Engineering Center
2232 Northmont Parkway
Duluth GA 30096
USA

Original Date:	12/14/2009	Document:	WavePro_ICD_OA.doc
Project Number:	2036	Sheet:	Page 1 of 31
Dwg. /Part Number /Id:	N/A		

Interface Control Document (ICD)

WavePro

DISTRIBUTION: Per document OPDP0003

ORIGINAL APPROVAL:

Prepared By: (Last, F I typed)	(Signature and date)	Approved By: (Last, F I typed)	(Signature and date)
Mavila, Lakshmi	*Mavila, L (2009/12/14)*	Bacellar, Luiz	*Bacellar, L (2009/12/16)*

REVISION APPROVAL RECORD:

Rev #	Rev Date (yyyy/mm/dd)	Revised By: (Last, F I typed)	Approved By: (Last, F I typed)	(Signature and date)
1	2010/03/03	Leyva, S.	Vahalia, K	*Kekti Vahalia (2010-03-10)*
1				
2				
3				

(Revision history on second page)

REVISION HISTORY:

Rev #	Rev Date (yyyy/mm/dd)	Revised By: (Last, F I typed)	Changes
1	2010/03/03	Leyva, S.	Separated the PDA from the Wireless ICD. Updated information surrounding a WLM firmware upgrade, including adding EVENTs, removing REQUESTs, and updating Appendix B.

Table of Contents

List of Tables

List of Figures

1 Introduction

1.1 Purpose

This document describes the application protocol for the WavePro system. This includes communication between

- Lock Management System and the wireless access point

1.2 Scope

This document assumes that readers are familiar with the functionalities of locking systems in general. The transport protocols between LMS and the wireless access point and the wireless access point and the WavePro lock are out of scope of this document. These are covered in other documents as specified in the references section.

1.3 Definitions, Acronyms and Abbreviations

Term / Abbreviation	Explanation / Description / Definition
CM	Control Module
ILS	Intelligent Lock System
IP	Internet Protocol
LAN	Local Area Network
LMS	Lock Management System
ACU	Access Control Unit (Lock)
NA	Not Applicable
ORP	Onity Reader Protocol
PSR	Packet Success Rate (measures % of successfully transmitted packets)
RSSI	Received Signal Strength Indicator
SSN	Secure Sensor Network (proprietary 900 MHz wireless network)
TCP	Transport Control Protocol
UTCFS	United Technologies Corporation Fire & Security
WAP	Wireless Access Point
WLM	Wireless Lock Module - refers to the radio module on the lock
WMC	WAP Main Controller
WWM	WAP Wireless Module - refers to the radio module on the WAP

Table 1: Definitions, Acronyms and Abbreviations

1.4 References

Ref	Number	Title	Version
A	ORP_OA	ORP Document	B

Table 2: References

2 System Overview

The WavePro locks will be managed through LMS running on a Windows PC. The communication path from LMS to the lock will be through a Wireless access point. The Wireless access point may connect to the LMS through either a WiFi-G network or through wired LAN and communicate over TCP/IP.

The wireless access point connects with the WavePro locks using proprietary protocol over the 915MHZ band. The system overview is depicted in **Error! Reference source not found.**.

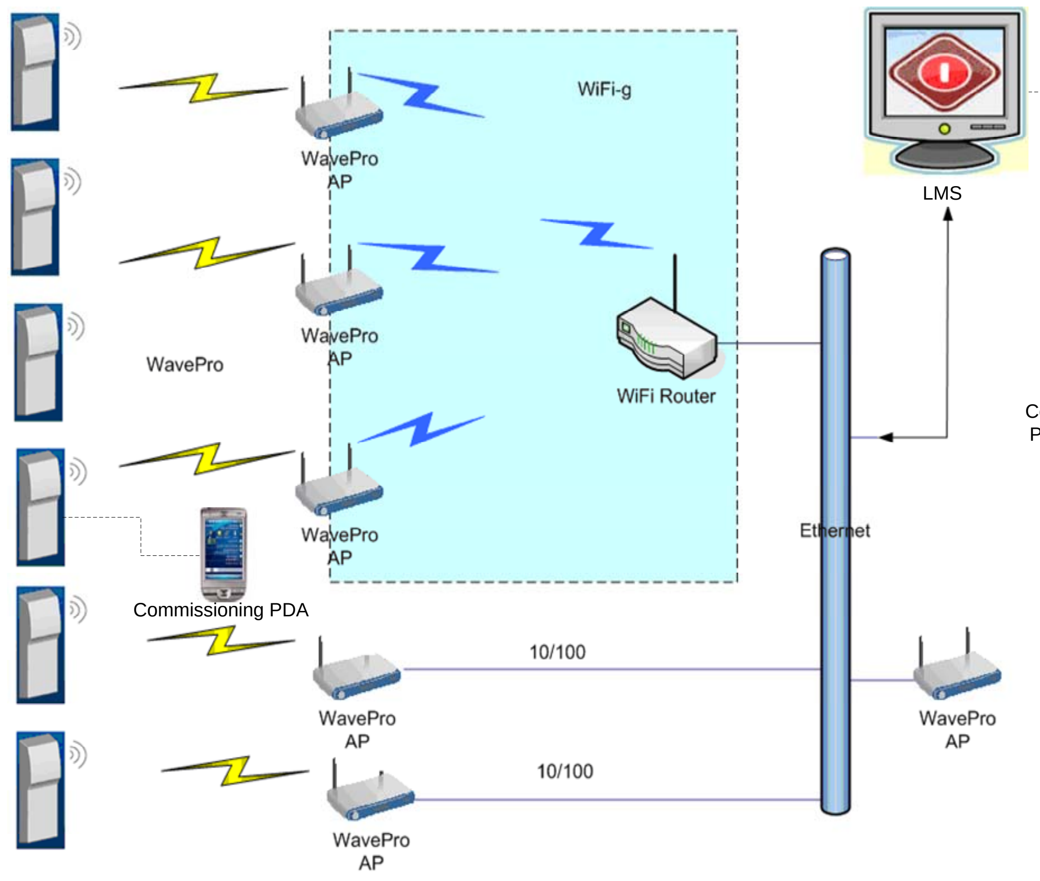

Figure 1: System Overview

The functional components of the system are depicted in Figure 2. Within the WAP and the WavePro lock, there are two micro-controllers – one main controller and the radio module micro controller. This adds additional hops for data that is exchanged between LMS and the WavePro lock main controller.

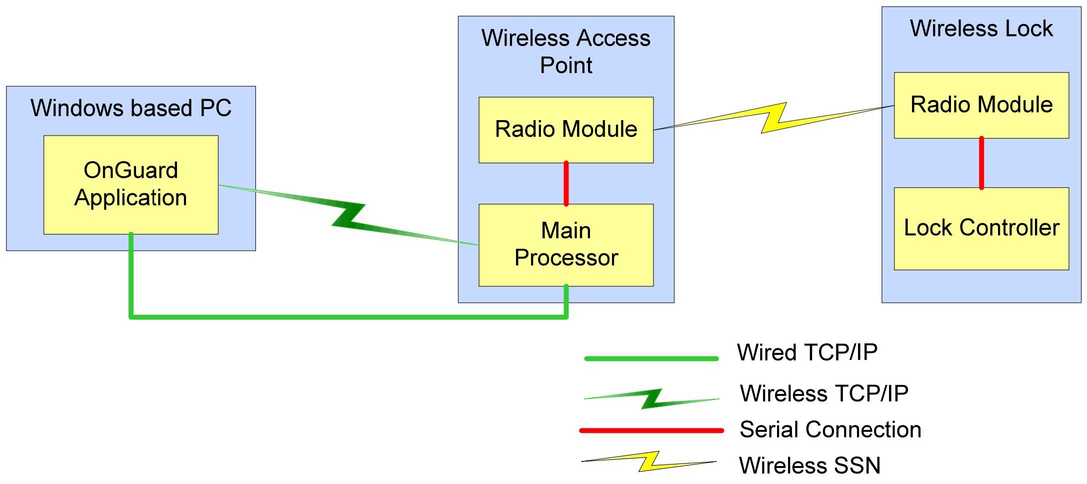

Figure 2: Functional Overview

3 Interface Requirements

3.1 Protocol Requirements

The protocol should allow data retrieval or data transmission, in a request / response format. One device shall send the request to “read” the data and the target device shall respond with the appropriate data.

The transport mechanism is TCP/IP and if the data size is larger than one TCP/IP packet size, then the protocol shall allow for devices to segment the data before sending and reassemble it at the receiving end.

The protocol shall allow asynchronous transmission of information, in case of events.

The protocol shall protect the privacy as well as the integrity of the data exchanged.

The protocol shall provide authentication of the source and target devices.

3.2 Broadcast

The protocol needs to support broadcast of commands to locks.

3.3 WAP Device

The WAP needs to be a low cost embedded device with limited resources. Hence it is desirable to keep it simple.

3.4 Communication between LMS and Lock

The communication between LMS and the lock is accomplished through WAP.

The lock provides its functionality through the ORP protocol [Reference **Error! Reference source not found.**]. The ORP protocol will be implemented at the application level between LMS and the lock controller.

LMS will embed the ORP packets as required by the lock CM in a TCP/IP packet to the WAP. The WAP will forward this to the lock via the WLM. The lock will send an ORP response, which the WLM forwards to the WAP over the wireless link in a SSN packet. The WAP in turn, forwards the lock ORP response to LMS in a TCP/IP packet. If a certain lock command involves multiple ORP packets, each ORP packet will flow between LMS and the ACU. This data flow is depicted in Figure 3.

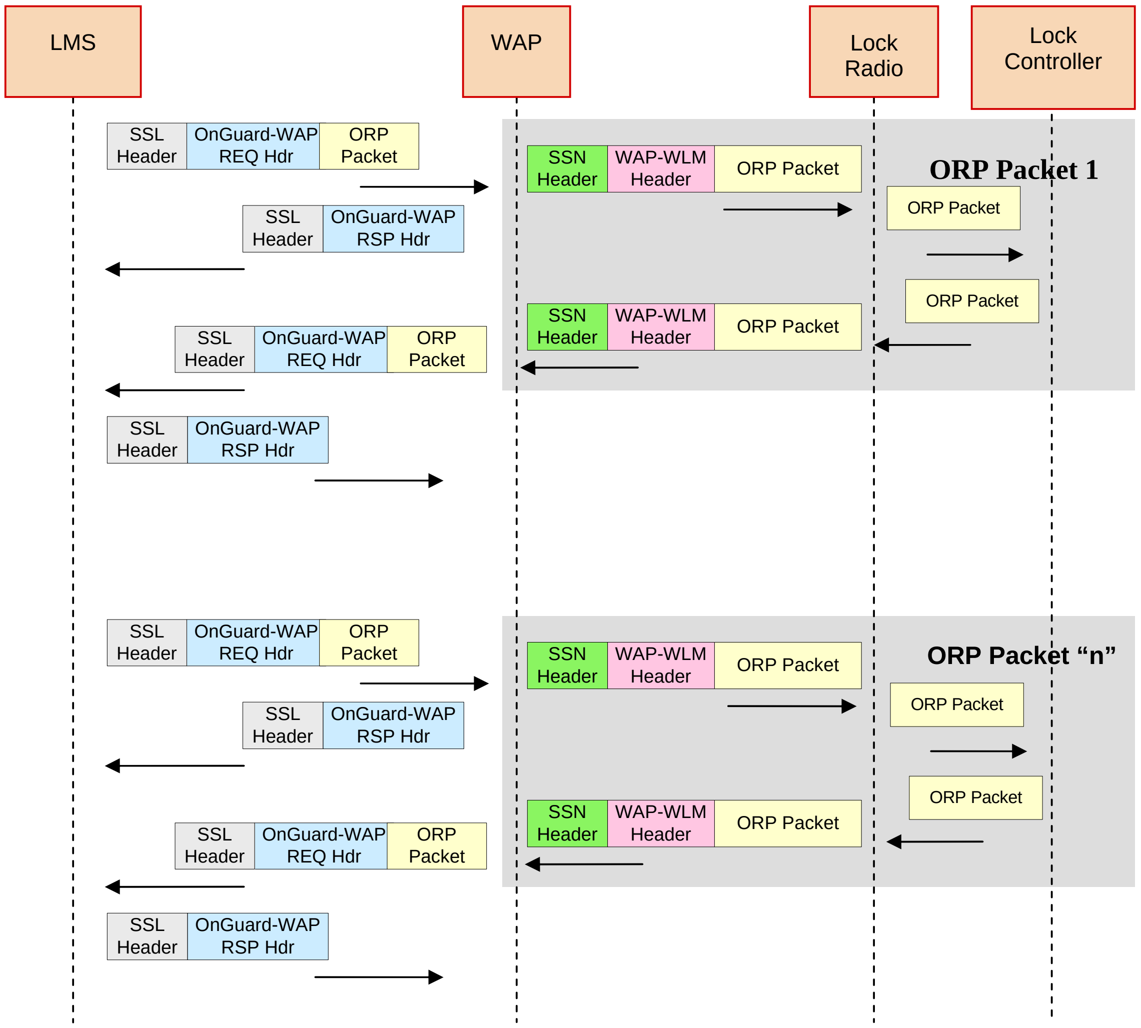

Figure 3: LMS to Lock Command Execution

When the CM needs to send data (response to an ORP command or an asynchronous event) to the LMS, it sends the ORP packets to the WLM. The WLM forwards them to the WAP, which in turn forwards them to the LMS. The LMS will then send ORP response to the lock via the WAP. This data flow is depicted in Figure 4.

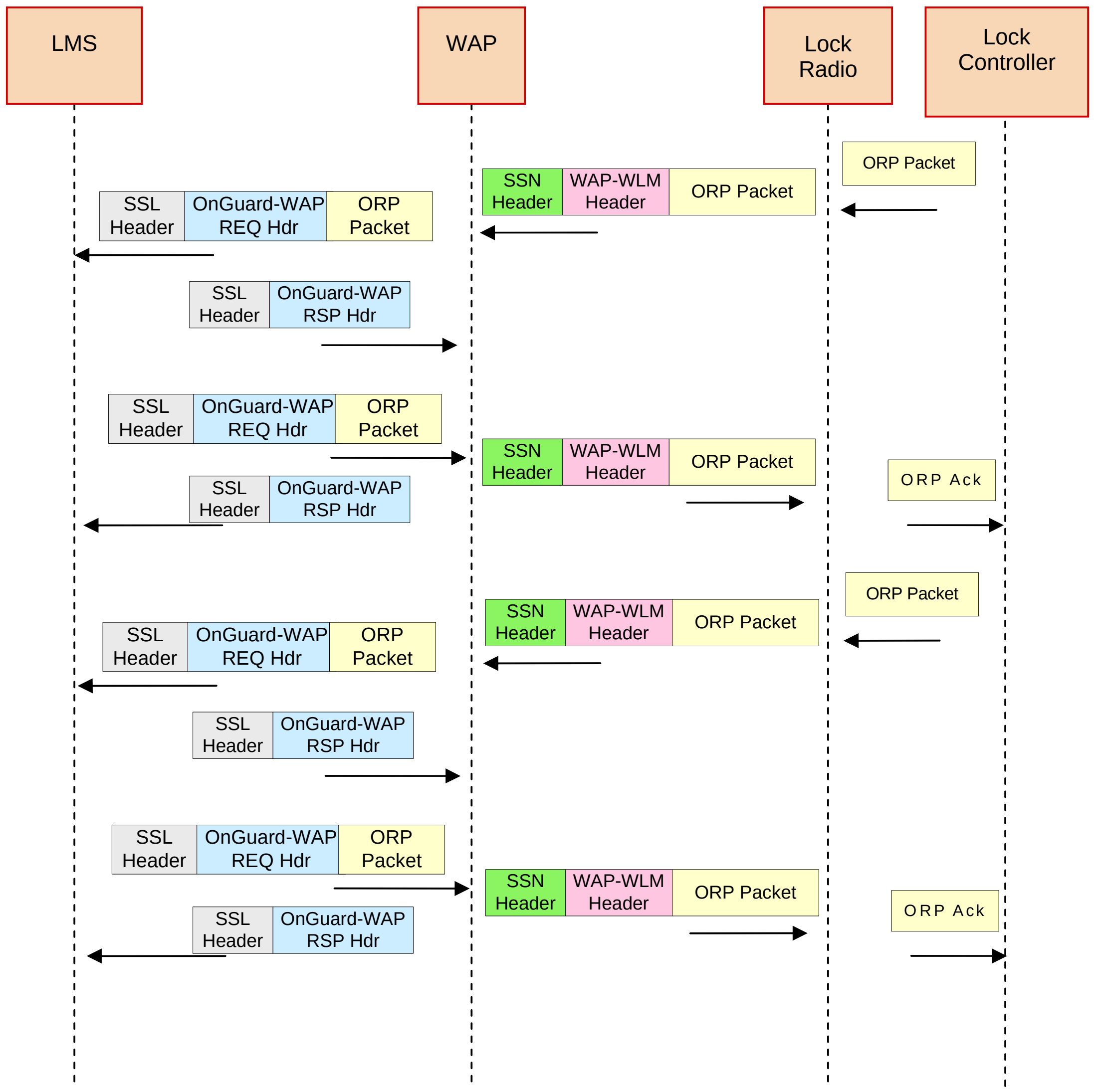

Figure 4: Lock to LMS Data Transfer

As is visible from the above flow diagrams, the ORP messages are interpreted only at LMS and Control Module – the WAP and the WLM have no knowledge of ORP. This scheme implies the following –

- It is possible to broadcast single ORP packet command from LMS to several locks connected to a WAP. Each lock will respond to the command separately and LMS will receive individual responses from each lock.

- It is not possible to broadcast lock commands comprised of multiple ORP packets. This is because the WAP does not cache the entire lock command (multiple ORP packets). LMS will need to send the command to one lock at a time.

3.5 Communication between WAP and Lock

The communication between the WAP and the locks is asynchronous. Only the lock can initiate the communication between the WAP and the lock. The lock sends periodic heartbeats to the WAP. On receiving the heartbeat, the WAP can inform the lock to stay awake for further exchange of data.

4 LMS – WAP Interface

The basic functionality of the WAP is to provide a communication link between LMS and the lock and allow the exchange of data between the two. Additionally the WAP allows reading and writing of WAP configuration from LMS, upgrading WAP firmware from LMS and maintaining lock communication status.

4.1 Connection Management

LMS and WAP shall communicate over TCP/IP, either over Ethernet or over Wi-Fi network. The communication protocol shall be based over SSL / TLS. The WAP shall host a SSL / TLS server listening on port 8008 by default. LMS shall connect to the WAP using a SSL / TLS client.

The SSL / TLS protocol provides -

- Encryption, authentication and integrity of data.
- Segmentation and reassembly of data

In order for the WAP to communicate with SSL enabled, it must have three certificates in PEM format:

1. HostRoot.pem - The Root CA certificate (public only)
2. Svrcert.pem - The WAP certificate (with both public and private keys)
3. WAPClient.pem - The Host certificate (public only)

The host vendor must provide these certificates to the WAP.

The format of the data that will be exchanged between LMS and the WAP over the SSL connection is detailed in the following sections.

4.2 Application Protocol Definition

The application protocol will support three types of packets:

REQUEST – Originated by a device requesting data or sending a command

EVENT – Originated by a device sending asynchronous event information (e.g., battery low event)

RESPONSE – Sent by the target device in response to a REQUEST or EVENT

The protocol header will identify the type of packet. The RESPONSE packet is sent only in response to a REQUEST or EVENT packet.

4.3 REQUEST Packet

A REQUEST packet shall consist of a Header Block that may be immediately followed by an optional Data block, which is formatted in accordance with the specifics identified in the Header Block.

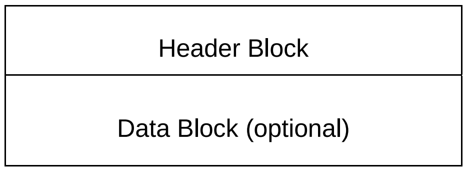

Figure 5: REQUEST Packet

4.3.1 REQUEST Header Block

The Header Block shall be formatted as shown in Table 3: **LMS-WAP REQUEST Header Block**

Offset	Field	Length (in bytes)
0	Version	2
2	FrameType	2
4	MessageID	2
6	SourceType	2
8	SourceID	4
12	CommandID	2
14	TargetType	2
16	TargetID	4
20	DataSize	4

Table 3: LMS-WAP REQUEST Header Block

The header fields shall be in big endian format.

4.3.1.1 REQUEST Header Block: Version

The version is a 2 bytes field and its value shall be 0x0001 for this definition of the interface protocol. If the definition of the interface changes in future, the version shall change as well.

4.3.1.2 REQUEST Header Block: FrameType

The FrameType shall be of type REQUEST. The possible values of FrameType are defined as follows.

Type	Value
FTYPE_REQUEST	0x0001
FTYPE_EVENT	0x0002
FTYPE_RESPONSE	0x8000

Table 4: FrameType Values

4.3.1.3 REQUEST Header Block: MessageID

The MessageID is a random 2 byte number that allows the requester to match a response to a request. The RESPONSE packet shall have the same MessageID as the corresponding REQUEST packet.

Note that it is the requester’s responsibility to make sure that two simultaneous requests don’t have the same MessageID.

4.3.1.4 REQUEST Header Block: SourceType

The SourceType defines the source of the command. The possible values are defined in Table 5: REQUEST **Header Block – SourceType Values**

Type	Value
MGMT_SYSTEM	0x0001
ACCESS_POINT	0x0002
LOCK	0x0004

Table 5: REQUEST Header Block – SourceType Values

4.3.1.5 REQUEST Header Block: SourceID

The SourceID identifies the source device of type "SourceType".

When the SourceType is MGMT_SYSTEM, the SourceID is 0 (zero) and it identifies the LMS system.

When the SourceType is ACCESS_POINT, the SourceID is the WAP ID.

When the SourceType is LOCK, the SourceID is the lock ID, identifying one particular lock.

4.3.1.6 REQUEST Header Block: CommandID

The CommandID is a 2 byte field and it defines the commands supported between LMS and WAP. The possible values for CommandID are enumerated in Table 6: **REQUEST Header Block – CommandID Values**

CommandID	Value	Description
WAP_READ_CONFIG	0x0001	Read configuration of the WAP
WAP_WRITE_CONFIG	0x0002	Modify configuration of the WAP
WAP_READ_LOCK_STATUS	0x0003	Read lock status
WAP_UPGRADE_FW	0x0004	Upgrade the firmware on the WAP main controller
WAP_UPGRADE_RM_FW	0x0005	Upgrade the firmware on the WAP radio module
WAP_REBOOT	0x0006	Reboot the WAP
WAP_READ_STATUS	0x0007	Get the WAP status information
WAP_HEARTBEAT	0x0008	Send periodic information from the WAP
WAP_SET_CURRENT_TIME	0x0009	Set the time at LMS on the WAP
TO_LOCK	0x1001	Send enclosed ORP packets to the lock
FROM_LOCK	0x1002	Forward ORP packets received from the lock
LOCK_UPGRADE_RM_FW	0x1003	Upgrade the firmware on the WLM
LOCK_ENTER_SLEEP_MODE	0x1005	Inform the lock that it can go back to sleep

Table 6: REQUEST Header Block – CommandID Values

4.3.1.7 REQUEST Header Block: TargetType

The TargetType defines the destination of the command. All the possible values for SourceType, as defined in Table 7: **REQUEST Header Block – TargetType Values**, are applicable to TargetType as well. An additional possible value for TargetType is LOCK_MULTICAST. All possible TargetType values are defined in Table 7: **REQUEST Header Block – TargetType Values**

Type	**Value**
MGMT_SYSTEM	0x0001
ACCESS_POINT	0x0002
LOCK	0x0004
LOCK_MULTICAST	0x0008

Table 7: REQUEST Header Block – TargetType Values

A TargetType of LOCK_MULTICAST indicates that the CommandID applies to more than a single lock connected to the WAP. A combination of the Lock ID and the Data Block of the packet shall contain details about which locks are the targets for the CommandID.

4.3.1.8 REQUEST Header Block: TargetID

The TargetID identifies the destination device of “destination type”.

When the TargetType is MGMT_SYSTEM, the TargetID is 0 (zero) and it identifies the LMS system.

When the TargetType is ACCESS_POINT, the TargetID is the WAP ID.

When the TargetType is LOCK, the TargetID is the Lock ID, identifying one particular lock.

When the TargetType is LOCK_MULTICAST, and the TargetID of 0xFFFFFFFF, indicates that the CommandID applies to all locks connected to the WAP.

When the TargetType is LOCK_MULTICAST, the TargetID of 0 (zero) indicates that the CommandID applies to only certain locks connected to the WAP. The data portion of the packet will identify the locks.

4.3.1.9 REQUEST Header Block: DataSize

This is a 4 byte field and provides the length in bytes of the Data Block of the request packet.

4.3.2 REQUEST Data Block

The Data Block shall be formatted in accordance with the specifics identified in the Header Block. Generally, the CommandID shall define the Data Block format. The Data Block shall not be required for every CommandID. Acceptable responses to each of the given commands are covered in the RESPONSE Packet Section, Section 4.5.

When the TargetType is LOCK_MULTICAST and the TargetID is zero, the Data Block contains information about which locks are addressed for the given command. In this case, the Data Block shall be formatted as follows:

Offset	**Field**	**Length**
0	Number of Locks (“N”)	4
4	Lock ID 1	4
8	Lock ID 2	4
	:	
“N” * 4	Lock ID “N”	4

Table 8: Data Block for LOCK_MULTICAST

The actual data (if present) shall start after the above lock addressing information.

4.3.2.1 REQUEST Data Block: WAP Read Data

WAP_READ_CONFIG

WAP_READ_LOCK_STATUS

For each of the above CommandIDs, the Data Block shall not be present and the DataSize shall be zero.

4.3.2.2 REQUEST Data Block: WAP Write Configuration

WAP_WRITE_CONFIGURATION

The Data Block shall contain the WAP configuration information. The WAP configuration format is shown in Table 9: **WAP Configuration**

Offset	**Length**	**Field Name**	**Field Type**
0	2	WAP Configuration Version (should be 1)	Integer
2	2	Size in bytes of the configuration	Integer
4	4	WAP ID	Integer
8	4	Site ID	Integer
12	4	Number of missed lock heartbeats that causes communication failure notification sent to LMS	Integer
16	2	WWM Channel ID (1 to 26 – default is 1)	Integer
18	1	WWM Output Power (1 to 4 – default is 3)	Integer
19	1	WWM Frequency Agility (0 or 1 – default is 1 / ON)	Integer

20	4	WAP heartbeat frequency in seconds	Integer
24	4	Number of Locks (“N”)	Integer
28	4	Lock ID 1	Integer
32	4	Lock 1 heartbeat frequency (1 to 1440 minutes)	Integer
Offset	**Length**	**Field Name**	**Field Type**
36	4	Lock ID 2	Integer
40	4	Lock 2 heartbeat frequency (1 to 1440 minutes)	Integer
44	4	Lock ID 3	Integer
48	4	Lock 3 heartbeat frequency (1 to 1440 minutes)	Integer
		:	
20+(8*N)	4	Lock ID “N”	Integer
24+(8*N)	4	Lock “N” heartbeat frequency (1 to 1440 minutes)	Integer

Table 9: WAP Configuration

4.3.2.3 REQUEST Data Block: WAP Reboot

WAP_REBOOT

For the reboot command, the Data Block shall not be present and the DataSize shall be zero.

4.3.2.4 REQUEST Data Block: WAP Status

WAP_READ_STATUS

For the reboot command, the Data Block shall not be present and the DataSize shall be zero.

4.3.2.5 REQUEST Data Block: WAP Firmware Upgrade

WAP_UPGRADE_FW

WAP_UPGRADE_RM_FW

For the above commands, the Data Block shall contain the new WAP main controller firmware or the new WAP radio module firmware in binary format.

WAP_UPGRADE_FW_DONE

WAP_UPGRADE_RM_FW_DONE

For the above two commands, the Data Block shall contain the version string of the new firmware.

4.3.2.6 REQUEST Data Block: WAP Heartbeat

WAP_HEARTBEAT

The WAP Heartbeat will be originated by the WAP. The Data Block contains the current time on the WAP, the current operating channel, as well as any new lock diagnostic information. For information on how this command works please refer to Appendix C – WAP Heartbeat. The Data Block for this message can be found in Table 10.

Offset	Length	Field Name	Field Type	Expected Values
0	10	WAP Time	WAP Time Type	See Table 11
10	1	WAP Operating Channel	Unsigned Integer	1-26
11	2	Number of Locks Diagnostics (“N”)	Unsigned Integer	0-32
13	4	Lock 1 ID	Unsigned Integer	
17	13	Lock 1 Diagnostics	Lock Diagnostic Type	See Table 27
		:		
17*(N-1) +13	4	Lock “N” ID	Unsigned Integer	
17*(N-1) +17	13	Lock “N” Diagnostics	Lock Diagnostic Type	See Table 27

Table 10: WAP Heartbeat Format

4.3.2.7 REQUEST Data Block: WAP SET CURRENT TIME

WAP_SET_CURRENT_TIME

The Data Block shall contain the current time at the LMS. The format of the block is of WAP Time Type found below in Table 11.

Offset	Length	Field Name	Field Type	Expected Values
0	2	Year	Unsigned Integer	Current Year
2	1	Month	Unsigned Integer	1-12
3	1	Day	Unsigned Integer	1-31
4	1	Hour	Unsigned Integer	0-23
5	1	Minute	Unsigned Integer	0-59

6	1	Second	Unsigned Integer	0-59
7	1	Day of Week	Unsigned Integer	0-6 (0 being Sunday)
8	2	Milliseconds	Unsigned Integer	0

Table 11: WAP Time Type

4.3.2.8 REQUEST Data Block: Lock Commands

TO_LOCK

FROM_LOCK

Only one ORP packet can be forwarded by the WAP at a time. For all the above commands, the Data Block shall contain a single ORP packet. The DataSize field in the header will provide the length of the ORP packet.

The ORP Packet data shall be formatted in accordance with the format expected by the lock as described in the ORP document [Reference **Error! Reference source not found.**].

LOCK_UPGRADE_RM_FW

For the above command, the Data Block shall contain the WLM firmware in binary format.

LOCK_ENTER_SLEEP_MODE

For the above command, the Data Block shall not be present and the DataSize shall be zero. This command is used to indicate to lock that LMS is done transmitting to the lock and it can go back to sleep to conserve battery.

4.4 EVENT Packet

An EVENT message shall be sent when the WAP detects a condition that needs to be notified to LMS. An EVENT message shall consist of a Header Block that may be followed by an optional Data Block, which is formatted in accordance with the specifics identified in the Header Block.

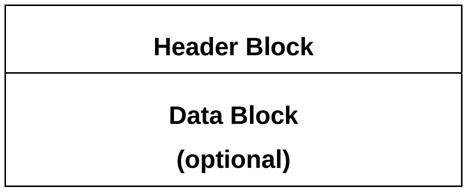

Figure 6: EVENT Packet

4.4.1 EVENT Header Block

The EVENT Header Block shall be formatted as defined in Section 4.3.1. The FrameType for a EVENT header block shall be of type FTYPE_EVENT.

The CommandID shall identify the unique event type. The acceptable CommandIDs are defined in Table 12: **EVENT Header Block - Command ID Values**

Table 12: EVENT Header Block - Command ID Values

4.4.2 EVENT Data Block

The Data Block shall be formatted in accordance with the specifics identified in the Header Block. Generally, the CommandID shall define the Data Block format. The Data Block shall not be required for every CommandID. Acceptable responses to each of the given commands are covered in Section 4.5

4.4.2.1 EVENT Data Block: Lock Communication Events

EV_LOCK_COMM_FAIL

EV_LOCK_COMM_RESTORED

For the above events the Data Block shall be formatted per Table 13.

Offset	Field Name	Length
0	Lock ID	4

Table 133: EVENT Data Block - Lock Communication Events

The lock ID shall define which lock lost / restored communication with the WAP.

4.4.2.2 EVENT Data Block: WLM Firmware Download Events

EV_WLM_FW_DOWNLOAD_START

This event sends an indication to the LMS that the WAP has begun transmitting the WLM firmware to the lock. For this event, the Data Block shall contain no payload. The source ID shall be that of the Lock that the firmware is presently being sent to.

EV_WLM_ FW_DOWNLOAD_DONE

The Data Block shall contain indication of whether the firmware upgrade was successfully completed or not. The Data Block shall contain a 2 byte integer, which can have one of the response result codes from Table 16: RESPONSE Header Block – Result

6. The format for this Data Block can be found in Table 144. The source ID shall be that of the Lock that the firmware is being sent to.

Offset	Field Name	Length
0	**Lock ID**	**4**

Table 144: EVENT Data Block - WLM Firmware Download Done Event

4.4.2.3 EVENT Data Block: WAP Events

EV_WAP_POWER_UP

This event sends an indication to LMS that the WAP just powered up. The WAP may have gone through an unexpected reboot, or may have experienced a short power failure. For this event, the Data Block size shall be zero.

4.5 RESPONSE Packet

A RESPONSE message shall be required for each REQUEST and EVENT message.

A RESPONSE packet shall consist of a Header Block that may be immediately followed by an optional Data block, which is formatted in accordance with the specifics identified in the Header Block.

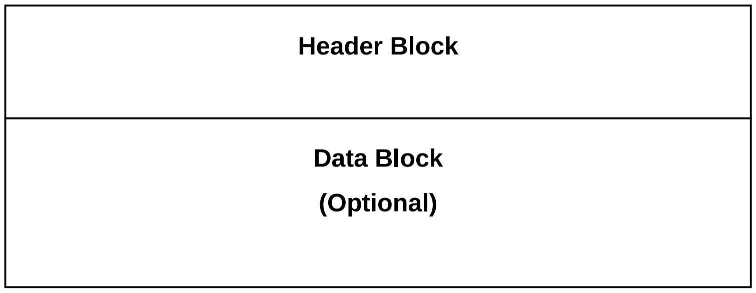

Figure 7: RESPONSE Packet

4.5.1 RESPONSE Header Block

The Header Block for RESPONSE packets shall be formatted as follows:

Offset	Field Name	Length
0	Version	2
2	FrameType	2
4	MessageID	2
6	SourceType	2
8	SourceID	4
12	CommandID	2
14	TargetType	2
16	TargetID	4
20	DataSize	4
24	Result	2
26	Reserved	2

Table 15: RESPONSE Header Block

The header fields shall be in big endian format.

4.5.1.1 RESPONSE Header Block: Version

The Version shall be a 2 byte field and its value shall be 0x0001 for this definition of the protocol.

4.5.1.2 RESPONSE Header Block: FrameType

The FrameType shall be of type FTYPE_RESPONSE.

4.5.1.3 RESPONSE Header Block: MessageID

The RESPONSE packet shall have the same MessageID as the corresponding REQUEST or EVENT packet.

4.5.1.4 RESPONSE Header Block: SourceType

The SourceType field shall reflect the responding device's type. This may simply be a copy of the TargetType field as specified in the REQUEST Header Block, but shall reflect the responding device's specific type if the Header Block specified multiple types. Acceptable types shall be any (one) of the values identified in the TargetType table, as specified in Section 4.3.1.7 REQUEST Header Block: TargetType, except LOCK_MULTICAST.RESPONSE Header Block: SourceID

The SourceID field shall reflect the responding device's ID. This may simply be a copy of the TargetID field as specified in the REQUEST Header Block, but shall reflect the responding device's specific ID if the REQUEST Header Block specified the LOCK_MULTICAST.

4.5.1.5 RESPONSE Header Block: SourceID

The SourceID field shall reflect the responding device's ID. This may simply be a copy of the TargetID field as specified in the REQUEST Header Block, but shall reflect the responding device's specific ID if the REQUEST Header Block specified the LOCK_MULTICAST

4.5.1.6 RESPONSE Header Block: CommandID

The CommandID field value shall be a direct copy of the CommandID field value as received in the REQUEST Header Block.

4.5.1.7 RESPONSE Header Block: TargetType

The TargetType field shall reflect the target type of the responding device. This shall be simply a copy of the SourceType field as specified in the REQUEST Header Block.

4.5.1.8 RESPONSE Header Block: TargetID

The TargetID field shall reflect the ID of the responding device's target. This shall simply be a copy of the SourceID field as specified in the REQUEST Header Block.

4.5.1.9 RESPONSE Header Block: DataSize

The DataSize field shall identify the size of the Data Block (in bytes). Note that this value may be 0. The sections below identify which CommandIDs shall result in a 0-length Data block.

4.5.1.10 RESPONSE Header Block: Result

The Result field shall indicate the responder's ability to respond to the particular request. Acceptable values shall be 0 (indicating success) or non-zero (indicating some sort of failure to comply with the request). No Data Block is required or expected if the Result field is non-zero (DataSize value in this case should be 0). The possible values for the Result field are defined in Table 16: **RESPONSE Header Block – Result**

Result	Description
0x0000	Request successfully processed
0x0001	Version not supported
0x0002	Invalid TargetID
0x0003	Invalid TargetType
0x0004	CommandID not supported
0x0005	Invalid request
0x0006	Internal error
0x0007	Resource error
0x0008	Not supported
0x0009	Packet is invalid
0x000A	Lock communication error
0x000B	A command is pending
0x000C	Data portion is invalid
0x000D	Error in WLM

Table 16: RESPONSE Header Block – Result

In response to a non-zero Result field, the application that issued the REQUEST could choose to retry the original command, but it is unlikely that the target will respond any differently. In most cases, the requester should have an appropriate exception handler in place for receipt of a non-zero Result (one that includes an appropriate, translatable error message if the application supports a UI).

4.5.2 RESPONSE Data Block

The Data Block shall be formatted in accordance with the specifics identified in the Header Block. Generally, the CommandID shall define the Data Block format. The Data Block shall not be required for every CommandID.

4.5.2.1 RESPONSE Data Block: Read Configuration

The Data Block shall contain the WAP configuration information. The WAP configuration format is described in Table 9: WAP Configuration

.

4.5.2.2 RESPONSE Data Block: Read WAP Status

The Data Block shall contain the WAP status information. The WAP status format is shown in Table 17: WAP **Status Structure**

Offset	Length	Field Name	Field Type
0	2	WAP Status Version (should be 1)	Integer
2	2	WAP SSL Port	Integer
4	2	WAP Type (Ethernet or WiFi)	Integer
6	2	DHCP Enabled	Integer
8	4	WAP Uptime in seconds	Integer
12	4	WAP IP Address	Integer
16	4	WAP Network Mask	Integer
20	4	Gateway IP Address	Integer
24	2	WAP DNS Name Len (dns_len)	Integer
26	2	WMC Version String Length (wmc_len)	Integer
28	2	WWM Version String Length (wwm_len)	Integer
30	Variable (dns_len)	WAP DNS Name	String
28+dns_len	Variable (wmc_len)	WMC Firmware Version	String
28+dns_len+wmc_vlen	Variable (wwm_len)	WWM Firmware Version	String

Table 17: WAP Status Structure

The WAP Type field can have one of the following values:

Description	Value
Ethernet WAP	0x0001
WiFi WAP	0x0002

Table 18: WAP Type Field

4.5.2.3 RESPONSE Data Block: Read Lock Status

The Data Block shall contain the status of the locks connected to the WAP. The status information shall be formatted as defined in Table 19: Lock Status Information

Offset	Length	Field Name	Field Type
0	2	Version (Should be 1)	Integer

2	2	Number of Locks (“N”)	Integer
Offset	**Length**	**Field Name**	**Field Type**
4	4	Lock 1 ID	Integer
8	4	Lock 1 Status	Integer
12	4	Lock 1 ID	Integer
16	4	Lock 1 Status	Integer
		:	Integer
		Lock “N” ID	
	4	Lock “N” Status	Integer

Table 19: Lock Status Information

The possible values for the lock status field are below:

Result	**Description**
0x0000	No communication with lock
0x0001	Active communication with lock

Table 20: Lock Status Values

4.5.2.4 RESPONSE Data Block: Write Configuration

For this command, the Data Block shall not be present and the DataSize shall be zero.

4.5.2.5 RESPONSE Data Block: Reboot

For this command, the Data Block shall not be present and the DataSize shall be zero.

4.5.2.6 RESPONSE Data Block: WAP Firmware Upgrade

WAP_UPGRADE_FW

For this command, the Data Block shall not be present and the DataSize shall be zero.

4.5.2.7 RESPONSE Data Block: WAP Radio Module Firmware Upgrade

WAP_UPGRADE_RM_FW

For this command, the Data Block shall not be present and the DataSize shall be zero.

4.5.2.8 RESPONSE Data Block: WAP Heartbeat

WAP_HEARTBEAT

For this command, the Data Block shall not be present and the DataSize shall be zero.

4.5.2.9 RESPONSE Data Block: WAP SET CURRENT TIME

WAP_SET_CURRENT_TIME

For this command, the Data Block shall not be present and the DataSize shall be zero.

4.5.2.10 RESPONSE Data Block: Lock Commands

For all the above commands, the Data Block shall not be present and the DataSize shall be zero.

4.6 Failure Conditions

If a RESPONSE packet is not received for a REQUEST or EVENT packet within a certain time period, the initiator may retry the REQUEST or the EVENT.

The WAP shall not remember the commands across reboots. If a WAP goes through an unplanned reboot while some commands are in progress, those commands should timeout and may be retried by LMS.

For lock related requests, the RESPONSE from WAP only indicates a successful receipt of the REQUEST packet. LMS shall wait for the next REQUEST from WAP with CommandID of FROM_LOCK. This is the actual response from the lock.

Appendix A – Packet Format

For the LMS to WAP interface, when the TargetType is LOCK_MULTICAST and the TargetID is 0, the application packet is formatted as shown in Figure 9: LMS Packet (TargetType=LOCK_MULTICAST, TargetID=0).

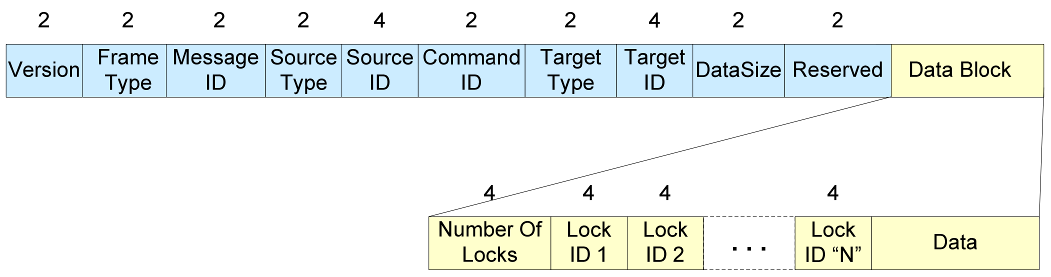

Figure 8: LMS - WAP Packet (TargetType=LOCK_MULTICAST, TargetID=0)

For all other TargetType and TargetID values, the application packet is formatted as shown in

.

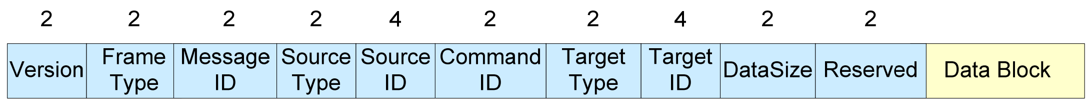

Figure 9: LMS - WAP Packet

Appendix B – WLM Firmware Upgrade

Firmware upgrade on the WLM from LMS is performed as follows:

- LMS sends the entire firmware to the WAP in a REQUEST and awaits a RESPONSE from the WAP indicating success or failure of receiving the firmware.
- The WAP caches the WLM firmware, awaiting the specified lock's next heartbeat
- Upon receiving heartbeat, WAP sends an EVENT to LMS indicating firmware download has started.
- WAP proceeds to send the cached firmware one, smaller, fragmented, packet to the WLM at a time until the entire firmware has been sent, and each packet acknowledged. The WLM will attempt to write the entire firmware to flash and validate the checksum, by sending a positive or negative indication to WAP.
- WAP sends an EVENT back to LMS indicating the result of the download.

The sequence of communication between LMS, WAP and WLM for WLM firmware upgrade functionality is depicted in **Error! Reference source not found.**.

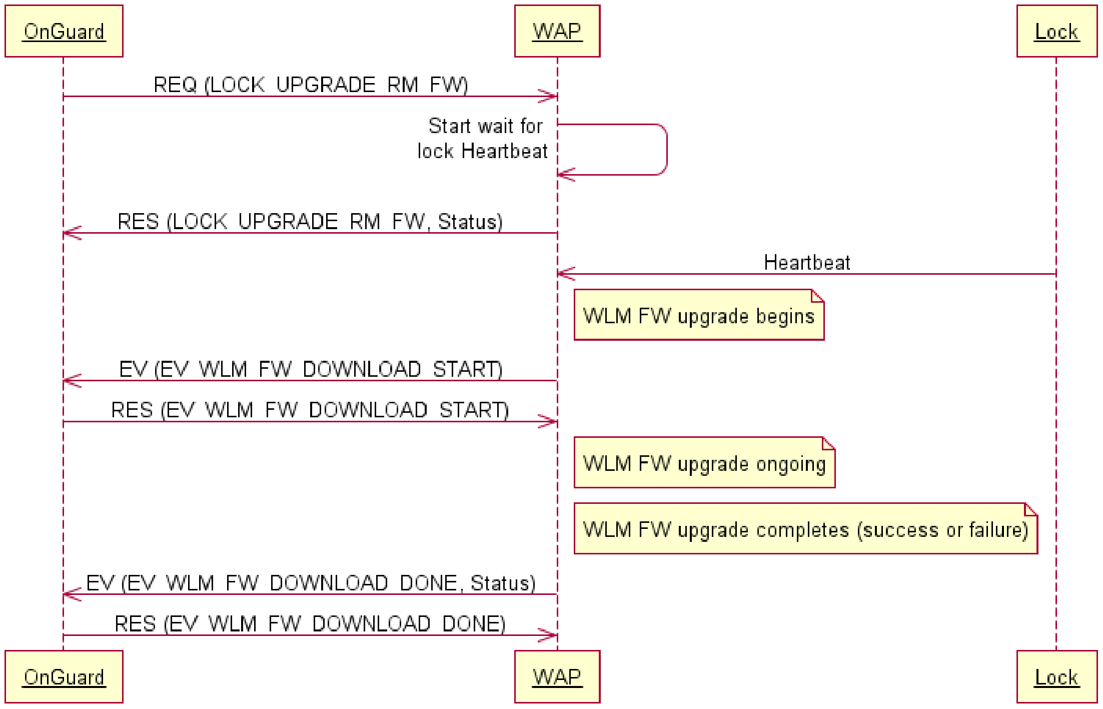
OnGuard
WAP
Lock
REQ (LOCK UPGRADE RM FW)
Start wait for lock Heartbeat
RES (LOCK UPGRADE RM FW, Status)
Heartbeat
WLM FW upgrade begins
EV (EV WLM FW DOWNLOAD START)
RES (EV WLM FW DOWNLOAD START)
WLM FW upgrade ongoing
WLM FW upgrade completes (success or failure)
EV (EV WLM FW DOWNLOAD DONE, Status)
RES (EV WLM FW DOWNLOAD DONE)
OnGuard
WAP
Lock

Figure 10: WLM Firmware Upgrade Flow Diagram

Appendix C – WAP Heartbeat & Diagnostics

The WAP will send a Heartbeat REQUEST to LMS at a regular, periodic interval and will contain lock diagnostic information. This interval is a configurable rate in seconds. Each device's role and responsibility regarding this message is explained below. A sequence diagram of this can be found in Figure 12: WAP Heartbeat Sequence Diagram

WAP Responsibility:

The payload of the WAP Heartbeat REQUEST shall contain the current time on the WAP, and any updated diagnostic information it has received from any of its Locks (WLMs) since the last WAP Heartbeat REQUEST was sent and acknowledged. Upon receiving a RESPOSNE (ACK) for this REQUEST the WAP shall reset its diagnostic information for any locks that it had just sent diagnostic information for, making certain that new diagnostic information for that lock has not been received in the interim of receiving LMS's response. This heartbeat shall be sent at all times, including during firmware download. This will ensure that when the heartbeat is not received it truly is offline.

LMS Responsibility:

Upon receiving the WAP Heartbeat REQUEST it shall send a RESPONSE with an empty payload giving acknowledgement of reception of the heartbeat. This will give an indication to LMS that the WAP is still alive. In addition it shall interpret the time within the payload and send a WAP_SET_CURRENT_TIME REQUEST to the WAP if the time has drifted by more than two seconds from the last set time by LMS. It will also interpret the diagnostic information for each lock that it receives. This diagnostic information is, in some cases, cumulative and therefore if LMS sees two heartbeats with same timestamp, it should conclude that it is receiving a duplicate heartbeat packet and not use the diagnostics data from that packet. LMS shall display and update in the Alarm Monitoring application the WAP Operating Channel received in each WAP heartbeat message. LMS shall store each lock's diagnostic information (see **Error! Reference source not found.** for data to be included) along with time and date information, and give the capability to extract the data for additional analysis in a graphing program such as Microsoft Excel. The data shall wrap around on itself after a configurable amount of time, with the default being 28 days (or 4 full weeks). The full content of the lock's diagnostic information is not necessarily required to be displayed and shall be addressed on a parameter by parameter basis.

WLM Responsibility:

The WLM's will send diagnostic information as part of their heartbeat payloads to the WAP. Upon a successful heartbeat with the WAP, these values will be reset.

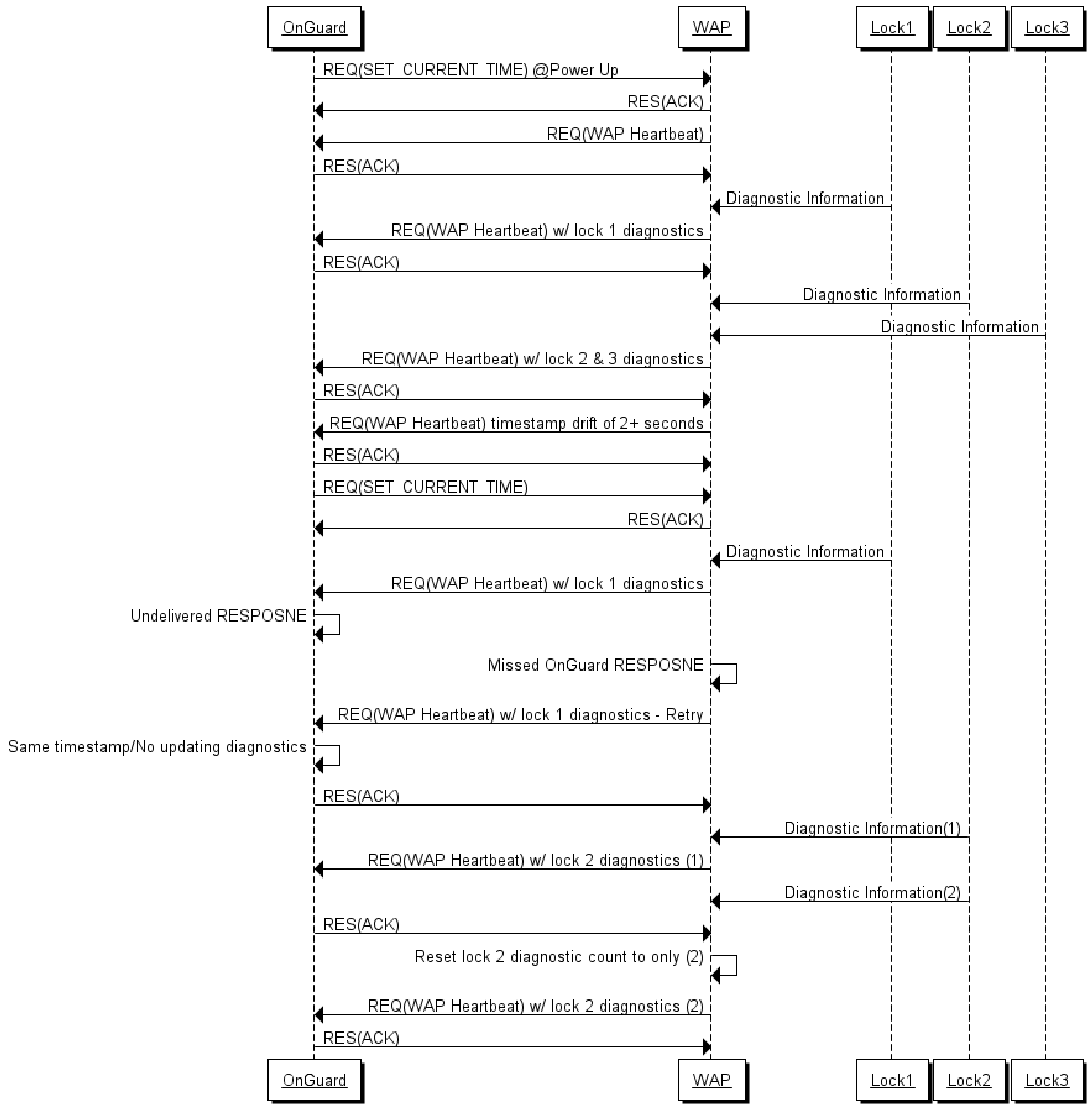
OnGuard
WAP
Lock1
Lock2
Lock3
REQ(SET_CURRENT_TIME) @Power Up
RES(ACK)
REQ(WAP Heartbeat)
RES(ACK)
Diagnostic Information
REQ(WAP Heartbeat) w/ lock 1 diagnostics
RES(ACK)
Diagnostic Information
Diagnostic Information
REQ(WAP Heartbeat) w/ lock 2 & 3 diagnostics
RES(ACK)
REQ(WAP Heartbeat) timestamp drift of 2+ seconds
RES(ACK)
REQ(SET_CURRENT_TIME)
RES(ACK)
Diagnostic Information
REQ(WAP Heartbeat) w/ lock 1 diagnostics
Undelivered RESPOSNE
Missed OnGuard RESPOSNE
REQ(WAP Heartbeat) w/ lock 1 diagnostics - Retry
Same timestamp/No updating diagnostics
RES(ACK)
Diagnostic Information(1)
REQ(WAP Heartbeat) w/ lock 2 diagnostics (1)
Diagnostic Information(2)
RES(ACK)
Reset lock 2 diagnostic count to only (2)
REQ(WAP Heartbeat) w/ lock 2 diagnostics (2)
RES(ACK)
OnGuard
WAP
Lock1
Lock2
Lock3

Figure 11: WAP Heartbeat Sequence Diagram