ports, used by Node.js (Express.js module), accessible only in internal network.

27017,28017 – MongoDB local ports, accessible only in internal network.

# 6. MongoDB Implementation

## 6.1 Data format

In the current first phase of this product no enhanced UI and reports will be implemented. UI requirements and scope of reports that will be available for end user is a subject to be discussed. After requirements approval new section with UI and reports will be added to this document.

MongoDB works with JSON and BSON (binary) formats – this feature uses JSON. Physically all the data is stored as MongoDB files (not in JSON), their format depends on MongoDB engine (MMAPv1/WirdeTiger).

### 6.1.1 Events

Protocol ID should distinguish different versions of our logging protocol as old logs generated by an earlier version need to be processed. The protocol will evolve over time.

Main events format:

Format :

"\_id": MonogDB row ID

"sessionid" : SessionID

"devid" : DeviceID

"created\_date": Date and time when server saves this event

"client\_date": Date and time when event was generated

“event\_type”: Event Type

"event\_name": Event Name

"event\_data": Event data (Array), contains Labels and value for every label (See table)

"tag": Some tag from MA

"phone\_number": Phone number (ooma account)

"phone\_ext": phone extension ( for office)

"remote\_ip": client remote IP Address

"db\_id": id from APP local DB

“app\_version”: application version

“os\_version”: Operation system version

“app\_name”: Application name

“hw\_info”: Hardware model

“os\_name”: Operation system name (Android,iOS)

**Events** collection example:

{      "\_id" : ObjectId("559463de56b4ac33079da8be"),

        "sessionid" : "559463dc56b4ac33079da6a8",

        "devid" : "496AE7ED-E84E-4FCE-A390-717154DC7436",

        "created\_date" : ISODate("2015-07-01T22:04:14.003Z"),

        "client\_date" : ISODate("2015-07-01T20:52:23.496Z"),

**“event\_type”:” SIP**”

**"event\_name" : " SIP\_CALL\_STATE ",**

        "event\_data" :[ {

**"value" : "Success",**

**"label" : " Status",**

                "action" : "",

                "category" : ""},

{**"value" : "0",**

**"label" : " Error code",**

                "action" : "",

                "category" : ""}],

        "tag" : "",

        "phone\_number" : "6506145128",

        "phone\_ext" : "107",

        "remote\_ip" : "70.214.15.37",

"db\_id": "546784",

        “app\_version”: “110795”,

        “os\_version”: “8.1.2”,

        “app\_name”: “mobile”,

         “hw\_info”: “Iphone6Splus”,

        “os\_name”: “iOS”

           }

Events

{      "\_id" : ObjectId("559463de56b4ac33079da8be"),

   "f": "APIKEY",

    "s": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9. ",

   "h": "HW\_GUID",

    "a": "6.0",

    "o": "9.0.2",

    "m": "home",

    "i": "iPhone6,1",

    "j": "iPhone OS",

            "p": "00000000",

    "x": "0000",

    "d": [

        {

            "c": "1446130381.000",

            "e": [

                {

                    "z": 127,

                    "k": "20%"

                }

            ],

            "t": 30011,

          "b": "233"

        },

        {

            "c": "1446130181.000",

            "e": [

                {

                    "z": 27,

                    "k": "19%"

                },

                {

                    "z": 29,

                    "k": "YES"

                }

            ],

            "t": 30011,

          "b": "234"

        }

    ]

}

Table of events

|  |  |  |  |
| --- | --- | --- | --- |
| **Event type** | **Event name** | **Optional data specification** | **Description** |
| SYSTEM | APP\_START | *Network status* – **enum**:  NotReachable = 0  ReachableWiFi = 1  ReachableCellular = 2  *Carrier name*- **string:**  “Some Carrier" | Application is started |
| SYSTEM | APP\_BG | N/A | Application did enter background mode |
| SYSTEM | APP\_FG | N/A | Application did enter foreground mode |
| SYSTEM | APP\_INACTIVE | N/A | Application is about to move from active to inactive state. This can occur for certain types of temporary interruptions (such as an incoming phone call or SMS message) or when the user quits the application and it begins the transition to the background state. |
| SYSTEM | APP\_ACTIVE | N/A | Application is about to move from inactive to active state. |
| SYSTEM | APP\_LMEM | *Free memory value in MB* – **double**, example 21.56  *Memory value allocated by mobile application in MB* – **double**, example 18.5 | Low memory warning |
| SYSTEM | APP\_RNOTIFY | *Remote notification data associated*– **string**, example “{data:“some\_value”}” | iOS remote notification |
| SYSTEM | APP\_LNOTIFY | *Local notification data associated*– **string**, example “some\_value” | iOS local notification |
| SYSTEM | APP\_CRASH | *Crash name* – **string**, example “SIGABRD”  *Crash reason* – **string**, example “Paging fault”  *Crash data* – **string** | Application crash report |
| SYSTEM | NET\_CHANGE | *Old Network status* – **enum**,  *New Network status* – **enum**:  NotReachable = 0,  ReachableWiFi = 1,  ReachableCellular = 2,  *Carrier name*- **string:**  “Some Carrier" | Occurs when network state is changed to WLAN/WiFi |
| SYSTEM | APP\_KEEPALIVE | N/A | Apps wakes up in background mode |
| SYSTEM | APP\_AUDIO\_INT\_START | N/A | Audio session interruption start |
| SYSTEM | APP\_AUDIO\_INT\_STOP | N/A | Audio session interruption stop |
| SYSTEM | APP\_STOP | N/A | Normal application termination |
| UI | UI\_VIEW | *Screen name* – **string**, example: “Login” | New application screen is opened |
| UI | UI\_EVENT | *UI control name* – **string**, example: “NumpadButton+” | Important Button/control tapped |
| UI | UI\_ALERT | Alert title – **string**, example “login failed”  Alert Text – **string**, example “The internet connection appears to be offline” | Alert message is shown (error message box etc) |
| IP | SIP\_INIT | *Initialization result* – **int**, example 0 (OOMA\_SUCCESS) | Sip initialization result |
| SIP | SIP\_DESTROY | *De-initialization result* – int, example 0 (OOMA\_SUCCESS) | Sip de-initialization result |
| SIP | SIP\_REG\_STATE | *Registration status* – **enum**, example:  REG\_INVALID = -1,  REG\_UNREGISTERED,  REG\_REGISTERED,  REG\_REGISTERING,  REG\_NETWORKCHANGED  *Error code* – **int**, example 200 (OK) | Sip registration/un-registration status |
| SIP | SIP\_CALL\_STATE | *Call status* – **enum**, example:  CALL\_INVALID = -1,  CALL\_INPROGRESS,  CALL\_EARLYMEDIA,  CALL\_CONNECTING,  CALL\_CONNECTED,  CALL\_DISCONNECTED,  CALL\_DISCONNECTING,  CALL\_INCOMING,  CALL\_DECLINE,  CALL\_TRANSFER\_SUCCESS,  CALL\_TRANSFER\_FAILED  *Error code* – **int**, example 200 (OK)  *Remote number* – **string**, example “12312372005”  *Is call anonymous* – **boolean**, example FALSE | Sip call status |
| SIP | SIP\_MEDIA\_STATE | *Media state* –**enum:**  ***0 - PJSUA\_CALL\_MEDIA\_NONE,***  ***/\*\****  ***\* The media is active***  ***\*/***  ***1 -PJSUA\_CALL\_MEDIA\_ACTIVE,***  ***/\*\****  ***\* The media is currently put on hold by local endpoint***  ***\*/***  ***2 -PJSUA\_CALL\_MEDIA\_LOCAL\_HOLD,***  ***/\*\****  ***\* The media is currently put on hold by remote endpoint***  ***\*/***  ***3 -PJSUA\_CALL\_MEDIA\_REMOTE\_HOLD,***  ***/\*\****  ***\* The media has reported error (e.g. ICE negotiation)***  ***\*/***  ***4 - PJSUA\_CALL\_MEDIA\_ERROR***  *Media type* – **enum :**  ***/\*\* Type is not specified. \*/***  ***0 - PJMEDIA\_TYPE\_NONE,***  ***/\*\* The media is audio \*/***  ***1 - PJMEDIA\_TYPE\_AUDIO,***  ***/\*\* The media is video. \*/***  ***2 - PJMEDIA\_TYPE\_VIDEO,***  ***/\*\* The media is application. \*/***  ***3 - PJMEDIA\_TYPE\_APPLICATION,***  ***/\*\* The media type is unknown or unsupported. \*/***  ***4 - PJMEDIA\_TYPE\_UNKNOWN***  *Media codec info* – **string**, example “G722, 22kbps” | Sip media status |
| SIP | SIP\_TRANSPORT\_STATE | *Transport state – enum :*  0 - PJSIP\_TP\_STATE\_CONNECTED,      /\*\*< Transport connected, applicable only  to connection-oriented transports  such as TCP and TLS.      \*/     1 -  PJSIP\_TP\_STATE\_DISCONNECTED,    /\*\*< Transport disconnected, applicable  only to connection-oriented  transports such as TCP and TLS.    \*/     2 -  PJSIP\_TP\_STATE\_SHUTDOWN,        /\*\*< Transport shutdown, either                                           due to TCP/TLS disconnect error                                           from the network, or when shutdown                                           is initiated by PJSIP itself.      \*/    3 -  PJSIP\_TP\_STATE\_DESTROY,         /\*\*< Transport destroy, when transport is about to be destroyed.  *Transport status – enum :*   /\*\* Status is OK. \*/  ***0 -*** ***PJ\_SUCCESS***  ***/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****  ***\* TRANSPORT ERRORS***  ***\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/***  ***/\*\****  ***\* @hideinitializer***  ***\* Unsupported transport type.***  ***\*/***  ***171060 - PJSIP\_EUNSUPTRANSPORT***  ***(PJSIP\_ERRNO\_START\_PJSIP + 60)***  ***/\*\****  ***\* @hideinitializer***  ***\* Buffer is being sent, operation still pending.***  ***\*/***  ***171061 - PJSIP\_EPENDINGTX***  ***(PJSIP\_ERRNO\_START\_PJSIP + 61)***  ***/\* \*/***  ***/\*\****  ***\* @hideinitializer***  ***\* Rx buffer overflow. See also PJSIP\_EMSGTOOLONG.***  ***\*/***  ***171062 - PJSIP\_ERXOVERFLOW***  ***(PJSIP\_ERRNO\_START\_PJSIP + 62)***  ***/\*  \*/***  ***/\*\****  ***\* @hideinitializer***  ***\* This is not really an error, it just informs application that***  ***\* transmit data has been deleted on return of pjsip\_tx\_data\_dec\_ref().***  ***\*/***  ***171063 -  PJSIP\_EBUFDESTROYED     (PJSIP\_ERRNO\_START\_PJSIP + 63)***  ***/\* \*/***  ***/\*\****  ***\* @hideinitializer***  ***\* Unsuitable transport selected. This error occurs when application***  ***\* has explicitly requested to use a particular transport/listener,***  ***\* but the selected transport is not suitable to send request to***  ***\* the specified destination.***  ***\*/***  ***171064 - PJSIP\_ETPNOTSUITABLE***  ***(PJSIP\_ERRNO\_START\_PJSIP + 64)***  ***/\* \*/***  ***/\*\****  ***\* @hideinitializer***  ***\* Transport not available. This error occurs for example when the SIP stack***  ***\* is trying to use a SIP transport while the transport is being paused by***  ***\* application.***  ***\*/***  ***171065 - PJSIP\_ETPNOTAVAIL***  ***(PJSIP\_ERRNO\_START\_PJSIP + 65)***  ***/\* \*/***  *Transport hostname – string, example “https://sip.oomamobile.com/….”* | Sip transport status |
| RUNTIME | RT\_WEB\_REQUEST | *Request API server* – **string**, example “https://apiv2.my.ooma.com/v1”  *Request API path* – **string**, example “/res/preferences/blacklist”  *Request type* – **string**, example “GET”  *Request data* - **string** | Web request sent |
| RUNTIME | RT\_WEB\_RESPONSE | *Error description* – **string**, example “Unauthorized, please check your user name or password”  *Response data* - **string** | Request response |
| RUNTIME | RT\_WARNING | *Warning message* – **string**, example “something happened” | Runtime warning message |
| RUNTIME | RT\_LOGIN\_START | N/A | Login and manual login start |
| RUNTIME | RT\_LOGIN\_RESULT | *Login result* – **string**, example “Login failed with error: 401 - Unauthorized” | Login result |
| RUNTIME | RT\_SIP\_REG\_START | N/A | Sip registration start |
| RUNTIME | RT\_SIP\_REG\_RESULT | *Sip registration result* – **string**, example “Sip registration failed with error: 401 - Unauthorized” | Sip registration result |
| RUNTIME | RT\_OCALL\_START | *Remote number* – **string**, example “12312372005”  *SIP call status code* – **int**, example 200 (OK)  *Network status* – **enum**:  NotReachable = 0,  ReachableWiFi = 1,  ReachableCellular = 2  Call\_id – **String ,** example 123e4567-e89b-12d3-a456-426655440000 | Outgoing call start. Calling PJSIP make call function. |
| RUNTIME | RT\_ICALL\_START | *Remote number* – **string**, example “12312372005”  *SIP call status code* – **int**, example 200 (OK)  *Network status* – **enum**:  NotReachable = 0  ReachableWiFi = 1  ReachableCellular = 2  Call\_id – **String ,** example 123e4567-e89b-12d3-a456-426655440000  *Carrier name*- **string:**  “Some Carrier" | Incoming call |
| RUNTIME | RT\_CALL\_ANSWER | *Remote number* – **string**, example “12312372005”  *Network status* – **enum**:  NotReachable = 0  ReachableWiFi = 1  ReachableCellular = 2  Call\_id – **String ,** example 123e4567-e89b-12d3-a456-426655440000  *Carrier name*- **string:**  “Some Carrier" | When calling PJSIP answer function |
| RUNTIME | RT\_CALL\_END | *Network status* – **enum**:  NotReachable = 0  ReachableWiFi = 1  ReachableCellular = 2  *Media packets lost %%* - **double**, example 23.67 (% of packets lost)  *Media packets reorder %%* - **double**, example 15.56 (% of packets reorder)  *Media packets discarded %%* - **double**, example 8.45 (% of discarded packets)  *Media packets duplicated %%* - **double**, example 5.89 (% of duplicated packets)  *Remote number* – string, example “12312372005” (\*this field is important, so it’s removing from array to main document fields by server API , to make queries much easier)  *Call\_duration* – int , example 145 (\*this field is important, so it is moving from array to main document body by server API to make queries much easier)  Call\_direction – **enum** : (\*this field is important, so it’s moving from array to main document fields by server API to make queries much easier)  Incoming = 0  Outgoing = 1  Call\_end\_status – **enum** :  Normally = 0  Abnormally = 1  Call\_end\_information – **string** ,example, NormalCallClearing  Call\_id – **String ,** example 123e4567-e89b-12d3-a456-426655440000  *Carrier name*- **string:**  “Some Carrier" | When calling PJSIP disconnect function |
| RUNTIME | RT\_BATTERY\_LVL | *Current battery level* – double, example 35.55 (%)  *Parameter\_info* – String , example ‘Keepalive end’ | Used before and after energy-consuming operations such as WebServer interaction, SIP calls, file system I/O operations etc. |
| RUNTIME | RT\_ERROR | *Error message* – **string**, example “Something went wrong” | Runtime error message |
| SYSTEM | APP\_SIZE | Size – **double**, example 11.57  (Size of the app on file system) | On keep alive  once in 12 hours |
| RUNTIME | RT\_CHANGE\_SERVER\_ADDRESS | *Address –* ***string,*** *example “[sim.ooma.com](http://sim.ooma.com)”*  *Server type –* ***enum:***  *WEB\_API\_SERVER = 0*  *SIP\_SERVER = 1*  *Is  address hardcoded IP – Boolean:*  *0 – DNS name*  *1 - Hardcoded IP*  *Network status* – **enum**:  NotReachable = 0  ReachableWiFi = 1  ReachableCellular = 2  *Carrier name*- **string:**  “Some Carrier" | Client started using other server address by some reason |
| RUNTIME | RT\_FIVE\_UNSUCCESSFUL\_LOGIN\_ATTEMPTS | N/A | User has made 5 unsuccessful login attempts during 5 minutes. |

### 6.1.2 Logs.

***Note! In a first phase of development an upload of logs will not be implemented. It will take place in a second phase.***

Format :

"\_id": MonogDB row ID

"sessionid" : SessionID

"devid" : DeviceID

"created\_date": Date and time when server saves this event

"client\_date": Date and time when event was generated

"log\_level" – level of this log

0: 'EMERGENCY'

1: 'ALERT'

2: 'CRITICAL'

3: 'ERROR'

4: 'WARNING'

5: 'NOTICE'

6: 'INFO'

7: 'DEBUG'

default: 'UNKNOWN'

"Log": Detailed log information

“process\_id” – ID of local process

"tag": Some tag from MA

"phone\_number": Phone number (ooma account)

"phone\_ext": phone extension (for office)

"remote\_ip": client remote IP Address

"db\_id": id from APP local DB

“app\_version”: Application version

“os\_version”: Operation system version

“app\_name”: Application name

“hw\_info”: Hardware model

“os\_name”: Operation system name (Android, iOS)

Example:

{

"\_id" : ObjectId("55677a0a1234600027b927c2"),

"sessionid" : "55677a091234600027b927c1",

"apikey" : "254556e0-0572-11e5-9f4f-8321894734a1",

"devid" : "AD386AB9-70F4-45F4-A46A-65F76BC7D5F5",

"created\_date" : ISODate("2015-05-28T20:26:50.891Z"),

"client\_date" : ISODate("2015-05-21T01:19:44.299Z"),

"level" : 7,

"log" : "OomaAPI.m[2097] updateSettings. api: /res/voicemail/folders\nurlString: https://api-qa.ooma.com/v1/res/voicemail/folders?access\_token=7eb6350e1c3a9200540cf27611af3ed6 ",

"processid" : "",

"tag" : "",

"phone\_number" : "6699007157",

"phone\_ext" : "",

"remote\_ip" : "10.66.10.254",

"db\_id":"3452",

"local\_ip" : "",

“app\_version”: “110795”,

        “os\_version”: “8.1.2”,

        “app\_name”: “mobile”,

         “hw\_info”: “Iphone6Splus”,

        “os\_name”: “iOS”

           }

### 6.1.3 Sessions

Format:

        "\_id" : MonogDB row ID

        "apikey" : Unique key, every type of Application has the same API key

        "devid" : DeviceID

        "created\_date" : Date and time when session was created

        "os\_name" – Operation system name (IOS / Android)

        "hw\_info" – Name of used device (iphone6splus etc.)

        "devManufacture": manufacturer of the device (e.g. "Apple")

        "osVersion" : Application Operation System Version

        "appVersion" : Application version

        "appName" : application name (office/mobile)

Example:

{

\_id: ObjectId("569fa276fa59ae877c5e3cd6")

 apikey: "sometestAPIkey"

 devid: "testDevID33"

 created\_date: 01/20/2016 06:06:30 PM (+0300)

 os\_name: "iOS"

 hw\_info: "iphone6plus"

 devManufacturer: "Apple"

 os\_version: "8.3"

 appVersion: "5.2(12345)"

 appName: "mobile"

}

## 6.2 Capacity

All values based on Usage Sept 2015 doc were increased fivefold.

MongoDB disk = 10TB

Required free space = 2TB (20%)

Available volume = 8TB

Active devices = 90k\*5=450k

Avg calls per month per active device = 2.5\*5 = 12.5

Avg events + logs per call = 60

Avg size of 1 log/event in WiredTiger storage engine with ‘snappy’ compression = 0.1kb (need to make more test to check it)

Data volume for 1 month 450k\*12.5\*60\*0.1kb = 32GB

Data volume for 1 year = 32GB\*12 = 384 GB

So, if all variables are correct, data can be stored for about 20 years, after that old data will need to be deleted.

## 6.3 Data lifecycle

When the free space on disk is less than 20%, Nagios monitoring service will send a notification to administrators to take an appropriate action.

## 6.4 Security

Mongo instance must work with “-auth” parameter. This parameter enables authorization to control user’s access to database resources and operations. When authorization is enabled, MongoDB requires all clients to authenticate themselves first in order to determine the access for the client. All mongo-users must have strong passwords and special mongoDB roles, which grant minimum of necessary accesses. Outside of role assignments, the user has no access to the system. All requests for changing privileges must be performed only through MongoDB DBA.

All collected data stores in mongoDB database files with ‘snappy’ compression provided by WiredTiger storage engine. These DB-files are stored in a special disk, without any encryption from MongoDB side. Since MongoDB doesn’t have encryption mechanism for db files, they can be encrypted by third-party solutions if it is necessary in the future. Some examples of these solutions can be found here (<https://docs.mongodb.org/manual/core/security-encryption-at-rest/>)

## 6.5 Additional info

The database supports two store engines – MMAPv1 and WiredTiger.

WiredTiger is more preferable because it supports data compression. Using this feature it can store 20 times more data than using MMAPv1.

To get data from mongo DB, the following components can be used:

1. Mongo-shell
2. Umongo and other third-party GUI

## 6.6 Working with duplicates / Handling connection problems

Events and logs may be duplicated due to various reasons (connection problems, server instability), that may cause incorrect reporting and other problems.

The easiest way to handle duplicate records is to add a new field from a mobile local database with ID value of every row in that DB (local ID) and use hardware id (hw\_id) and the client\_date field which contains data of when log/event is generated. If a log with the same hardware id and the same local ID or client\_date is uploaded, MongoDB will discard such events or logs.

Even if a customer reinstalls an application, the counter returns to zero and duplicates are generated, Mongo will discard duplicates and not insert them in DB by using this logic. App will just get “OK” response and delete duplicates from local DB.

# 7. API Implementation

## 7.1 Authorization

For upload of logs and events Auth Token received from WebAPI is used. Before a creation of a new session, CSL checks if a token is valid in local MongoDB token storage/WebAPI.

1)      If a token is valid, CSL server generates session\_id for the app without any limitation on loading logs and events and sends it to the app

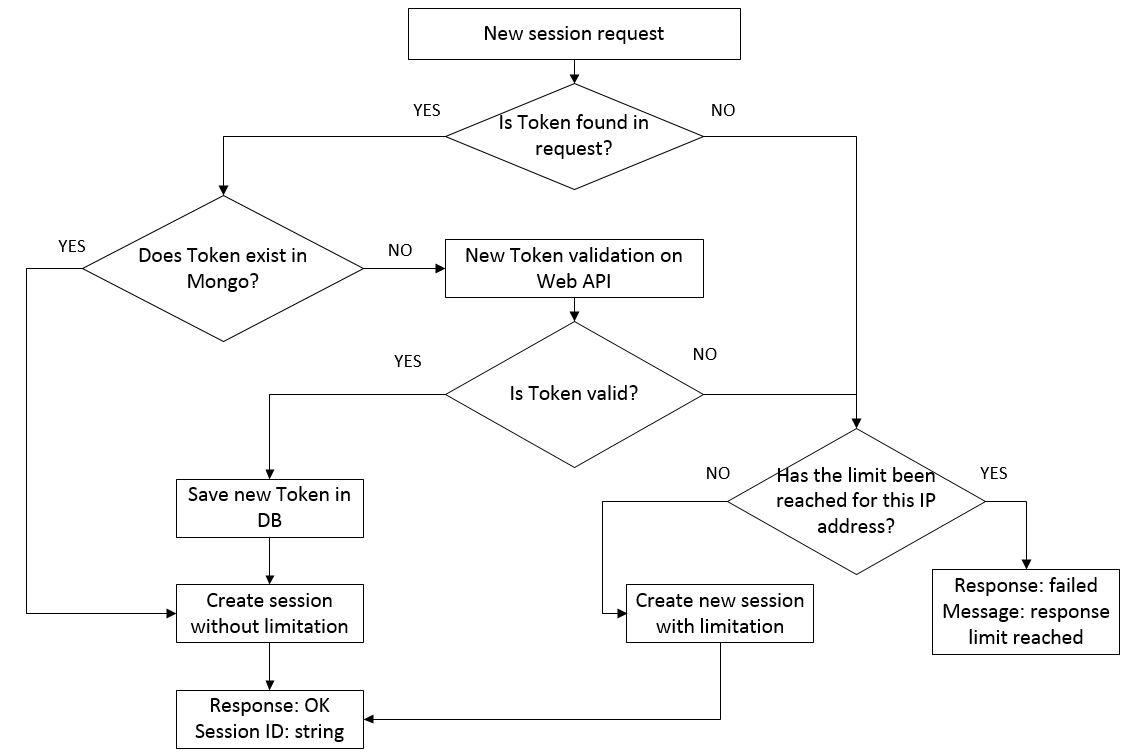
2)      If a token is not valid:

a)      If user(s) has not reached a limit for uploading data from this IP address (1000 documents (rows)) CSL server generates session\_id for the app with a limitation flag and sends it to the app (if session\_id limitation flag is set, every loaded log/event is counted and this number is added to a special collection for IP address, from which session was generated. When this number reaches hardcoded limitation, CSL server rejects a request for creating a new session without an access token from these IPs).

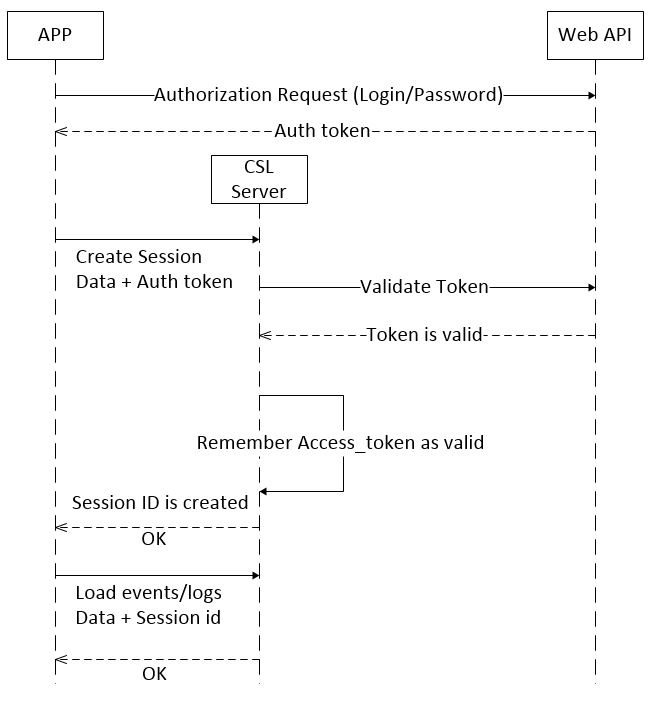
b)      If user has reached a server limit for uploading data from this IP address, then CSL server rejects request for a new session.

All limitation based on counter are deleted if user passes authorization using an access token and counter is created again when user comes without a token.

(CSL-REQ1.1).



**Figure 4. Request For session\_id (Session creation event in Figure 2. Application diagram)**



**Figure 5. Authorization Diagram**

## 7.2 Protocol and data compressing

All requests to API must be sent via HTTPS, using gzip data compression. Additional compression mechanism must be implemented on both Client and Server side to decrease average size of logs and events. For this purposes maps and shortened parameter names must be used.

## 7.3 API methods

API main page /api/v1

### 7.3.1 Session

Create a Session (/api/v1/session)

**Description**:

Create a temporary session; output key is used to validate request input body.

**Input (Short names from web requests – full name (Data Format): Description**).

“f” - apikey (String) : apikey

“w” – access token (String): Web API access Token

“h” - devid (String) : Device id

“j” - Os\_name (String) : For example: IOS, ANDROID, WP, BLACKBERRY

“i” - hw\_info (String) : For Example [iPhone 5s, iPhone 5c, iPhone 6, Nexus 6, galaxy 6]

“o” - osVersion (String): Device operating system version for example: 8.3, 5.2

“a” - AppVersion(String): application version

“m” - appName(String): application name (mobile/office)

“q” – devManufacturer(String): Device manufacturer (Apple, Samsung etc.)

**Example of session:**

**output**:

sessionid (String) : Used when upload log or event

**method**:

POST

### 7.3.2 Log

Save log (/api/v1/log)

Description:

Upload log to mobile log server

Input (First character is a short name for each value):

“f” - apikey (String) : apikey

“w” – access token (String): Web API access Token

“s” - sessionid (String) : Session id from create session

“h” - devid (String) : Device id

“a” - app\_version: Application version

“o” - os\_version: Operation system version

“m” - app\_name: Application name

“q” – devManufacturer(String): Device manufacturer (Apple, Samsung etc.)

“i” - hw\_info: Hardware model

“j” - os\_name: Operation system name (Android, iOS)

“d” - data (log objects) : Array of log objects

log objects:

[{

“v” - level: int,

“l” - log: String,

“y” - processid: String,

“g” - tag: String,

“b”- db\_id: String,

“r”- local\_ip: String, //Local IP address of device

“c” - client\_date: date //Local date of device when log was generated

}];

**Log example:**

{

    "f": "sometestAPIkey",

    "s": "Session\_key",

    "h": "testDevID33",

    "a": "6.0(12345)",

    "o": "6.0.2",

    "m": "mobile",

    "i": "iPhone6",

    "j": "iOS",

    "q": "Apple",

            "p": "00000000",

    "x": "0000",

    "d": [

        {

          "c": "1446130381.000",

          "v": 4,

          "l":"Test log message",

          "y":"Internal process\_id",

          "r":"192.192.192.129",

          "g":"test\_tag",

          "b": "322",

          "r":"Some local\_ip"

          }

    ]

}

**methods**:

POST

**Output**:

HTTP 201 – success

or

HTTP 500 - error

### 7.3.3 Event

Save event (/api/v1/event)

**Description**:

Upload events to mobile log server

**input**:

“f” - apikey (String) : apikey

“w” – access token (String): Web API access Token

“s” - sessionid (String) : Session id from create session

“h” - devid (String) : Device id

“a” - app\_version: Application version

“o” - os\_version: Operation system version

“m” - app\_name: Application name

“q” – devManufacturer(String): Device manufacturer (Apple, Samsung etc.)

“i” - hw\_info: Hardware model

“j” - os\_name: Operation system name (Android, iOS)

event objects:

[{

“t” - event\_type and event\_name: String, (both values in one number, description below)

“e” - event\_data: Event data object(array),

“g” - tag: String,

“b” - db\_id: String //row id from app local Database

“r” - local\_ip: String, //Local ip address of device

“c” - client\_date: date //Local date of device event was generated

}];

event data object:

[ {

“z” - label: String,

“k” - value: String

}]

**methods:**

POST

**Event example:**

{

    "f": "sometestAPIkey",

    "s": " Session\_key",

    "h": "testDevID33",

    "a": "6.0(12345)",

    "o": "6.0.2",

    "m": "mobile",

    "i": "iPhone6",

    "j": "iPhone OS",

    "q": "Apple",

            "p": "00000000",

    "x": "0000",

    "d": [

        {

          "c": "1446130381.000",

          "r":"192.192.192.129",

          "b": "623",

          "t": 30011,

          "g":"test\_tag",

          "e": [

                {

                    "z": 8,

                    "k": "test\_value\_1"

                },

                {

                    "z": 21,

                    "k": "test\_value\_2"

                }

            ]}

    ]

}

**Output**:

HTTP 201 – success

or

HTTP 500 – error

See Appendix 15.2 Event Name and Event Type Mapping.

# 8. Deployment and upgrade

All modules, libraries and source code must be done through Perforce or Ooma local rpm repository.

## 8.1 MongoDB

There is no necessity to install mongoDB, it can be just downloaded from the official website, unzipped and run. To upgrade mongo to a newer version it must be stopped and an executable file must be replaced on the newer version.

After mongoDB is installed, it should be configured for CSL. Deployment instructions are as follows:

1.       Create directory, where MongoDB stores DB files **(mkdir -p /db/mongodb/data)**

2.       Run mongo -   **mongod --dbpath /db/mongodb/data/  &**

3.       Connect to mongo with a deployment script that is located here **//internal/software/mobile/ServerSide/mobilelogs/script/Deploy/mongodb\_deployment.js**

Command to connect: **mongo --shell mongodb\_deployment.js**

4.       Run 2 deployment scripts

a.       **deployment.deployment\_start()** This script will create users for node.js app with necessary privileges  and some collections with unique indexes.

b.      **deployment.create\_admin("Admin\_user\_name","Admin\_password"**

This script creates DBA user in mongoDB with a specified username and password.

5.       Exit from mongo shell.

6.       Find and terminate mongod process.

7.       Start mongo in auth mode using the following command:

**mongod --dbpath /db/mongodb/data --storageEngine wiredTiger --auth --logpath /var/log/mongodb/mongod.log &**

or using mongod daemon **- /etc/init.d/mongod start (in p.8.5 there are some instructions how to add it)**

8.       Connect to mongo using admin username and password **mongo -u USERNAME -p --authenticationDatabase admin**

9.      Check that authentication is successful, try to make a query.

10.   Check that 3 databases are successfully created (notification, express, applog) and users exist in each db.

11.   Check that unique indexes were created in applog db for collections: events and log data

## 8.2 Node.js and npm (node.js package manager)

Node.js must be installed from the official website; npm can be installed from the repository.

All additional modules for node.js must be installed via npm lists in package.json file in the main source-code file directory. Upgrade of all modules must be also performed using (or through/via) npm.

The log server is self-contained and does not have to access the internet for libraries, etc. that node.js implementation needs.

## 8.3 Nginx

Nginx can be installed and upgraded from the repository.

## 8.4 Perforce

To deploy perforce it must be downloaded from the official web site – HELIX P4: COMMAND-LINE for RHEL6 OS via rpm – (perforce-cli-base.x86\_64 file). FTP server with these files can be found here - [Link](http://package.perforce.com/yum/rhel/6/x86_64/)

Then run - yum install NAME\_OF\_RPM\_PACKAGE

After install of perforce

1)    Add two new paths to /etc/profile

P4PORT=10.66.12.106:6666

P4CLIENT=mobilelogs-app

2)    p4 client can be added by command – p4 client. Root-path for this client must be set - /root/mobilelogs-app

## 8.5 CSL Application

Application source code is located on perforce server, so deployment and upgrade operation must be performed using perforce.

            To get latest version of the code – run

1)    export P4USER=YOUR\_PERFORCE\_USERNAME

2)    p4 info

In the output of this step

Client name must be mobilelogs-app

User name must be YOUR\_PERFORCE\_USERNAME

3)    p4 sync //internal/software/mobile/ServerSide/...

After these commands perforce will download all app files, if they already exist, they will be upgraded to the latest version.

On the first install app daemon (file mobilelogs in …/script/Deploy/ ) must be copied to /etc/init.d/

And Permissions for executing need to be added (chmod +x /etc/init.d/mobilelogs)

The same operations must be performed for mongod daemon (mongod file is in the same directory)

# 9. Troubleshooting

## 9.1 Logs

### 9.1.1 Mongo DB logs

/var/log/mongodb

### 9.1.2 UI logs

 /root/.pm2/logs

“Live” mode – run command ‘pm2 logs’

### 9.1.3 Nginx logs

/usr/local/nginx/logs

## 9.2 How to check mongoDB status

1. Connect to a server using SSH, and run mongo shell using command – mongo --port 27017 , if error is returned, then type /etc/inid.d/mongod to restart.
2. Use third-party tools to connect to mongo from remote computer.

In a 2nd phase of development tool to monitor the status of the server continuously must be implemented. – To be checked with Nagios

## 9.3 How to restart UI

/etc/init.d/mobilelogs restart

## 9.4 How to restart MongoDB

/etc/init.d/mongod restart

## 9.5 How to check that scripts for monitoring free space and deletion of duplicates work

Check cron logs in /var/log/cron

## 9.6 How to add a new event (server side)

To do this, dictionaries must be updated. All these dictionaries are placed in perforce in fole decode.js. This file is located here - //internal/software/mobile/ServerSide/mobilelogs/api/decode.js

In this file Dictionaries are global values

1)      Event\_type\_dict – contains values, possible for event\_type field

2)      Event\_name\_dict – contains values, possible for event\_name filed

3)      Labeldic – contains values for labels fields in event\_data array – in fact they are possible label names

4)      Values\_dic – dictionary-partner for labeldic has the same structure, but contains possible values of every label for decoding information

e.g.

        0: {"0":"Not Reachable",

            "1":"Reachable Wifi",

      "2":"Reachable Cellular"},//'Network status'

            For label = 0 (NetWork status) this dictionary has decoding information about three values

(0 ,1 and 2) , in fact app receives value in numbers, but using this decoding rules replaces it with logical human-reading information.

            Some values in this dictionary have the value = **undefined.** It means that, these values must not be decoded (e.g. it contains readable information from the beginning)

            So, to add a new event:

1)      Event\_type\_dict must be updated if a new event\_type is added

2)      New event\_name must be added in Event\_name\_dict

3)      Labeldic must be updated if this new event contains new labels

4)      Values\_dic must be updated if some values must be decoded for new labels

# 10. Mobile Application CSL Implementation

## 10.1 Requirements

CSL framework provides:

1. API to be used within mobile application which allows to:
   1. Log message
   2. Log event
2. Local storage of messages and events
3. Uploading messages and event to log server

## 10.2 Messages and Events

Message is a custom text string logged by the application. Message consists of:

1. Message string
2. Timestamp
3. Logged in user
4. Hardware ID
5. Hardware model
6. OS name
7. OS version
8. Application name
9. Application version

Event is information of specific conditions logged by the application. Event consists of:

1. Event category (as defined in MA HLD)
2. Event name (as defined in MA HLD)
3. Event data (as defined in MA HLD)
4. Timestamp
5. Logged in user
6. Hardware ID
7. Hardware model
8. OS name
9. OS version
10. Application name
11. Application version

The following is considered for both messages and events:

* If no user is logged in at the moment of message or event, the field is empty string.
* Timestamp includes year, month, day, hours, minutes, seconds, milliseconds and time zone.
* Hardware ID is unique GUID associated to installation. I.e. the same application will have the same Hardware GUID on multiple runs. However after application reinstallation it may be changed.
* Application name is target name (f.e., “home”, “office”)

## 10.3 API

CSL framework provides two methods:

- (void)logEvent:(CSLEventType)type withData:(NSDictionary \*)data;

- (void)logMessage:(CSLLevel)level andMessage(NSString \*)message;

- (void)logMessage:(NSString \*)message;

The following is assumed:

1. type is valid
2. data can be nil
3. message is non-nil and valid.
4. if no log level specified, DEBUG is assumed

## 10.4 Local Storage

All events and messages are stored on the device locally. SQLite DB is used as a local storage. SQLite DB contains two tables: `event` and `log`. The structure of the `event` table is as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Name | Type | Size | Primary Key | Nullable |
| id | INTEGER |  | true | NO |
| created\_date | DATETIME |  |  | NO |
| type | INTEGER |  |  | NO |
| data | TEXT |  |  | YES |
| username | VARCHAR | 255 |  | YES |
| hwUuid | VARCHAR | 36 |  | YES |
| hwModel | VARCHAR | 255 |  | YES |
| osName | VARCHAR | 25 |  | YES |
| osVersion | VARCHAR | 25 |  | YES |
| appName | VARCHAR | 25 |  | YES |
| appVersion | VARCHAR | 25 |  | YES |
| status | INTEGER |  |  | NO |

**id** is a unique auto increment number.

**type** is an integer, a combination of event category and event subtype (name). Every category can contain 10000 of subtypes. Hence events of first category start numbering from 0, events of second category start numbering from 10000, etc. For example, event 20003 is 4th event of 3rd category according to MA HLD – <SIP:SIP_MEDIA_STATE>

**data** is JSON encoded dictionary.

**status** is used to track upload state (refer to section “Uploading” below).

The structure of the `log` table is:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Name | Type | Size | Primary Key | Nullable |
| id | INTEGER |  | true | NO |
| created\_date | DATETIME |  |  | NO |
| message | TEXT |  |  | YES |
| level | INTEGER |  |  | NO |
| username | VARCHAR | 255 |  | YES |
| hwUuid | VARCHAR | 36 |  | YES |
| hwModel | VARCHAR | 255 |  | YES |
| osName | VARCHAR | 25 |  | YES |
| osVersion | VARCHAR | 25 |  | YES |
| appName | VARCHAR | 25 |  | YES |
| appVersion | VARCHAR | 25 |  | YES |
| status | INTEGER |  |  | NO |

**id** is a unique auto increment number.

**status** is used to track upload state (refer to the section “Uploading” below)

**level** defines log level (0 – 'EMERGENCY', 1– 'ALERT', 2 – 'CRITICAL', 3 – 'ERROR'; break, 4 – 'WARNING', 5 – 'NOTICE', 6 – 'INFO', 7 – 'DEBUG')

All DB requests are done in a queue.

Whenever message or event is generated, it is queued to be stored in DB.

LOCAL STORAGE ROTATION

There should be a maximum of records stored in DB. Currently this is limited to 1 MB and should be easily changed in the code. A file handler is installed by CSL framework to monitor file size and once exceeded, 30% of the oldest records (both `log` and `event` table) should be removed.

UPLOADING

Local storage is uploaded to log server once in 10 minutes or after 500 messages/events are received (whatever comes first).

Log servers are defined in Settings manager; there can be several of them. Log servers should be changeable without rebuild of the application (downloaded as settings).

Uploading is done to one of the server, if server is not available or error is returned, upload is done to the next server from the list according to round-robin algorithm.

If no server from the list is responding, another try should happen in 10 minutes while messages and events are collected locally as described above.

Record status of “0” indicates the record if new and not uploading. Whenever record is selected to be uploaded, its status is changed to “1”. After record is uploaded it is removed from DB. In case of upload error, status changes back to “0” for next upload try.

SERVER API

Server API implements 3 methods over HTTPS:

* /session
* /log
* /event

All requests may be sent with GZIP compression (if supported by server).

Before actual upload application should receive session key using API key (stored in Settings manager). If recently received session key is expired, new session key should be received.

/log and /event request contains JSON encoded payload in HTTP body and uploads array of messages and events accordingly.

/log request body example is:

{

"apikey":"APIKEY",

"sessionid":"SESSIONID",

"devid":"HW\_GUID",

"app\_version": "6.0",

"os\_version": "9.0.2",

"app\_name": "home",

"hw\_info": " iPhone6,1",

"os\_name": "iPhone OS",

"data":[

{

"created\_date":"2015-03-03T11:33:33.000-0700",

"log":"Hello world",

"level":1,

"phone\_number" : "000000000",

"phone\_ext":"0000"

},

{

"created\_date":"2015-03-03T11:34:34.000-0700",

"log" : "Foo bar",

"level":1,

"phone\_number" : "000000000",

"phone\_ext":"0000"

}

]

}

/event request body example is:

{

"apikey":"APIKEY",

"sessionid":"SESSIONID",

"devid":"HW\_GUID",

"app\_version": "6.0",

"os\_version": "9.0.2",

"app\_name": "home",

"hw\_info": "iPhone6,1",

"os\_name": "iPhone OS",

"data":[

{

"created\_date":"2015-03-03T11:33:33.000-0700",

"event\_data":

[

{

"label":"BatteryLvl",

"value":"20%"

}

],

"event\_name":"RUNTIME",

"event\_type":"RT\_BATTERY\_LVL",

"phone\_number":"00000000",

"phone\_ext":"0000"

},

{

"created\_date":"2015-03-03T11:34:34.000-0700",

"event\_data":

[

{

"label":"BatteryLvl",

"value":"19%"

},

{

"label":"Charging",

"value":"YES"

}

],

"event\_name":"RUNTIME",

"event\_type":"RT\_BATTERY\_LVL",

"phone\_number":"00000000",

"phone\_ext":"0000"

}

]

}

## 10.5 CSLManager (Client side logging)

The CSLManager will be created as one of Core components available for both Residential and Office targets. This component is responsible for log events and messages collection, local storage and uploading to the remote Ooma log server.

CSLManager Lifecycle:

|  |  |
| --- | --- |
| Phase | Actions |
| **Application start** | Logger initialization right after application start before the app is connected to the Ooma service. So, if the user cannot login it is known why. |
| Uploading previously collected events and logs (if any) |
| **Application running** | Runtime logs and events collection (provided from UI and core components by calling Logger API) |
| Uploading events and logs collected by timeout (timeout interval – TBD. Logger feature needs to be tested and collected logs size needs to be estimated to specify an appropriate timeout) |
| Uploading events and logs collected by reaching log file size limit (size limit – TBD. Logger feature needs to be tested and collected logs size needs to be estimated to specify a limit value). |
| **Application termination** | Logger de-initialization |

Application crashes notice: In case of Application crash, the appropriate log line will be passed to Logger component, and then App will be killed by iOS and restarted. This application crash related log data will be uploaded to the log server on this automatic app restart. This is native iOS behavior; not any additional logger functionality is required in case of application crashes.

Logs uploading notice: the collected logs should be uploaded in case of WiFi network connection is established and no active call is placed.

## 10.6 CSLEvent description

The following data have to be included to CSLEvent for future logs sorting, analysis and issues debugging:

* **<TIME\_STAMP>** - GMT date and time of event occurrence
* **<HW\_ID>** - Uniquely user HW identify used to associate logs with a specific device.
* **<HW\_INFO>** - HW device model name (iPhone4/5/6, iPad)
* **<OS\_NAME>** - Operation system name – iOS/Android
* **<OS\_VERSION>** - Operation system version (7.1.1, 8.0, 9.0 etc)
* **<APP\_TYPE>** - Application type (Home/Office) to associate logs with specific application type.
* **<APP\_VERSION>** - Application build version (5.1.9 (65789), 1.8 (123456) etc)
* **<USER\_NAME>** - Username (Ooma account) to associate logs with specific user.
* **<EVENT\_TYPE>**
  + **SYSTEM** – iOS related events such as application start/stop/crash, network status change, battery/memory alerts etc
  + **UI** – UI surfing and other user related activities
  + **RUNTIME** – application runtime internal logic events and important conditions to be highlighted (critical code sections logging, runtime errors, WebAPI requests and responses etc)
  + **SIP** - The SIP stack related events (registration state, call state etc)
* **<EVENT\_NAME>** - Uniquely identify of event occurred (EVT\_APP\_START, EVT\_CALL\_END etc)
* **<EVENT\_DATA>** - Associated data user for logs analysis and issues debugging

**PRIVACY WARNING: All end-user private data such as user name (possible), password, first/last/.. names, billing info etc have to be excluded from logged data.**

Example 1: User tries to listen voicemail, but application shows some error message.

Logs collected:

…..

8/17/2015 12:45:33;<HW\_ID>;<APP\_TYPE>;<USER\_NAME>;UI;UI\_VIEW;VoicemailViewController

8/17/2015 12:45:34;<HW\_ID>;<APP\_TYPE>;<USER\_NAME>;UI;UI\_VIEW;VoicemailDetailsViewController

8/17/2015 12:45:35;<HW\_ID>;<APP\_TYPE>;<USER\_NAME>;UI;UI\_TAP;Play button tapped

8/17/2015 12:45:35;<HW\_ID>;<APP\_TYPE>;<USER\_NAME>;RT; RT\_WEB\_REQUEST;Type=GET URL=http://my.ooma.com/v2/vm?id=…. Error=200 Ok

8/17/2015 12:45:45;<HW\_ID>;<APP\_TYPE>;<USER\_NAME>;RT; RT\_WEB\_RESPONSE;Response data=[“Error: 408-Request timeout”]

8/17/2015 12:45:45;<HW\_ID>;<APP\_TYPE>;<USER\_NAME>;RT; RT\_ERROR;Web request filed with error 408

8/17/2015 12:45:45;<HW\_ID>;<APP\_TYPE>;<USER\_NAME>;RT; RT\_WARNING;Network status is: WiFi

8/17/2015 12:45:45;<HW\_ID>;<APP\_TYPE>;<USER\_NAME>;RT; RT\_WARNING;Error alert shown

……

Logs analysis:

1. User opens voicemails list (VoicemailViewController)
2. User opens voicemail details (VoicemailDetailsViewController)
3. User presses “Play” button to listen voicemail selected
   1. Voicemail Web request is sent successfully
   2. Web request is failed with 408 error (request timeout)
   3. Network status is – Reachable, WiFi

Analysis result: VM Web request is failed possibly due to

1. Poor WiFi signal/Firewall restrictions
2. VM server issues

Action items: To check VM server logs according to appropriate timestamp

Example 2: User tries to listen voicemail, but application is closed unexpectedly.

Logs collected:

…..

8/17/2015 12:45:33;<HW\_ID>;<APP\_TYPE>;<USER\_NAME>;UI;UI\_VIEW;VoicemailViewController

8/17/2015 12:45:34;<HW\_ID>;<APP\_TYPE>;<USER\_NAME>;UI;UI\_VIEW;VoicemailDetailsViewController

8/17/2015 12:45:35;<HW\_ID>;<APP\_TYPE>;<USER\_NAME>;UI;UI\_TAP;Play button tapped

8/17/2015 12:45:35;<HW\_ID>;<APP\_TYPE>;<USER\_NAME>;RT; RT\_WEB\_REQUEST;Type=GET URL=http://my.ooma.com/v2/vm?id=…. Error=200 Ok

8/17/2015 12:45:45;<HW\_ID>;<APP\_TYPE>;<USER\_NAME>;RT; APP\_LMEM;Low memory!

8/17/2015 12:45:45;<HW\_ID>;<APP\_TYPE>;<USER\_NAME>;RT; APP\_CRASH; Type=SIGKILL Stack=….

……

Logs analysis:

1. User opens voicemails list (VoicemailViewController)
2. User opens voicemail details (VoicemailDetailsViewController)
3. User presses “Play” button to listen voicemail selected
   1. Voicemail Web request is sent successfully
   2. iOS kills application due to low memory issue

Action items: To check application for memory leaks

## 10.7 CSLMessage description

The following data have to be included to CSLMessage for future logs sorting, analysis and issues debugging:

* **<TIME\_STAMP>** - GMT date and time of event occurrence
* **<HW\_ID>** - Uniquely user HW identify used to associate logs with a specific device.
* **<HW\_INFO>** - HW device model name (iPhone4/5/6, iPad)
* **<OS\_NAME>** - Operation system name – iOS/Android
* **<OS\_VERSION>** - Operation system version (7.1.1, 8.0, 9.0 etc)
* **<APP\_TYPE>** - Application type (Home/Office) to associate logs with specific application type.
* **<APP\_VERSION>** - Application build version (5.1.9 (65789), 1.8 (123456) etc)
* **<USER\_NAME>** - Username (Ooma account) to associate logs with specific user.
* **<SEVERITY\_LEVEL >**
  + **ERROR –** high severity message requires attention
  + **WARNING –** moderate severity message
  + **INFO –**  runtime message
  + **DEBUG –** low level debug message
* **<MESSAGE\_TEXT>** - Text string

# 11. Future plans

## 11.1 UI implementation

## 11.2 Reports implementation

# 12. Testing considerations

API

Normal work

Speed

Loosing connection

Error conditions

# 13. Additional info

## 13.1 Perforce

All latest versions of the source code can be found here: //internal/software/mobile/ServerSide/mobilelogs

MA HLD: //internal/software/mobile/ios/docs/res\_new\_ui\_hld/Ooma\_Res\_New\_UI\_HLD.docx

CSL Req Document: //internal/software/mobile/ServerSide/mobilelogs/doc/CSL requirements 04.docx

# 14. Server Information

## 14.1 Disk

/root – 5.8 GB ext4

/var – 100 GB ext4

/db  - 11TB xfs

## 14.2 Memory

RAM – 32GB

Swap – 16 GB

## 14.3 CPU

2 processors, 4 core each - Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz

## 14.4 Software versions

Linux - Scientific Linux release 6.7 (Carbon)

Nginx - v1.9.2

Node.js & npm -  v0.10.33

MongoDB - 3.2

Perforce - v1319959

## 14.5 Server addresses and ports

Internal – mobilelogs-new.corp.ooma.com

External – mobilelogs.ooma.com (443 and 80 ports opened to public internet)

IP address - 10.66.12.79

Server API path - <https://mobilelogs.ooma.com/api/v1>

MongoDB address - mobilelogs-new.corp.ooma.com

MongoDB port – 27017

App backend ports – 8000-8003

# 15. Appendix

## 15.1 Mapping Dictionary

Below is the mapping dictionary of the label fields stored in event data array. Using this dictionary, label names can be decoded. For example: “z": 8 will be decoded into “z”: 'Old network status'.

    0:'Network status',

    1:'Free memory',

    2:'Allocated memory',

    3:'Remote notification text',

    4:'Local notification text',

    5:'Crash name',

    6:'Crash reason',

    7:'Crash data',

    8:'Old network status',

    9:'New network status',

    10:'Screen name',

    11:'Control name',

    12:'Alert title',

    13:'Alert text',

    14:'SIP initialization result',

    15:'SIP de-initialization result',

    16:'SIP registration status',

    17:'Error code',

    18:'SIP call status',

    19:'Remote number',

    20:'Anonymous call',

    21:'Media state',

    22:'Media type',

    23:'Media codec info',

    24:'Transport state',

    25:'Transport status',

    26:'Transport hostname',

    27:'API server',

    28:'API path',

    29:'Request type',

    30:'Request data',

    31:'Response error',

    32:'Response data',

    33:'Warning message',

    34:'Login result',

    35:'SIP registration result',

    36:'Media packet loss',

    37:'Media packet reorder',

    38:'Media packet discarded',

    39:'Media packet duplicated',

    40:'Current battery level',

    41:'Error message',

    42:'Parameter Info',

    43:'Call duration',

    44:'Call end status',

    45:'Call direction',

    46:'Call\_id',

    47:'Call end information',

    48:'Size',

    49:'Address',

    50:'Server type',

    51:'Hardcoded\_ip'

52:’Carrier name'

## 15.2 Event Name and Event Type Mapping

Event type and event name mapping are saved in one value (for example: "t": 30011):

Type is a rounded number of value divided by 10000

0:'SYSTEM',

1:'UI',

2:'SIP',

3:'RUNTIME'

Name is a value specified in a table below:

    0:'APP\_START',

    1:'APP\_BG',

    2:'APP\_FG',

    3:'APP\_INACTIVE',

    4:'APP\_ACTIVE',

    5:'APP\_LMEM',

    6:'APP\_RNOTIFY',

    7:'APP\_LNOTIFY',

    8:'APP\_CRASH',

    9:'NET\_CHANGE',

    10:'APP\_KEEPALIVE',

    11:'APP\_AUDIO\_INT\_START',

    12:'APP\_AUDIO\_INT\_STOP',

    13:'APP\_STOP',

    14:'APP\_SIZE',

    10000:'UI\_VIEW',

    10001:'UI\_EVENT',

    10002:'UI\_ALERT',

    20000:'SIP\_INIT',

    20001:'SIP\_DESTROY',

    20002:'SIP\_REG\_STATE',

    20003:'SIP\_CALL\_STATE',

    20004:'SIP\_MEDIA\_STATE',

    20005:'SIP\_TRANSPORT\_STATE',

    30000:'RT\_WEB\_REQUEST',

    30001:'RT\_WEB\_RESPONSE',

    30002:'RT\_WARNING',

    30003:'RT\_LOGIN\_START',

    30004:'RT\_LOGIN\_RESULT',

    30005:'RT\_SIP\_REG\_START',

    30006:'RT\_SIP\_REG\_RESULT',

    30007:'RT\_OCALL\_START',

    30008:'RT\_ICALL\_START',

    30009:'RT\_CALL\_ANSWER',

    30010:'RT\_CALL\_END',

    30011:'RT\_BATTERY\_LVL',

    30012:'RT\_ERROR',

    30013:'RT\_CHANGE\_SERVER\_ADDRESS',

    30014:'RT\_FIVE\_UNSUCCESSFUL\_LOGIN\_ATTEMPTS'