RFID READER SOFTWARE INTERFACE

Control Guide

ZEBRA

72E-131718-12EN

RFID READER SOFTWARE INTERFACE

CONTROL GUIDE

72E-131718-12EN

Revision A

November 2024

For Australia Only

For Australia Only. This warranty is given by Zebra Technologies Asia Pacific Pte. Ltd., 71 Robinson Road, #05-02/03, Singapore 068895, Singapore. Our goods come with guarantees that cannot be excluded under the Australia Consumer Law. You are entitled to a replacement or refund for a major failure and compensation for any other reasonably foreseeable loss or damage. You are also entitled to have the goods repaired or replaced if the goods fail to be of acceptable quality and the failure does not amount to a major failure.

Zebra Technologies Corporation Australia's limited warranty above is in addition to any rights and remedies you may have under the Australian Consumer Law. If you have any queries, please call Zebra Technologies Corporation at +65 6858 0722. You may also visit our website: zebra.com for the most updated warranty terms.

Terms of Use

Proprietary Statement

This manual contains proprietary information of Zebra Technologies Corporation and its subsidiaries ("Zebra Technologies"). It is intended solely for the information and use of parties operating and maintaining the equipment described herein. Such proprietary information may not be used, reproduced, or disclosed to any other parties for any other purpose without the express, written permission of Zebra Technologies.

Product Improvements

Continuous improvement of products is a policy of Zebra Technologies. All specifications and designs are subject to change without notice.

Liability Disclaimer

Zebra Technologies takes steps to ensure that its published Engineering specifications and manuals are correct; however, errors do occur. Zebra Technologies reserves the right to correct any such errors and disclaims liability resulting therefrom.

Limitation of Liability

In no event shall Zebra Technologies or anyone else involved in the creation, production, or delivery of the accompanying product (including hardware and software) be liable for any damages whatsoever (including, without limitation, consequential damages including loss of business profits, business interruption, or loss of business information) arising out of the use of, the results of use of, or inability to use such product, even if Zebra Technologies has been advised of the possibility of such damages. Some jurisdictions do not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you.

Publication Date

November 21, 2024

Revision History

Changes to the original manual are listed below:

Change	Date	Description
-01 Rev A	11/2009	Initial release
-02 Rev A	07/2010	Added reader management custom extensions, LLRP custom messages, and LLRP custom parameters; added SNMP chapter, updated XML schema.
-03 Rev A	06/2011	Added Fujitsu commands
-04 Rev A	02/2012	Added: RM and LLRP custom extension tables by product Removed: MOTO_C2_COMMAND, MOTO_C2_COMMAND_RESPONSE.
-05 Rev A	01/2014	Added support for FX7500
-06 Rev A	02/2015	Zebra Re-Branding
-07 Rev A	02/2016	Page 3-22 - updates for SNAP
-08 Rev A	11/2017	Added: FX9600; getPowerNegotiation, setPowerNegotiation, getAllowGuestStatus, setAllowGuestStatus; error codes 281-293.
-09 Rev A	11/2017	Added support for ATR7000
-10EN Rev A	03/2020	Updated RM and LLRP custom extension tables by product. Added RM and LLRP custom extension tables by product, error codes 294-306.
-11EN Rev A	12/2022	Add the following: ReaderDevice.get802.1xEAPInterfaces, ReaderDevice.get802.1xEAPTypes, ReaderDevice.get802.1xEAPInnerTypes, ReaderDevice.get802.1xEAPProperties, ReaderDevice.userAdd, ReaderDevice.userDel, ReaderDevice.userMod, ReaderDevice.set802.1xEAPProperties
-12EN Rev A	11/2024	Added: ReaderDevice. importSSHKey ReaderDevice. exportSSHKey

Table of Contents

ABOUT THIS GUIDE

Introduction

This Software Interface Control Guide provides information for RFID system integrators and software developers for evaluating and applying Zebra RFID products in RFID applications, and describes Low Level Reader Protocol (LLRP) extensions for RFID control, and Reader Management (RM) protocol extensions which use XML over HTTP methods for reader web page control.

Configurations

This guide applies to the following RFID configurations:

- FX7400-42350A30-US: 4-Port RFID Reader, US
- FX7400-22350A30-US: 2-Port RFID Reader, US
- FX7400-42310A30-WR: 4-Port RFID Reader, Global
- FX7400-22310A30-WR: 2-Port RFID Reader, Global
- FX7500-42320A50-US: 4-Port FCC
- FX7500-22320A50-US: 2-Port FCC
- FX7500-42325A50-WR: 4-Port Worldwide
- FX7500-22325A50-WR: 2-Port Worldwide
- FX9500-41324D41-US: 4-port configuration, US and Canada
- FX9500-41324D41-WW: 4-port configuration, International
- FX9500-81324D41-US: 8-port configuration, US and Canada
- FX9500-81324D41-WW: 8-port configuration, International
- FX9600-42320A50-US: 4-port RFID Reader, US and Canada
- FX9600-42325A50-WR: 4-port RFID Reader, Worldwide
- FX9600-42320A50-JP: 4-port RFID Reader, Japan
- FX9600-82320A50-US: 8-port RFID Reader, US and Canada
- FX9600-82325A50-WR: 8-port RFID Reader, Worldwide
- FX9600-82320A50-JP: 8-port RFID Reader, Japan
- MC3090Z-LC48HBAQE1: RFID Mobile Computer, US
- MC3090Z-LC48HBAQE2: RFID Mobile Computer, US and Canada

- MC319Z-GL4H24E0W: Laser, RFID, Worldwide
- MC319Z-GL4H24E0E: Laser, RFID, EU
- MC319Z-GI4H24E0W: Imager, RFID, Worldwide
- MC319Z-GI4H24E0E: Imager, RFID, EU
- MC9090-GJ0HJEQZ1US: Laser, RFID, US and Canada
- MC9090-GK0HJEQZ1US: 2D imager, RFID, US and Canada
- MC9090-GU0HJEQZ1US: 1D imager, RFID, US and Canada
- MC9090-GJ0HJEQZ4ER: Laser, RFID, EU
- MC9090-GK0HJEQZ4ER: 2D imager, RFID, EU
- MC9090-GU0HJEQZ4ER: 1D, RFID, EU
- MC919Z-GA0SWEQZ1WR: 1D, RFID, Worldwide
- MC919Z-GA0SWEQZ2EU: 1D, RFID, EU
- MC919Z-G30SWEQZ1WR: 2D imager, RFID, Worldwide
- MC919Z-G30SWEQZ2EU: 2D imager, RFID, EU
- MC919Z-G50SWEQZ1WR: 2D DPM Imager, RFID, Worldwide
- MC919Z-G50SWEQZ2EU: 2D DPM Imager, RFID, EU
- MC919Z-GJ0SWEQZ1WR: 1D LRX, RFID, Worldwide
- MC919Z-GJ0SWEQZ2EU: 1D LRX, RFID, EU
- MC919Z-GA0SWEQZ12R: 1D, RFID, Worldwide
- MC919Z-GA0SWEQZ22R: 1D, RFID, EU
- ATR7000-P1100A0-US

Chapter Descriptions

Topics covered in this guide are as follows:

- Getting Started provides an overview of Zebra extensions.
- LLRP Custom Extensions Operation provides general information on the Low Level Reader Protocol (LLRP) application.
- Reader Management Custom Extensions describes Reader Management (RM) custom extensions.
- LLRP Custom Extensions describes custom messages and parameters and provides the binary packet format for these.
- SNMP describes reader support for RFC1213 (MIB for Network Management of TCP/IP-based internets: MIB-II), RM MIB, and a Zebra custom MIB.
- Appendix , XML Schema for RM Extensions provides XML schema as a simple method of exercising RM commands.

Notational Conventions

The following conventions are used in this document:

- “RFID Reader” or “reader” refers to the Zebra FX Series RFID reader or RFID enabled MC Series mobile computers.
- Bullets (•) indicate:
  - Action items
  - Lists of alternatives
  - Lists of required steps that are not necessarily sequential.
- Sequential lists (e.g., those that describe step-by-step procedures) appear as numbered lists.

Related Documents and Software

The following documents provide more information about the RFID readers mentioned in this guide.

- *FX Series RFID Reader Integrator Guide*
- *FX7400 Series RFID Reader Regulatory Guide*
- *FX7500 RFID Reader Quick Start Guide*
- *FX7500 RFID Reader Regulatory Information*,
- *FX9500 RFID Reader User Guide*
- *FX9500 RFID Reader Regulatory Guide*
- FX9600 RFID Reader Quick Start Guide
- *MC3090Z RFID Mobile Computer Integrator Guide*
- *MC3090Z RFID Mobile Computer Quick Start Guide*
- *MC3090Z RFID Mobile Computer Regulatory Guide*
- *MC3191Z RFID Mobile Computer Integrator Guide*
- *MC3191Z Mobile Computer Quick Start Guide*
- *MC3191Z Mobile Computer Regulatory Guide*
- *MC9090 Mobile Computer Integrator Guide*
- *MC9090 Mobile Computer User Guide*
- *MC9090 Mobile Computer Quick Start Guide*
- *MC9090 Mobile Computer Regulatory Guide*
- *MC919Z Mobile Computer User Guide*
- *MC919Z Mobile Computer Quick Start Guide*
- *MC919Z Mobile Computer Regulatory Guide*
- *ATR7000 Advanced Array RFID Reader Integration Guide*
- *ATR7000 Advanced Array RFID Reader Quick Reference Guide*

For the latest version of all software and guides, go to: zebra.com/support.

Service Information

If you have a problem with your equipment, contact Zebra support for your region. Contact information is available at: zebra.com/support.

When contacting Zebra support, please have the following information available:

- Serial number of the unit
- Model number or product name
- Software type and version number

Zebra responds to calls by e-mail, telephone or fax within the time limits set forth in service agreements.

If your problem cannot be solved by Zebra support, you may need to return your equipment for servicing and will be given specific directions. Zebra is not responsible for any damages incurred during shipment if the approved shipping container is not used. Shipping the units improperly can possibly void the warranty.

If you purchased your business product from a Zebra business partner, please contact that business partner for support.

Getting Started

Introduction

This guide describes LLRP protocol extensions for RFID control, and RM protocol extensions which use XML over HTTP methods for reader web page control. These extensions, in addition to LLRP and RM standard protocols, constitute the native and most direct software interfaces to the RFID readers and RFID enabled mobile computers, and as such are used by the Zebra RFID3 API for C, .NET, and Java.

The RFID Reader software interface consists of the following two components.

RFID Control and Data Plane

The FX Series RFID reader and RFID enabled MC Series mobile computers support EPCGlobal Inc.'s, LLRP (low level reader) protocol v1.0.1 standard. In addition to standard LLRP support, the readers supports LLRP custom extensions detailed in this guide.

Reader (Device) Management Plane

The FX Series supports a number of software interfaces such as SNMP, Web Services, and Reader Management (RM) protocol v1.0.1 by EPCGlobal Inc. For the management interface, this guide includes XML extensions over HTTP that support a reader web page interface. Since the FX Series supports XML over HTTP message transport binding as defined in global's RM protocol, the XML custom extensions can be regarded as RM custom extensions. The MC Series mobile computers do not support RM protocol.

Audience

The target audience for this guide are RFID system integrators, RFID middleware software developers, and RFID application software developers who want to control the reader directly while bypassing the RFID3 API interface, particularly developers familiar with LLRP and RM protocols.

Zebra Extensions

The Reader currently supports Gen 2 v 1.2 new features, such as extended protocol control word (XPC), block permalocking, and re-commissioning using LLRP custom extensions. In the future, new functionality will be added to a new LLRP standard.

Another important set of LLRP extensions supports asynchronous events which enable efficient tag reporting of visibility events, typically referred to as XR autonomous mode. Because this method allows for more efficient use of network resources, and because it was the preferred method for many XR customers, the FX and MC series supports it over the LLRP interface through a set of LLRP custom extensions.

The RM protocol addresses basic RFID device management functionality. RM custom extensions address the need to extend the support of management interfaces beyond what is defined in the RM protocol, such as firmware upgrade and downgrade. The RM specification enables defining new vendor commands in a separate XML namespace. The RM custom extensions define the command interface (using XML over HTTP) to the reader's web management functions, available on the reader's web page. For example, the reader's web interface allows managing user (login) accounts, read point control, and region control.

NOTE: MC Series readers do not support RM.

LLRP Custom Extensions Operation

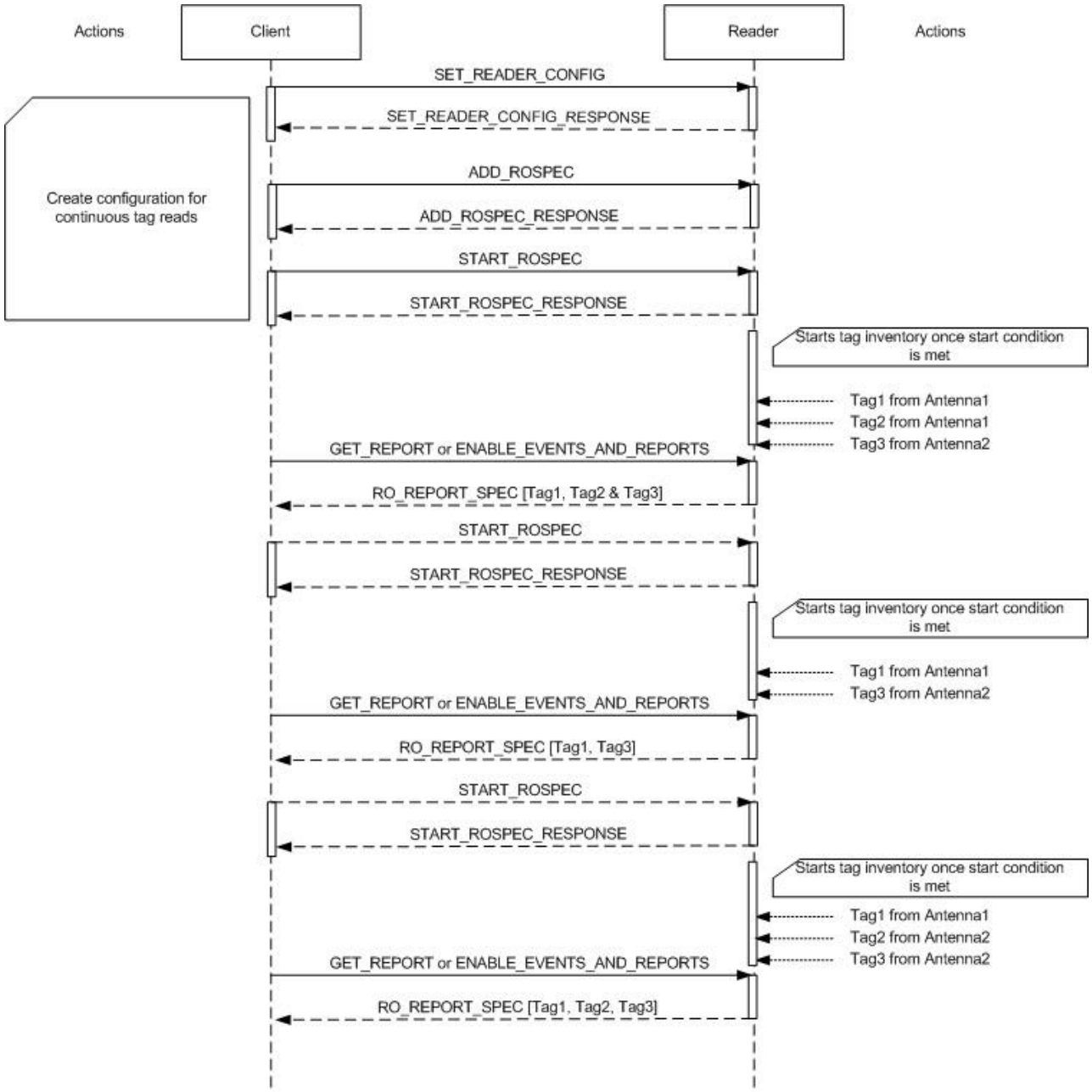
Introduction

The Low Level Reader Protocol (LLRP) application provides necessary features for operating and monitoring the LLRP server and the lower level reader entities. In order to minimize changes on the LLRP client side, it is strongly recommended to issue a capability query before using custom extensions as the capability value set can change in each software release.

Asynchronous Tag Events in Autonomous Mode Operation

Traditional LLRP operations use a client controller model where the LLRP client instructs the reader to perform an operation and the reader reports the results of the operation. Class 1 Gen 2 provisions limit the level of data filtering that the reader returns so the reader reports all results of the operation to the client. In typical continuous read modes where tags remain in the field of view for many read cycles, this type of operation can introduce overhead on the network as the same set of tag information is reported to the client across multiple operation cycles. Figure 1 illustrates a typical usage model.

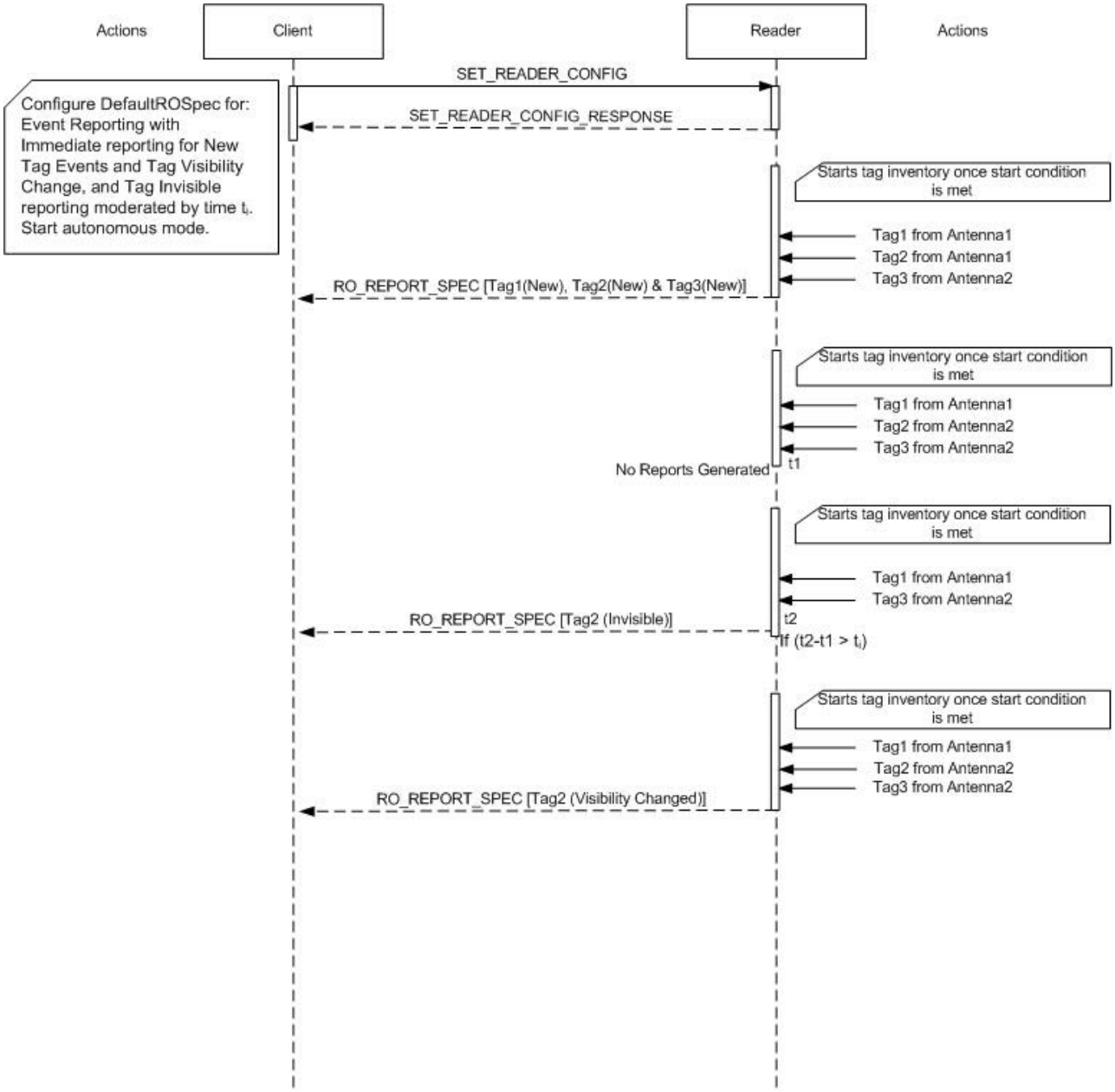
Figure 1 Command Driven Reader Operation Model



Readers support an event driven model called autonomous mode with event reporting, via custom extensions to LLRP. This enables the client to put reader into continuous operation mode and configure it to report only changes

in tag visibility states. A pre-configured ROSpec is provided as a custom extension to reader configuration. See MotoDefaultSpec on page 153. Use **GET_READER_CONFIG** and **SET_READER_CONFIG** to respectively retrieve and manipulate this extension. Use custom extensions to ROReportSpec associated with the ROSpec to control the format and criteria for reporting tags. After setting a reporting preference, use the **MotoAutonomousState** custom parameter in the reader configuration to enable autonomous mode on the reader. Figure 2 provides an example usage under this model.

Figure 2 Autonomous Event Driven Reader Operation Model



See LLRP Custom Extensions for information on the message and parameter extensions associated with autonomous mode of operation.

Filtering Tags Based on RSSI and Time

In addition to the Class 1 Gen 2 tag filtering mechanism, the reader can filter tags reported to the client based on the received signal strength, the time that the reader read the tag, or a combination of both. Use multiple rules to create a custom filter which can also filter multiple ranges of data. See LLRP Custom Extensions for details on the message and parameter extensions for custom filtering.

Class 1 Generation 2 (C1G2) Extensions Operation

Extensions support features such as block permalocking, XPC, and re-commissioning. There are two custom OpSpecs that support block permalock and re-commissioning access commands: **MotoC1G2BlockPermalock** and the OpSpec result enable block permalock operations. **MotoC1G2Recommission** and the OpSpec result enable re-commissioning operations. When re-commissioning a tag during an inventory operation, the reader sees the re-commissioning command type previously issued to the tag via the XPC (refer to the Gen 2 v1.2 specification for details on re-commissioning types). Therefore, the LLRP **TagReportData** parameter is extended to include XPC information.

All rules and functionality that apply to the OpSpec within an AccessSpec also apply to custom OpSpecs. For example, use the **C1G2TagSpec** parameter in conjunction with the OpSpec within an AccessSpec to specify tag filter data.

Before using a custom OpSpec, send the **GET_READER_CAPABILITIES** message specifying **0** or **All** in the **RequestedData** field. Alternatively, to avoid getting the entire capability set, specify the custom capability parameter subset to get using the **MotoGeneralRequestCapability** parameter in the extension point of the **GET_READER_CAPABILITIES** message. Use the **RequestData** field in **MotoGeneralRequestCapability** to return the following custom capability sets: All (custom capabilities), general capabilities, autonomous mode capabilities, filtering capabilities, persistent capabilities, and C1G2 V1.2 capabilities. Querying for C1G2 v1.2 capabilities returns the reader's ability to support the custom C1G2 command.

Currently, tags supporting re-commissioning are not available. Therefore, the **CanSupportRecommissioning** field in the **MotoAirProtocolCapabilities** parameter is set to **FALSE** even though it is implemented. In the near future, **CanSupportRecommissioning** will be set to TRUE when this feature is tested with real tags.

Refer to global's Gen v1.2 specification for operational details of XPC, block permalock, and e-commissioning.

Reader Management Custom Extensions

Introduction

This chapter describes Reader Management (RM) custom extensions.

NOTE: MC Series readers do not support RM.

RM over HTTP/XML transport binding mandates that a valid user is logged in before using RM commands. In addition, the user can use only one session for the RM over HTTP/XML transport interface.

NOTE: FX Series readers do not support commands under the AntennaReadPoint object.

Reader support for RM over SNMP transport binding is limited to mandatory features defined in GSI RM Spec 1.0.1. Refer to this specification for the MIB definition.
Go to gs1.org/standards/epc-rfid/reader-management/1-0-1 for more details.
Extensions described in this chapter are available only for RM over HTTP/XML transport binding.
See Appendix , XML Schema for RM Extensions for XML schema definitions of the extensions defined in this chapter.

The RM commands return the following errors when a valid user is not logged in or the current session is not valid:

- sessionTimeout
- notCurrentSession
- notLoggedIn

RM commands involving configuration changes mandate that a user with administrator privileges is logged in before making these changes, otherwise the following error returns:

- noEdit

During reader software update, all RM commands except **getFirmwareUpdateProgress** return the following error:

- osUpdateInProgress

When the reader is in diagnostic mode, all RM commands except the **viewSystemLog** command return the following error:

- diagnosticsInProgress

RM Extensions by Product

The RM commands supported by the various RFID fixed readers are outlined in Table 1.

Table 1 RM Extensions by Product

RM Command	FX7400	FX7500	FX9500	FX9600	ATR7000	Page
ReaderDevice.getCPUUsage	Y	Y	Y	Y	Y	28
ReaderDevice.getRAMUsage	Y	Y	Y	Y	Y	28
ReaderDevice.doFirmwareUpdate	Y	Y	Y	Y	Y	29
ReaderDevice.setFirmwareUpdateParams	Y	Y	N	Y	Y	29
ReaderDevice.updateCertificate	Y	Y	N	Y	Y	30
ReaderDevice.setUserLED	Y	Y	Y	Y	N	31
ReaderDevice.getFlashMemoryUsage	Y	Y	N	Y	Y	31
ReaderDevice.getFirmwareUpdateProgress	Y	Y	N	Y	Y	32
ReaderDevice.getUserList	Y	Y	N	Y	Y	32
ReaderDevice.doAddUser	Y	N	N	N	N	33
ReaderDevice.doDelUser	Y	N	N	N	N	34
ReaderDevice.doChangePassword	Y	Y	Y	Y	Y	34
ReaderDevice.doChangeUserRole	Y	N	N	Y	N	35
ReaderDevice.doLogin	Y	Y	Y	Y	Y	35
ReaderDevice.doLogout	Y	Y	Y	Y	Y	36
ReaderDevice.doChangeDefaultUserPassword	Y	N	N	N	N	36
ReaderDevice.getSupportedRegionList	Y	Y	N	Y	Y	37
ReaderDevice.getRegionStandardList	N	Y	N	Y	N	37
ReaderDevice.getActiveRegion	Y	Y	Y	Y	Y	38
ReaderDevice.setActiveRegion	Y	Y	N	Y	N	39
ReaderDevice.getMaxAntennasSupported	Y	Y	Y	Y	Y	39
ReaderDevice.getAlarmNotificationSNMPHost	Y	Y	N	Y	Y	40
ReaderDevice.setAlarmNotificationSNMPHost	Y	Y	N	Y	Y	40
ReaderDevice.getNetworkInterfaceSettings	Y	Y	Y	Y	Y	41
ReaderDevice.setNetworkInterfaceSettings	Y	Y	N	Y	Y	42
ReaderDevice.setDHCPConfig	Y	Y	N	Y	Y	43
ReaderDevice.getWebServerSecuritySetting	N	Y	N	Y	Y	43
ReaderDevice.setBTConfig	N	Y	N	Y	N	44

RM Commands are applicable for fixed readers only.
3 = Supported / X = Not Supported

Table 1 RM Extensions by Product

RM Command	FX7400	FX7500	FX9500	FX9600	ATR7000	Page
ReaderDevice.getWebServerSecuritySetting	Y	Y	N	Y	Y	44
ReaderDevice.setWebServerSecuritySetting	Y	Y	N	Y	Y	45
ReaderDevice.getShellStatus	Y	Y	Y	Y	Y	45
ReaderDevice.setShellStatus	Y	Y	N	Y	Y	46
ReaderDevice.getFTPStatus	Y	Y	Y	Y	Y	46
ReaderDevice.setFTPStatus	Y	Y	N	Y	Y	47
ReaderDevice.getUSBMode	Y	Y	N	Y	N	47
ReaderDevice.setUSBMode	Y	Y	N	Y	N	48
ReaderDevice.getLLRPConfig	Y	Y	Y	Y	Y	49
ReaderDevice.setLLRPConfig	Y	Y	Y	Y	Y	50
ReaderDevice.isLLRPRunning	Y	Y	N	Y	Y	50
ReaderDevice.isLLRPConnected	Y	Y	N	Y	Y	51
ReaderDevice.ConnectLLRP	Y	Y	Y	Y	Y	51
ReaderDevice.viewSystemLog	Y	Y	N	Y	Y	52
ReaderDevice.viewAccessLog	Y	Y	N	Y	Y	52
ReaderDevice.viewCurrentCertificateDetails	Y	Y	N	Y	Y	53
ReaderDevice.setNTPConfig	Y	Y	N	Y	Y	53
ReaderDevice.getWatchdogStatus	Y	Y	N	Y	Y	54
ReaderDevice.setWatchdogStatus	Y	Y	N	Y	Y	54
ReaderDevice.shutDown	Y	Y	Y	Y	Y	55
ReaderDevice.getExtAntennaMode	Y	Y	Y	Y	Y	55
ReaderDevice.setExtAntennaMode	Y	Y	Y	Y	Y	56
ReaderDevice.getReaderVersionInfo	Y	Y	Y	Y	Y	56
ReaderDevice.getManufacturer	Y	Y	Y	Y	Y	57
ReaderDevice.getModel	Y	Y	Y	Y	Y	57
ReaderDevice.getName	Y	Y	Y	Y	Y	58
ReaderDevice.setName	Y	Y	Y	Y	Y	58
ReaderDevice.getDebounceTime	Y	Y	Y	Y	Y	59
ReaderDevice.setDebounceTime	Y	Y	Y	Y	Y	59
ReaderDevice.getTimeTicks	Y	Y	Y	Y	Y	60
ReaderDevice.getLocalTime	Y	Y	Y	Y	Y	60

RM Commands are applicable for fixed readers only.
3 = Supported / X = Not Supported

Table 1 RM Extensions by Product

RM Command	FX7400	FX7500	FX9500	FX9600	ATR7000	Page
ReaderDevice.setLocalTime	Y	Y	Y	Y	Y	61
ReaderDevice.getAllReadPoints	Y	Y	Y	Y	Y	61
ReaderDevice.saveConfigChanges	Y	Y	Y	Y	Y	62
ReaderDevice.discardConfigChanges	Y	Y	N	Y	Y	62
ReaderDevice.hasConfigChanged	Y	Y	N	Y	Y	63
ReaderDevice.getUncommitedConfigChangesDescription	Y	Y	N	Y	Y	63
ReaderDevice.getTimeZones	Y	Y	Y	Y	Y	64
ReaderDevice.setTimeZone	Y	Y	Y	Y	Y	64
ReaderDevice.getReaderProfileList	Y	Y	Y	Y	Y	65
ReaderDevice.setProfileActive	Y	Y	Y	Y	Y	65
ReaderDevice.deleteProfile	Y	Y	Y	Y	Y	66
ReaderDevice.importProfileToReader	Y	Y	N	Y	Y	67
ReaderDevice.exportProfileFromReader	Y	Y	N	Y	Y	68
ReaderDevice.getSerialTimeout	Y	N	N	N	N	68
ReaderDevice.setSerialTimeout	Y	N	N	N	N	69
ReaderDevice.getAntennaCheck	Y	Y	N	Y	Y	69
ReaderDevice.setAntennaCheck	Y	Y	N	Y	Y	70
ReaderDevice.getReaderDetails	Y	Y	Y	Y	Y	70
ReaderDevice.firmwareRevertBack	N	Y	N	Y	Y	71
ReaderDevice.addIPSecParams	N	Y	N	Y	Y	71
ReaderDevice.removeIPSecParams	N	Y	N	Y	Y	72
ReaderDevice.getGPIPortStatus	N	Y	N	Y	Y	72
ReaderDevice.setGPOPinStatus	N	Y	N	Y	Y	73
ReaderDevice.getEnableRevertBackStatus	N	Y	N	Y	Y	73
ReaderDevice.getGPOPortStatus	N	Y	N	Y	Y	74
ReaderDevice.getIPSecParamsList	N	Y	N	Y	Y	74
ReaderDevice.getIdleModeTimeout	N	Y	N	Y	Y	75
ReaderDevice.setIdleModeTimeout	N	Y	N	Y	Y	75
ReaderDevice.processResponseFile	Y	Y	N	Y	Y	76
ReaderDevice.startOSupdate	N	Y	N	Y	N	76
ReaderDevice.getMaxUserApps	N	Y	N	Y	Y	77

RM Commands are applicable for fixed readers only.
3 = Supported / X = Not Supported

Table 1 RM Extensions by Product

RM Command	FX7400	FX7500	FX9500	FX9600	ATR7000	Page
ReaderDevice.installUserApp	N	Y	N	Y	Y	77
ReaderDevice.startUserApp	N	Y	N	Y	Y	78
ReaderDevice.autostarUserApp	N	Y	N	Y	Y	78
ReaderDevice.uninstalluserapp	N	Y	N	Y	Y	79
ReaderDevice.getInstalledApps	N	Y	N	Y	Y	79
ReaderDevice.getCurrentRunStatus	N	Y	N	Y	Y	80
ReaderDevice.generateCustomerSupportDataFile	N	Y	N	Y	Y	80
ReaderDevice.purgeLogs	N	Y	N	Y	Y	81
ReaderDevice.getwirelessnwlist	N	Y	N	Y	Y	81
ReaderDevice.addwirelessnw	N	Y	N	Y	Y	82
ReaderDevice.getwirelessnwproperties	N	Y	N	Y	Y	82
ReaderDevice.getwirelessconfiguredparams	N	Y	N	Y	Y	83
ReaderDevice.getGPIOSettings	N	Y	N	Y	Y	84
ReaderDevice.setGPIOSettings	N	Y	N	Y	Y	85
ReaderDevice.resetToFactoryDefaults	N	Y	N	Y	Y	85
ReaderDevice.getSystemLogConfiguration	N	Y	N	Y	Y	86
ReaderDevice.setSystemLogConfiguration	N	Y	N	Y	Y	85
ReaderDevice.getRadioModuleOnTime	N	Y	N	Y	Y	86
ReaderDevice.setDiagnosticMode	N	Y	N	Y	Y	87
ReaderDevice.startReaderDiagnostics	N	Y	N	Y	Y	88
ReaderDevice.getEventAmbientTemperatureHighAlarmCount	N	Y	N	Y	Y	88
ReaderDevice.getEventAmbientTemperatureCriticalAlarmCount	N	Y	N	Y	Y	89
ReaderDevice.getEventPATemperatureHighAlarmCount	N	Y	N	Y	Y	89
ReaderDevice.getEventPATemperatureCriticalAlarmCount	N	Y	N	Y	Y	90
ReaderDevice.getEventForwardPowerHighAlarmCount	N	Y	N	Y	Y	90
ReaderDevice.getEventForwardPowerLowAlarmCount	N	Y	N	Y	Y	91
ReaderDevice.getEventReversePowerHighAlarmCount	N	Y	N	Y	Y	91
RM Commands are applicable for fixed readers only. 3 = Supported / X = Not Supported						

Table 1 RM Extensions by Product

RM Command	FX7400	FX7500	FX9500	FX9600	ATR7000	Page
ReaderDevice.getEventEchoThresholdAlarmCount	N	Y	N	Y	Y	92
ReaderDevice.getEventDatabaseWarningCount	N	Y	N	Y	Y	92
ReaderDevice.getEventDatabaseErrorCount	N	Y	N	Y	Y	93
ReaderDevice.getEventGPIOInformationCount	N	Y	N	Y	Y	93
ReaderDevice.getRadioPowerState	N	Y	N	Y	Y	94
ReaderDevice.getUSBState	N	Y	N	Y	Y	94
ReaderDevice.viewMACErrorLog	Y	N	N	N	N	95
ReaderDevice.getPowerNegotiation	N	Y	N	Y	Y	95
ReaderDevice.setPowerNegotiation	N	Y	N	Y	Y	96
ReaderDevice.getAllowGuestStatus	N	Y	N	Y	Y	96
ReaderDevice.setAllowGuestStatus	N	Y	N	Y	Y	97
ReaderDevice.manageLicense	N	Y	N	Y	N	97
ReaderDevice.getNodeJSPortnum	N	Y	N	Y	N	98
ReaderDevice.setNodeJSPortnum	N	Y	N	Y	N	98
ReaderDevice.setLEDFirmwareUpdate	N	Y	N	Y	N	99
ReaderDevice.getInstalledLicenseList	N	Y	N	Y	N	99
ReaderDevice.manageFXEasyConnection	N	Y	N	Y	N	100
ReaderDevice.getSerialConfig	N	Y	N	Y	N	101
ReaderDevice.setSerialConfig	N	Y	N	Y	N	102
ReaderDevice.getTempSensorData	N	Y	N	Y	N	102
ReaderDevice.get802.1xEAPInterfaces	N	Y	N	Y	Y	103
ReaderDevice.get802.1xEAPTypes	N	Y	N	Y	Y	103
ReaderDevice.get802.1xEAPInnerTypes	N	Y	N	Y	Y	103
ReaderDevice.get802.1xEAPProperties	N	Y	N	Y	Y	104
ReaderDevice.set802.1xEAPProperties	N	Y	N	Y	Y	106
ReaderDevice.userAdd	N	Y	N	Y	Y	104
ReaderDevice.userDel	N	Y	N	Y	Y	105
ReaderDevice.userMod	N	Y	N	Y	Y	105
ReaderDevice.enrollToCloud	N	Y	N	Y	Y	106
ReaderDevice.disEnrollFromCloud	N	Y	N	Y	Y	107
ReaderDevice.isEnrolledToCloud	N	Y	N	Y	Y	107

RM Commands are applicable for fixed readers only.
3 = Supported / X = Not Supported

Table 1 RM Extensions by Product

RM Command	FX7400	FX7500	FX9500	FX9600	ATR7000	Page
ReaderDevice.isConnectedToCloud	N	Y	N	Y	Y	108
ReaderDevice.connectToCloud	N	Y	N	Y	Y	108
ReaderDevice.disconnectFromCloud	N	Y	N	Y	Y	108
ReaderDevice.autoConnectToCloud	N	Y	N	Y	Y	109
ReaderDevice.isAutoConnectToCloud	N	Y	N	Y	Y	109
ReaderDevice.importCloudConfigToReader	N	Y	N	Y	Y	110
ReaderDevice.exportCloudConfigFromReader	N	Y	N	Y	Y	110
ReaderDevice.exportGPIOConfigFromReader	N	Y	N	Y	Y	110
ReaderDevice.manageCloudEndpoints	N	Y	N	Y	Y	111
ReaderDevice.cloudEndpointsMapping	N	Y	N	Y	Y	111
ReaderDevice.registerIoTConnector	N	Y	N	Y	Y	116
ReaderDevice.autoEnrollIoTConnector	N	Y	N	Y	Y	116
ReaderDevice.importOperatingModeToReader	N	Y	N	Y	Y	116
ReaderDevice.exportOperatingModeFromReader	N	Y	N	Y	Y	116
ReaderDevice.addCAcert	N	Y	N	Y	Y	116
ReaderDevice.deleteCAcert	N	Y	N	Y	Y	116
ReaderDevice.listCAcerts	N	Y	N	Y	Y	116
AntennaReadPoint.getSupportedAirProtocols	Y	Y	N	Y	Y	116
AntennaReadPoint.getCurrentAirProtocol	Y	Y	N	Y	Y	116
AntennaReadPoint.setAirProtocol	Y	Y	N	Y	Y	117
AntennaReadPoint.getTransmitPowerLevel	Y	Y	N	Y	Y	117
AntennaReadPoint.setTransmitPowerLevel	Y	Y	N	Y	Y	118
AntennaReadPoint.getCableLossCompensation	N	Y	N	Y	N	118
AntennaReadPoint.setCableLossCompensation	N	Y	N	Y	N	119
AntennaReadPoint.getCRCErrors	Y	Y	N	Y	Y	119
AntennaReadPoint.resetCRCErrors	Y	Y	N	Y	Y	120
AntennaReadPoint.getRFOnTime	Y	Y	N	Y	Y	120
AntennaReadPoint.getGen2OptionalOperCounts	Y	Y	N	Y	Y	121
AntennaReadPoint.getNXPCustomOperCounts	Y	Y	N	Y	Y	122
AntennaReadPoint.getFujitsuCustomOperCounts	Y	N	N	N	N	123
AntennaReadPoint.getImpinjCustomOperCounts	Y	Y	N	Y	Y	124

RM Commands are applicable for fixed readers only.
3 = Supported / X = Not Supported

ReaderDevice.getCPUUsage

Get CPU usage information.

Usage

ReaderDevice.getCPUUsage(void): user: int, system: int

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **user** - Data type: integer. Percent of CPU utilization for user processes.
- **system** - Data type: integer. Percent of CPU utilization for system processes.

Possible Error Conditions

N/A

ReaderDevice.getRAMUsage

Get RAM usage information.

Usage

ReaderDevice.getRAMUsage(void): total: unsignedInt, used: unsignedInt, free: unsignedInt

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **total** - Data type: unsignedInt. Total RAM.
- **used** - Data type: unsignedInt. RAM used.
- **free** - Data type: unsignedInt. RAM available for use.

Possible Error Conditions

- operationFailed

ReaderDevice.doFirmwareUpdate

Initiate firmware update.

Usage

ReaderDevice.doFirmwareUpdate(void): void

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

- nosuchFileOrPath
- operationFailed

ReaderDevice.setFirmwareUpdateParams

Set firmware image location parameters.

Usage

ReaderDevice.setFirmwareUpdateParams(imageLocationURL: string, userName: string, password: string, updateAllPartitions: boolean): void

Parameter(s)

- **imageLocationURL** - Data type: string. FTP or secure FTP server location of image files.
- **userName** - Data type: string. User login name for FTP or secure FTP server.
- **password** - Data type: string. Password for user login.
- **updateAllPartitions** - Data type: boolean. Flag indicating if all partitions must be updated.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

- malformedFTPURL
- invalidUser

ReaderDevice.updateCertificate

In secure mode, update the security certificate on the reader using secure FTP.

Usage

ReaderDevice.updateCertificate(ftpsUrl: string, ftpsUserName: string, ftpsPassword: string, pfxPassword: string, restartConfirm: boolean): void

Parameter(s)

- **ftpsUrl** - Data type: string. URL of the secure FTP server.
- **ftpsUserName** - Data type: string. User name for the secure FTP server.
- **ftpsPassword** - Data type: string. Password for the secure FTP server.
- **pfxPassword** - Data type: string. Public key for the certificate.
- **restartConfirm** - Data type: boolean. Specifies whether to restart the FTPS and SSH services.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

- noRestartPermission
- noMaint
- notInSecureMode
- malformedFTPSURL
- malformedFTPURL
- invalidIpAddr
- invalidUser
- nosuchFileOrPath
- operationFailed
- wrongPFXPassword
- couldNotUninstallCert
- wrongPFXPassword
- noPrivateKeyFound
- failedCertImport
- errorReadingPassFile
- invalidDataInPfx
- couldNotInstallCertificate

ReaderDevice.setUserLED

Set the user LED.

Usage

ReaderDevice.setUserLED(ledColor: string, duration: int, blink: boolean): void

Parameter(s)

- **ledColor** - Data type: string. Color for the user LED.
- **duration** - Data type: integer. Duration in which the LED is turned on.
- **blink** - Data type: boolean. Flag indicating blink status.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

N/A

ReaderDevice.getFlashMemoryUsage

Get the flash memory usage information.

Usage

ReaderDevice.getFlashMemoryUsage(void): list of <value: motorm:FlashPartitionParamType>

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **value** - **Data Type**: **motorm:FlashPartitionParamType**. Flash partition usage information.

Possible Error Conditions

- operationFailed

ReaderDevice.getFirmwareUpdateProgress

Get the progress status on the firmware update.

Usage

ReaderDevice.getFirmwareUpdateProgress(void): progressValue: motorm:firmwareUpdtProgress

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **progressValue** - **Data Type**: **motorm:firmwareUpdtProgress**. Progress value.

Possible Error Conditions

- FirmwareUpdateNotStarted
- startingFirmwareUpdate
- failedToGetUpdateProgress

ReaderDevice.getUserList

Get the list of the configured user along with the permission.

Usage

ReaderDevice.getUserList(void): list of <value: **motorm:UserInfoValueParamType>**

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **value** - **Data Type: motorm:UserInfoValueParamType**. List of user with current permission level.

Possible Error Conditions

- dbOpenFailed

ReaderDevice.doAddUser

Add the user along with the password and permission specified.

Usage

ReaderDevice.doAddUser(userName: string, password: string, isAdmin: boolean): void

Parameter(s)

- **userName** - Data type: string. Name of the user to add.
- **password** - Data type: string. Password for user login.
- **isAdmin** - Data type: boolean. Indicates whether the user is an administrator.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

- invalidName
- invalidPassword
- invalidAccess
- addAdminUser
- dbOpenFailed
- duplicatedUserName
- dbPutFailed

ReaderDevice.doDelUser

Delete the user specified by **userName**.

Usage

ReaderDevice.doDelUser(userName: string): void

Parameter(s)

- **userName** - Data type: string. Name of the user to delete.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

- invalidSelection
- dbOpenFailed

ReaderDevice.doChangePassword

Change the password for the user specified by **userName**.

Usage

ReaderDevice.doChangePassword(userName: string, oldPassword: string, newPassword: string): void

Parameter(s)

- **userName** - Data type:string. Name of the user whose password is to change.
- **oldPassword** - Data type: string. Existing password of the user.
- **newPassword** - Data type: string. New password specified for the user.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

- invalidSelection
- invalidPassword
- newPswdSameAsOld
- dbOpenFailed
- invalidName
- wrongOldPswd
- notFindUser

ReaderDevice.doChangeUserRole

Change the user role for the user specified by **userName**.

Usage

ReaderDevice.doChangeUserRole(userName: string, isAdmin: boolean): void

Parameter(s)

- **userName** - Data type: string. Name of the user whose role is to change.
- **isAdmin** - Data type: boolean. Indicates whether the user is an administrator.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

- invalidSelection
- dbOpenFailed
- operationFailed

ReaderDevice.doLogin

Log in to the management console.

Usage

ReaderDevice.doLogin(username: string, password: string): sessionID: string

Parameter(s)

- **username** - Data type: string. User name.
- **password** - Data type: string. Password for the user.

Return Value(s)

- **sessionID** - Data type: string. Session ID used for this login session.

Possible Error Conditions

- invalidUser
- userLoggedIn
- AdminLoggedIn
- notCurrentSession

ReaderDevice.doLogout

Logout current session from issuing host over HTTP interface.

Usage

ReaderDevice.doLogout(forceLogout: boolean): void

Parameter(s)

- **forceLogout** - Data type: boolean. Force logout even if there are configuration changes.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

- pendingChanges

ReaderDevice.doChangeDefaultUserPassword

Change the default user name and password on the first login.

Usage

ReaderDevice.doChangeDefaultUserPassword(defaultUsername: string, defaultPassword: string, newUsername: string, newPassword: string): void

Parameter(s)

- **defaultUsername** - Data type: string. Default user name.
- **defaultPassword** - Data type: string. Default password for the user.
- **newUsername** - Data type: string. New user name.
- **newPassword** - Data type: string. New password for the user.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

- invalidUser
- addAdminUser
- dbOpenFailed
- defaultUserNotPresent
- invalidName
- invalidPassword
- duplicatedUserName
- dbPutFailed

ReaderDevice.getSupportedRegionList

Get the supported region list on the reader.

Usage

ReaderDevice.getSupportedRegionList(void): list of <value: string>

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **value** - Data type: string. List of regions supported.

Possible Error Conditions

N/A

ReaderDevice.getRegionStandardList

Get the supported communication standard list along with details for the chosen region.

Usage

ReaderDevice.getRegionStandardList(regionName: string):
list of <value: motorm:RegionInfoValueParamType>

Parameter(s)

- **regionName** - Data type: string. Name of the chosen region.
- r**equestChannelList** - Data Type: Boolean. Request to include the channel list as part of the response.

Return Value(s)

- **value** - Data type: **motorm:RegionInfoValueParamType**. List of supported standards for the region with supported settings for each region.

Possible Error Conditions

- unsupportedRegion

ReaderDevice.getActiveRegion

Get details of the active region.

Usage

ReaderDevice.getActiveRegion(void): regionName: string, standardName: string, list of <channelUsed: int>, isLBTOn: boolean, isHoppingOn: boolean

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **regionName** - Data type: string. Name of the active country.
- **standardName** - Data type: string. Name of the active region.
- **channelUsed** - Data type: integer. One instance of the channel used in the region.
- **isLBTOn** - Data type: boolean. Indicates if LBT is on for the active region.
- **isHoppingOn** - Data type: boolean. Indicates if frequency hopping is on for the active region.

Possible Error Conditions

N/A

ReaderDevice.setActiveRegion

Set the active region.

Usage

ReaderDevice.setActiveRegion(regionName: string, standardName: string, list of<channelUsed:int>, doLBT: boolean, doHopping: boolean): void

Parameter(s)

- **regionName** - Data type: string. Name of the active region.
- **standardName** - Data type: string. Name of the active communication standard.
- **channelUsed** - Data type: integer. One instance of channel used in the region.
- **doLBT** - Data type: boolean. Indicates whether to use LBT for this region.
- **doHopping** - Data type: boolean. Indicates whether to use frequency hopping for this region.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

- radioNotInitialised
- unsupportedStandard
- unsupportedRegion
- invalidValue

ReaderDevice.getMaxAntennasSupported

Get the maximum number of antennas supported by the reader.

Usage

ReaderDevice.getMaxAntennasSupported(void): maxAntennasSupported: int

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

maxAntennasSupported - Data type: integer. Maximum number of antennas the reader supports.

Possible Error Conditions

N/A

ReaderDevice.getAlarmNotificationSNMPHost

Get host information for the currently registered SNMP-based alarm notification receiver.

Usage

ReaderDevice.getAlarmNotificationSNMPHost(void): hostIP: string, version: string, community: string, sendServerHeartBeat: boolean

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **hostIP** - Data type: string. IP address of listener.
- **version** - Data type: string. SNMP version.
- **community** - Data type: string. SNMP community string.
- **sendServerHeartBeat** - Data type: boolean. Send the server heartbeat.

Possible Error Conditions

N/A

ReaderDevice.setAlarmNotificationSNMPHost

Method to set SNMP trap listener host to which reader alarms must be issued.

Usage

ReaderDevice.setAlarmNotificationSNMPHost(hostIP: string, version: string, community: string, sendServerHeartBeat: boolean): void

Parameter(s)

- **hostIP** - Data type: string. IP address of the SNMP trap listener.
- **version** - Data type: string. SNMP protocol version number the host supports.
- **community** - Data type: string. SNMP community string.
- **sendServerHeartBeat** - Data type: boolean. Send the server heartbeat.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

- invalidIpAddr
- valueOverSize
- invalidSnmpVersion

ReaderDevice.getNetworkInterfaceSettings

Retrieve network interface settings on the reader.

Usage

ReaderDevice.getNetworkInterfaceSettings(Interface: motorm:InterfaceType, isCoreConfig: boolean): isDHCPEnabled: boolean, IPAddress: string, MACAddress: string, SubnetMask: string, Gateway: string, DNSServer: string, isCoreConfig: boolean, IPVersion: string, isDHCPv6Enabled: boolean, IPV6Address: string, IPV6Suffix: string, IPV6DNS: string, IPV6GateWay: string, EnableRAPAckets: boolean

Parameter(s)

- **Interface** - Data type:motorm:InterfaceType. Network interface to be queried for settings. Possible values are ETH, WIFI, or BT.
- **isCoreConfig** - Data type: boolean. Is the core config asked for.

Return Value(s)

- **isDHCPEnabled** - Data type: boolean. Indicates whether DHCP is enabled on the reader.
- **IPAddress** - Data type: string. IP address of the reader.
- **MACAddress** - Data type: string. MAC address of the reader.
- **SubnetMask** - Data type: string. Subnetmask of the reader.
- **Gateway** - Data type: string. Gateway of the reader.
- **DNSServer** - Data type: string. DNS server of the reader.
- **isCoreConfig** - Data type: boolean. Indicates whether the core configuration is requested.
- **IPVersion** - Data type: string. Indicates IP version enabled - IPV4/IPV6/Both.
- **isDHCPv6Enabled** - Data type: boolean. Is DHCPv6 enabled on the reader.
- **IPV6Address** - Data type: string. IPV6 IP address.
- **IPV6Suffix** - Data type: string. IPV6 suffix or netmask.
- **IPV6DNS** - Data type: string. IPV6 DNS server.
- **IPV6GateWay** - Data type: string. IPV6 gateway.
- **EnableRAPAckets** - Data type: boolean. Enable the acceptance of RA packets.

Possible Error Conditions

- operationFailed

ReaderDevice.setNetworkInterfaceSettings

Set the network interface parameters on the reader.

Usage

ReaderDevice.setNetworkInterfaceSettings(Interface: motorm:InterfaceType, IPAddress: string, SubnetMask: string, Gateway: string, DNSServer: string, IPV6Address: string, IPV6Suffix: string, IPV6GateWay: string, IPV6DNS: string, IPVersion: string, EnableRAPackets: boolean): void

Parameter(s)

- **Interface** - Data type: motorm:InterfaceType. Network interface to which the settings apply. Possible values are ETH, WIFI, or BT.
- **IPAddress** - Data type: string. IP address of the reader.
- **SubnetMask** - Data type: string. Subnetmask of the reader.
- **Gateway** - Data type: string. Gateway of the reader.
- **DNSServer** - Data type: string. DNS server of the reader.
- **IPV6Address** - Data type: string. IPV6 address of the reader.
- **IPV6Suffix** - Data type: string. IPV6 network suffix.
- **IPV6GateWay** - Data type: string. IPV6 gateway of the reader.
- **IPV6DNS** - Data type: string. IPV6 DNS server of the reader.
- **IPVersion** - Data type: string. Indicates IP version enabled - IPV4/IPV6/Both.
- **EnableRAPackets** - Data type: boolean. Enable the acceptance of RA packets.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

- invalidIpAddr
- invalidNetworkMask

Notes

- **InterfaceType** currently supports only ETH.
- IPV4 elements are mandatory.
- IPV6 elements are optional.

ReaderDevice.setDHCPConfig

Modify (enable or disable) the DHCP configuration on the reader.

Usage

ReaderDevice.setDHCPConfig(Interface: motorm:InterfaceType, enableDHCP: boolean, enableDHCPv6: boolean): void

Parameter(s)

- **Interface** - Data type: motorm:InterfaceType. Network interface to which the DHCP setting apply. Possible values are ETH, WIFI, or BT.
- **enableDHCP** - Data type: Boolean. Indicates whether to enable or disable DHCP.
- **enableDHCPV6** - Data type: Boolean. Indicates whether to enable or disable DHCPv6.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

N/A

Notes

- **InterfaceType** supports Ethernet, Wi-Fi, and Bluetooth.

ReaderDevice.getBTConfig

Get Bluetooth specific parameters from the reader.

Usage

ReaderDevice. getBTConfig (void) : isDiscoverable: Boolean, isPairable: Boolean, isPasskeyenabled: Boolean, startIP: string, endIP: string

Parameter(s)

- **isCoreConfig** - Data type: boolean. Is the core configuration requested.

Return Value(s)

- **isDiscoverable** - Data type: boolean. Flag indicating if the Bluetooth is discoverable.
- **isPairable** - Data type: boolean. Is Bluetooth pairable.
- **isPasskeyenabled** - Data type: boolean. Is passkey required for pairing with the reader.
- **startIP** - Data type: string. Start of the IP address range assigned for BT client devices.
- **endIP** - Data type: string. End of the IP address range assigned for BT client devices.

Possible Error Conditions

N/A

ReaderDevice.setBTConfig

Modify Bluetooth specific parameters on the reader.

Usage

ReaderDevice. setBTConfig (Discoverable: Boolean, Pairable: Boolean, enablePasswordPairing: string, setPassword: string, startIP: string, endIP: string): void

Parameter(s)

- **Discoverable** - Data type: boolean. Enable Bluetooth discoverable mode.
- **Pairable** - Data type: boolean. Enable pairing of Bluetooth devices.
- **enablePasswordPairing** - Data type: boolean. Enable password based authentication while pairing.
- **setPassword** - Data type: string. Set the password used for pairing.
- **startIP** - Data type: string. Set the start of the IP address range for Bluetooth client devices.
- **endIP** - Data type: string. Set the end of the IP address range for Bluetooth client devices.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

- invalidIpAddr

ReaderDevice.getWebServerSecuritySetting

Retrieve if the web server is configured as secure on the reader.

Usage

ReaderDevice.getWebServerSecuritySetting(isCoreConfig: boolean): IsSecure: boolean, isCoreConfig: boolean

Parameter(s)

- **isCoreConfig** - Data type: boolean. Indicates whether the core configuration is requested.

Return Value(s)

- **IsSecure** - Data type: boolean. Flag indicating if the web server is secure.
- **isCoreConfig** - Data type: boolean. Indicates whether the core configuration is requested.

Possible Error Conditions

N/A

ReaderDevice.setWebServerSecuritySetting

Set the web server to be secure or nonsecure on the reader.

Usage

ReaderDevice.setWebServerSecuritySetting(IsSecure: boolean): void

Parameter(s)

- **IsSecure** - Data type: boolean. Flag indicating if the web server is secure.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

N/A

ReaderDevice.getShellStatus

Get the shell status.

Usage

ReaderDevice.getShellStatus(void): shellState: motorm:ShellMode

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **shellState** - Data type: **motorm:ShellMode**. Shell state configured on the reader.

Possible Error Conditions

N/A

ReaderDevice.setShellStatus

Modify the shell status on the reader.

Usage

ReaderDevice.setShellStatus(shellState: motorm:ShellMode): void

Parameter(s)

- **shellState** - Data type: **motorm:ShellMode**. Sets the shell mode.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

N/A

ReaderDevice.getFTPStatus

Get the FTP status.

Usage

ReaderDevice.getFTPStatus(void): ftpState: motorm:FileServerMode

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **ftpState** - Data type: **motorm:FileServerMode**. The FTP state configured on the reader.

Possible Error Conditions

N/A

ReaderDevice.setFTPStatus

Modify the FTP status on the reader.

Usage

ReaderDevice.setFTPStatus(ftpState: motorm:FileServerMode): void

Parameter(s)

- **ftpState** - Data type: **motorm:FileServerMode**. Sets the FTP mode.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

N/A

ReaderDevice.getUSBMode

Get the USB operation mode.

Usage

ReaderDevice.getUSBMode(void): usbMode: motorm:USBOperationMode, allowLLRPConnectionOverride: boolean

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **usbMode** - Data type: **motorm:USBOperationMode**. Get the USB operation mode configured on the reader.
- **allowLLRPConnectionOverride** - Data type: boolean. Allow overriding the LLRP connection established on the other interface. Does not apply when using USB ActiveSync.

Possible Error Conditions

- unknownUSBMode

ReaderDevice.setUSBMode

Modify the USB operation mode on the reader.

Usage

ReaderDevice.setUSBMode(usbMode: motorm:USBOperationMode,
allowLLRPConnectionOverride: boolean): void

Parameter(s)

- **usbMode** - Data type: **motorm:USBOperationMode**. Set the USB operation mode based on this element.
- **allowLLRPConnectionOverride** - Data type: boolean. Set the value for allowing override of the LLRP connection established on the other interface. Does not apply when using USB ActiveSync.

Return Value(s)

- void. This command does not return a value.

Possible Error Conditions

- noEdit
- invalidOption

ReaderDevice.getLLRPConfig

Retrieve the LLRP configuration items from the reader.

Usage

ReaderDevice.getLLRPConfig(isCoreConfig: boolean): portNum: int, IsSecure: boolean, ValidatePeerInSecureMode: boolean, IsClient: boolean, serverIP: string, ShouldReconnect: boolean, isCoreConfig: boolean

Parameter(s)

- **isCoreConfig** - Data type: boolean. Indicates whether the core configuration is requested.

Return Value(s)

- **portNum** - Data type: integer. The LLRP port number configured on the reader.
- **IsSecure** - Data type: boolean. Indicates whether LLRP is configured for secure mode.
- **ValidatePeerInSecureMode** - Data type: boolean. Indicates whether peer certificate validation is enabled in secure mode.
- **IsClient** - Data type: boolean. Indicates whether LLRP is configured as a client.
- **serverIP** - Data type: string. The IP address of the LLRP server to which the reader is connecting.
- **ShouldReconnect** - Data type: boolean. Indicates whether the reader should attempt to reconnect to the server.
- **isCoreConfig** - Data type: boolean. Indicates whether the core configuration is requested.

Possible Error Conditions

N/A

ReaderDevice.setLLRPConfig

Set one or more LLRP configuration items from the reader.

Usage

ReaderDevice.setLLRPConfig(portNum: int, IsSecure: boolean, ValidatePeerInSecureMode: boolean, IsClient: boolean, serverIP: string, ShouldReconnect: Boolean): void

Parameter(s)

- **portNum** - Data type: integer. The LLRP port number configured on the reader.
- **IsSecure** - Data type: boolean. Indicates whether LLRP is configured for secure mode.
- **ValidatePeerInSecureMode** - Data type: boolean. Indicates whether peer certificate validation is enabled in secure mode.
- **IsClient** - Data type: boolean. Indicates whether LLRP is configured as a client.
- **serverIP** - Data type: string. The IP address of the LLRP server to which the reader is connecting.
- **ShouldReconnect** - Data type: boolean. Indicates whether the reader should attempt to reconnect to the server.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

- invalidValue
- StandardportConflict
- invalidIpAddr

ReaderDevice.isLLRPRunning

Check if the LLRP service is running on the reader.

Usage

ReaderDevice.isLLRPRunning(void): LLRPStartStatus: boolean

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **LLRPStartStatus** - Data type: boolean. Indicates if LLRP is running.

Possible Error Conditions

N/A

ReaderDevice.isLLRPConnected

Check if the reader is connected over LLRP to host.

Usage

ReaderDevice.isLLRPConnected(void): LLRPConnectStatus: boolean

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **LLRPConnectStatus** - Data type: boolean. Indicates if the reader is connected over LLRP.

Possible Error Conditions

N/A

ReaderDevice.ConnectLLRP

Connect or disconnect the reader using LLRP to host.

Usage

ReaderDevice.ConnectLLRP(LLRPConnectAction: boolean): void

Parameter(s)

- **LLRPConnectAction** - Data type: boolean. Connect or disconnect LLRP.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

- llrpServerMode
- llrpConnectFailed
- llrpDisconnectFailed

ReaderDevice.viewSystemLog

View the system log on the reader.

Usage

ReaderDevice.viewSystemLog(filterRules: motorm:systemLogFilterRules, saveFilterSettings: unsignedInt): systemLogContent: string, filterRules: motorm:systemLogFilterRules

Parameter(s)

- **filterRules** - Data type: motorm:systemLogFilterRules. Specify the filter parameters to apply before sending back the system log. Currently supports Severity and Process based filtering.
- **saveFilterSettings** - Data type: unsignedInt. Indicates whether to save the filter configuration.

Return Value(s)

- **systemLogContent** - Data type: string. Current system log on the reader.
- **filterRules** - Data type: motorm:systemLogFilterRules. Provides the filter rules used for filtering.

Possible Error Conditions

- generatefilteredSyslogFailed
- generatefilteredSyslogStarted

ReaderDevice.viewAccessLog

View the access log on the reader.

Usage

ReaderDevice.viewAccessLog(void): accessLogContent: string

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **accessLogContent** - Data type: string. Current access log on the reader.

Possible Error Conditions

N/A

ReaderDevice.viewCurrentCertificateDetails

In secure mode, view the current security certificate on the reader.

Usage

ReaderDevice.viewCurrentCertificateDetails(void): subjectName: string, issuerName: string, validityStart: string, validityEnd: string, serial: string, installTime: string

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **subjectName** - Data type: string. Subject name of the current certificate.
- **issuerName** - Data type: string. Issuer name of the current certificate.
- **validityStart** - Data type: string. Validity start of the current certificate.
- **validityEnd** - Data type: string. Validity end of the current certificate.
- **serial** - Data type: string. Serial of the current certificate.
- **installTime** - Data type: string. Install date and time of the current certificate.

Possible Error Conditions

- failedCertImport

ReaderDevice.setNTPConfig

Set the system time with the option to use NTP-based time synchronization.

Usage

ReaderDevice.setNTPConfig(Server: string): void

Parameter(s)

- **Server** - Data type: string. NTP server IP or name.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

- notFindHost
- invalidIpAddr

ReaderDevice.getWatchdogStatus

Response to **getWatchdogStatus** command.

Usage

ReaderDevice.getWatchdogStatus(void): enableWatchdog: boolean

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **enableWatchdog** - Data type: boolean. Indicates the status of the watchdog (enabled or disabled).

Possible Error Conditions

N/A

ReaderDevice.setWatchdogStatus

Modify (enable or disable) the watchdog on the reader.

Usage

ReaderDevice.setWatchdogStatus(enableWatchdog: boolean): void

Parameter(s)

- **enableWatchdog** - Data type: boolean. Enables or disables the watchdog.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

N/A

ReaderDevice.shutDown

Shutdown or restart the reader.

Usage

ReaderDevice.shutDown(restartNeeded: boolean, forceShutdown: boolean): void

Parameter(s)

- **restartNeeded** - Data type: boolean. Indicates if a restart is needed.
- **forceShutdown** - Data type: boolean. Forces the shutdown even if there are configuration changes.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

- pendingChanges

ReaderDevice.getExtAntennaMode

Get the reader's external antenna mode (monostatic or bistatic).

Usage

ReaderDevice.getExtAntennaMode(void): returnValue: motorm:ExtAntennaMode

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **returnValue** - Data type: **motorm:ExtAntennaMode**. Indicates whether the external antenna is set to monostatic, bistatic, or mixed mode.

Possible Error Conditions

N/A

ReaderDevice.setExtAntennaMode

Set the reader's external antenna to monostatic or bistatic.

Usage

ReaderDevice.setExtAntennaMode(extAntennaMode: motorm:ExtAntennaMode): void

Parameter(s)

- **extAntennaMode** - Data type: **motorm:ExtAntennaMode**. Sets the external antenna to monostatic, bistatic, or mixed mode.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

- invalidOption

ReaderDevice.getReaderVersionInfo

Get the version of software on the reader.

Usage

ReaderDevice.getReaderVersionInfo(void): list of <value: motorm:VersionInfoParamType>

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **value** - Data type: **motorm:VersionInfoParamType**. Flash partition usage information.

Possible Error Conditions

N/A

ReaderDevice.getManufacturer

Get the manufacturer of the reader.

Usage

ReaderDevice.getManufacturer(void): returnValue: string

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **returnValue** - Data type: string. Name of the manufacturer.

Possible Error Conditions

N/A

ReaderDevice.getModel

Get the reader model.

Usage

ReaderDevice.getModel(void): returnValue: string

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **returnValue** - Data type: string. Model name.

Possible Error Conditions

N/A

ReaderDevice.getName

Get the name of the reader.

Usage

ReaderDevice.getName(void): returnValue: string

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **returnValue** - Data type: string. Name of the reader.

Possible Error Conditions

N/A

ReaderDevice.setName

Set the name of the reader.

Usage

ReaderDevice.setName(name: string): void

Parameter(s)

- **name** - Data type: string. Sets a new name for the reader.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

- invalidName
- valueOverSize

ReaderDevice.getDebounceTime

Get the GPI debounce time.

Usage

ReaderDevice.getDebounceTime(void): returnValue: int

Parameter(s)

- Data type: void. This command takes no parameters.

Return Value(s)

- **returnValue** - Data type: int. Current Debounce time.

Possible Error Conditions

- notLoggedIn
- sessionTimeout
- notCurrentSession

ReaderDevice.setDebounceTime

Set the GPI debounce time.

Usage

ReaderDevice.setDebounceTime(dbtime: int): void

Parameter(s)

- **dbtime** - Data Type: int. new debounce time to set.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

- noEdit
- valueOverSize
- invalidData

ReaderDevice.getTimeTicks

Get the time in ticks at the reader.

Usage

ReaderDevice.getTimeTicks(void): returnValue: unsignedLong

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **returnValue** - Data type: unsignedLong. Indicates the uptime in ticks.

Possible Error Conditions

N/A

ReaderDevice.getLocalTime

Get the local time of the reader.

Usage

ReaderDevice.getLocalTime(void): returnValue: dateTime

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **returnValue** - Data type: **dateTime**. Local time of the reader.

Possible Error Conditions

N/A

ReaderDevice.setLocalTime

Set the UTC time of the reader.

Usage

ReaderDevice.setLocalTime(dateTime: dateTime): void

Parameter(s)

- **dateTime** - Data type:**dateTime**. Sets the local time.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

- invalidValue
- operationFailed

ReaderDevice.getAllReadPoints

Get all read points in the reader.

Usage

ReaderDevice.getAllReadPoints(maintenanceMode: boolean, refreshInterval: int): list of <value: string, readPointId: int>

Parameter(s)

- **maintenanceMode** - Data type: boolean. Flag indicating if maintenance mode is enabled.
- **refreshInterval** - Data type: int. Refresh interval value to update status.

Return Value(s)

- **value** - Data type: string. Read point name
- **readPointId** - Data type: integer. Read point ID.

Possible Error Conditions

N/A

ReaderDevice.saveConfigChanges

Commit configuration changes.

Usage

ReaderDevice.saveConfigChanges(void): void

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

- operationFailed
- httpErr
- reinitWait
- changeNeedsReboot
- noChangesCommit

ReaderDevice.discardConfigChanges

Discard configuration changes.

Usage

ReaderDevice.discardConfigChanges(void): void

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

- noChangesDiscard

ReaderDevice.hasConfigChanged

Response to **hasConfigChanged**.

Usage

ReaderDevice.hasConfigChanged(void): configChanged: boolean

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **configChanged** - Data type: boolean. Indicates if the configuration changed.

Possible Error Conditions

N/A

ReaderDevice.getUncommitedConfigChangesDescription

Get the description of configuration changes not yet committed.

Usage

ReaderDevice.getUncommitedConfigChangesDescription(void): list of <configChangeInfo: string>

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **configChangeInfo** - Data type: string. Indicates additional information based on the configuration changes.

Possible Error Conditions

N/A

ReaderDevice.getTimeZones

Get the list of time zones supported in the reader along with the current time zone.

Usage

ReaderDevice.getTimeZones(void): list of <TimeZoneDescription: string>, currentTimeZone: int

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **TimeZoneDescription** - Data type: string. Indicates the time zone description.
- **currentTimeZone** - Data type: integer. Index of the current time zone.

Possible Error Conditions

N/A

ReaderDevice.setTimeZone

Set the time zone in the reader.

Usage

ReaderDevice.setTimeZone(timeZoneIndex: int): void

Parameter(s)

- **timeZoneIndex** - Data type: integer. Index of the time zone to set.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

invalidOption

ReaderDevice.getReaderProfileList

Get the profile present in the reader.

Usage

ReaderDevice.getReaderProfileList(void): list of <value: string, isStandardProfile: boolean>, activeProfileName: string

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **value** - Data type: string. Specifies a single profile name.
- **isStandardProfile** - Data type: boolean. Indicates whether the profile is a standard profile.
- **activeProfileName** - Data type: string. Name of the active profile on the reader, if any. If absent, no profile is active.

Possible Error Conditions

N/A

ReaderDevice.setProfileActive

Activate the chosen profile on the reader.

Usage

ReaderDevice.setProfileActive(ProfileName: string): void

Parameter(s)

- **ProfileName** - Data type: string. Name of the profile.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

- currentProfile
- operationFailed
- nosuchFileOrPath
- invalidProfile

ReaderDevice.deleteProfile

Delete the chosen profile from the reader.

Usage

ReaderDevice.deleteProfile(ProfileName: string): void

Parameter(s)

- **ProfileName** - Data type: string. Name of the profile.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

- currentProfile
- nosuchFileOrPath
- operationFailed

ReaderDevice.importProfileToReader

Import a profile to the reader.

Usage

ReaderDevice.importProfileToReader(ProfileName: string, ProfileData: string, doSetActive: boolean, doSaveChange: boolean, doForceOverwrite: boolean): void

Parameter(s)

- **ProfileName** - Data type: string. Name of the profile.
- **ProfileData** - Data type: string. Content of the profile file.
- **doSetActive** - Data type: boolean. Set the profile active after importing it.
- **doSaveChange** - Data type: boolean. Commit the changes after setting the profile active. Use this if **doSetActive** is true.
- **doForceOverwrite** - Data type: boolean. Flag indicating whether to forcefully overwrite the profile on the reader.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

- currentProfile
- stdProfileNoImport
- maxProfilesPresent
- operationFailed
- nosuchFileOrPath
- invalidProfile

ReaderDevice.exportProfileFromReader

Retrieve the profile file contents from the reader.

Usage

ReaderDevice.exportProfileFromReader(ProfileName: string): ProfileName: string, ProfileData: string

Parameter(s)

- **ProfileName** - Data type: string. Name of the profile.

Return Value(s)

- **ProfileName** - Data type: string. Name of the profile.
- **ProfileData** - Data type: string. Content of the profile file.

Possible Error Conditions

- operationFailed
- nosuchFileOrPath

ReaderDevice.getSerialTimeout

Get the serial connection timeout on the reader.

Usage

ReaderDevice.getSerialTimeout(void): timeOutValue: unsignedInt

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **timeOutValue** - Data type: unsignedInt. Value of the serial timeout in seconds. **0** is no timeout.

Possible Error Conditions

- unsupportedCommand

ReaderDevice.setSerialTimeout

Set the serial connection timeout on the reader.

Usage

ReaderDevice.setSerialTimeout(timeOutValue: unsignedInt): void

Parameter(s)

- **timeOutValue** - Data type: unsignedInt. Value of the serial timeout in seconds. **0** is no timeout.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

- unsupportedCommand
- invalidSerialTimeOut

ReaderDevice.getAntennaCheck

Get whether the antenna check is enabled on the reader.

Usage

ReaderDevice.getAntennaCheck(void): antennaCheckStatus: boolean

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **antennaCheckStatus** - Data type: boolean. Indicates whether to enable or disable the antenna check.

Possible Error Conditions

N/A

ReaderDevice.setAntennaCheck

Enable or disable the antenna check on the reader.

Usage

ReaderDevice.setAntennaCheck(antennaCheckStatus: boolean): void

Parameter(s)

- **antennaCheckStatus** - Data type: boolean. Indicates whether to enable or disable the antenna check.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

N/A

ReaderDevice.getReaderDetails

Get the reader details including **HostName**, **IP-Address**, **Serial Number**, **Software Version**, **NumPorts**.

Usage

ReaderDevice.getReaderDetails(void): HostName: string, IPAddress: string, SerialNumber: string, SoftwareVersion: string, NumPorts: int

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **HostName** - Data type: string. Host name of the reader.
- **IPAddress** - Data type: string. IP address of the reader.
- **SerialNumber** - Data type: string. Serial number of the reader.
- **SoftwareVersion** - Data type: string. Software version of the reader.
- **NumPorts** - Data type: integer. Number of ports in the reader.
- **PowerSource** - Power source type used to power the reader. 0 = 24V DC power, 2 = POE Standard.

Possible Error Conditions

N/A

ReaderDevice.firmwareRevertBack

Revert firmware to last known bootable configuration.

Usage

ReaderDevice.firmwareRevertBack(void): void

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

- bootdatareadfailed
- bootdatawritefailed
- revertBackNotAllowed
- revertbackFailed

ReaderDevice.addIPSecParams

Set the parameters for IPSec.

Usage

ReaderDevice.addIPSecParams(IPMode: string, IPAddress: string, PassCode: string): void

Parameter(s)

- **IPMode** - Data type: string. Mode for IPSec Tunnel/Transport.
- **IPAddress** - Data type: string. IP address of the peer.
- **PassCode** - Data type: string. Pre-shared key.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

- invalidIpAddr
- addIPSecParamsFailed

ReaderDevice.removeIPSecParams

Remove the parameters for IPSec.

Usage

ReaderDevice.removeIPSecParams(IPAddress: string): void

Parameter(s)

- **IPAddress** - Data type: string. IP address of the peer.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

- invalidIpAddr
- removeIPSecParamsFailed

ReaderDevice.getGPIPortStatus

Get the PIN status for all GPI pins.

Usage

ReaderDevice.getGPIPortStatus(void): GPIPortStatus: int

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **GPIPortStatus** - Data type: int. Value of all GPI pins. LSB maps to Pin1, etc.

Possible Error Conditions

- getGPIPortStatusFailed

ReaderDevice.setGPOPinStatus

Set the status of GPO pin.

Usage

ReaderDevice.setGPOPinStatus(PinNumber: int, GPOPintStatus: boolean): void

Parameter(s)

- **PinNumber** - Data type: int. GPO pin number.
- **GPOPintStatus** - Data type: boolean. GPO pIN status. TRUE indicates PIN HIGH, FALSE indicates PIN LOW.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

N/A

ReaderDevice.getEnableRevertBackStatus

Indicates if revert back can be enabled.

Usage

ReaderDevice.getEnableRevertBackStatus(void): canRevertBackEnabled: boolean

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **canRevertBackEnabled** - Data type: boolean. Status for canRevertBackEnabled.

Possible Error Conditions

- getEnableRevertBackStatusFailed

ReaderDevice.getGPOPortStatus

Get the PIN status for all GPO pins.

Usage

ReaderDevice.getGPOPortStatus(void): GPOPortStatus: int

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **GPOPortStatus** - Data type: int. Value of all GPO pins. LSB maps to Pin1, etc.

Possible Error Conditions

- getGPOPortStatusFailed

ReaderDevice.getIPSecParamsList

Get list of parameters added for IPSEC.

Usage

ReaderDevice.getIPSecParamsList(void): IpsecParamsList: string

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **IpsecParamsList** - Data type: string. List of IPSec parameters added.

Possible Error Conditions

- getIPSecParamsListFailed

ReaderDevice.getIdleModeTimeout

Get the idle mode timeout.

Usage

ReaderDevice.getIdleModeTimeout(void): timeoutValue: int

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **timeoutValue** - Data type: int. Current idle mode timeout.

Possible Error Conditions

- notLoggedIn
- sessionTimeout
- notCurrentSession

ReaderDevice.setIdleModeTimeout

Set idle mode timeout.

Usage

ReaderDevice.setIdleModeTimeout(timeOutValue: int): void

Parameter(s)

- **timeoutValue** - Idle mode timeout in seconds. Radio turns off when the reader is idle (there is no RF operation) for the specified time interval. When the radio turns off, the antenna check feature is not supported if inventory is not occurring. Set this value to 0 to disable this feature. The minimum allowed value is 10 and the maximum is 60000 seconds.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

- noEdit valueOverSize
- invalidData

ReaderDevice.processResponseFile

Start file based firmware update.

Usage

ReaderDevice.processResponseFile(updateAllPartitions: boolean): list of <value: string>

Parameter(s)

- **updateAllPartitions** - Data type: boolean. Flag indicating if all the partitions need to be updated.

Return Value(s)

- **value** - Data type: string. List of files to upload.

Possible Error Conditions

N/A

ReaderDevice.startOSupdate

Initiate file based firmware update.

Usage

ReaderDevice.startOSupdate(void): void

Parameter(s)

- **updateCommand** - Data Type: string. Update command with file name to be flashed
- **isForceUpdate** - Data Type: int. Set 1 if all the partitions need to be updated

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

N/A

ReaderDevice.installUserApp

Install the customer application.

Usage

ReaderDevice.installUserApp(appName: string): void

Parameter(s)

- **appName** - Data type: string. Name of the application.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

- packageinstallfailed
- packageDoesNotExist
- operationFailed

ReaderDevice.getMaxUserApps

Get the number of user apps installed in the reader.

Usage

ReaderDevice.getMaxUserApps(void): maxApps: int

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **maxApps** - Data type: int. Maximum number of user apps.

Possible Error Conditions

N/A

ReaderDevice.startUserApp

Start/Stop customer application.

Usage

ReaderDevice.startUserApp(appName: string, start: boolean): void

Parameter(s)

- **appName** - Data type: string. Name of the application.
- **start** - Data type: boolean. Set TRUE to start the application, FALSE to stop.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

- appAlreadyRunning
- startupFileDoesNotExist
- operationFailed

ReaderDevice.autostarUserApp

Autostart the customer application.

Usage

ReaderDevice.autostarUserApp(appName: string, autostart: boolean): void

Parameter(s)

- **appName** - Data type: string. Name of the application.
- **autostart** - Data type: boolean. Set TRUE to auto start the application on boot up.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

- operationFailed

ReaderDevice.uninstalluserapp

Unnstall the customer application.

Usage

ReaderDevice.uninstalluserapp(appName: string): void

Parameter(s)

- **appName** - Data type: string. Name of the application.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

- uninstallAppFailed
- insufficientpermissions

ReaderDevice.getInstalledApps

Get list of installed applications.

Usage

ReaderDevice.getInstalledApps(AppIndex: int): appName: string, runningStatus: boolean, autostart: boolean, metadata: string

Parameter(s)

- **AppIndex** - Data type: int. Index of applications.

Return Value(s)

- **appName** - Data type: string. Maximum number of user applications.
- **runningStatus** - Data type: boolean. Current running status.
- **autostart** - Data type: boolean. Autostart status.
- **metadata** - Data type: string. Metadata of application.

Possible Error Conditions

- operationFailed

ReaderDevice.getCurrentRunStatus

Get the current running status.

Usage

ReaderDevice.getCurrentRunStatus(appName: string): runstatus: boolean

Parameter(s)

- **appName** - Data type: string. Name of the application.

Return Value(s)

- **runstatus** - Data type: boolean. TRUE if the application is running, else set to FALSE

Possible Error Conditions

N/A

ReaderDevice.generateCustomerSupportDataFile

Generates the customer support data file.

Usage

ReaderDevice.generateCustomerSupportDataFile(void): void

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

- generateCSDFailed
- generateCSDStarted

ReaderDevice.purgeLogs

Purges the system logs and temporary files created in log path.

Usage

ReaderDevice.purgeLogs(void): void

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

- purgeLogsFailed

ReaderDevice.getwirelessnwlist

Get the list of wireless networks.

Usage

ReaderDevice. Getwirelessnwlist(void) : essid: string, signalStrength: string, metadata: string

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **essid** - Data type: string. ESSID of the of the available networks.
- **signalStrength** - Data type: string. Signal strength of the available networks.
- **metadata** - Data type: string. Reserved for future use.

Possible Error Conditions

- wirelessScanErr

ReaderDevice.addwirelessnw

Add a Wifi net work setting.

Usage

ReaderDevice.addwirelessnw(essid: string, passkey: string, autoconnect: boolean, metadata: string): void

Parameter(s)

- **essid** - Data type: string. ESSID of the network.
- **passkey** - Data type: string. Passkey of the network.
- **autoconnect** - Data type: boolean. Connect automatically on restart.
- **metadata** - Data type: string. Reserved for future use.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

- operationFailed

ReaderDevice.getwirelessnwproperties

Get the details of the connected wireless network.

Usage

ReaderDevice.getwirelessnwproperties(void): essid: string, signalstrength: string, connectionstatus: motorm:WirelessConnectionStatus, ipaddress: string

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **essid** - Data type: string. ESSID of the network.
- **signalstrength** - Data type: string. Signal strength of the connected network. Possible values are poor, average, excellent, or in % terms.
- **connectionstatus** - Data type: motorm:WirelessConnectionStatus. Get wireless network connection status.
- **ipaddress** - Data type: string. IP address.

Possible Error Conditions

- getGPOPortStatusFailed

ReaderDevice.getwirelessconfiguredparams

Get the details of parameters configured for the wireless network.

Usage

ReaderDevice.getwirelessconfiguredparams(void): essid: string, passkey: string, autoconnect: boolean

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **essid** - Data type: string. ESSID of the configured network.
- **passkey** - Data type: string. Passkey of the configured network.
- **autoconnect** - Data type: boolean. Indicates whether to autoconnect to the network after reader restart.

Possible Error Conditions

- operationFailed

ReaderDevice.disconnectwirelessnw

Disconnect the Wifi connection.

Usage

ReaderDevice.disconnectwirelessnw(void): void

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

- addwirelessnwfailed

ReaderDevice.getGPIOSettings

Get the mapping information of GPIO.

Usage

ReaderDevice.getGPIOSettings(void): isGPI1MappedToRadioGPIO1: boolean, isGPI2MappedToRadioGPIO2: boolean, isGPO1MappedToRadioGPIO1: boolean, isGPO2MappedToRadioGPIO2: boolean, maxNumOfGPIOs: int

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **isGPI1MappedToRadioGPIO1** - Data type: boolean. Indicates whether GPI1 is mapped with Radio GPIO1.
- **isGPI2MappedToRadioGPIO2** - Data type: boolean. Indicates whether GPI2 is mapped with Radio GPIO2.
- **isGPO1MappedToRadioGPIO1** - Data type: boolean. Indicates whether GPO1 is mapped with Radio GPIO1.
- **isGPO2MappedToRadioGPIO2** - Data type: boolean. Indicates whether GPO2 is mapped with Radio GPIO2.
- **maxNumOfRadioGPIOs** - Data type: int. Maximum number of radio GPIOs supported in this reader.

Possible Error Conditions

N/A

ReaderDevice.setGPIOSettings

Set GPIO mapping information.

Usage

ReaderDevice.setGPIOSettings(isGPI1MappedToRadioGPIO1: boolean, isGPI2MappedToRadioGPIO2: boolean, isGPO1MappedToRadioGPIO1: boolean, isGPO2MappedToRadioGPIO2: boolean): void

Parameter(s)

- **isGPI1MappedToRadioGPIO1** - Data type: boolean. Indicates whether GPI1 is mapped with Radio GPIO1.
- **isGPI2MappedToRadioGPIO2** - Data type: boolean. Indicates whether GPI2 is mapped with Radio GPIO2.
- **isGPO1MappedToRadioGPIO1** - Data type: boolean. Indicates whether GPO1 is mapped with Radio GPIO1.
- **isGPO2MappedToRadioGPIO2** - Data type: boolean. Indicates whether GPO2 is mapped with Radio GPIO2.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

N/A

ReaderDevice.resetToFactoryDefaults

Return the reader to factory default configuration after a system reboot.

Usage

ReaderDevice.resetToFactoryDefaults(void): void

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

N/A

ReaderDevice.setSystemLogConfiguration

Set system log configuration.

Usage

ReaderDevice.setSystemLogConfiguration(remoteServerIP: string, remoteServerPort: int,remoteServerMinSeverity: int): void

Parameter(s)

- **remoteServerIP** - Data type: string. IP address of remote logger.
- **remoteServerPort** - Data type: int. Port number of remote logger.
- **remoteServerMinSeverity** - Data type: int. Minimum severity above which to send to remote logger.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

- invalidIpAddr
- valueOverSize
- invalidSnmpVersion

ReaderDevice.getSystemLogConfiguration

Get system log configuration.

Usage

ReaderDevice.getSystemLogConfiguration(void): remoteServerIP: string, remoteServerPort: int, remoteServerMinSeverity: int

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- remoteServerIP - Data Type: string. IP address of Remote Logger
- remoteServerPort - Data Type: int. Port Number of Remote Logger
- remoteServerMinSeverity - Data Type: int. Minimum Severity above which to be send to Remote Logger

Possible Error Conditions

N/A

ReaderDevice.getRadioModuleOnTime

Get duration in seconds, since power-on of the reader, for which the radio module was powered on.

Usage

ReaderDevice.getRadioModuleOnTime(void): radioModuleOnTime: int

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **radioModuleOnTime** - Data type: int. Time elapsed in seconds since radio module was turned on.

Possible Error Conditions

N/A

ReaderDevice.setDiagnosticMode

Configure diagnostic mode parameters on the reader.

Usage

ReaderDevice.setDiagnosticMode(enableDiagnostics: boolean, extended: boolean): void

Parameter(s)

- **enableDiagnostics** - Data type: boolean. Generic flag, indicating which diagnostics to turn on in the reader.
- **extended** - Data type: boolean. Enable extended diagnostic monitoring functions. Applies only if **enableDiagnostic** is true. This parameter is for internal use only.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

N/A

ReaderDevice.startReaderDiagnostics

Start the diagnostics of the reader.

Usage

ReaderDevice.startReaderDiagnostics(void): void

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

N/A

ReaderDevice.getEventAmbientTemperatureHighAlarmCount

Get the number of the ambient temperature high alarm events.

Usage

ReaderDevice.getEventAmbientTemperatureHighAlarmCount(void): returnValue: int

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **returnValue** - Data Type: int.

Possible Error Conditions

N/A

ReaderDevice.getEventAmbientTemperatureCriticalAlarmCount

Get the number of the ambient temperature critical alarm events.

Usage

ReaderDevice.getEventAmbientTemperatureCriticalAlarmCount(void): returnValue: int

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **returnValue** - Data Type: int.

Possible Error Conditions

N/A

ReaderDevice.getEventPATemperatureHighAlarmCount

Get the number of the PA temperature high alarm events.

Usage

ReaderDevice.getEventPATemperatureHighAlarmCount(void): returnValue: int

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **returnValue** - Data Type: int.

Possible Error Conditions

N/A

ReaderDevice.getEventPATemperatureCriticalAlarmCount

Get the number of the PA temperature critical alarm events.

Usage

ReaderDevice.getEventPATemperatureCriticalAlarmCount(void): returnValue: int

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **returnValue** - Data Type: int.

Possible Error Conditions

N/A

ReaderDevice.getEventForwardPowerHighAlarmCount

Get the number of the forward power high alarm events.

Usage

ReaderDevice.getEventForwardPowerHighAlarmCount(void): returnValue: int

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **returnValue** - Data Type: int.

Possible Error Conditions

N/A

ReaderDevice.getEventForwardPowerLowAlarmCount

Get the number of the forward power low alarm events.

Usage

ReaderDevice.getEventForwardPowerLowAlarmCount(void): returnValue: int

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **returnValue** - Data Type: int.

Possible Error Conditions

N/A

ReaderDevice.getEventReversePowerHighAlarmCount

Get the number of the reverse power high alarm events.

Usage

ReaderDevice.getEventReversePowerHighAlarmCount(void): returnValue: int

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **returnValue** - Data Type: int.

Possible Error Conditions

N/A

ReaderDevice.getEventEchoThresholdAlarmCount

Get the number of the echo threshold alarm events.

Usage

ReaderDevice.getEventEchoThresholdAlarmCount(void): returnValue: int

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **returnValue** - Data Type: int.

Possible Error Conditions

N/A

ReaderDevice.getEventDatabaseWarningCount

Get the number of the database warning events.

Usage

ReaderDevice.getEventDatabaseWarningCount(void): returnValue: int

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **returnValue** - Data Type: int.

Possible Error Conditions

N/A

ReaderDevice.getEventDatabaseErrorCount

Get the number of the database error events.

Usage

ReaderDevice.getEventDatabaseErrorCount(void): returnValue: int

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **returnValue** - Data Type: int.

Possible Error Conditions

N/A

ReaderDevice.getEventGPIOInformationCount

Get the number of the GPIO information events.

Usage

ReaderDevice.getEventGPIOInformationCount(void): returnValue: int

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **returnValue** - Data Type: int.

Possible Error Conditions

N/A

ReaderDevice.getRadioPowerState

Get the current radio power state information.

Usage

ReaderDevice.getRadioPowerState(void): returnValue: int

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **returnValue** - Data Type: int. int. The current state (On = 0 / Off = 1) of the radio.

Possible Error Conditions

N/A

ReaderDevice.getUSBState

Get the current USB device status information.

Usage

ReaderDevice.getUSBState(void): PortStatus: motorm:USBPortStatus, list of <DeviceInfo:motorm:USBDeviceInfo>

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **PortStatus** - Data Type: motorm:USBPortStatus. The status of the USB connection (Connected/Unconnected/Powered off).
- **DeviceInfo** - Data Type: motorm:USBDeviceInfo. Device Information of each USB device connected.
  1. Type of device ("Mass Storage Device", "Wireless LAN adapter" or "Bluetooth adapter")
  2. Model Name of the USB device connected
  3. Vendor ID of the USB device manufacturer
  4. Serial Number of the connected USB Device.

Possible Error Conditions

N/A

ReaderDevice.viewMACErrorLog

View the MAC error log on the reader.

Usage

ReaderDevice.viewMACErrorLog(void): MACErrorLogContent: string

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **MACErrorLogContent** - Data type: string. Current MAC Error log on the reader.

Possible Error Conditions

N/A

ReaderDevice.getPowerNegotiation

Get whether Power Negotiation is enabled on the Reader.

Usage

ReaderDevice. getPowerNegotiation (void): powerNegotiationStatus : boolean

Parameter(s)

- Data Type: void. This command takes no parameters.

Return Value(s)

- powerNegotiationStatus - Data Type: boolean. The state on whether powerNegotiationStatus is enabled or disabled.

Possible Error Conditions

N/A

ReaderDevice.setPowerNegotiation

Set whether Power Negotiation is to be enabled on the Reader.

Usage

ReaderDevice. setPowerNegotiation (powerNegotiationStatus : boolean): void

Parameter(s)

- powerNegotiationStatus Data Type: boolean. Power Negotiation status to be set on the reader.

Return Value(s)

- Data Type: void. This command does not return a value.

Possible Error Conditions

N/A

ReaderDevice.getAllowGuestStatus

Get whether Guest login is allowed to reader.

Usage

ReaderDevice. getAllowGuestStatus (void): setAllowGuestStatus: boolean

Parameter(s)

- Data Type: void. This command takes no parameters.

Return Value(s)

- setAllowGuestStatus- Data Type: boolean. The state on whether Guest login is enabled or disabled.

Possible Error Conditions

N/A

ReaderDevice.setAllowGuestStatus

Set whether Guest login is to be enabled on the Reader.

Usage

ReaderDevice. setAllowGuestStatus (allowGuestStatus: boolean): void

Parameter(s)

o allowGuestStatusData Type: boolean. Allow Guest status to be set on the reader.

Return Value(s)

o Data Type: void. This command does not return a value.

Possible Error Conditions

"insufficientpermissions

ReaderDevice.manageLicense

RM Command to manage license on the FX Series RFID Reader. Supports activating and returning license for a software feature.

Usage

ReaderDevice.manageLicense(operationType : string, isOfflineSource : Boolean, serverURL : string, activationId : string, installApp : Boolean): void

Parameter(s)

- **operationType** - Data Type: string. Operation to be performed related to managing license. Can be ActivateLicense or ReturnLicense
- **isOfflineSource** - Data Type: Boolean. License server source type to be used in managing the license. In case of Offline source license bin file should be transferred to reader before this command is called
- **serverURL** – Data Type: string. URL of the license server that is to be used Activate or Return License
- **activationId** - Data Type: string. Activation Id of the customer that the license is linked to.
- **installApp** - Data Type: Boolean. Activation Id of the customer that the license is linked to.

Return Value(s)

Data Type: void. This command does not return a value.

Possible Error Conditions

- licenseCheckFailed
- onlineLicenseAcquireFailed
- offlineLicenseAcquiredFailed
- releaseLicenseFailed

ReaderDevice.getNodeJSPortnum

Query the reader for current Node JS port Number.

Usage

ReaderDevice.getNodeJSPortnum (isCoreConfig : boolean): portNum: int

Parameter(s)

isCoreConfig - Data Type: boolean. Is the core config asked for.

Return Value(s)

portNum - Data Type: int. Node JS port from Reader.

Possible Error Conditions

N/A

ReaderDevice.setNodeJSPortnum

Set Node JS port number.

Usage

ReaderDevice.setNodeJSPortnum(portNum : int): void

Parameter(s)

portNum - Data Type: int. Node JS port number

Return Value(s)

Data Type: void. This command does not return a value.

Possible Error Conditions

N/A

ReaderDevice.setLEDFirmwareUpdate

Control BOOT LED for firmware update process

Usage

ReaderDevice.setLEDFirmwareUpdate(ledColor : string, duration : int, blink : boolean): void

Parameter(s)

- **ledColor** - Data Type: string. Define the color for which LED should glow
- **duration** - Data Type: int. Duration in seconds, for which LED should perform the operation
- **blink** - Data Type: Boolean. Set TRUE to make LED blink

Return Value(s)

Data Type: void. This command does not return a value.

Possible Error Conditions

N/A

ReaderDevice.getInstalledLicenseList

Command to get the installed licenses on the reader.

Usage

ReaderDevice.getInstalledLicenseList(void): list of <value : string>

Parameter(s)

Data Type: void. This command takes no parameters.

Return Value(s)

Value - Data Type: string. List of licenses that have been installed on the reader with the relevant information.

Possible Error Conditions

- evalLicenseDateValidityFailed
- getInstalledLicenseListFailed

ReaderDevice.manageFXEasyConnection

Command to manage the FX Connect settings on the reader.

Usage

ReaderDevice.manageFXEasyConnection(isOperationSe : boolean, mode : int, httpPostConnectConfig : motorm: httpPostConnectSettings, httpPostProxy : motorm: httpPostProxySettings, outputDataFormat : motorm: outputDataFormatSettings, TCPIPConfig : motorm: TCPIPConfigSettings, tagMetaData : int, inventoryControl : string, heartBeatPeriod : int, isAutostart : boolean, list of< antennaConfig : motorm: AntennaConfiguration>): isOperationSe : boolean, mode : int, httpPostConnectConfig : motorm: httpPostConnectSettings, httpPostProxy : motorm: httpPostProxySettings, outputDataFormat : motorm: outputDataFormatSettings, TCPIPConfig : motorm: TCPIPConfigSettings, tagMetaData : int, inventoryControl : string, heartBeatPeriod : int, isAutostart : boolean, list of< antennaConfig : motorm: AntennaConfiguration>

Parameter(s)

- isOperationSe - Data Type: boolean. Indicates whether the user is trying to set or get the FX Easy Connect settings
- mode – Data Type: int. FX Easy Connect mode
- httpPostConnectConfig – Data Type: motorm: httpPostConnectSettings. HTTP POST Server Settings Parameters for FX Connect in HTTP Post mode
- httpPostProxy – Data Type: motorm: httpPostProxySettings. Connection Settings Parameters for FX Connect in HTTP Post mode
- outputDataFormat – Data Type: motorm:outputDataFormatSettings. Format of Output data for FX Connect.
- TCPIPConfig – Data Type: motorm:TCPIPConfigSettings. TCP/IP Generic socket configuration
- tagMetaData – Data Type: int. Meta Data information for FX Connect
- inventoryControl – Data Type: string. Inventory Control Parameters for FX Connect
- heartBeatPeriod – Data Type: string. Period in seconds after which the heartbeat message needs to be sent for FX Connect. 0 will be disable heartbeat.
- isAutostart – Data Type: boolean. If Autostart enabled Inventory will run on boot-up
- antennaConfig – Data Type: motorm:AntennaConfiguration. Antenna configuration for antennas used with FX Connect.

Return Value(s)

- isOperationSe - Data Type: boolean. Indicates whether the user is trying to set or get the FX Easy Connect settings
- mode – Data Type: int. FX Easy Connect mode
- httpPostConnectConfig – Data Type: motorm: httpPostConnectSettings. HTTP POST Server Settings Parameters for FX Connect in HTTP Post mode
- httpPostProxy – Data Type: motorm: httpPostProxySettings. Connection Settings Parameters for FX Connect in HTTP Post mode
- outputDataFormat – Data Type: motorm:outputDataFormatSettings. Format of Output data for FX Connect.
- TCPIPConfig – Data Type: motorm:TCPIPConfigSettings. TCP/IP Generic socket configuration
- tagMetaData – Data Type: int. Meta Data information for FX Connect

- inventoryControl – Data Type: string. Inventory Control Parameters for FX Connect
- heartBeatPeriod – Data Type: string. Period in seconds after which the heartbeat message needs to be sent for FX Connect. 0 will be disable heartbeat.
- isAutostart – Data Type: boolean. If Autostart enabled Inventory will run on boot-up
- antennaConfig – Data Type: motorm:AntennaConfiguration. Antenna configuration for antennas used with FX Connect.

Possible Error Conditions

N/A

ReaderDevice.getSerialConfig

Get Serial Port Configuration.

Usage

ReaderDevice.getSerialConfig (isCoreConfig : boolean): Mode : int, Baudrate : int, Databits : int, Parity : string, Stopbits : int, Flowcontrol : string, TagMetaData : int, InventoryControl : string, isAutostart : boolean

Parameter(s)

isCoreConfig- Data Type: boolean. Set TRUE if core config is asked for

Return Value(s)

- Mode - Data Type: int. Configure Serial Port mode.
- Baudrate – Data Type: int. Configure Serial Port Baudrate.
- Databits – Data Type : int. Configure Serial Port Databits
- Parity – Data Type: string. set Parity for Serial Port
- Stopbits – Data Type: int. Set Stopbits for Serial Port
- Flowcontrol – Data Type: string. set Flowcontrol for serial Port
- TagMetaData – Data Type: int. Meta Data information for Serial Port Push Data
- InventoryControl – Data Type: string. set Inventory Control Parameters for serial Port
- isAutostart - Data Type: boolean. Set TRUE to auto start Inventory on boot-up.

Possible Error Conditions

N/A

ReaderDevice.setSerialConfig

Modify Serial Port specific parameters on the reader.

Usage

ReaderDevice.setSerialConfig (Mode : int, Baudrate : int, Databits : int, Parity : string, Stopbits : int, Flowcontrol : string, TagMetaData : int, InventoryControl : string, isAutostart : boolean): void

Parameter(s)

- Mode - Data Type: int. Configure Serial Port mode.
- Baudrate - Data Type: int. Configure Serial Port Baudrate.
- Databits - Data Type : int. Configure Serial Port Databits
- Parity - Data Type: string. set Parity for Serial Port
- Stopbits - Data Type: int. Set Stopbits for Serial Port
- Flowcontrol - Data Type: string. set Flowcontrol for serial Port
- TagMetaData - Data Type: int. Meta Data information for Serial Port Push Data
- InventoryControl - Data Type: string. set Inventory Control Parameters for serial Port
- isAutostart - Data Type: boolean. Set TRUE to auto start Inventory on boot-up.

Return Value(s)

Data Type: void. This command does not return a value.

Possible Error Conditions

N/A

ReaderDevice.getTempSensorData

Query the reader for current PA Temp and ambient temp.

Usage

ReaderDevice.getTempSensorData (:void): paTemp: float, ambientTemp: float

Parameter(s)

Data Type: void. This command takes no parameters.

Return Value(s)

- **paTemp** - Data Type: float. Pa Temp sensor data collected from Reader.
- **ambientTemp** - Data Type: float. Ambient Temp sensor data collected from Reader.

Possible Error Conditions

N/A

ReaderDevice.get802.1xEAPInterfaces

Gets the 802.1x supported network interfaces.

Usage

ReaderDevice.get802.1xEAPInterfaces (void): list of <value: string>

Parameters(s)

Data Type: void. This command takes no parameters.

Return Value(s)

- Interfaces - Data Type: string. List of network Interfaces supporting 802.1x.

Possible Error Conditions

operationFailed

ReaderDevice.get802.1xEAPTypes

Gets the list of supported EAP Authentication methods.

Usage

ReaderDevice. get802.1xEAPTypes (interface: string): list of <value: string>

Parameters(s)

- Interface - Data Type: string. The Interface for which to retrieve the supported EAP methods.

Return Value(s)

- EAPType - Data Type: string. List of support EAP authentication methods.

Possible Error Conditions

operationFailed

ReaderDevice.get802.1xEAPInnerTypes

Gets the list of inner authentication methods for a given interface and outer authentication type.

Usage

ReaderDevice. get802.1xEAPInnerTypes (interface: string, EAPType: string): list of <value: string>

Parameters(s)

- Interface - Data Type: string. The Interface for which to retrieve the supported EAP methods.
- EAPType - Data Type: string. Type of outer authentication method.

Return Value(s)

EAPInnerType - Data Type: string. List of support EAP Inner authentication methods.

Possible Error Conditions

operationFailed

ReaderDevice.get802.1xEAPProperties

Get the details of specific interface 802.1xEAP configuration.

Usage

ReaderDevice. get802.1xEAPInnerTypes (interface: string): isEAPconfigured: Boolean, EAPType: string, EAPInnerType: string, username: string, passkey: string, certname: string, certificateStatus: string, autoconnect: int

Parameters(s)

- Interface - Data Type: string. The Interface for which to retrieve the supported EAP properties.

Return Value(s)

- isEAPconfigured - Data Type: boolean. Is EAP is configured for specific interface or not.
- EAPType - Data Type: string. EAP Type configured.
- EAPInnerType - Data Type: string. EAP Inner Type configured.
- username - Data Type: string. Username for authentication.
- passkey - Data Type: string. Password for authentication
- certname - Data Type: string. Name of the certificate to be used for authentication.
- certificateStatus - Data Type: string. Certificate Status valid or not else not found.
- autoconnect - Data Type: int. Specifies whether the device automatically connects to 802.1x network or not.

Possible Error Conditions

operationFailed

ReaderDevice.userAdd

Add the user along with password and permission specified.

Usage

ReaderDevice.userAdd(userName: string, isPassLoginEnabled: Boolean, password: string, isKeyLoginEnabled: Boolean, keyFilePath:" string): void

Parameter(s)

- userName - Data Type: string. Name of the user to be added.

- isPassLoginEnabled - Data Type: Boolean. Is the user password login enabled.
- password - Data Type:string. Password for user login.
- isKeyLoginEnabled - Data Type:Boolean. Is the user key based login enabled.
- keyFilePath - Data Type:string. key file path.

Return Value(s)

- Data Type: void. This command does not return a value.

Possible Error Conditions

invalidName

invalidPassword

invalidAccess

ReaderDevice.userDel

Delete the user specified by the username

Usage

ReaderDevice.userDel(username: string, isRmHome: boolean): void

Parameter(s)

- username - Data Type: string.Name of the user to be deleted.
- isRmHome - Data Type:Boolean. user home dir need to be removed.

Return Value(s)

Data Type: void. This command does not return any value.

Possible Error Conditions

UserDoesNotExits

ReaderDevice.userMod

Modify the user's password or login key.

Usage

ReaderDevice.userMod(userName: string, changePasswd: string, isPassLoginEnabled: string, isKeyLoginEnabled: string, keyFilePath: string): void

Parameter(s)

- userName - Data Type: string. Name of the user to be added.
- changePasswd - Data Type: string. Password for the user.

- isPassLoginEnabled - Data Type: Boolean. Is the user password login enabled.
- password - Data Type:string. Password for user login.
- isKeyLoginEnabled - Data Type:Boolean. Is the user key based login enabled.
- keyFilePath - Data Type:string. key file path.

Return Value(s)

Data Type: void. This command does not return a value.

Possible Error Conditions

UserDoesNotExits

ReaderDevice.set802.1xEAPProperties

Configures the 802.1xEAP network.

Usage

ReaderDevice. set802.1xEAPInnerTypes (interface: string, EAPType: string, EAPInnerType: string, username: string, passkey: string, certname: string, autoconnect: int): void

Parameters(s)

- Interface - Data Type: string. The Interface for which to retrieve the supported EAP properties.
- EAPType - Data Type: string. EAP Type configured.
- EAPInnerType - Data Type: string. EAP Inner Type configured.
- username - Data Type: string. Username for authentication.
- passkey - Data Type: string. Password for authentication
- certname - Data Type: string. Name of the certificate to be used for authentication.
- autoconnect - Data Type: int. Specifies whether the device automatically connects to 802.1x network or not.

Return Value(s)

- Data Type: void. This command does not return a value.

Possible Error Conditions

operationFailed

ReaderDevice.enrollToCloud

Initiate enrollment to cloud service.

Usage

ReaderDevice. enrollToCloud (provider: int, code: string, autoConnect: boolean): void

Parameter(s)

- provider - Data Type: int. Cloud service provider ID. default 0 for Zebra Savanna over GCP.
- code - Data Type: string. Cloud enrollment claim code.
- autoConnect - Data Type: Boolean. Initiate connection after enrollmen

Return Value(s)

Data Type: void. This command does not return a value..

Possible Error Conditions

operationFailed

ReaderDevice.disEnrollFromCloud

Remove enrollment with cloud service.

Usage

ReaderDevice. disEnrollFromCloud (code: string): isDisEnrolled: boolean

Parameter(s)

- code - Data Type: string. Disenrollment code
- Return Value(s)
- isDisEnrolled - Data Type: boolean. Flag indicating status of dis-enrollment from cloud.

Possible Error Conditions

operationFailed

ReaderDevice.isEnrolledToCloud

Check if reader is enrolled to cloud service.

Usage

ReaderDevice. isEnrolledToCloud (void): isEnrolled: Boolean, provider: int

Parameter(s)

- Data Type: void. This command does not take a parameter.

Return Value(s)

- isEnrolled - Data Type: boolean. Flag indicating status of enrollment.

- Provider - Data type: int. Cloud service to which reader is enrolled.

Possible Error Conditions

operationFailed

ReaderDevice.isConnectedToCloud

Check if reader is connected to cloud service.

Usage

ReaderDevice. isConnectedToCloud (void): isConnected: Boolean, ConnectionStatus: string

Parameter(s)

- Data Type: void. This command does not take a parameter.

Return Value(s)

- isConnected - Data Type: boolean. Flag indicating status of cloud service connection.
- ConnectionStatus - Data type: string. Cloud connection status for Data, Control and Management path as JSON object.

Possible Error Conditions

operationFailed

ReaderDevice.connectToCloud

Connect to cloud service.

Usage

ReaderDevice. connectToCloud (void): isConnected: Boolean

Parameter(s)

- Data Type: void. This command does not take a parameter.

Return Value(s)

- isConnected - Data Type: boolean. Flag indicating status of cloud service connection.

Possible Error Conditions

operationFailed

ReaderDevice.disconnectFromCloud

Disconnect from cloud service.

Usage

ReaderDevice. disconnectFromCloud (void): isDisconnected: Boolean

Parameter(s)

- Data Type: void. This command does not take a parameter.

Return Value(s)

- isDisconnected- Data Type: boolean. Flag indicating status of cloud service disconnection.

Possible Error Conditions

operationFailed

ReaderDevice.autoConnectToCloud

Set auto connect for cloud connect mode.

Usage

ReaderDevice. autoConnectToCloud (void): autoConnect: Boolean

Parameter(s)

- autoConnect - Data Type: Boolean. auto connect flag.

Return Value(s)

- Data Type: void. This command does not return a value.

Possible Error Conditions

operationFailed

ReaderDevice.isAutoConnectToCloud

Check if auto connect is enabled for cloud.

Usage

ReaderDevice. isAutoConnectToCloud (void): isAutoConnect: Boolean

Parameter(s)

- Data Type: void. This command does not accept any parameters.

Return Value(s)

- isAutoConnect - Data Type: Boolean. Flag indicating auto connect state.

Possible Error Conditions

operationFailed

ReaderDevice.importCloudConfigToReader

Import cloud configuration file content onto the reader.

Usage

ReaderDevice. importCloudConfigToReader (CloudConfigData: string, GPIOConfigData: string): void

Parameter(s)

- CloudConfigData - Data Type: string. Content of the configuration file.
- GPIOConfigData - Data Type: string. Content of the GPIO configuration.

Return Value(s)

- Data Type: void. This command does not return a value.

Possible Error Conditions

operationFailed

invalidConfig

ReaderDevice.exportCloudConfigFromReader

Retrieve current cloud configuration from the reader.

Usage

ReaderDevice. exportCloudConfigFromReader (void): ConfigData: string

Parameter(s)

- Data Type: void. This command does not take a parameter.

Return Value(s)

- CloudConfigData - Data Type: string. Content of the configuration file.

Possible Error Conditions

operationFailed

nosuchFileOrPath

ReaderDevice.exportGPIOConfigFromReader

Retrieve GPIO configuration from the reader.

Usage

ReaderDevice. exportGPIOConfigFromReader (void): GPIOConfigData: string

Parameter(s)

- Data Type: void. This command does not take a parameter.

Return Value(s)

- GPIOConfigData - Data Type: string. Content of the GPIO configuration.

Possible Error Conditions

operationFailed

nosuchFileOrPath

ReaderDevice.manageCloudEndpoints

Cloud Endpoints management.

Usage

ReaderDevice. manageCloudEndpoints (operation: string, data: string): data: string

Parameter(s)

- operation - Data Type: string. ADD/DEL/UPDATE/VIEW Endpoint Configuration.
- Data - Data Type: string. data for add, del and empty.

Return Value(s)

- data - Data Type: string. Success or Failure reason for ADD/DEL/UPDATE command and Endpoints for VIEW

Possible Error Conditions

operationFailed

ReaderDevice.cloudEndpointsMapping

Map endpoints to cloud configuration (generate cloud.json from endpoints).

Usage

ReaderDevice. cloudEndpointsMapping (operation: string, data: string): data: string

Parameter(s)

- operation - Data Type: string. VIEW/UPDATE Endpoints mapped.
- Data - Data Type: string. endpoints data.

Return Value(s)

- data - Data Type: string. configuration data.

Possible Error Conditions

operationFailed

ReaderDevice.registerIoTConnector

Registers device to provided registration service and configures IoT Connector on successful registration.

Usage

ReaderDevice. registerIoTConnector (registrationDetails: string): void

Parameter(s)

- registerIoTConnector - Data Type: string. JSON object with registration parameters.

Return Value(s)

- Data Type: void. This command does not return a value.

Possible Error Conditions

operationFailed

ReaderDevice.autoEnrollIoTConnector

Trigger auto enrollment

Usage

ReaderDevice. autoEnrollIoTConnector (void): void

Parameter(s)

- Data Type: void. This command does not take a parameter.

Return Value(s)

- Data Type: void. This command does not return a value.

Possible Error Conditions

operationFailed

ReaderDevice.importOperatingModeToReader

Import Operating Mode configuration.

Usage

ReaderDevice. importOperatingModeToReader (OperatingMode: string): void

Parameter(s)

- OperatingMode - Data Type: string. JSON configuration.

Return Value(s)

- Data Type: void. This command does not return a value.

Possible Error Conditions

operationFailed

invalidConfig

ReaderDevice.exportOperatingModeFromReader

Retrieve current Operating Mode configuration from the reader.

Usage

ReaderDevice. exportOperatingModeFromReader (void) OperatingMode: string

Parameter(s)

- Data Type: void. This command does not take a parameter.

Return Value(s)

- OperatingMode - Data Type: string. Content of the Operating Mode configuration.

Possible Error Conditions

operationFailed

nosuchFileOrPath

ReaderDevice.addCAcert

Adds the CA certificate to the trusted store.

Usage

ReaderDevice. addCAcert (CAcertName: string, CAfileContent: string) void

Parameter(s)

- CAcertName - Data Type: string. CA certificate name.
- CAfileContent - Data Type: string. Contents of the CA certificate in pem format.

Return Value(s)

- Data Type: void. This command does not return a value.

Possible Error Conditions

operationFailed

ReaderDevice.deleteCAcert

Delete the specified CA certificate.

Usage

ReaderDevice. deleteCAcert (CAcertName: string) void

Parameter(s)

- CAcertName - Data Type: string. Name of the CA certificate to delete

Return Value(s)

- Data Type: void. This command does not return a value.

Possible Error Conditions

operationFailed

ReaderDevice.listCAcerts

Lists the user installed CA certificates on the reader.

Usage

ReaderDevice. listCAcerts (void) certificates: list of <string>

Parameter(s)

- Data Type: void. This command does not take a parameter.

Return Value(s)

- certificates - Data Type: list of <string>. List of CA certificates that have been installed on the reader

Possible Error Conditions

operationFailed

ReaderDevice. importSSHKey

Import SSH keys into the reader.

Usage

ReaderDevice. importSSHKey (publicKey: string, privateKey: string) void

Parameter(s)

- **publicKey - Data Type:** string. Public key string.
- **privateKey - Data Type:** string. Private key string

Return Value(s)

- **Data Type:** void. This command does not return a value.

Possible Error Conditions

noEdit

operationFailed

notRSASSHKey

invalidSSHKeyLength

invalidSSHKeyLength

mismatchedSSHKey

fifoOpenFailed

nosuchFileOrPath

ReaderDevice. exportSSHKey

Export SSH keys from the reader.

Usage

ReaderDevice. exportSSHKey (void) PublicKey: string

Parameter(s)

- **Data Type:** void. This command does not take a parameter.

Return Value(s)

- **Public key - Data Type:** string. Returns the reader's public SSH key.

Possible Error Conditions

noEdit

nosuchFileOrPath

AntennaReadPoint.getSupportedAirProtocols

Get the supported air protocols.

Usage

AntennaReadPoint.getSupportedAirProtocols(void): list of <value: int>

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **value** - Data type: integer. Specifies a single air protocol.

Possible Error Conditions

N/A

AntennaReadPoint.getCurrentAirProtocol

Get the current air protocol.

Usage

AntennaReadPoint.getCurrentAirProtocol(void): currentAirProtocol: int

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- currentAirProtocol - Data type: integer. The current air protocol set in the reader.

Possible Error Conditions

N/A

AntennaReadPoint.setAirProtocol

Set the air protocol for the read point.

Usage

AntennaReadPoint.setAirProtocol(airProtocolType: int): void

Parameter(s)

- **airProtocolType** - Data type: integer. Air protocol type to set to the readpoint.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

N/A

AntennaReadPoint.getTransmitPowerLevel

Get the transmit power level for the readpoint.

Usage

AntennaReadPoint.getTransmitPowerLevel(void): transmitPowerLevel: int

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **transmitPowerLevel** - Data type: integer. The transmit power level for the readpoint.

Possible Error Conditions

N/A

AntennaReadPoint.setTransmitPowerLevel

Set the transmit power level for the readpoint.

Usage

AntennaReadPoint.setTransmitPowerLevel(transmitPowerLevel: int): void

Parameter(s)

- **transmitPowerLevel** - Data type: integer. Transmit power level to set to the readpoint.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

N/A

AntennaReadPoint.getCableLossCompensation

Get the configured cable loss rating and length of the cable for the readpoint.

Usage

AntennaReadPoint.getCableLossCompensation(void): void

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

N/A

AntennaReadPoint.setCableLossCompensation

Set the cable loss compensation for the readpoint from the cable-loss rating and length of the cable.

Usage

AntennaReadPoint.setCableLossCompensation(cableLossPerHundredFt: float, cableLength: float):void

Parameter(s)

- **cableLossPerHundredFt** - Data type: float. Cable loss rating of the cable used for the readpoint in dBm/100ft.
- **cableLength** - Data type: float. Length of the cable at the readpoint in feet.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

N/A

AntennaReadPoint.getCRCErrors

Get number of CRC errors encountered.

Usage

AntennaReadPoint.getCRCErrors(void): void

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

N/A

AntennaReadPoint.resetCRCErrors

Reset current value of CRC error counter.

Usage

AntennaReadPoint.resetCRCErrors(void): void

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

N/A

AntennaReadPoint.getRFOnTime

Get duration in seconds, since power on of the reader, for which RF was turned on by the reader.

Usage

AntennaReadPoint.getRFOnTime(void): void

Parameter(s)

- **Data Type**: void. This command takes no parameters.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

N/A

AntennaReadPoint.getGen2OptionalOperCounts

Get the operational counts for optional Gen2 operations across an AntennaReadPoint.

Usage

AntennaReadPoint.getGen2OptionalOperCounts(successCount: boolean, requestedOperType: motorm:gen2OptionalOperType): void

Parameter(s)

- **successCount** - Data type: boolean. Flag indicating if the requested operation count is for success or failure. If omitted success count is returned.
- **requestedOperType** - Data type: motorm:gen2OptionalOperType. Flag indicating the specific operation whose count is requested. If omitted or "All" all the related operations count is returned.

Return Value(s)

- **successCount** - Data type: boolean. Flag indicating if the requested operation count is for success or failure.
- **blockEraseCount** - Data type: int. The count of the successful or failed block erase operations on this antenna.
- **blockWriteCount** - Data type: int. The count of the successful or failed block write operations on this antenna.
- **blockPermalockCount** - Data type: int. The count of the successful or failed block perma-lock operations on this antenna.

Possible Error Conditions

- invalidOption

AntennaReadPoint.getNXPCustomOperCounts

Get the operational counts for NXP Custom operations across an AntennaReadPoint.

Usage

AntennaReadPoint.getNXPCustomOperCounts(successCount: boolean, requestedOperType: motorm:NXPOperType): void

Parameter(s)

- **successCount** - Data type: boolean. Flag indicating if the requested operation count is for success or failure. If omitted success count is returned.
- **requestedOperType** - Data type: **motorm:NXPOperType**. Flag indicating the specific operation whose count is requested. If omitted or "All" all the related operations count is returned.

Return Value(s)

- **successCount** - Data type: boolean. Flag indicating if the operation count returned is for success or failure.
- **changeEASCount** - Data type: int. The count of the successful or failed change EAS operations on this antenna.
- **EASAlarmCount** - Data type: int. The count of the successful or failed EAS alarms received on this antenna.
- **setQuietCount** - Data type: int. The count of the successful or failed set quiet operations on this antenna.
- **resetQuietCount** - Data type: int. The count of the successful or failed reset quiet operations on this antenna.
- **calibrateCount** - Data type: int. The count of the successful or failed calibrate operations on this antenna.

Possible Error Conditions

- invalidOption

AntennaReadPoint.getFujitsuCustomOperCounts

Get the operational counts for Fujitsu Custom operations across an AntennaReadPoint.

Usage

AntennaReadPoint.getFujitsuCustomOperCounts(successCount: boolean, requestedOperType: motorm:FujitsuOperType): void

Parameter(s)

- **successCount** - Data type: boolean. Flag indicating if the requested operation count is for success or failure. If omitted success count is returned.
- **requestedOperType** - Data type: **motorm:FujitsuOperType**. Flag indicating the specific operation whose count is requested. If omitted or "All" all the related operations count is returned.

Return Value(s)

- **successCount** - Data type: boolean. Flag indicating if the operation count returned is for success or failure.
- **ChangeWordLockCount** - Data type: int. The count of the successful or failed Change Word Lock operations on this antenna.
- **ChangeBlockLockCount** - Data type: int. The count of the successful or failed Change Block Lock operations on this antenna.
- **ReadBlockLockCount** - Data type: int. The count of the successful or failed Read Block Lock operations on this antenna.
- **ChangeBlockOrAreaGroupPasswordCount** - Data type: int. The count of the successful or failed Change Block Or Area Group Password operations on this antenna.
- **BurstWriteCount** - Data type: int. The count of the successful or failed Burst Write operations on this antenna.
- **BurstEraseCount** - Data type: int. The count of the successful or failed Burst Erase operations on this antenna.
- **AreaReadLockCount** - Data type: int. The count of the successful or failed Area Read Lock operations on this antenna.
- **AreaWriteLockCount** - Data type: int. The count of the successful or failed Area Write Lock operations on this antenna.
- **AreaWriteLockWOPasswordCount** - Data type: int. The count of the successful or failed Area Write Lock (Without Password) operations on this antenna.

Possible Error Conditions

- invalidOption

AntennaReadPoint.getImpinjCustomOperCounts

Get the operational counts for Impinj Custom operations across an AntennaReadPoint.

Usage

AntennaReadPoint.getImpinjCustomOperCounts(successCount: boolean, requestedOperType: motorm:ImpinjOperType): void

Parameter(s)

- **successCount** - Data type: boolean. Flag indicating if the requested operation count is for success or failure. If omitted success count is returned.
- **requestedOperType** - Data type: **motorm:ImpinjOperType**. Flag indicating the specific operation of the requested count. If omitted or **All**, all related operations' count is returned.

Return Value(s)

- **Data Type**: void. This command does not return a value.

Possible Error Conditions

N/A

Reader Management Custom Error Codes

Table 2 RM Custom Error Codes

Error Number	Error Code	Error Description
1	notLoggedIn	Access denied - host is not logged in
12	operationFailed	Operation failed
13	dbOpenFailed	DB open failed
14	dbPutFailed	DB put failed
17	invalidUser	You have entered an invalid user name and/or password - try again
18	sessionTimeout	Invalid session - log in again
19	invalidName	Invalid name
23	invalidIpAddr	Not a legal IP address (1.0.0.0 - 255.255.255.255)
28	invalidPassword	Invalid password
31	valueOverSize	The value size is over the limit
38	invalidAccess	Invalid user access value
39	nameUsed	This name is already in use
40	ipAddressUsed	This IP address is already in use
43	invalidSelection	You must select an item from the list
48	invalidOption	Option is not valid
49	notFindReadPoint	Cannot find the specified read point
54	newPswdNotMatch	Failed to confirm the new password
55	newPswdSameAsOld	The new password is the same as the old one
56	wrongOldPswd	The old password is not correct
57	notFindUser	The user name is not correct
62	notCurrentSession	Another administrator is logged in - try again later
65	notFindHost	Cannot find the specified host address
70	delAdminUser	Cannot delete the **admin** user account
75	firmwareParamsNotSet	Firmware update parameters were not set for the update
76	missingFirmwareFile	Missing firmware files for the readers
78	invalidSnmpHostLink	The SNMP host link is not valid
79	invalidSnmpVersion	Invalid SNMP version number

Table 2 RM Custom Error Codes (Continued)

Error Number	Error Code	Error Description
80	invalidDescription	Description invalid
83	invalidLocation	Location invalid
84	invalidContact	Contact invalid
89	duplicatedUserName	The user name is already in use
91	invalidNetworkMask	Invalid network mask
93	invalidAdminUserAccess	Can not reduce the access level for the **admin** user
94	noEdit	View-only user can not make configuration changes
100	noMaint	This action requires maintenance user privilege
101	notEnoughPrivilege	The user does not have privileges for this action
105	delCurrentUser	Cannot modify the active user account
112	malformedFTPURL	Malformed FTP URL
113	nosuchFileOrPath	File or path does not exist or is inaccessible, or incorrect username/password
116	noChangesCommit	No changes to commit
117	noChangesDiscard	No changes to discard
119	reinitWait	Reinitializing - please wait
120	missingFreq	Specify at least one frequency
121	invalidCrypt	Invalid encrypted string
122	incompleteRegion	Region not completely specified
132	invalidValue	Invalid parameter value
161	antennaFault	Antenna fault
182	malformedFTPSURL	Invalid secure FTP server path
183	wrongPFXPassword	Wrong PFX password
184	noPrivateKeyFound	Certificate does not have an associated private key
185	failedCertImport	Failed to import certificate
186	errorReadingPassFile	Error reading password file
187	invalidDataInPfx	Invalid data in certificate file
188	couldNotInstallCertificate	Could not install the certificate
189	noRestartPermission	Certificate cannot be installed without restarting SSH/FTPS
190	notInSecureMode	Certificate update allowed only in secure mode
197	userLoggedIn	Another user is logged in - try again later

Table 2 RM Custom Error Codes (Continued)

Error Number	Error Code	Error Description
198	AdminLoggedIn	The administrator is logged in - try again later
199	llrpConnected	LLRP is already connected - disconnect LLRP and try again
200	llrpServerMode	LLRP is running in server mode
202	StandardportConflict	Conflict of port - LLRP can not use port number 21/22/23/80/443
204	llrpTaskInitFailed	LLRP task initialization failed
205	statsOverflow	Statistics overflow error - reset statistics to retrieve correct statistics
206	webServerRefresh	Web server is reinitializing
207	failedToGetStats	Failed to retrieve read point statistics
224	unsupportedCountry	Specified country is not supported
225	unsupportedRegion	Specified region is not supported
226	invalidNumChannels	Invalid number of channels for this region
227	currentProfile	Current profile cannot be imported/activated/deleted
228	invalidProfile	Invalid profile received
229	invalidSerialTimeOut	Invalid serial timeout value (minimum allowed is 15 seconds)
230	couldNotUninstallCert	Could not uninstall existing certificate
231	radioNotInitialised	Radio initialization failed, could not perform operation
232	FirmwareUpdateNotStarted	Firmware update was not started
233	startingFirmwareUpdate	Firmware update is starting, please wait
234	failedToGetUpdateProgress	Failed to get firmware update progress
235	llrpConnectFailed	LLRP connect request failed
236	llrpDisconnectFailed	LLRP disconnect request failed
237	addAdminUser	Cannot add the **admin** user account
238	defaultUser	Credentials (user name and password) must be changed for reader access
239	defaultUserNotPresent	Default user not present
240	pendingChanges	Changes not committed - operation not performed
241	stdProfileNoImport	Standard profile, can not be overwritten
242	stdProfileNoDelete	Standard profile, can not be deleted
243	unsupportedCommand	Command not supported on this device

Table 2 RM Custom Error Codes (Continued)

Error Number	Error Code	Error Description
244	maxProfilesPresent	Maximum number of supported profiles are present in the reader
245	cannotImportActiveProfile	Active profile cannot be overwritten in the reader
246	profileExists	Specified profile already exists in the reader
247	invalidProfileName	Invalid profile name specified
248	unknownUSBMode	Specified USB operating mode is not known
249	invalidData	Specified data is invalid
250	loginfailed	Login failed
251	filedownloadfailed	File download failed
252	bootdatareadfailed	Unable to read bootdata
253	bootdatawritefailed	Unable to write bootdata
254	invalidPreSharedKey	Invalid pre-shared key
255	invalidPolicyFile	Invalid p-olicy file
256	policyAlreadyAdded	Policy is already added
257	invlaidIPSecConfFile	Invalid IPSEC configuration file
258	WrongIDType	Wrong ID Type
259	FunctionNotExecuted	Function not executed
260	packageDoesNotExist	Package installation file does not exist
261	packageInstallFailed	Installation of package failed
262	startupFileDoesNotExist	Start or stop script does not exist
263	generatefilteredSyslogStarted	Filtering system log started
264	generatefilteredSyslogIn Progress	Filtering system log is in progress
265	generatefilteredSyslogFailed	Filtering system log failed
266	generateCSDStarted	Filtering customer support data file started
267	generateCSDInProgress	Filtering customer support data file in progress
268	generateCSDFailed	Filtering customer support data file failed
269	purgeLogsFailed	Failed to clear system and intermediate logs
270	limitExceeded	IPSEC configuration limit exceeded, operation not allowed
271	wirelessScanErr	Wireless scan failed, try again later
272	resetToFactoryDefaultsFailed	Factory reset failed, refer to the log

Table 2 RM Custom Error Codes (Continued)

Error Number	Error Code	Error Description
273	unableToGenerateLogFile	Unable to generate log file, change the filter options and try again
274	llrpDiagnosticsStartFailed	Starting of LLRP diagnostics failed
275	llrpDiagnosticsFailed	LLRP diagnostics failed
276	appAlreadyRunning	Cannot start, application is already running
277	cannotChangePassword	Cannot change password for guest
278	invalidnetworkadapter	Adapter not found
279	invalidessid	ESSID is invalid
280	unableToConnect	Unable to connect to wireless network
281	sshNotEnabled	SSH is not enabled
282	appRunningFailedToUninstall	Application is running, cannot be uninstalled
283	appRunningFailedToInstall	Application is running, cannot be installed again
284	appNotInstalled	Application is not installed
285	fipsModeAttributeChanged	FIPS mode attribute changed in XML. Reader is rebooting.
286	failedtostopracoon	Failed to stop racoon
287	failedtostartracoon	Failed to start racoon
288	setkeyfailed	Setkey failed
289	wirelessNotConnected	Wireless is not connected
290	hoppingNotSupported	Hopping not supported for single channel selection
291	singleChannelNeeded	Single channel should be selected when hopping is not configured
292	powerNegotiationInProgress	Another power negotiation activity is in progress
293	guestlogindisabled	Guest login is disabled and can be enabled only by the admin user
294	radioFWUpgradeInProgress	Radio firmware update is being updated
295	licenseCheckFailed	Not a valid license installed
296	onlineLicenseAcquireFailed	Failed to acquire license online
297	offlineLicenseAcquiredFailed	Failed to install offline license
298	releaseLicenseFailed	Failed to release the installed license
299	getInstalledLicenseListFailed	Failed to get the installed license list
300	evalLicenseDateValidityFailed	Evaluation license validity failed
301	importProfileChecksumFailed	Checksum is failed to match for importing profile

Table 2 RM Custom Error Codes (Continued)

Error Number	Error Code	Error Description
302	errorAccessingHardwareResource	Failed to access hardware resource
303	missingUpdateFile	One or more files are missing for firmware update
304	readerModelNotSupported	Feature is not supported on this model
305	applicationInstallRunDisabled	Cannot install and run application on license install time
306	invalidActivationID	License activation ID is not valid

LLRP Custom Extensions

Introduction

This chapter describes custom messages and parameters and provides the binary packet format for these.

LLRP Custom Messages Per Product

The LLRP custom messages supported by the FX7400, FX7500, FX9500, FX9600, and ATR7000 RFID fixed readers and MC3000/MC9000 Series mobile computer are outlined in Table 3.

Table 3 LLRP Custom Messages Per Product

LLRP Custom Parameters	FX7400	FX7500	FX9500	FX9600	ATR7000	MC3000/ MC9000 Series
MOTO_GET_TAG_EVENT_REPORT	Y	Y	Y	Y	Y	Y
MOTO_PURGE_TAGS	Y	Y	Y	Y	Y	Y
MOTO_PURGE_TAGS_RESPONSE	Y	Y	Y	Y	Y	Y
MOTO_TAG_EVENT_NOTIFY	Y	Y	Y	Y	Y	Y
MOTO_UPDATE_RADIO_FIRMWARE	N	N	N	N	N	Y
MOTO_UPDATE_RADIO_FIRMWARE_RESPONSE	N	N	N	N	N	Y
MOTO_UPDATE_RADIO_CONFIG	N	N	N	N	N	Y
MOTO_UPDATE_RADIO_CONFIG_RESPONSE	N	N	N	N	N	Y
MOTO_GET_RADIO_UPDATE_STATUS	N	N	N	N	N	Y
MOTO_GET_RADIO_UPDATE_STATUS_RESPONSE	N	N	N	N	N	Y
Y = Supported / N= Not Supported						

The LLRP custom messages are as follows.

MOTO_GET_TAG_EVENT_REPORT

The client sends this message to the reader to retrieve consolidated event information associated with each tag read during autonomous read mode.

During autonomous reader operation mode, events associated with read tags can be configured to accumulate on the reader. If events that are not yet reported are associated with one or more tags known to the reader, the reader sends an **RO_ACCESS_REPORT** message with multiple **TagReportData**, each having a custom event list parameter, to the client as a response to this message. The reader clears the event list associated with each tag once it sends a response to this message. The reader still maintains tag identification unless explicitly purged using the **MOTO_PURGE_TAGS** message.

MOTO_GET_TAG_EVENT_REPORT

Vendor Identifier: 161

Message Subtype: 2

MOTO_PURGE_TAGS

This message is sent from the client to the reader. It is used to purge tags or optionally only events associated with the tag from the tag database in the reader. Only tags generated using custom event reporting can be purged using this message. .

MOTO_PURGE_TAGS

Vendor Identifier: 161

Message Subtype: 3

PurgeTagEventStateOnly - Boolean value:

- **True** - purge only the event list associated with tags matching the list specified in the **<Data>** parameter from the reader database. If the **<Data>** parameter list is empty or not specified, purge the event list associated with all tags.
- **False** - purge all tags and associated events matching the list specified in the **<Data>** parameter from the reader database. If the **<Data>** parameter list is empty or not specified, purge all tags and associated event lists.

Data: Parameter specifying ID of tags to match and select for purge operation. Refer to the LLRP specification for definition of Data.

NOTE: Purging an explicit list of tags is *Not* supported in the FX9500 and FX9600.

MOTO_PURGE_TAGS_RESPONSE

The reader sends this message to the client in response to a **MOTO_PURGE_TAGS** message, indicating the status of the tag and/or event purging operation.

MOTO_PURGE_TAGS_RESPONSE

Vendor Identifier: 161

Message Subtype: 4

LLRPStatus: Parameter specifying the status of the operation.

MOTO_TAG_EVENT_NOTIFY

The reader sends this message to the client to indicate a visibility event change for one or more tags. This message occurs when the **MotoTagReportMode** parameter is set to **Report Notifications**.

MOTO_TAG_EVENT_NOTIFY
Vendor Identifier: 161
Message Subtype: 5

MOTO_UPDATE_RADIO_FIRMWARE

The client sends this message to the reader to initiate the radio firmware upgrade according to the file specified in the path.

MOTO_UPDATE_RADIO_FIRMWARE
Vendor Identifier: 161
Message Subtype: 10
FirmwareFilePath: UTF-8 String. Path where the firmware file is present.

MOTO_UPDATE_RADIO_FIRMWARE_RESPONSE

The reader sends this message to the client in response to a **MOTO_UPDATE_RADIO_FIRMWARE** message, indicating the status of the firmware update.

MOTO_UPDATE_RADIO_FIRMWARE_RESPONSE
Vendor Identifier: 161
Message Subtype: 11
LLRPStatus: LLRPStatus parameter. Status of operation.

MOTO_UPDATE_RADIO_CONFIG

The client sends this message to the reader to initiate the radio configuration upgrade according to the file specified in the path.

MOTO_UPDATE_RADIO_CONFIG
Vendor Identifier: 161
Message Subtype: 12
ConfigFilePath: UTF-8 String.

MOTO_UPDATE_RADIO_CONFIG_RESPONSE

The reader sends this message to the client in response to a **MOTO_UPDATE_RADIO_CONFIG** message, indicating the status of the configuration update.

MOTO_UPDATE_RADIO_CONFIG_RESPONSE

Vendor Identifier: 161

Message Subtype: 13

LLRPStatus: LLRPStatus parameter. Status of operation.

MOTO_GET_RADIO_UPDATE_STATUS

The client sends this message to the reader to obtain the status of the radio firmware or configuration update.

MOTO_GET_RADIO_UPDATE_STATUS

Vendor Identifier: 161

Message Subtype: 14

MOTO_GET_RADIO_UPDATE_STATUS_RESPONSE

The reader sends this message to the client in response to a **MOTO_GET_RADIO_UPDATE_STATUS** message, indicating the status of the radio files (firmware or configuration) update.

MOTO_GET_RADIO_UPDATE_STATUS_RESPONSE

Vendor Identifier: 161

Message Subtype: 15

LLRPStatus: LLRPStatus parameter. Status of operation. **0** for Success, otherwise error code returned from radio communicated.

MotoRadioUpdateStatusInfo: MotoRadioUpdateStatusInfo parameter. Radio update status information.

LLRP Custom Parameters Per Product

The LLRP custom parameters supported by the FX7400, FX7500, FX9500, and FX9600 RFID fixed readers and MC3000/MC9000 Series mobile computer are outlined in Table 4.

Table 4 LLRP Custom Parameters Per Product

LLRP Custom Parameters	FX7400	FX7500	FX9500	FX9600	ATR7000	MC3000/ MC9000 Series
MotoGeneralRequestCapabilities	Y	Y	Y	Y	Y	Y
MotoGeneralCapabilities	N	Y	N	Y	Y	N
MotoAutonomousCapabilities	Y	Y	Y	Y	Y	Y
MotoTagEventsGenerationCapabilities	Y	Y	Y	Y	Y	Y
MotoLocationCapabilities	See page 141	N	N	N	Y	Y
MotoFilterCapabilities	Y	Y	Y	Y	Y	Y
MotoPersistenceCapabilities	Y	Y	See page 142	Y	Y	N
MotoAdvancedCapabilities	N	Y	N	Y	Y	N
MotoRadioTransmitDelay	N	Y	N	Y	N	N
MotoGeneralGetParams	Y	Y	Y	Y	Y	Y
MotoRadioPowerState	N	Y	N	Y	Y	Y
MotoRadioUpdateStatusInfo	N	N	N	N	N	Y
MotoRadioDutyCycle	N	N	N	N	Y	Y
MotoRadioDutyCycleTable	N	N	N	N	Y	Y
MotoVersion	Y	Y	N	Y	Y	Y
MotoVersionList	Y	Y	N	Y	Y	Y
MotoSledBatteryStatus	N	N	N	N	N	N
MotoFilterRule	N	Y	N	Y	Y	N
MotoFilterTimeOfDay	Y	Y	Y	Y	Y	Y
MotoFilterTimeRange	Y	Y	Y	Y	Y	Y
MotoUTCTimestamp	Y	Y	Y	Y	Y	Y
MotoFilterRSSIRange	Y	Y	Y	Y	Y	Y
MotoFilterTagList	N	Y	N	Y	N	N
MotoFindItem		N	N	N	N	Y
MotoLocationResult	N	N	N	N	N	Y
Y = Supported / N= Not Supported						

Table 4 LLRP Custom Parameters Per Product

LLRP Custom Parameters	FX7400	FX7500	FX9500	FX9600	ATR7000	MC3000/ MC9000 Series
MotoAutonomousState	Y	Y	Y	Y	Y	Y
MotoTagEventSelector	Y	Y	Y	Y	Y	Y
MotoTagReportMode	Y	Y	Y	Y	Y	Y
MovingStationaryTagReport	N	Y	N	Y	N	N
MotoFilterList	Y	Y	Y	Y	Y	Y
MotoPersistenceSaveParams	Y	Y	see page 153	Y	Y	N
MotoDefaultSpec	Y	Y	Y	Y	Y	Y
MotoTagEventList	Y	Y	Y	Y	Y	Y
MotoTagEventEntry	Y	Y	Y	Y	Y	Y
MotoPersistenceCapabilities	N	Y	N	Y	Y	N
MotoROReportTrigger	N	Y	N	Y	Y	N
MotoC1G2LLRPCapabilities	N	Y	N	Y	Y	N
MotoC1G2ExtendedPC	Y	Y	Y	Y	Y	Y
MotoC1G2Recommission	N	Y	N	Y	Y	N
MotoC1G2RecommissionOpSpecResult	N	Y	N	Y	Y	N
MotoC1G2BlockPermalock	Y	Y	Y	Y	Y	Y
MotoC1G2BlockPermalockOpSpecResult	Y	Y	Y	Y	Y	Y
MotoNXPChangeEAS	Y	Y	Y	Y	Y	Y
MotoNXPChangeEASOpSpecResult	Y	Y	Y	Y	Y	Y
MotoNXPSetQuiet	Y	Y	Y	Y	Y	Y
MotoNXPSetQuietOpSpecResult	Y	Y	Y	Y	Y	Y
MotoNXPResetQuiet	Y	Y	Y	Y	Y	Y
MotoNXPResetQuietOpSpecResult	Y	Y	Y	Y	Y	Y
MotoNXPCalibrate	Y	Y	Y	Y	Y	Y
MotoNXPCalibrateOpSpecResult	Y	Y	Y	Y	Y	Y
MotoNXPEASAlarmSpec	Y	Y	Y	Y	Y	Y
MotoNXPEASAlarmNotification	Y	Y	Y	Y	Y	Y
MotoConnectionFailureReason	Y	Y	N	Y	Y	Y
MotoCustomCommandOptions	Y	Y	Y	Y	Y	Y
Y = Supported / N= Not Supported						

Table 4 LLRP Custom Parameters Per Product

LLRP Custom Parameters	FX7400	FX7500	FX9500	FX9600	ATR7000	MC3000/ MC9000 Series
MotoFujitsuChangeWordLock	Y	N	N	N	N	Y
MotoFujitsuChangeWordLockOpSpecResult	Y	N	N	N	N	Y
MotoFujitsuChangeBlockLock	Y	N	N	N	N	Y
MotoFujitsuChangeBlockLockOpSpecResult	Y	N	N	N	N	Y
MotoFujitsuReadBlockLock	Y	N	N	N	N	Y
MotoFujitsuReadBlockLockOpSpecResult	Y	N	N	N	N	Y
MotoFujitsuChangeBlockOrAreaGroupPassword	Y	N	N	N	N	Y
MotoFujitsuChangeBlockOrAreaGroupPasswordOpSpecResult	Y	N	N	N	N	Y
MotoFujitsuBurstWrite	Y	N	N	N	N	Y
MotoFujitsuBurstWriteOpSpecResult	Y	N	N	N	N	Y
MotoFujitsuBurstErase	Y	N	N	N	N	Y
MotoFujitsuBurstEraseOpSpecResult	Y	N	N	N	N	Y
MotoFujitsuAreaReadLock	Y	N	N	N	N	Y
MotoFujitsuAreaReadLockOpSpecResult	Y	N	N	N	N	Y
MotoFujitsuAreaWriteLock	Y	N	N	N	N	Y
MotoFujitsuAreaWriteLockOpSpecResult	Y	N	N	N	N	Y
MotoFujitsuAreaWriteLockWOPassword	Y	N	N	N	N	Y
MotoFujitsuAreaWriteLockWOPasswordOpSpecResult	Y	N	N	N	N	Y
MotoNXPChangeConfig	Y	Y	N	Y	Y	Y
MotoNXPChangeConfigOpSpecResult	Y	Y	N	Y	Y	Y
MotoImpinjQT	Y	Y	N	Y	Y	Y
QTData	Y	Y	N	Y	Y	Y
MotoImpinjQTOpSpecResult	Y	Y	N	Y	Y	Y
MotoC1G2Authenticate	N	Y	N	Y	N	N
MotoC1G2AuthenticateOpSpecResult	N	Y	N	Y	N	N
MotoC1G2ReadBuffer	N	Y	N	Y	N	N
MotoC1G2ReadBufferOpSpecResult	N	Y	N	Y	N	N
Y = Supported / N= Not Supported						

Table 4 LLRP Custom Parameters Per Product

LLRP Custom Parameters	FX7400	FX7500	FX9500	FX9600	ATR7000	MC3000/ MC9000 Series
MotoC1G2Untraceable	N	Y	N	Y	N	N
MotoC1G2UntraceableOpSpecResult	N	Y	N	Y	N	N
MotoC1G2Crypto	N	Y	N	Y	N	N
MotoC1G2CryptoOpSpecResult	N	Y	N	Y	N	N
MotoTagGPS	N	Y	N	Y	N	N
MotoAntennaConfig	N	Y	N	Y	Y	N
MotoAntennaStopCondition	N	Y	N	Y	Y	N
MotoAntennaPhysicalPortConfig	N	Y	N	Y	Y	N
MotoTagReportContentSelector	N	Y	N	Y	Y	N
MotoTagPhase	N	Y	N	Y	Y	N
MotoAntennaQueryConfig	N	Y	N	Y	Y	N
NXPBrandIDCheckConfig	N	Y	N	Y	N	N
BrandIDCheckStatus	N	Y	N	Y	N	N
ZebraROTriggerSpec	N	Y	N	Y	N	N
ZebraROSpecStartTrigger	N	Y	N	Y	N	N
ZebraTimelapseStart	N	Y	N	Y	N	N
ZebraDistance	N	Y	N	Y	N	N
ZebraROSpecStopTrigger	N	Y	N	Y	N	N
ZebraTimelapseStop	N	Y	N	Y	N	N
Y = Supported / N= Not Supported						

The LLRP custom parameters are as follows.

MotoGeneralRequestCapabilities

The client sends this parameter to the reader as part of a **GET_READER_CAPABILITIES** message to select custom capabilities that the reader reports in the corresponding **GET_READER_CAPABILITIES_RESPONSE** message. General capabilities currently include autonomous mode operation, additional filtering based on time or RSSI, and saving reader configuration, tag data, and events over LLRP.

MotoGeneralRequestCapabilities

Vendor Identifier: 161

Parameter Subtype: 50

RequestedData: Unsigned character. Specifies capabilities to report. Possible values are:

- **0** - All
- **1** - General capabilities
- **2** - Autonomous mode capabilities
- **3** - Tag events generation capabilities
- **4** - Filtering capabilities
- **5** - Persistence capabilities
- **6** - C1G2 v1.2 capabilities
- **7** - Tag locating capabilities
- **8** - Radio duty cycle capabilities
- **9** - Versions capabilities
- **10** - Advanced capabilities

MotoGeneralCapabilities

This parameter extends reader capabilities to report custom capabilities supported by Zebra readers. The reader sends this to the client as a custom parameter to the **GET_READER_CAPABILITIES_RESPONSE** message when the client issues a **GET_READER_CAPABILITIES_REQUEST** with custom reporting of general capabilities enabled. See MotoGeneralRequestCapabilities. The general capabilities indicate if the reader can report the radio version, part number, and whether the reader supports MotoGeneralGetParams on page 144.

MotoGeneralCapabilities

Vendor Identifier: 161

Parameter Subtype: 1

Version: Unsigned integer. Version of custom capability.

CanGetGeneralParams: Boolean value. If true, the reader can return **MotoGeneralGetParams** if it is requested in the **GET_READER_CONFIG** message.

CanReportPartNumber: Boolean value. If true, the reader can report its part number.

CanReportRadioVersion: Boolean value. If true, the reader can report its radio version information.

CanSupportRadioPowerState: Boolean value. If true, the reader can report if the radio module is on or off, and the radio state can also be controlled.

CanSupportRadioTransmitDelay: Boolean value. If true, the reader can control the radio rest mode, which can force NGE to low power mode in case of no tag.

CanSupportZebraTrigger: Boolean value. If true, the reader has other triggers such as Timelapse and GPS distance.

MotoAutonomousCapabilities

This parameter reports the reader's ability to operate in autonomous mode. The reader sends this to the client as a custom parameter to the **GET_READER_CAPABILITIES_RESPONSE** message when the client issues a **GET_READER_CAPABILITIES_REQUEST** with custom reporting of autonomous mode capabilities enabled.

In autonomous mode, the reader continuously operates without requiring intervention from the client to re-initiate an operation. Autonomous mode and associated tag reporting is useful for continuous inventory, but can also be used for access operations. For efficiency, configure this mode for tag reporting when tag visibility changes during repeated inventory, rather than reporting the same tag data continuously. See MotoTagEventsGenerationCapabilities on page 140 for the reader's support of event reporting, MotoTagEventSelector on page 150 and MotoTagReportMode on page 151 for configuring reporting criteria, and MotoDefaultSpec on page 153 and MotoAutonomousState on page 149 for configuring and controlling autonomous mode.

MotoAutonomousCapabilities

Vendor Identifier: 161

Parameter Subtype: 100

Version: Unsigned integer. Version of custom capability.

CanSupportAutonomousMode: Boolean value. If true, the reader is capable of autonomous mode.

MotoTagEventsGenerationCapabilities

This parameter reports the reader's ability to report tag visibility state changes. The reader sends this to the client as a custom parameter to the **GET_READER_CAPABILITIES_RESPONSE** message when the client issues a **GET_READER_CAPABILITIES_REQUEST** with custom reporting of tag events generation capabilities enabled.

This capability makes the autonomous mode of operation more efficient by reporting only event state changes rather than reporting tags continuously.

MotoTagEventsGenerationCapabilities

Vendor Identifier: 161

Parameter Subtype: 120

Version: Unsigned integer. Version of custom capability.

CanSelectTagEvents: Boolean value. If true, the client can choose the tag event state changes to monitor and report. Event state options are **New Tag, Tag Invisible,** or **Tag Visibility Changed.**

CanSelectTagReportingFormat: Boolean value. If true, the client can choose what to report when an event generation criterion is met. Reporting format options are **No Reporting, Report a notification message,** or **Report events with the tag report.**

CanSelectMovingEvent: Boolean value. If true, the client can enable the tag moving/stationary algorithm. Event state can be **Moving Tag**.

MotoLocationCapabilities

This parameter reports the reader's ability to locate specific tags. The reader sends this to the client as a custom parameter to the **GET_READER_CAPABILITIES_RESPONSE** message when the client issues a **GET_READER_CAPABILITIES_REQUEST** with custom reporting of tag locating capabilities enabled. See MotoGeneralRequestCapabilities on page 139.

MotoLocationCapabilities

Vendor Identifier: 161

Parameter Subtype: 130

Version: Unsigned integer. Version of custom capability.

CanSupportMotoFindItem: Boolean value. If true, the client can locate an item with a specific tag identified by the EPC ID.

NOTE: The FX7400 and FX7500 return false for **CanSupportMotoFindItem**.

MotoFilterCapabilities

This parameter reports the reader's ability to filter tags based on the tags' RSSI, the time of day the tags were read, or the UTC Timestamp when the tags were read. The reader sends this to the client as a custom parameter to the **GET_READER_CAPABILITIES_RESPONSE** message when the client issues a **GET_READER_CAPABILITIES_REQUEST** with custom reporting of filtering capabilities enabled.

MotoFilterCapabilities

Vendor Identifier: 161

Parameter Subtype: 200

Version: Unsigned integer. Version of custom capability.

CanFilterTagsBasedOnRSSI: Boolean value. If true, the client can set a filter on the reader to report tags based on the signal strength of the back-scattered signal from the tag.

CanFilterTagsBasedOnTimeOfDay: Boolean value. If true, the client can set a filter on the reader to report tags based on the time of day (based on 24 hour clock time, with microsecond resolution) the reader read the tag.

CanFilterTagsBasedOnUTCTimeStamp: Boolean value. If true, the client can set a filter on the reader to report tags based on the UTC Timestamp (with microsecond resolution) the reader read the tag.

MotoPersistenceCapabilities

This parameter reports the reader's ability to save the configuration, save the tag, and save the associated event information. Saved information is retained across reader reboots. The reader sends this to the client as a custom parameter to the **GET_READER_CAPABILITIES_RESPONSE** message when the client issues a **GET_READER_CAPABILITIES_REQUEST** with custom reporting of persistence capabilities enabled.

MotoPersistenceCapabilities

Vendor Identifier: 161

Parameter Subtype: 300

Version: Unsigned integer. Version of custom capability.

CanSaveConfiguration: Boolean value. If true, the reader can save configurations across reboots.

CanSaveTags: Boolean value. If true, the reader can save tags across reboots. Note that only tags not yet reported and those generated as a result of the autonomous mode of operation can be configured to save for future reporting.

CanSaveEvents: Boolean value. If true, the reader can save events associated with tags across reboots. Note that only tags generated as a result of the autonomous mode of operation can have associated events that can be saved for future reporting.

NOTE: The FX9500 is able to save LLRP configuration for SET_READER_CONFIG message only. The capabilities for the FX9500 are set as follows: CanSaveConfiguration = true, CanSaveTags = false, which implies that tag data and event data cannot be saved in the FX9500.

MotoAdvancedCapabilities

This parameter extends reader capabilities to report custom advanced capabilities supported by Zebra readers. The reader sends it to the client as a custom parameter to the **GET_READER_CAPABILITIES_RESPONSE** message when the client issues a **GET_READER_CAPABILITIES_REQUEST** with custom reporting of general capabilities enabled (see MotoGeneralRequestCapabilities on page 139).

The advanced capabilities indicate if the reader can support phase, zone, antenna RF configuration, etc.

MotoAdvancedCapabilities

Vendor Identifier: 161

Parameter Subtype: 110

Version: Unsigned integer. Version of custom capability.

CanReportPhase: Boolean value. If true, the reader can report phase information along with the tag data if the same is enabled.

CanReportGPS: Boolean value. If true, the reader can report the GPS co-ordinates (longitude, latitude and altitude) information along with the tag data if the same is enabled.

CanSupportZone: Boolean value. If true, the reader can configure zones and support zone based operations.

CanSupportAntennaRFConfig: Boolean value. If true, the reader can configure the RF Config extensions such as Stop condition for antenna, SL_All, AB Flip, and physical port configuration.

CanSupportPeriodicTagReports: Boolean value. If true, the reader supports periodic reporting of tags. The reader can be configured to report a tag the first time it is seen, and then periodically based on a configured time period. The reader reports the tag if it is still read after the period.

CanSupportSledBatteryStatus: Boolean value. If true, the reader can report sled battery status as battery level and battery state.

CanSupportLogicalAntenna: If true, reader is capable of reporting Sled battery status as battery level and battery state.

MotoRadioTransmitDelay

This parameter sets or gets the radio transmit delay of the reader.

If this parameter is in SET_READER_CONFIG, reader passes the configuration to NGE for transmit delay.

If using this parameter in GET_READER_CONFIG_RESPONSE, the value indicates the current configuration of radio transmit delay.

MotoRadioTransmitDelay

Vendor Identifier: 161

Parameter Subtype: 511

RadioTransmitDelay: Indicates whether to turn on or turn off the radio transmit delay. Possible values are:

- **0** - Off
- **1** – On_No_Tag
- **2** – On_No_Unique_Tag

MotoGeneralGetParams

The client sends this parameter to the reader as part of the **GET_READER_CONFIG** message to retrieve custom LLRP configurations on the reader.

MotoGeneralGetParams

Vendor Identifier: 161

Parameter Subtype: 51

RequestedData: Unsigned character. Specifies custom parameters to report. Possible values are:

- **0** - All
- **1** - Autonomous mode state
- **2** - Filter list
- **3** - Persistency parameters
- **4** - Default spec for autonomous mode
- **5** - Radio power state
- **6** - Radio duty cycle
- **7** - Custom command options state
- **9** - Sled battery status

MotoRadioPowerState

This parameter enables or disables radio power. If this parameter is set to ON (logic 1) in **SET_READER_CONFIG**, radio power must be enabled (ON) or this parameter should be disabled (OFF). If using this parameter in **GET_READER_CONFIG_RESPONSE**, the value indicates the current state of radio power.

MobileRadioPowerState

Vendor Identifier: 161

Parameter Subtype: 500

RadioPowerState: Boolean. Possible values are:

- **0** - Off
- **1** - On

MotoRadioUpdateStatusInfo

This parameter obtains information on the progress of firmware or configuration download requests.

MotoRadioUpdateStatusInfo

Vendor Identifier: 161

Parameter Subtype: 501

PercentageComplete: Unsigned character. Indicates the percent complete of the firmware or configuration update.

RadioUpdateStatusCode: Unsigned character. Indicates success or failure of the operation.

StatusDescription: UTF-8 String. Describes the status of the operation.

MotoRadioDutyCycle

This parameter sets the duty cycle value of the radio. If included in **SET_READER_CONFIG**, its value is used as an index to the duty cycle table is set to the radio.

MotoRadioDutyCycle
Vendor Identifier: 161
Parameter Subtype: 502
DutyCycleIndex: Unsigned character. Index in the duty cycle table as reported in the reader capabilities.

MotoRadioDutyCycleTable

This parameter reports the reader's ability to support different duty cycles on the radio. The reader reports this to the client as a custom parameter to the **GET_READER_CAPABILITIES_RESPONSE** message when the client issues a **GET_READER_CAPABILITIES_REQUEST** with custom reporting of radio duty cycle capabilities enabled. See MotoGeneralRequestCapabilities on page 139.

This parameter provides a table of duty cycle values the radio supports. If zero, the duty cycle modification is not allowed.

MotoRadioDutyCycleTable
Vendor Identifier: 161
Parameter Subtype: 503
DutyCyclePercentage: Unsigned short vector. List of duty cycle percentages the reader allows.

MotoVersion

This parameter reports the name and version of a module. This currently reports the Radio Firmware and OEM versions.

MotoVersion

Vendor Identifier: 161

Parameter Subtype: 256

ModuleName: UTF-8 String. Name of the module.

ModuleVersion: UTF-8 String. Version of the module

MotoVersionList

This parameter reports the versions of reader components. If entry is zero, then version reporting is not supported.

MotoVersionList

Vendor Identifier: 161

Parameter Subtype: 504

MotoVersion: List of <MotoVersion Parameter >

MotoSledBatteryStatus

This parameter provides battery status for the sled.

MotoSledBatteryStatus

Vendor Identifier: 161

Parameter Subtype: 508

BatteryLevel: Unsigned integer.

Status: Unsigned character. Possible values are:

- **0** - Charging
- **1** - Discharging
- **2** - Critical level
- **-1** - Status unknown

MotoFilterRule

This parameter defines a single filter rule. Use MotoFilterList on page 152 to combine one or more of these rules to create a filter list.

Specify at least an RSSI range (see MotoFilterRSSIRange on page 148) or a time range (see MotoFilterTimeRange on page 147) in the rule. If specifying both, the filter criteria must satisfy both, i.e., an AND rule is applied.

MotoFilterRule

Vendor Identifier: 161

Parameter Subtype: 254

RuleType: Unsigned character. Specifies an action to perform on a tag that matches the filter rule. Possible values are:

- **0: Inclusive** - Report only tags matching this rule to the client.
- **1: Exclusive** - Report only tags that do not match this rule to the client.
- **2: Continue** - Create multiple filter rules and apply these in a sequence. If a tag matches this rule, do not report the tag yet, but continue processing subsequent rules in the rule list. If the tag doesn't match the rule, it is excluded from reporting without processing other rules. If this is the last rule in the rule list and the tag matches the rule, report the tag.

  All except the last **MotoFilterRule** in the **MotoFilterList** parameter have **RuleType** set to **Continue** for all rules to apply. If an interim rule does not match, the tag is not reported to the client.

MotoFilterRSSIRange: <MotoFilterRSSIRange Parameter>[Optional]

MotoFilterTimeRange: <MotoFilterTimeRange Parameter>[Optional]

MotoFilterTagList: < MotoFilterTagList Parameter>[Optional]

MotoFilterTimeOfDay

This parameter defines a time value with respect to the 24 hour clock. Use this to define bounds on time to filter on the time of day.

MotoFilterTimeOfDay

Vendor Identifier: 161

Parameter Subtype: 251

Microseconds: Number of microseconds since 0:00:00:000 midnight.

MotoFilterTimeRange

This parameter represents a time-based tag filter. Base this filter on either a UTC timestamp or on the time of day the tag read occurred. Specify two entries in the **MotoFilterTimeFormatChoice** sub-parameter: one for the lower time bound and one for the upper bound.

MotoFilterTimeRange

Vendor Identifier: 161

Parameter Subtype: 252

TimeFormat: Unsigned character. Selection for time format. Possible values are:

- **0** - Time of the day
- **1** - UTC Timestamp

Match: Unsigned character. Selection for match range. Possible values are:

- **0** - Within range; between lower and upper time limits, lower and upper limit inclusive.
- **1** - Outside range; outside lower and upper time limits, lower and upper limit inclusive.
- **2** - Greater than lower limit; Greater than lower time limit, lower limit inclusive, upper limit ignored.
- **3** - Lower than upper limit; Lower than upper time limit, upper limit inclusive, lower limit ignored.

MotoFilterTimeFormatChoice: List of <MotoFilterTimeOfDay Parameter> or list of <MotoUTCTimestamp Parameter>

MotoUTCTimestamp

This parameter represents a custom UTC timestamp representation format. The **Microseconds** field represents the number of microseconds elapsed since EPOC.

MotoUTCTimestamp

Vendor Identifier: 161

Parameter Subtype: 250

Microseconds: Number of microseconds since EPOC.

MotoFilterRSSIRange

This parameter represents an RSSI-based tag filter. Specify two entries in the **PeakRSSI** sub-parameter: one for the lower bound and one for the upper bound.

MotoFilterRSSIRange

Vendor Identifier: 161

Parameter Subtype: 253

Match: Unsigned character. Selection for match range. Possible values are:

- **0** - Within range; between lower and upper RSSI limits, lower and upper limit inclusive.
- **1** - Outside range; outside lower and upper RSSI limits, lower and upper limit inclusive.
- **2** - Greater than lower limit; Greater than lower RSSI limit, lower limit inclusive, upper limit ignored.
- **3** - Lower than upper limit; Lower than upper RSSI limit, upper limit inclusive, lower limit ignored.

PeakRSSI: List of <PeakRSSI Parameter>

MotoFilterTagList

This parameter represents a tag-list based tag filter.
The tag-list based filter checks the received tag based on Match and EPCData list.
When Match is Inclusive and the tag is included in the EPCData list, report the tag. Otherwise drop the tag.
When Match is Exclusive and the tag is included in the EPCData list, drop the tag. Otherwise report the tag.

MotoFilterRSSIRange

Vendor Identifier: 161

Parameter Subtype: 258

Match: Selection for match tag list. Possible values are:

- **0** - Inclusive; Tag matching this rule is reported to the client.
- **1** - Exclusive; Tag matching this rule is excluded from being reported to client.

EPCData: A Tag list used for filter.

MotoFindItem

This parameter locates a tag in the reader's field of view using a reader-supported tag locating algorithm. Currently the reader supports one algorithm. To invoke this set the **Mode** field to 0. Other **Mode** and **ModeParam** field values are reserved for future use.

MotoFindItem

Vendor Identifier: 161

Parameter Subtype: 270

Mode: Unsigned short. Reserved for future use. Provide value as 0.

ModeParam: Unsigned short. Reserved for future use. Provide value as 0.

AntennaIDs: Unsigned short vector. Antenna on which to perform locationing. The default logic supports locationing only on a single antenna.

MotoLocationResult

This parameter indicates the result of the tag locating operation. The **RelativeDistance** field reports the relative distance of a tag on a scale of 0-100. Zero indicates that the tag cannot be detected; 1 is the lowest tag detection frequency (the tag is very far); 100 indicates maximum detection frequency (the tag is very close).

MotoLocationResult

Vendor Identifier: 161

Parameter Subtype: 271

RelativeDistance: Signed short. Indicates the relative distance of the tag on a scale of 0-100.

MotoAutonomousState

This parameter enables or disables autonomous mode of operation and is sent as a custom parameter to the **SET_READER_CONFIG** message. The client can retrieve the current autonomous mode state from **GET_READER_CONFIG_RESPONSE** by requesting reporting of the corresponding custom parameter in the **GET_READER_CONFIG** message.

MotoAutonomousState

Vendor Identifier: 161

Parameter Subtype: 101

AutonomousModeState: Boolean value. If true, enable autonomous mode, otherwise disable it.

NOTE: In the FX9500, autonomous events with asynchronous tag events has undefined behavior in state aware mode. Autonomous tag events work only in state unaware mode in the FX9500.

MotoTagEventSelector

This parameter configures events for the reader to report to the client for tag state changes. This is an optional sub-parameter in the **ROReportSpec** parameter. The **MotoTagReportMode** parameter controls enabling reporting and its format.
The default event selector setting is to report new tag events and tag visibility change events immediately, and tag invisible events by a moderation timeout of 8 seconds.

MotoTagEventSelector

Vendor Identifier: 161

Parameter Subtype: 121

ReportNewTagEvent: Unsigned character. Selects reporting of new tag events. Possible values are:

- **0: Never** - Disable new tag event reporting.
- **1: Immediate** - Send event immediately.
- **2: Moderated** - Send event based on **NewTagEventModeratedTimeout** setting.

NewTagEventModeratedTimeout: Unsigned short. Timeout in milliseconds for moderating new tag event reporting. Use this only when **ReportNewTagEvent** is set to **Moderated.**

ReportTagInvisibleEvent: Unsigned character. Selects reporting for tag invisible event. Possible values are:

- **0: Never** - Disable tag invisible event reporting.
- **1: Immediate** - Send event immediately.
- **2: Moderated** - Send event based on **TagInvisibleEventModeratedTimeout** setting.

TagInvisibleEventModeratedTimeout: Unsigned short. Timeout in milliseconds for moderating tag invisible event reporting. Use this only when **ReportTagInvisibleEvent** is set to **Moderated.**

ReportTagVisibilityChangeEvent: Unsigned character. Possible values are:

- **0: Never** - Disable tag visibility change event reporting.
- **1: Immediate** - Send event immediately.
- **2: Moderated** - Send event based on **TagVisibilityChangeEventModeratedTimeout** setting.

TagVisibilityChangeEventModeratedTimeout: Unsigned short. Timeout in milliseconds for moderating tag visibility change event reporting. Use this only when **MotoTagEventSelectorReportTagVisibilityChangeEvent** is set to **Moderated.**

MotoTagReportMode

This parameter enables and disables event reporting and controls the format for event reporting. This is an optional sub-parameter in the **ROReportSpec** parameter. Use the **MotoTagEventSelector** parameter to specify the events to monitor. The default report format is **Report_Notifications**.

MotoTagReportMode

Vendor Identifier: 161

Parameter Subtype: 122

ReportFormat: Unsigned character. Specifies reporting format. Possible values are:

- **0: No reporting** - The reader issues no notification on event changes to the client. Clients can periodically issue **MOTO_GET_TAG_EVENT_REPORT** to retrieve tag data reports with event information.
- **1: Report Notification** - The reader issues a **MOTO_TAG_EVENT_NOTIFY** message to the client on a tag event change. Upon notification, the client can use the **MOTO_GET_TAG_EVENT_REPORT** message to retrieve tag data reports with event information.
- **2: Report events** - The reader issues all accumulated **TagReportData** with event extensions (see MotoTagEventList on page 156) to the client as part of the **RO_ACCESS_REPORT** response. This is the optimal mode in which the reader issues only changes in events since the last reporting of **TagReportData** with event extensions to the client. The reader does not send notifications to the client. It sends the report as soon as an event occurs and event generation criteria are met.

MovingStationaryTagReport

This parameter controls enanbles or disables event reporting and controls the format for event reporting. This is an optional sub-parameter in the ROReportSpec parameter. Use the MotoTagEventSelector parameter to specify the events to monitor. The default report format is Report_Notifications.

MovingStationaryTagReport

Vendor Identifier: 161

Parameter Subtype: 122

ReportFormat: Unsigned character. Specifies reporting format. Possible values are:

MovingStationaryTagReport: Parameters are:

- **ReportMovingTag**: Selects reporting for tag moving event. If TagEventSelectorReportMovingTag is enabled, the Tag Moving event is used to report whenever the 'New Tag Visible", "Tag Not Visible', and "Tag Visibility Changed" events take place. And, stationary tags can be obtained by sending the GET_REPORT command to the reader.
- **StrayTagModeratedTimeout**: Timeout in milliseconds for changing tag to stationary state.

MotoFilterList

Use this parameter to configure filter settings on the reader to filter tags based on the received signal strength of the tag, the time the tag was read, or both. The filter list is composed of one or more filter rules. Configure multiple filter rules as a chain of rules using the **Continue** flag for **RuleType**. See MotoFilterRule on page 146. This allows continuing rule matching for subsequent rules if the current rule passes.\

MotoFilterList

Vendor Identifier: 161

Parameter Subtype: 255

UseFilter: Boolean value. If true, enable filtering based on this filter list; otherwise disable filtering.

MotoFilterRule: List of <MotoFilterRule Parameter>

NOTE: In the FX9500, getting configuration (even if MotoGeneralGetParams is specified) does *Not* return MotoFilterList details because there is no preconfigured default set of MotoFilterList.

Notes

- The filter list accommodates a maximum of 10 filter rules.
- The default filter is disabled at initialization. Modify and enable this to start filtering for both autonomous mode and regular ROSpecs.
- The default filter has a single filter rule of type inclusive (add matching tag to database) with an RSSI and time of day based filter. Both rules allow any RSSI value (-128 to 127 dbm) and a full day's time coverage (0 to 86400000000 microseconds since midnight) by default.

MotoPersistenceSaveParams

This parameter configures the reader to save and restore reader configurations performed over LLRP, save tag data and non-reported events in internal flash, and restore these if the reader power cycles.

SaveTagData and **SaveTagEventData** enable saving tag and event data persistently on the reader across graceful reader shutdowns and reboots. Saving tag events also saves tag data and events associated with it. This saves only tags stored in the reader's internal tag database generated by ROSpec (either autonomous mode or regular ROSpecs) that are set to use custom reporting preference. The following conditions must be met to save a tag data report:

- The tag report must have an ROSpecID, antenna ID, first seen, or last seen timestamp.
- The ROReportSpec associated with the RO that generated this tag data report must have the reporting trigger set to none.
- The ROReportSpec should have custom parameters for event reporting configured (see MotoTagEventSelector on page 150 and MotoTagReportMode on page 151).

MotoPersistenceSaveParams

Vendor Identifier: 161

Parameter Subtype: 350

SaveConfiguration: Boolean value. If true, persist configurations made over LLRP (**SET_READER_CONFIG**) across reader shutdown or restart; otherwise do not save the configuration.

SaveTagData: Boolean value. If true, persist tags read by executing autonomous mode or ROSpec with custom event reporting across reader shutdown or restart; otherwise discard read tags.

SaveTagEventData: Boolean value. If true, persist the event list and the tags read by executing autonomous mode or reader operation with custom event reporting across reader shutdown or restart; otherwise do not save the event list.

NOTE: The FX9500 supports only SaveConfiguration. The FX9500 does *Not* always save configurations when the reader is shutdown gracefully. To make sure configuration is saved, LLRP must be disconnected. Disconnecting LLRP triggers the FX9500 to save the contents of SET_READER_CONFIG.

MotoDefaultSpec

This parameter defines a pre-configured ROSpec that the reader uses.

MotoDefaultSpec

Vendor Identifier: 161

Parameter Subtype: 102

UseDefaultSpecForAutoMode: Boolean value. If true, the default spec is used for autonomous (continuous read) mode of operation; otherwise it can be used for any operation.

ROSpec: <ROSpec Parameter>

AccessSpec: List of <AccessSpec Parameter>

NOTE: In the FX9500, if multiple access specs associated with MotoDefaultSpec are configured, GET_READER_CONFIG retrieves only one auto access spec instead of all access specs.

The default ROSpec and list of access specs in the **MotoDefaultSpec** custom extension point list applies to autonomous or non-autonomous mode depending on whether the **UseDefaultSpecForAutoMode** flag is set. By default the reader is in autonomous mode with one pre-configured ROSpec for inventory only. An ROReport spec parameter is always associated with autonomous mode ROSpec and this is mandatory. By default, the ROReportSpec parameter associated with **MotoDefaultSpec** sets the custom parameter for **MotoTagReportMode** to **Report_Notifications**, an optimized reporting mode where the reader issues only the **MOTO_TAG_EVENT_NOTIFY** message to the client. **MotoTagEventSelector** configures the default event selector to report new tag events and tag visibility change events immediately and tag invisible events by a moderation timeout of 8 seconds.

The default ROReportSpec configurable in **SET_READER_CONFIG** is NOT used for autonomous mode of operation.

To define custom operations for autonomous mode, reconfigure the default ROSpec, ROReportSpec, and/or the list of AccessSpecs sub-parameters associated with the **MotoDefaultSpec** parameter in the **SET_READER_CONFIG** message.

To enable autonomous mode, the client sets the **MotoAutonomousState** parameter in the **SET_READER_CONFIG** message to true. To disable autonomous mode, issue **SET_READER_CONFIG** with **MotoAutonomousState** set to false. The reader default configuration has **AutonomousState** set to false. The following restrictions apply to autonomous mode ROSpec and AccessSpecs. These do not apply if the **UseDefaultSpecForAutoMode** flag is turned off in **MotoDefaultSpec**.

- In the previous custom extension, ROSpec by default has an **ROSpecID** of 1001. This **ROSpecID** is not reserved. When autonomous mode is active, the number of regular ROSspecs that can be added is reduced by one. For example, if the **MaxNumROSpecs** reported in reader capabilities is **N** and if autonomous mode ROSpec is active, then only **N-1** regular ROSpecs can be added.
- The maximum number of AISpec allowed in autonomous mode ROSpec is 1. The default AISpec ID is 1001. This ID is not reserved.
- Restrictions on the maximum number of specs per ROSpec, the number of IPSpec per AISpec, and the number of OpSpecs per access specs used for autonomous mode ROSpec are as per those reported in reader capabilities.
- The maximum number of AccessSpecs (M) in the list of AccessSpecs is limited by the number of access specs allowed on the reader, as specified in reader capabilities. Access specs associated with autonomous mode are not reserved and users can delete it.
- The maximum number of OpSpecs in an access spec is 48.
- Add Autonomous Mode ROSpec only with priority 7. No other priority is allowed, in order to enable executing any regular ROSpec with higher priority, even if autonomous mode is active.
- The ROSpec start trigger for autonomous mode cannot be NULL (it can be immediate, periodic, or trigger based).
- If autonomous mode is active, the **DELETE_ROSPEC** message can not delete the autonomous mode ROSpec.
- **ROReportTrigger** for the ROReport associated with the autonomous mode ROSpec has trigger type none (**ROSpecStopTriggerType** = **0**). Events during autonomous mode are generated based on tag visibility events rather than LLRP standard's **ROReportTrigger** values.
- **TagReportContentSelector** has the following contents in the report enabled (set to true): **EnableROSpecID**, **EnableAntennaID**, **EnableFirstSeenTimestamp**, **EnableLastSeenTimestamp**.
- AccessSpecs states which are part of the autonomous mode ROSpec are always active.
- OPSpecs other than **C1G2Read** are not currently supported in autonomous mode access specs.
- AccessSpecs that are part of autonomous mode support **AccessSpecStopTrigger** type **0** (no stop trigger) only.
- AccessSpecs that are part of autonomous mode support Access Report spec with trigger type **0** only (whenever ROReport is generated).

- Enabling or disabling autonomous mode ROSpec is not allowed.
- Enabling or disabling access specs in **MotoDefaultSpec** is not allowed when autonomous mode is active.
- The AccessSpec associated with **MotoDefaultSpec** must use the same **ROSpecID** as the ROSpec in **MotoDefaultSpec** when autonomous mode is active. **AccessSpecStopTrigger** values other than Null are not allowed for access specs associated with autonomous mode ROspec.

RO Specs

To enable regular RO specs (those added via an **ADD_ROSPEC** message) for custom event or notification reporting, configure the custom parameters in either the default ROReportSpec in **SET_READER_CONFIG** or by using its own ROReportSpec. The following restrictions apply when using the ROReportSpec for event or notification reporting (when the **MotoReportFormat** custom parameter is set to **No_Reporting**, **Report_Notifications**, or **Report_Events**):

- **ROReportTrigger** must set trigger type to none.
- **TagReportContentSelector** must have the following contents in the report enabled (set to true): **EnableROSpecID**, **EnableAntennaID**, **EnableFirstSeenTimestamp**, **EnableLastSeenTimestamp.**

If initially using the default ROReportSpec in the reader configuration for event reporting mode for a regular ROSpec (**ROReportTrigger** set to none and content selectors **EnableROSpecID**, **EnableAntennaID**, **EnableFirstSeenTimestamp**, **EnableLastSeenTimestamp** set to true), configuring the default ROReportSpec to change **SET_READER_CONFIG** values for **MotoReportFormat**, **ROReportTrigger**, and/or one of the restricted content selectors to values other than those allowed for event reporting, changes the existing ROSpecs reporting behavior to regular tag reporting mode.

A regular ROSpec with or without event or notification mode of reporting can co-exist with an autonomous mode ROSpec. ROSpecs execute based on priority. Regular ROSpec execution generates tag reports based on the ROReportSpec. Tag report data is not generated for the **RO_ACCESS_REPORT** due to the execution of autonomous mode RO or regular RO configured for none, event notification, or event mode of reporting (as set in **MotoTagReportMode**).

When an ROSpec is set to execute under custom event reporting mode, tags are reported in the reader and events generated based on the custom reporting criteria when the visibility status changes. Such tags are not reported as a response to a **GET_REPORT** message. **MOTO_GET_TAG_EVENT_REPORT** can retrieve these tags if the tag has events associated with it. Reporting events clears the event list associated with that tag, so subsequent **MOTO_GET_TAG_EVENT_REPORT** won't report that tag. The reader still knows tags generated as a result of configuring custom notification in ROSpec execution, unless the custom **PURGE_TAGS** message purges them. The tag database maintains tags as known tags in order to continue generating visibility changes or invisible events.

MotoTagEventList

This is a custom parameter in **TagReportData** if a tag has associated events. This is generated as a response to **MOTO_GET_TAG_EVENT_REPORT** or during asynchronous reporting when the **MotoReportFormat** custom parameter is set to **Report_Events**. Each event entry of type **MotoTagEventEntry** defines a particular event that happened to the tag reported in **TagReportData**.

MotoTagEventList

Vendor Identifier: 161

Parameter Subtype: 123

MotoTagEventEntry: List of <MotoTagEventEntry Parameter>

MotoTagEventEntry

This parameter defines a tag event entry. It provides the type of event and a timestamp for that event. One or more such event entries can be associated with a tag and defined in the **MotoTagEventList** associated with **TagReportData** for a particular tag.

MotoTagEventEntry

Vendor Identifier: 161

Parameter Subtype: 124

EventType: Unsigned character. Type of event. Possible values are:

- **0** - Unknown state
- **1** - New Tag Visible
- **2** - Tag Not Visible
- **3** - Tag Visibility Changed
- **4** - Tag_Moving
- **5** -Tag_Stationary

Microseconds: Unsigned long integer. UTC Timestamp when the event happened.

MotoROReportTrigger

This parameter is an extension of the ROReportTrigger and allows configuring report triggers outside the standard ROReportTriggers supported in LLRP. **Moto_Upon_N_Seconds_Or_End_Of_AISpec** or **Moto_Upon_N_Seconds_Or_End_Of_ROSpec** can be specified only if the **ROReportSpec ROReportTriggerType** is none.

MotoROReportTrigger

Vendor Identifier: 161

Parameter Subtype: 125

MotoReportTrigger: Unsigned character. Zebra custom report trigger to control reporting of tags outside the standard ROReportTriggers supported in LLRP. Currently supports periodic reporting of tags if the tag continues to be read after the period elapses. The following restrictions apply when using a periodic report trigger on the reader:
1. Access operation results are not sent as part of the periodic report trigger. The reader sends an exception event when access operations are added when the reader is configured for periodic reporting of tags.
2. AISpec stop trigger of type TagObservationTrigger with Upon_Seeing_N_Tags_Or_Timeout is not supported when the reader is configured for periodic reporting of tags.
3. Phase reported back is 0 for tags reported periodically.

Possible values are:

- **0** - **Moto_None**: No Moto Trigger condition
- **1** - **Moto_Upon_N_Seconds_Or_End_Of_AISpec**: Tag is reported immediately upon read and is reported only after **N** seconds if the tag is read continuously. Read tags are also reported at the end of AISpec. Use this setting to reduce traffic to the host; the value of **N** is taken from RO Report **N**. If N = 0, tags are reported at the end of AISpec.
- **2** - **Moto_Upon_N_Seconds_Or_End_Of_ROSpec**: Tag is reported immediately upon read and is reported only after **N** seconds if the tag is read continuously. Read tags are also reported at the end of ROSpec. Use this setting to reduce the traffic to the host; the value of **N** is taken from RO Report **N**.
  If N = 0, tags are reported at the end of ROSpec

MotoC1G2LLRPCapabilities

MotoC1G2LLRPCapabilities

Vendor Identifier: 161

Parameter Subtype: 400

Version: Unsigned integer. Version of custom capability.

CanSupportBlockPermalock: Boolean value. If true, the reader supports **BlockPermaLock** command; if false the reader does not support this.

CanSupportRecommissioning: Boolean value. If true, the reader supports tag recommissioning; if false the reader does not support this.

CanWriteUMI: Boolean value. If true, the reader supports writing user memory bit; if false the reader does not support this.

CanSupportNXPCuxtomCommands: Boolean value. If true, the reader supports NXP Custom commands; if false the reader does not support this.

CanSupportFujitsuCuxtomCommands: Boolean value. If true, the reader supports Fujitsu Custom commands; if false the reader does not support this.

CanSupportG2V2Commands: Boolean value. If true, reader supports G2V2 commands; if false the reader does not support this.

MotoC1G2ExtendedPC

This parameter is a custom extension to **TagReportData** and is reported only when the **enabl** field is set to true in the **C1G2MemorySelector** parameter. It is reported along with the PC in the tag report if the tag transmits XPC data during inventory.

MotoC1G2ExtendedPC

Vendor Identifier: 161

Parameter Subtype: 450

XPC: Unsigned short vector. XPC1 word is the most significant word position followed by XPC2 word.

MotoC1G2Recommission

Use this C1G2 OpSpec parameter for tag re-commissioning. It is used in AccessSpecs like other OpSpec parameters. At the reader this parameter translates into the **C1G2 Kill** command with re-commissioning bits (tags are not killed). Refer to the Class 1 Gen2 V1.2 kill command specification for more details. The reader honors this command when the reader reports **CanSupportRecommissioning** as true in **GET_READER_CAPABILITIES_RESPONSE**.

MotoC1G2Recommission

Vendor Identifier: 161

Parameter Subtype: 451

OpSpecID: Unsigned short. OpSpec ID.

KillPassword: Unsigned integer. The kill password is re-used for re-commissioning.

Operation: Unsigned character. It determines the re-commissioning type or mode. Possible values are:

- **1** - Recommission undoing and disabling block permalock functions.
- **2** - Recommission disabling user memory.
- **3** - Same as **2**.
- **4** - Recommission unlocking mem (undoing lock command) and making kill and access passwords permanently unreadable.
- **5** - Combines functions **1** and **4**.
- **6** - Combines functions **2** and **4**.
- **7** - Same as **6**.

NOTE: The MotoC1G2Recommission parameter is deprecated.

MotoC1G2RecommissionOpSpecResult

This C1G2 OpSpec result parameter returns the result of **MotoC1G2Recommission** within the **TagReportData** parameter.

MotoC1G2RecommissionOpSpecResult

Vendor Identifier: 161

Parameter Subtype: 452

Result: Unsigned character. Result of recommissioning. Possible values are:

- **0** - Success
- **1** - Zero kill password error
- **2** - Insufficient power to perform kill operation
- **3** - Non-specific tag error
- **4** - No response from tag
- **5** - Non-specific reader error

OpSpecID: Unsigned short. OpSpec ID.

NOTE: The MotoC1G2RecommissionOpSpecResult parameter is deprecated.

MotoC1G2BlockPermalock

This C1G2 OpSpec parameter enables support for the block permalock command. It is used in AccessSpecs like other OpSpec parameters. The reader honors this command only when it sets the **CanSupportBlockPermalock** parameter to true in the **GET_READER_CAPABILITIES_RESPONSE** message. If **ReadLock** is zero, the reader ignores the content of **Mask**. If **ReadLock** is set, **Mask** contains a bit mask specifying the blocks to permalock. The **Mask** array size provides the bit mask size and **ReadBlockRange** is ignored. Note that **ReadBlockRange** and **Mask** must be included even though the reader can ignore them depending on the value of **ReadLock**.

MotoC1G2BlockPermalock

Vendor Identifier: 161

Parameter Subtype: 453

OpSpecID: Unsigned short. OpSpec ID.

AccessPassword: Unsigned integer. Access password.

MB: Integer. Memory bank specifying whether **BlockPermaLock** applies to , TID, or user memory.

ReadLock: Boolean value. If false, read the permalock status; if true perform permalock.

BlockPointer: Unsigned short. Block pointer specifying the start address of the mask.

ReadBlockRange: Unsigned short. Block range specifying the range of the mask.

Mask: Unsigned short vector. Mask specifying the memory blocks in the tag to permalock.

MotoC1G2BlockPermalockOpSpecResult

This C1G2 OpSpec result parameter returns the result of **MotoC1G2BlockPermalock** within the **TagReportData** parameter.

MotoC1G2BlockPermalockOpSpecResult

Vendor Identifier: 161

Parameter Subtype: 454

Result: Unsigned character. Result of **BlockPermaLock**. Possible values are:

- **0** - Success
- **1** - Insufficient power to perform lock operation
- **2** - Non-specific tag error
- **3** - No response from tag
- **4** - Non-specific reader error

OpSpecID: Unsigned short. OpSpec ID.

Status: Unsigned short vector. Returns bit mask status bit when ReadLock is set to zero.

MotoNXPChangeEAS

This parameter is a C1G2 Custom OpSpec and can be used in AccessSpecs as with any other C1G2 OpSpec parameter. It enables support for NXP custom changeEAS command. This command applies only when the reader sets the **CanSupportNXPCuxtomCommands** parameter to TRUE in the **GET_READER_CAPABILITIES_RESPONSE** message. If the reader sees this command when **CanSupportNXPCuxtomCommands** parameter is false, the reader returns an error message. The response to the Change EAS OPSpec is the presence of

MotoNXPChangeEASOpSpecResult in the Tag Report indicating the result of the Change EAS operation. Once the EAS State is set on a tag the tag backscatters alarm code for the EAS Alarm request.

MotoNXPChangeEAS
Vendor Identifier: 161
Parameter Subtype: 455
OpSpecID: Unsigned short. OpSpec ID.
AccessPassword: Unsigned integer. Access password.
EASState: Boolean value. If true, set the EAS flag on the chosen tags, otherwise reset the EAS flag.

MotoNXPChangeEASOpSpecResult

This is a C1G2 Custom OpSpec result parameter. This parameter returns the result of **MotoNXPChangeEAS** within the **TagReportData** parameter.

MotoNXPChangeEASOpSpecResult
Vendor Identifier: 161
Parameter Subtype: 456
Result: Unsigned character. Result of **NXPChangeEAS**. Possible values are:

- **0** - Success
- **1** - Insufficient power to perform custom operation
- **2** - Non-specific tag error
- **3** - No response from tag
- **4** - Non-specific reader error

OpSpecID: Unsigned short. OpSpec ID.

MotoNXPSetQuiet

This parameter is a C1G2 Custom OpSpec and can be used in **AccessSpecs** as with any other C1G2 OpSpec parameter. It enables support for NXP custom SetQuiet command. This command applies only when the reader sets the **CanSupportNXPCuxtomCommands** parameter to TRUE in the **GET_READER_CAPABILITIES_RESPONSE** message. If the reader sees this command when **CanSupportNXPCuxtomCommands** parameter is false, the reader returns an error message. The response to the SetQuiet OPSpec is the presence of **MotoNXPSetQuietOpSpecResult** in the Tag Report indicating the result of the SetQuiet operation. Once a set quiet is performed on the tag, the tag overrides all memory data and CRC's with Zero. No access operation works on the tag.

MotoNXPSetQuiet
Vendor Identifier: 161
Parameter Subtype: 457
OpSpecID: Unsigned short. OpSpec ID.
AccessPassword: Unsigned integer. Access password.

NOTE: In the FX9500 and FX9600, NXP tags in quiet mode are not exposed because IDs equal to all zeroes are n*ever* reported.

NOTE: NXP Set Quiet Tag and Reset Quiet commands are disabled by default on the reader.

MotoNXPSetQuietOpSpecResult

This is a C1G2 Custom OpSpec result parameter. This parameter returns the result of **MotoNXPSetQuiet** within the **TagReportData** parameter.

MotoNXPSetQuietOpSpecResult

Vendor Identifier: 161

Parameter Subtype: 458

Result: Unsigned character. Result of **NXPChangeEAS**. Possible values are:

- **0** - Success
- **1** - Insufficient power to perform custom operation
- **2** - Non-specific tag error
- **3** - No response from tag
- **4** - Non-specific reader error

OpSpecID: Unsigned short. OpSpec ID.

MotoNXPResetQuiet

This parameter is a C1G2 Custom OpSpec and can be used in **AccessSpecs** as with any other C1G2 OpSpec parameter. It enables support for NXP custom ResetQuiet command. This command applies only when the reader sets the **CanSupportNXPCuxtomCommands** parameter to TRUE in the **GET_READER_CAPABILITIES_RESPONSE** message. If the reader sees this command when **CanSupportNXPCuxtomCommands** parameter is false, the reader returns an error message. The response to the ResetQuiet OPSpec is the presence of **MotoNXPResetQuietOpSpecResult** in the Tag Report indicating the result of the ResetQuiet operation. A successful reset quiet operation opens the tag for inventory and access operations.

MotoNXPResetQuiet

Vendor Identifier: 161

Parameter Subtype: 459

OpSpecID: Unsigned short. OpSpec ID.

AccessPassword: Unsigned integer. Access password.

NOTE: NXP Set Quiet Tag and Reset Quiet commands are disabled by default on the reader.

MotoNXPResetQuietOpSpecResult

This is a C1G2 Custom OpSpec result parameter. This parameter returns the result of **MotoNXPResetQuiet** within the **TagReportData** parameter.

MotoNXPResetQuietOpSpecResult

Vendor Identifier: 161

Parameter Subtype: 460

Result: Unsigned character. Result of **NXPChangeEAS**. Possible values are:

- **0** - Success
- **1** - Insufficient power to perform custom operation
- **2** - Non-specific tag error
- **3** - No response from tag
- **4** - Non-specific reader error

OpSpecID: Unsigned short. OpSpec ID.

MotoNXPCalibrate

This parameter is a C1G2 Custom OpSpec and can be used in **AccessSpecs** as with any other C1G2 OpSpec parameter. It enables support for NXP custom Calibrate command. This command applies only when the reader sets the **CanSupportNXPCuxtomCommands** parameter to TRUE in the **GET_READER_CAPABILITIES_RESPONSE** message. If the reader sees this command when **CanSupportNXPCuxtomCommands** parameter is false, the reader returns an error message. The response to the Calibrate OPSpec is the presence of **MotoNXPCalibrateOpSpecResult** in the Tag Report indicating the result of the calibrate operation. The reader sends back 512 bits (looped) of user memory data in the calibrate opspec result upon success.

MotoNXPCalibrate

Vendor Identifier: 161

Parameter Subtype: 461

OpSpecID: Unsigned short. OpSpec ID.

AccessPassword: Unsigned integer. Access password.

MotoNXPCalibrateOpSpecResult

This is a C1G2 Custom OpSpec result parameter. This parameter returns the result of **MotoNXPCalibrate** within the **TagReportData** parameter.

MotoNXPCalibrateOpSpecResult

Vendor Identifier: 161

Parameter Subtype: 462

Result: Unsigned character. Result of **NXPChangeEAS**. Possible values are:

- **0** - Success
- **1** - Insufficient power to perform custom operation
- **2** - Non-specific tag error
- **3** - No response from tag
- **4** - Non-specific reader error

OpSpecID: Unsigned short. OpSpec ID.

ReadData: Unsigned short vector. Data returned by the calibrate command. The first 512 bits.

MotoNXPEASAlarmSpec

This parameter is a custom spec in LLRP InventoryParameter Spec. It enables support for NXP custom EAS Alarm command. This command applies only when the reader sets the **CanSupportNXPCuxtomCommands** parameter to true in the **GET_READER_CAPABILITIES_RESPONSE** message. If the reader sees this command when **CanSupportNXPCuxtomCommands** parameter is false, the reader returns an error message. The presence of this parameter allows the reader to use the **InventoryParameterSpec** to perform an EAS alarm operation on the specified subset of antennas (among the antennas specified in AISpec). On the other antennas (specified in the AISpec), the reader performs inventory. If an antenna is specified in **MotoNXPEASAlarmSpec** but not in AISpec, EAS alarm is not performed on this antenna. The EAS alarm does not use the **C1G2TagInventoryMask** specified in the antenna configuration to filter the tags (for sending alarms). The response to the EAS Alarm is a custom notification data of type **MotoNXPEASAlarmNotification**. This is sent back in the LLRP **ReaderEventNotificationData** parameter.

MotoNXPEASAlarmSpec

Vendor Identifier: 161

Parameter Subtype: 463

AntennaIDs: Unsigned short vector. Antenna IDs on which to perform the EAS Alarm operation. If set to zero, this EAS Alarm spec uses all antennas specified in the AISpec. Otherwise, EAS Alarm is performed on the subset of antennas of AISpec as indicated by this field.

MotoNXPEASAlarmNotification

This custom reader event notification data parameter is returned when an alarm code is returned in response to the **MotoNXPEASAlarm** request.

MotoNXPEASAlarmNotification

Vendor Identifier: 161

Parameter Subtype: 464

EASAlarmCode: Unsigned long integer. EAS Alarm code returned by a tag.

AntennaID: <AntennaID Parameter >[Optional].

MotoConnectionFailureReason

This custom reader event notification data parameter is the additional error information sent to the client when connection to the reader fails with a status code value 3 (Failed (any reason other than a connection already exists) in **ConnectionAttemptEvent**).

MotoConnectionFailureReason

Vendor Identifier: 161

Parameter Subtype: 465

ErrorCode: Unsigned character. Error code describing the additional information for LLRP connection request failure. Possible values are:

- **0** - Unknown error
- **1** - Region not configured error

ErrorDescription: UTF-8 String. String describing the reason and additional information for LLRP connection request failure.

MotoCustomCommandOptions

This parameter enables or disables specific custom command options. It is sent as a custom parameter to the **SET_READER_CONFIG** message. The client can retrieve the current custom options state from **GET_READER_CONFIG_RESPONSE** by requesting the reporting of the corresponding custom parameter in **GET_READER_CONFIG** message.

MotoCustomCommandOptions

Vendor Identifier: 161

Parameter Subtype: 466

EnableNXPSetAndResetQuietCommands: Boolean. If true, the NXP commands SetQuiet and ResetQuiet are enabled on the reader. The NXP tags that are set quiet have 0 and 0 CRC. Enable this option in order to use NXP Set and Reset Quiet commands on the reader. The default state of this option is disabled and the reader reports Unsupported parameter when **MotoNXPSetQuiet** and **MotoNXPResetQuiet** Opspecs are included in the AccessSpec.

MotoFujitsuChangeWordLock

This parameter is a C1G2 Custom OpSpec and can be used in **AccessSpecs** as with any other C1G2 OpSpec parameter. It enables support for Fujitsu custom **ChangeWordLock** command. This command applies only when the reader sets the **CanSupportFujitsuCustomCommands** parameter to TRUE in the **GET_READER_CAPABILITIES_RESPONSE** message. If the reader sees this command when the **CanSupportFujitsuCustomCommands** parameter is false, the reader returns an error message. The response to the **ChangeWordLock** OPSpec is the presence of **MotoFujitsuChangeWordLockOpSpecResult** in the tag report indicating the result of the **ChangeWordLock** operation. **ChangeWordLock** can set/reset the word lock flags of up to two consecutive words. Setting a word's word lock flag to high prevents any future writes to the word until the word lock flag is reset to low. This operation is secured by the password for enclosing block group and is currently supported only for words in the user memory bank.

MotoFujitsuChangeWordLock

Vendor Identifier: 161

Parameter Subtype: 467

OpSpecID: Unsigned short. OpSpec ID.

WordPointer: Unsigned short. Specifies the word offset for the words to lock.

MB: Integer. Specifies which memory bank contains the word(s) to lock.

PayloadMask: Integer. A 2 bit pattern that specifies whether to apply or ignore the lock action for each of the two words.

PayloadAction: Integer. A 2 bit pattern that specifies the lock action for each of the two consecutive words at the **WordPointer**.

BlockGroupPassword: Unsigned integer. Password of the block group enclosing the word(s) to lock.

MotoFujitsuChangeWordLockOpSpecResult

This is a C1G2 Custom OpSpec result parameter and returns the result of **MotoFujitsuChangeWordLock** within the **TagReportData** parameter.

MotoFujitsuChangeWordLockOpSpecResult

Vendor Identifier: 161

Parameter Subtype: 468

Result: Unsigned character. Result of Fujitsu custom command **ChangeWordLock**. Possible values are:

- **0** - Success
- **1** - Insufficient power to perform custom operation
- **2** - Non-specific tag error
- **3** - No response from tag
- **4** - Non-specific reader error
- **5** - Tag memory overrun error
- **6** - Tag memory locked error

OpSpecID: Unsigned short. OpSpec ID.

MotoFujitsuChangeBlockLock

This parameter is a C1G2 Custom OpSpec and can be used in **AccessSpecs** as with any other C1G2 custom OPSpec parameter. It enables support for Fujitsu's custom **ChangeBlockLock** command. This command applies only when the reader sets **CanSupportFujitsuCustomCommands** parameter to TRUE in the **GET_READER_CAPABILITIES_RESPONSE** message. If the reader sees this command when the **CanSupportFujitsuCustomCommands** parameter is false, the reader returns an error message.

The response to the **ChangeBlockLockOPSpec** is the presence of **MotoFujitsuChangeBlockLockOpSpecResult** in the tag report indicating the result of the **ChangeBlockLock** operation. Every block of a Fujitsu tag has a **BlockLock** flag which, when set high, write locks any unlocked words inside the block after a write operation to that word. In other words, deasserting the **WordLock** flag for a word inside the block sets the word to high after a write operation, provided the **BlockLock** flag for the enclosing block is set high. The **MotoFujitsuChangeBlockLock** OPSpec can change the state of **BlockLock** flags for all blocks inside a block group.

MotoFujitsuChangeBlockLock

Vendor Identifier: 161

Parameter Subtype: 469

OpSpecID: Unsigned short. OpSpec ID.

BlockGroupPointer: Unsigned character. Specifies the offset for the block group whose blocks are block locked.

PayloadMask: Unsigned short. Specifies to which blocks of the block group the **BlockLock** action applies.

PayloadAction: Unsigned short. Specifies the blocks to be acted upon and the lock action to perform.

BlockGroupPassword: Unsigned integer. Password of the enclosing block group.

MotoFujitsuChangeBlockLockOpSpecResult

This is a C1G2 Custom OpSpec result parameter and returns the result of **MotoFujitsuChangeBlockLock** within the **TagReportData** parameter.

MotoFujitsuChangeBlockLockOpSpecResult

Vendor Identifier: 161

Parameter Subtype: 470

Result: Unsigned character. Result of Fujitsu custom command **ChangeBlockLock**. Possible values are:

- **0** - Success
- **1** - Insufficient power to perform custom operation
- **2** - Non-specific tag error
- **3** - No response from tag
- **4** - Non-specific reader error
- **5** - Tag memory overrun error
- **6** - Tag memory locked error

OpSpecID: Unsigned short. OpSpec ID.

MotoFujitsuReadBlockLock

This parameter is a C1G2 Custom OpSpec and can be used in **AccessSpecs** as with any other C1G2 custom OPSpec parameter. It enables support for Fujitsu's custom **ReadBlockLock** command, and applies only when the reader sets the **CanSupportFujitsuCustomCommands** parameter to TRUE in the **GET_READER_CAPABILITIES_RESPONSE** message. If the reader sees this command when the **CanSupportFujitsuCustomCommands** parameter is false, the reader returns an error message. The response to the **ReadBlockLockOPSpec** is the presence of **MotoFujitsuReadBlockLockOpSpecResult** in the tag report indicating the result of the **ReadBlockLock** operation. **ReadBlockLock** reads the block-lock status of the blocks inside the specified block group.

MotoFujitsuReadBlockLock

Vendor Identifier: 161

Parameter Subtype: 471

OpSpecID: Unsigned short. OpSpec ID.

BlockGroupPointer: Unsigned character. Specifies the offset for the block group whose blocks are to be operated upon.

MotoFujitsuReadBlockLockOpSpecResult

This is a C1G2 Custom OpSpec result parameter and returns the result of **MotoFujitsuReadBlockLock** within the **TagReportData** parameter.

MotoFujitsuReadBlockLockOpSpecResult

Vendor Identifier: 161

Parameter Subtype: 472

Result: Unsigned character. Result of Fujitsu custom command **ReadBlockLock**. Possible values are:

- **0** - Success
- **1** - Insufficient power to perform custom operation
- **2** - Non-specific tag error
- **3** - No response from tag
- **4** - Non-specific reader error
- **5** - Tag memory overrun error
- **6** - Tag memory locked error

OpSpecID: Unsigned short. OpSpec ID.

BlockLockStatus: Unsigned short. A 16-bit pattern indicating the block status of each of the 16 blocks inside the specified block group.

MotoFujitsuChangeBlockOrAreaGroupPassword

This parameter is a C1G2 Custom OpSpec and can be used in **AccessSpecs** as with any other C1G2 custom OPSpec parameter. It enables support for Fujitsu's custom **ChangeBlockGroupPassword** or **ChangeAreaGroupPassword** command. This command applies only when the reader sets the **CanSupportFujitsuCustomCommands** parameter to TRUE in the **GET_READER_CAPABILITIES_RESPONSE** message. If the reader sees this command when the **CanSupportFujitsuCustomCommands** parameter is false, the reader returns an error message. The response to the **ChangeBlockOrAreaGroupPasswordOPSpec** is the presence of **MotoFujitsuChangeBlockOrAreaGroupPasswordOpSpecResult** in the tag report indicating the result of the **ChangeBlockOrAreaGroupPassword** operation. The **ChangeBlockOrAreaGroupPassword** operation changes the password of the specified block group or area group. It is secured by the current block group or area group password.

MotoFujitsuChangeBlockOrAreaGroupPassword

Vendor Identifier: 161

Parameter Subtype: 473

OpSpecID: Unsigned short. OpSpec ID.

BlockOrAreaGroupPointer: Unsigned character. Specifies the offset for the block or area group whose blocks are to be operated upon.

currentPassword: Unsigned integer. Current block group password.

newPassword: Unsigned integer. New block group password.

MotoFujitsuChangeBlockOrAreaGroupPasswordOpSpecResult

This is a C1G2 Custom OpSpec result parameter and returns the result of **MotoFujitsuChangeBlockOrAreaGroupPassword** within the **TagReportData** parameter.

MotoFujitsuChangeBlockOrAreaGroupPasswordOpSpecResult

Vendor Identifier: 161

Parameter Subtype: 474

Result: Unsigned character. Result of Fujitsu custom command **ChangeBlockOrAreaGroupPassword**. Possible values are:

- **0** - Success
- **1** - Insufficient power to perform custom operation
- **2** - Non-specific tag error
- **3** - No response from tag
- **4** - Non-specific reader error
- **5** - Tag memory overrun error
- **6** - Tag memory locked error

OpSpecID: Unsigned short. OpSpec ID.

MotoFujitsuBurstWrite

This parameter is a C1G2 Custom OpSpec and can be used in **AccessSpecs** as with any other C1G2 custom OPSpec parameter. It enables support for Fujitsu's custom **BurstWrite** command and applies only when the reader sets **CanSupportFujitsuCustomCommands** parameter to TRUE in the **GET_READER_CAPABILITIES_RESPONSE** message. If the reader sees this command when the **CanSupportFujitsuCustomCommands** parameter is false, the reader returns an error message. The response to the **BurstWriteOPSpec** is the presence of **MotoFujitsuBurstWriteOpSpecResult** in the tag report indicating the result of the **BurstWrite** operation. The **BurstWrite** operation writes an even number of words and returns the number of words unsuccessfully written.

MotoFujitsuBurstWrite

Vendor Identifier: 161

Parameter Subtype: 475

OpSpecID: Unsigned short. OpSpec ID.

AccessPassword: Unsigned integer.

MB: Integer. Memory bank.

WordPointer: Unsigned short. Word offset, must be an even number.

BurstWriteData: Unsigned short vector. Data to write, must have an even length.

MotoFujitsuBurstWriteOpSpecResult

This is a C1G2 Custom OpSpec result parameter and returns the result of **MotoFujitsuBurstWrite** within the **TagReportData** parameter.

MotoFujitsuBurstWriteOpSpecResult

Vendor Identifier: 161

Parameter Subtype: 476

Result: Unsigned character. Result of Fujitsu custom command **BurstWrite**. Possible values are:

- **0** - Success
- **1** - Insufficient power to perform custom operation
- **2** - Non-specific tag error
- **3** - No response from tag
- **4** - Non-specific reader error
- **5** - Tag memory overrun error
- **6** - Tag memory locked error

OpSpecID: Unsigned short. OpSpec ID.

WordsNotWritten: Unsigned character. Number of words unsuccessfully written.

MotoFujitsuBurstErase

This parameter is a C1G2 Custom OpSpec and can be used in **AccessSpecs** as with any other C1G2 custom OPSpec parameter. It enables support for Fujitsu's custom **BurstErase** command. This command applies only when the reader sets **CanSupportFujitsuCustomCommands** parameter to TRUE in the **GET_READER_CAPABILITIES_RESPONSE** message. If the reader sees this command when the **CanSupportFujitsuCustomCommands** parameter is false, the reader returns an error message. The response to the **BurstEraseOPSpec** is the presence of **MotoFujitsuBurstEraseOpSpecResult** in the tag report indicating the result of the **BurstWrite** operation. The **BurstWrite** operation erases an even number of words and returns the number of words unsuccessfully erased.

MotoFujitsuBurstErase

Vendor Identifier: 161

Parameter Subtype: 477

OpSpecID: Unsigned short. OpSpec ID.

AccessPassword: Unsigned integer.

MB: Integer. Memory bank.

WordPointer: Unsigned short. Word offset, must be an even number.

WordCount: Unsigned character. Number of words to erase, must be an even number.

MotoFujitsuBurstEraseOpSpecResult

This is a C1G2 Custom OpSpec result parameter and returns the result of **MotoFujitsuBurstErase** within the **TagReportData** parameter.

MotoFujitsuBurstEraseOpSpecResult

Vendor Identifier: 161

Parameter Subtype: 478

Result: Unsigned character. Result of Fujitsu custom command **BurstErase**. Possible values are:

- **0** - Success
- **1** - Insufficient power to perform custom operation
- **2** - Non-specific tag error
- **3** - No response from tag
- **4** - Non-specific reader error
- **5** - Tag memory overrun error
- **6** - Tag memory locked error

OpSpecID: Unsigned short. OpSpec ID.

WordsNotErased: Unsigned character. Number of words unsuccessfully erased.

MotoFujitsuAreaReadLock

This parameter is a C1G2 Custom OpSpec and can be used in **AccessSpecs** as with any other C1G2 custom OPSpec parameter. It enables support for Fujitsu's custom **AreaReadLock** command. This command applies only when the reader sets the **CanSupportFujitsuCustomCommands** parameter to TRUE in the **GET_READER_CAPABILITIES_RESPONSE** message. If the reader sees this command when the **CanSupportFujitsuCustomCommands** parameter is false, the reader returns an error message.

The response to the **AreaReadLockOPSpec** is the presence of **MotoFujitsuAreaReadLockOpSpecResult** in the tag report indicating the result of the **AreaReadLock** operation. The **AreaReadLock** command specifies the **AreaReadLock** status in the control memory with password.

MotoFujitsuAreaReadLock

Vendor Identifier: 161

Parameter Subtype: 479

OpSpecID: Unsigned short. OpSpec ID.

AreaGroupPointer: Unsigned character. Specifies the offset for the area group whose area is to be read-locked.

AreaReadLockMask: Unsigned short. Specifies for which areas of the area group the **ReadLock** action applies. Bit value 0 indicates ignore the associated action field and retain the current setting.
Bit value 1 indicates implement the associated action field and overwrite the current **AreaReadLock** setting.

AreaReadLockAction: Unsigned short. Specifies the lock action on the areas as the **AreaReadLockMask** specifies. Bit value 0 indicates deassert **AreaReadLock**, and 1 indicates assert **AreaReadLock**.

AreaGroupPassword: Unsigned integer. Password of the enclosing area group.

MotoFujitsuAreaReadLockOpSpecResult

This is a C1G2 Custom OpSpec result parameter and returns the result of **MotoFujitsuAreaReadLock** within the **TagReportData** parameter.

MotoFujitsuAreaReadLockOpSpecResult

Vendor Identifier: 161

Parameter Subtype: 480

Result: Unsigned character. Result of Fujitsu custom command **AreaReadLock**. Possible values are:

- **0** - Success
- **1** - Insufficient power to perform custom operation
- **2** - Non-specific tag error
- **3** - No response from tag
- **4** - Non-specific reader error
- **5** - Tag memory overrun error
- **6** - Tag memory locked error

OpSpecID: Unsigned short. OpSpec ID.

MotoFujitsuAreaWriteLock

This parameter is a C1G2 Custom OpSpec and can be used in **AccessSpecs** as with any other C1G2 custom OPSpec parameter. It enables support for Fujitsu's custom **AreaWriteLock** command. This command applies only when the reader sets the **CanSupportFujitsuCustomCommands** parameter to TRUE in the **GET_READER_CAPABILITIES_RESPONSE** message. If the reader sees this command when the **CanSupportFujitsuCustomCommands** parameter is false, the reader returns an error message.

The response to **AreaWriteLockOPSpec** is the presence of **MotoFujitsuAreaWriteLockOpSpecResult** in the tag report indicating the result of the **AreaWriteLock** operation. The **AreaWriteLock** command specifies the **AreaWriteLock** status in the control memory with password.

MotoFujitsuAreaWriteLock

Vendor Identifier: 161

Parameter Subtype: 481

OpSpecID: Unsigned short. OpSpec ID.

AreaGroupPointer: Unsigned character. Specifies the offset for the area group whose area is to be write-locked.

AreaWriteLockMask: Unsigned short. Specifies for which areas of the area group the **WriteLock** action applies. Bit value 0 indicates ignore the associated action field and retain the current setting.
Bit value 1 indicates implement the associated action field and overwrite the current **AreaWriteLock** setting.

AreaWriteLockAction: Unsigned short. Specifies the lock action on the areas as the **AreaWriteLockMask** specifies. Bit value 0 indicates deassert **AreaWriteLock**, and 1 indicates assert **AreaWriteLock**.

AreaGroupPassword: Unsigned integer. Password of the enclosing area group.

MotoFujitsuAreaWriteLockOpSpecResult

This is a C1G2 Custom OpSpec result parameter and returns the result of **MotoFujitsuAreaWriteLock** within **TagReportData** parameter.

MotoFujitsuAreaWriteLockOpSpecResult

Vendor Identifier: 161

Parameter Subtype: 482

Result: Unsigned character. Result of Fujitsu custom command **AreaWriteLock**. Possible values are:

- **0** - Success
- **1** - Insufficient power to perform custom operation
- **2** - Non-specific tag error
- **3** - No response from tag
- **4** - Non-specific reader error
- **5** - Tag memory overrun error
- **6** - Tag memory locked error

OpSpecID: Unsigned short. OpSpec ID.

MotoFujitsuAreaWriteLockWOPassword

This parameter is a C1G2 Custom OpSpec and can be used in **AccessSpecs** as with any other C1G2 custom OPSpec parameter. It enables support for Fujitsu's custom **AreaWriteLockWOPassword** command. This command applies only when the reader sets **CanSupportFujitsuCustomCommands** parameter to TRUE in the **GET_READER_CAPABILITIES_RESPONSE** message. If the reader sees this command when the **CanSupportFujitsuCustomCommands** parameter is false, the reader returns an error message.

The response to the **AreaWriteLockWOPasswordOPSpec** is the presence of **MotoFujitsuAreaWriteLockWOPasswordOpSpecResult** in the tag report indicating the result of the **AreaWriteLockWOPassword** operation. The **AreaWriteLockWOPassword** command can set the **AreaWriteLock** status in the control memory without password. This command can not reset the **AreaWriteLock** status.

MotoFujitsuAreaWriteLockWOPassword

Vendor Identifier: 161

Parameter Subtype: 483

OpSpecID: Unsigned short. OpSpec ID.

AreaGroupPointer: Unsigned character. Specifies the offset for the area group whose area is to be write-locked.

AreaWriteLockWOPasswordAction: Unsigned short. Specifies the lock action on the areas in the group as specified by **AreaGroupPointer**. Bit value 0 indicates no action and 1 indicates assert **AreaWriteLock**.

MotoFujitsuAreaWriteLockWOPasswordOpSpecResult

This is a C1G2 Custom OpSpec result parameter and returns the result of **MotoFujitsuAreaWriteLockWOPassword** within the **TagReportData** parameter.

MotoFujitsuAreaWriteLockOpSpecResult

Vendor Identifier: 161

Parameter Subtype: 484

Result: Unsigned character. Result of Fujitsu custom command **AreaWriteLockWOPassword**. Possible values are:

- **0** - Success
- **1** - Insufficient power to perform custom operation
- **2** - Non-specific tag error
- **3** - No response from tag
- **4** - Non-specific reader error
- **5** - Tag memory overrun error
- **6** - Tag memory locked error

OpSpecID: Unsigned short. OpSpec ID.

MotoNXPChangeConfig

This parameter is a C1G2 Custom OpSpec. And, it can be used in AccessSpecs just like any other C1G2 custom OPSpec parameter. It enables support for NXP's custom ChangeConfig command. This command is honored only when reader sets CanSupportNXPCustomCommands parameter to TRUE in GET_READER_CAPABILITIES_RESPONSE message. If this command is seen at the reader when CanSupportNXPCustomCommands parameter is advertised as false by the reader, the reader shall return an error message. The response to the ChangeConfigOPSpec shall be the presence of MotoNXPChangeConfigOpSpecResult in the Tag Report indicating the result of the ChangeConfig operation.

MotoNXPChangeConfig Parameter

Vendor Identifier: 161

Parameter Subtype: 485

OpSpecID: Unsigned Short. OpSpec ID.

AccessPassword: Unsigned Integer. Access Password

NXPChangeConfigWord: Unsigned Short. NXP Change Config Word.

MotoNXPChangeConfigOpSpecResult

This is a C1G2 Custom OpSpec result parameter. This parameter returns the result of MotoNXPChangeConfig within TagReportData parameter.

MotoNXPChangeConfigOpSpecResult Parameter

Vendor Identifier: 161

Parameter Subtype: 486

Result: Unsigned Character. Result of NXP custom command ChangeConfig.

Possible Values:

Value	Definition
0	Success
1	Insufficient power to perform custom operation
2	Non-specific tag error
3	No response from tag
4	Non-specific reader error
5	Tag Memory Overrun error
6	Tag Memory Locked error

OpSpecID: Unsigned Short. OpSpec ID.

NXPChangeConfigWord: Unsigned Short. Current NXP Change Config Word. This is valid only when the Result of the operation is Success

MotoImpinjQT

This parameter is a C1G2 Custom OpSpec. And, it can be used in AccessSpecs just like any other C1G2 custom OPSpec parameter. It enables support for Impinj's custom QT command. This command is honored only when reader sets CanSupportImpinjCustomCommands parameter to TRUE in GET_READER_CAPABILITIES_RESPONSE message. If this command is seen at the reader when CanSupportImpinjCustomCommands parameter is advertised as false by the reader, the reader shall return anerror message. The response to the MotoImpinjQT shall be the presence of MotoImpinjQTOpSpecResult in theTag Report indicating the result of the QT operation.

MotoImpinjQT Parameter

Vendor Identifier: 161

Parameter Subtype: 487

OpSpecID: Unsigned Short. OpSpec ID.

AccessPassword: Unsigned Integer.

QT_Write: Boolean. Indicates whether the QT command is called for read or write of the QT Control data

QT_Persist: Boolean. Indicates whether the QT control is written to nonvolatile (NVM) or volatile memory

QTData: <QTData Parameter >[Optional]

QTData

This parameter is a Custom parameter and can be used in MotoImpinjQT and MotoImpinjQTOpSpecResult parameter. When used with MotoImpinjQT this parameter is used to set the QT Control data for QT_Write Operation. This parameter is ignored when QT_Write is set to 0. When used with MotoImpinjQTOpSpecResult this parameter is used to indicate the current setting QT Control data when QT_Write is set to 0.

QTData Parameter

Vendor Identifier: 161

Parameter Subtype: 488

QT_Control: Unsigned Short. QT Control bits. Bit 15 controls the Short Range Feature and Bit 14 Controls the Public or Private Memory Map.

MotoImpinjQTOpSpecResult

This is a C1G2 Custom OpSpec result parameter. This parameter returns the result of MotoImpinjQT within TagReportData parameter.

MotoImpinjQTOpSpecResult Parameter

Vendor Identifier: 161

Parameter Subtype: 489

Result: Unsigned Character. Result of Impinj custom command QT

Possible Values:

Value	Definition
0	Success
1	Insufficient power to perform custom operation
2	Non-specific tag error
3	No response from tag
4	Non-specific reader error
5	Tag Memory Overrun error
6	Tag Memory Locked error

OpSpecID: Unsigned Short. OpSpec ID.

QT_Write: Boolean. Indicates whether the QT command has been called for read or write of the QT Control data

QTData: <QTData Parameter >[Optional]

MotoC1G2Authenticate

This is a C1G2 OpSpec and is used in AccessSpecs like other OpSpec parameters. It enables support for Authenticate command. This command applies only when the reader sets CanSupportG2V2Commands parameter to TRUE in GET_READER_CAPABILITIES_RESPONSE message.

MotoC1G2Authenticate Parameter

Vendor Identifier: 161

Parameter Subtype: 490

OpSpecID: OpSpec ID.

AccessPassword: The access password.

SenResp: Specifies whether a tag backscatters its response or stores the response in its ResponseBuffer.

IncRespLen: Specifies whether a tag omits or includes the length in its reply.

CSI: Selects the cryptographic suite that Tag and Interrogator use for the authentication as well as for all subsequent communications.

Message: The parameters for the authentication in the bits format.

MotoC1G2AuthenticateOpSpecResult

This is a C1G2 OpSpec result parameter. This parameter returns the result of MotoC1G2Authenticate within TagReportData parameter.

MotoC1G2AuthenticateOpSpecResult Parameter

Vendor Identifier: 161

Parameter Subtype: 491

Result: Result of Authenticate. List of possible results:

- 0 - Success
- 1 - Insufficient power to perform Read operation
- 2 - Non-specific tag error
- 3 - No response from tag
- 4 - Non-specific reader error
- 5 - in_process: Still working
- 6 - in_process: Successful stored response without length
- 7 - in_process: Successful stored response with length
- 8 - in_process: Successful send response without length
- 9 - in_process: Successful send response with length
- 10 - in_process: Error code stored response without length
- 11 - in_process: Error code stored response with length
- 12 - in_process: Error code send response without length
- 13 - in_process: Error code send response with length

OpSpecID: OpSpec ID.

DataBits: Returns bit string, could be an error code, or an authenticate response.

MotoC1G2ReadBuffer

This is a C1G2 OpSpec and is used in AccessSpecs like other OpSpec parameters. It enables support for Read Buffer command. This command applies only when the reader sets CCanSupportG2V2Commands parameter to TRUE in GET_READER_CAPABILITIES_RESPONSE message.

MotoC1G2ReadBuffer Parameter

Vendor Identifier: 161

Parameter Subtype: 492

OpSpecID: OpSpec ID.

AccessPassword: The access password.

WordPtr: Word Pointer. This specifies the starting address for the read.

BitCount: Bit Count. This specifies the number of bits to read.

MotoC1G2ReadBufferOpSpecResult

This is a C1G2 OpSpec result parameter. This parameter returns the result of MotoC1G2ReadBuffer within TagReportData parameter.

MotoC1G2ReadBufferOpSpecResult Parameter

Vendor Identifier: 161

Parameter Subtype: 493

Result: Result of ReadBuffer. List of possible results:

- 0 - Success
- 1 - Insufficient power to perform Read operation
- 2 - Non-specific tag error
- 3 - No response from tag
- 4 - Non-specific reader error

OpSpecID: OpSpec ID.

DataBits: Returns bit string, could be an error code, or an authenticate response.

MotoC1G2Untraceable

This is a C1G2 OpSpec and is used in AccessSpecs like other OpSpec parameters. It enables support for Untraceable command. This command applies only when the reader sets CanSupportG2V2Commands parameter to TRUE in GET_READER_CAPABILITIES_RESPONSE message.

MotoC1G2Untraceable Parameter

Vendor Identifier: 161

Parameter Subtype: 494

OpSpecID: OpSpec ID.

AccessPassword: The access password.

U: Specifies a value for the U bit in XPC_W1.

EPC: This show or hide bit (MSB) and 5 length bits (5 LSBs).

TID: Specifies the TID memory that a Tag untraceably hides.

User: Specifies whether a Tag untraceably hides User memory.

Range: Specifies a Tag’s operating range.

MotoC1G2UntraceableOpSpecResult

This is a C1G2 OpSpec result parameter. This parameter returns the result of MotoC1G2Untraceable within TagReportData parameter.

MotoC1G2UntraceableOpSpecResult Parameter

Vendor Identifier: 161

Parameter Subtype: 495

Result: Result of Untraceable. List of possible results:

- 0 - Success
- 1 - Insufficient power to perform Read operation
- 2 - Non-specific tag error
- 3 - No response from tag
- 4 - Non-specific reader error

OpSpecID: OpSpec ID.

DataBits: Returns bit string, could be an error code, or an authenticate response.

MotoC1G2Crypto

This is a C1G2 OpSpec and is used in AccessSpecs like other OpSpec parameters. It enables support for NXP crypto command. This command applies only when reader sets CanSupportG2V2Commands parameter to TRUE in GET_READER_CAPABILITIES_RESPONSE message.

MotoC1G2Crypto Parameter

Vendor Identifier: 161

Parameter Subtype: 496

OpSpecID: OpSpec ID.

AccessPassword: The access password.

KeyID: The key that should be used by the tag in its response.

IChallenge: This is an array of 3-32 bits words. The challenge should first fill the most significant bit of the array. The first 32 bits should be in IChallenge[0], the next 32 bits should be in IChallenge[1], and the final 16 bits should be the most significant bits in IChallenge[2].

CustomData: 0 indicates no custom data and tag authentication only. 1 indicates the data are included in response.

Profile: A 4-bit pointer that selects a memory profile for the additional custom data. Values above 15 returns an error message.

Offset: Specifies a 12-bit offset (in multiple of 64-bit blocks) that needs to be added to the address that is specified by Profile. Values above 4095 returns an error message.

BlockCount: A 4-bit number that defines the size of the customer data as a number of 64-bit blocks. Values above 15 returns an error message.

ProtMode: A 4-bit value that select the operation mode that is used to process the custom data. Values above 15 returns an error message.

MotoC1G2CryptoOpSpecResult

This is a C1G2 OpSpec result parameter. This parameter returns the result of MotoC1G2Crypto within TagReportData parameter.

MotoC1G2CryptoOpSpecResult Parameter

Vendor Identifier: 161

Parameter Subtype: 497

Result: Result of Crypto. List of possible results:

- 0 - Success
- 1 - Insufficient power to perform Read operation
- 2 - Non-specific tag error
- 3 - No response from tag
- 4 - Non-specific reader error
- 5 - in_process: Still working
- 6 - in_process: Successful stored response without length
- 7 - in_process: Successful stored response with length
- 8 - in_process: Successful send response without length
- 9 - in_process: Successful send response with length
- 10 - in_process: Error code stored response without length
- 11 - in_process: Error code stored response with length
- 12 - in_process: Error code send response without length
- 13 - in_process: Error code send response with length

OpSpecID: OpSpec ID.

DataBits: Returns bit string, could be an error code, or an authenticate response.

MotoTagGPS

This parameter has the GPS information for the tag.

MotoTagGPS Parameter

Vendor Identifier: 161

Parameter Subtype: 1000

GPS: GPS co-ordinates the longitude, latitude and altitude reported by the reader when the tag is seen.

MotoAntennaConfig

This is a custom parameter in LLRP C1G2InventoryCommand, and allows specifying antenna configuration extensions for finer control over the physical antenna configuration. Currently this parameter is added as part of C1G2InventoryCommand since the AntennaConfiguration does not support custom extensions.

MotoAntennaConfig Parameter

Vendor Identifier: 161

Parameter Subtype: 703

MotoAntennaStopCondition: <MotoAntennaStopCondition Parameter >[Optional]

MotoAntennaPhysicalPortConfig: <MotoAntennaPhysicalPortConfig Parameter >[Optional]

MotoAntennaQueryConfig: <MotoAntennaQueryConfig Parameter >[Optional]

MotoAntennaStopCondition

This is a custom parameter in LLRP C1G2InventoryCommand, and allows specifying the stop condition for each antenna. Currently this supports stop conditions based on the dwell time and number of inventory cycles.

MotoAntennaStopCondition Parameter

Vendor Identifier: 161

Parameter Subtype: 704

AntennaStopTrigger: Unsigned character. Specifies the type of stop trigger used for this antenna.

Possible Values:

Value	Definition
0	Dwell_Time
1	Number_Inventory_Cycles

AntennaStopConditionValue: Unsigned short. Stop condition value used to stop inventory on the specific antenna. This value depends on the stop trigger specified in the AntennaStopTrigger field.
If the stop condition is Dwell_Time this value specifies the number of milliseconds for which to perform inventory operation on the antenna. If the stop condition is Number_Inventory_Cycles this values specifies the number of rounds of inventory to perform on this antenna.

MotoAntennaPhysicalPortConfig

This parameter is a Custom parameter in LLRP C1G2InventoryCommand. This parameter allows the user to specify the physical port configuration for the antenna.

MotoAntennaPhysicalPortConfig Parameter

Vendor Identifier: 161

Parameter Subtype: 705

PhysicalTransmitPort: Unsigned short. Specifies the physical transmit port to use for this antenna.

PhysicalReceivePort: Unsigned short. Specifies the physical receive port to use for this antenna.

MotoTagReportContentSelector

This parameter configures the optional parameters reported back as part of the tag report data. This is an optional sub-parameter in the ROReportSpec parameter.

The default event selector setting reports new tag events and tag visibility change events immediately, and reports tag invisible events by a moderation timeout of 8 seconds.

MotoTagReportContentSelector Parameter

Vendor Identifier: 161

Parameter Subtype: 708

EnableZoneID: Boolean. Enables reporting the ZoneID of the antenna on which the tag is inventoried. If the antenna is not part of a zone, the zone ID is not reported.

EnableZoneName: Boolean. Enables reporting the ZoneName of the antenna on which the tag is inventoried. If the antenna is not part of a zone, the zone ID is not reported.

EnableAntennaPhysicalPortConfig: Boolean. Enables reporting the physical port configuration of the antenna on which the tag is inventoried.

EnablePhase: Boolean. Enables reporting the phase information of the antenna on which the tag is inventoried.

EnableGPS: Boolean. Enables reporting the GPS co-ordinates (longitude, latitude and altitude) information that this tag is inventoried on.

EnableMLTReport: Boolean value. Enables reporting the MLT algorithm output values.

MotoTagPhase

This parameter holds the phase information for the tag.

MotoTagPhase Parameter

Vendor Identifier: 161

Parameter Subtype: 709

Phase: Signed short. Phase information the reader reported when this tag was seen. The phase angle is reported in radians varying from 0 to 360^{o}. Phase angle is part of the LLRP tag report data and possible values are 0x8000 = -pi, 0x7fff = +pi (minus a bit)

MotoAntennaQueryConfig

This is a custom parameter in LLRP C1G2InventoryCommand, and allows specifying the SL all and A/B flip feature.

MotoAntennaQueryConfig Parameter

Vendor Identifier: 161

Parameter Subtype: 710

EnableSLAll: Boolean. Specifies inventory all tags irrespective of SL Flag for this antenna.

EnableABFlip: Boolean. Specifies alternate inventory flag for this antenna during inventory.

NXPBrandIDCheckConfig

This is a custom parameter in LLRP C1G2InventoryCommand and allows instructing the reader to perform a BrandID check operation as part of Inventory.

NOTE: This feature works only on NXP UCode-8 and above that supports BrandID feature

NXPBrandIDCheckConfig Parameter

Vendor Identifier: 161

Parameter Subtype: 711

NXPBrandIDCheckConfig: BrandID. Possible values are:

- **0** - Fail
- **1** – Pass

BrandIDCheckStatus

This holds status of BrandID check on the inventory tag.

BrandIDCheckStatus Parameter

Vendor Identifier: 161

Parameter Subtype: 712

NXPBrandIDCheckConfig: BrandID. Possible values are:

- **0** - Fail
- **1** – Pass

ZebraROTriggerSpec

This parameter defines extra triggers, which includes the start trigger and the stop trigger.

ZebraROTriggerSpec Parameter

Vendor Identifier: 161

Parameter Subtype: 801

ZebraROSpecStartTrigger: The extra start trigger.

ZebraROSpecStopTrigger: The extra stop trigger.

ZebraROSpecStartTrigger

This parameter defines the extra start trigger, which has the timelapse trigger or the GPS distance trigger.

ZebraROSpecStartTrigger Parameter

Vendor Identifier: 161

Parameter Subtype: 802

ZebraTimelapseStart: The timelapse start trigger.

ZebraDistance: The distance trigger.

ZebraTimelapseStart

This parameter defines the timelapse start trigger, which includes start point of time of day to trigger periodically.

ZebraTimelapseStart Parameter

Vendor Identifier: 161

Parameter Subtype: 803

TimeOfDay: The specific time of day to start trigger. The time format in [0~23]:[0~59]:[0:59]. For example, 08:32:14. When the string is empty, the value is equivalent to 0:0:0, the timeofday is on the mid-night.

Period: Certain regular interval in second unit. A cycle of inventory and idle. If =0, not periodic and inventory cannot restart after stop.

ZebraDistance

This parameter defines the distance trigger, which includes value to set GPS threshold to trigger inventory operation.

ZebraDistance Parameter

Vendor Identifier: 161

Parameter Subtype: 804

Value: Define distance value, over which Inventory starts running.

ZebraROSpecStopTrigger

This parameter defines the extra stop trigger, which has the timelapse stop trigger.

ZebraROSpecStopTrigger Parameter

Vendor Identifier: 161

Parameter Subtype: 805

ZebraTimelapseStop: The timelapse stop trigger.

ZebraTimelapseStop

This parameter defines the timelapse stop trigger, which includes total running duration with inventory periodic duration.

ZebraTimelapseStop Parameter

Vendor Identifier: 161

Parameter Subtype: 806

TotalDuration: Specific total running duration in a second unit. If = 0, no stop in duration. If work with the GPI and GPS start trigger, the GPI and GPS start trigger provide similar timeofday set in trigger enable time.

PeriodicDuration: Define the inventory duration in a second unit within a periodic window. If =0, no stop in inventory.

Binary Packet Format for Custom Parameters

This section provides the binary packet format for the previous custom messages and parameters.

MOTO_GET_TAG_EVENT_REPORT

0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Rsvd			Ver			Message Type = 1023										Message Length [31:16]															
Message Length [15:00]																Message ID [31:16]															
Message ID [15:00]																Vendor Identifier [31:16]															
Vendor Identifier [15:00] = 161																Message Subtype = 2															

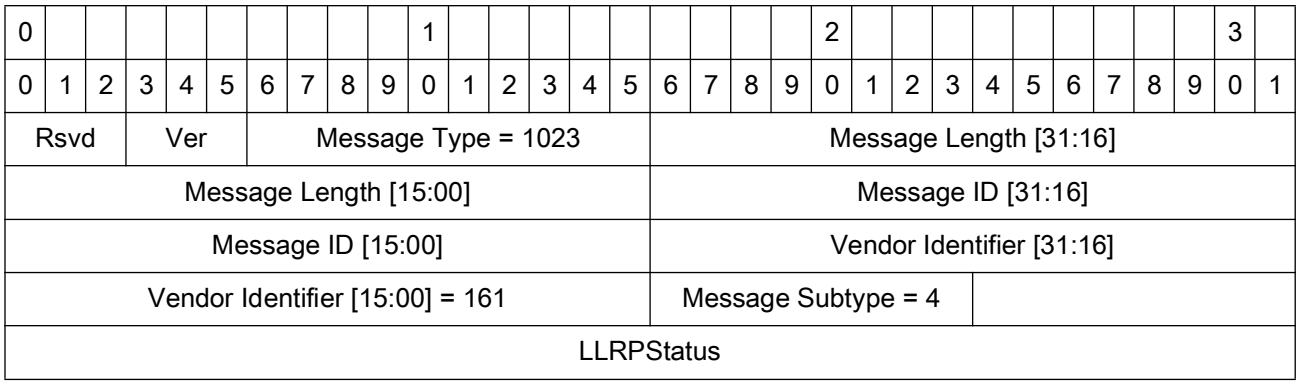
MOTO_PURGE_TAGS

0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Rsvd			Ver			Message Type = 1023										Message Length [31:16]															
Message Length [15:00]																Message ID [31:16]															
Message ID [15:00]																Vendor Identifier [31:16]															
Vendor Identifier [15:00] = 161																Message Subtype = 3			P	Reserved											
Data (0-n)																															

Abbreviations:

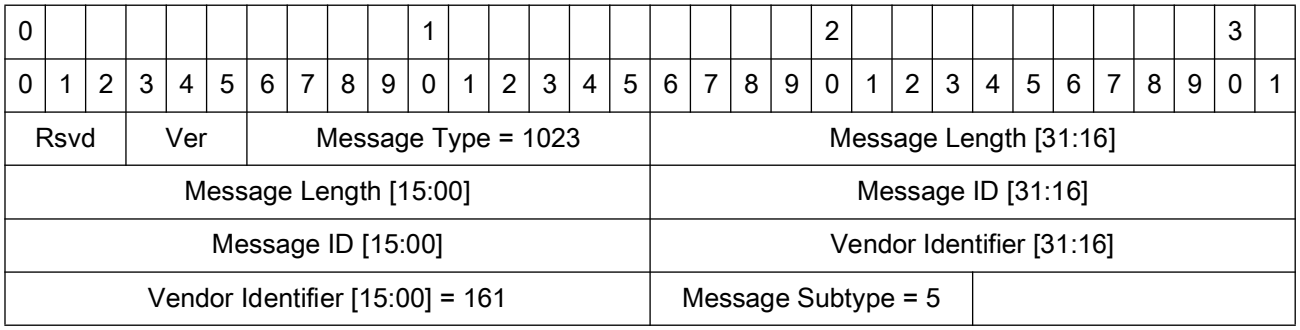
- P - Purge event list only

MOTO_PURGE_TAGS_RESPONSE



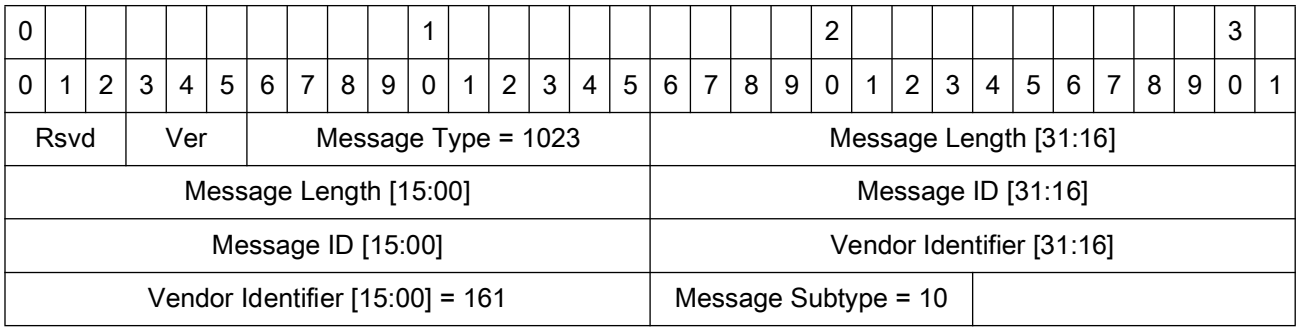
0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Rsvd			Ver			Message Type = 1023										Message Length [31:16]															
Message Length [15:00]																Message ID [31:16]															
Message ID [15:00]																Vendor Identifier [31:16]															
Vendor Identifier [15:00] = 161																Message Subtype = 4															
LLRPStatus																															

MOTO_TAG_EVENT_NOTIFY



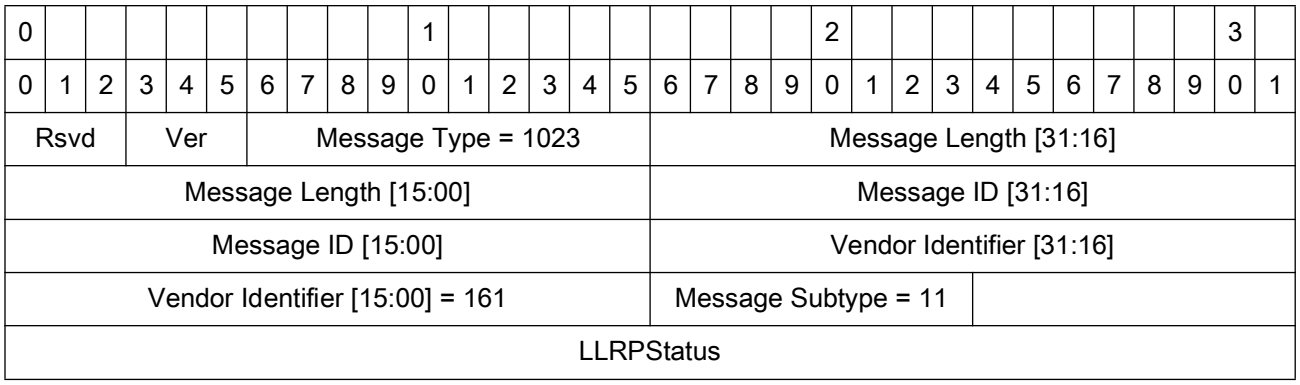
0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Rsvd			Ver			Message Type = 1023										Message Length [31:16]															
Message Length [15:00]																Message ID [31:16]															
Message ID [15:00]																Vendor Identifier [31:16]															
Vendor Identifier [15:00] = 161																Message Subtype = 5															

MOTO_UPDATE_RADIO_FIRMWARE



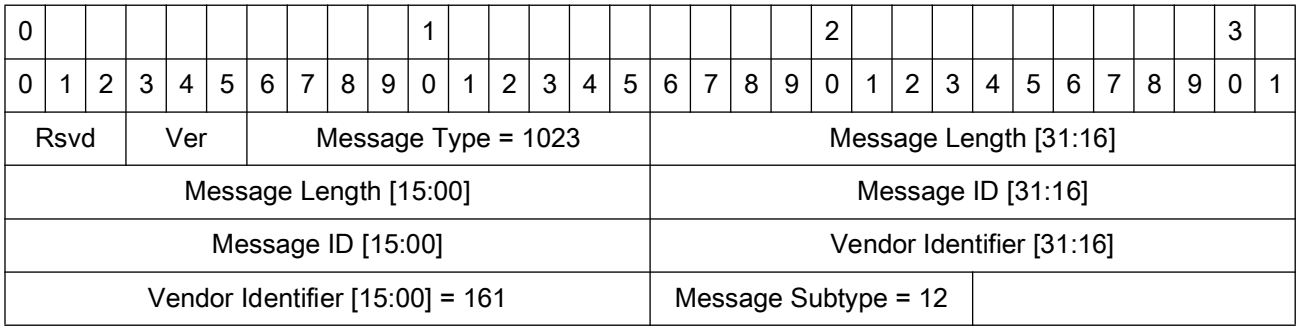
0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Rsvd			Ver			Message Type = 1023										Message Length [31:16]															
Message Length [15:00]																Message ID [31:16]															
Message ID [15:00]																Vendor Identifier [31:16]															
Vendor Identifier [15:00] = 161																Message Subtype = 10															

MOTO_UPDATE_RADIO_FIRMWARE_RESPONSE



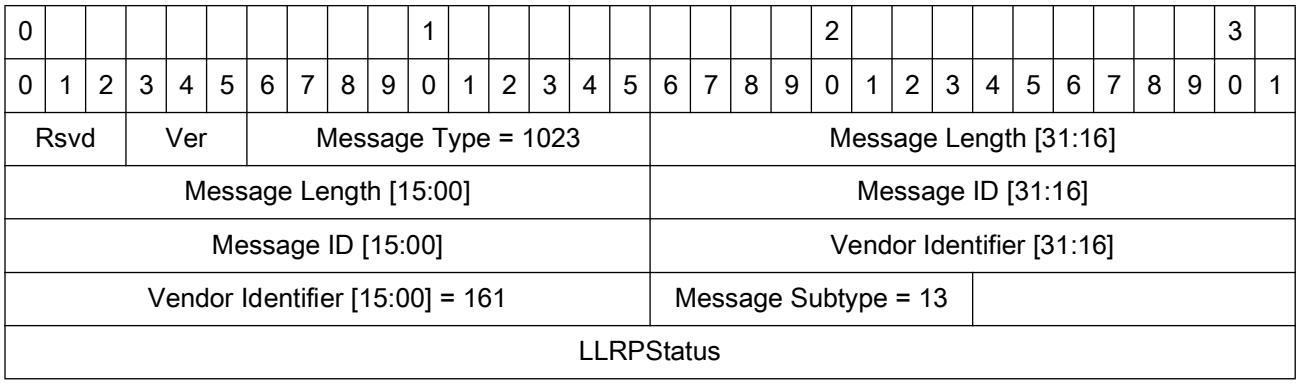
0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Rsvd			Ver			Message Type = 1023										Message Length [31:16]															
Message Length [15:00]																Message ID [31:16]															
Message ID [15:00]																Vendor Identifier [31:16]															
Vendor Identifier [15:00] = 161																Message Subtype = 11															
LLRPStatus																															

MOTO_UPDATE_RADIO_CONFIG



0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Rsvd			Ver			Message Type = 1023										Message Length [31:16]															
Message Length [15:00]																Message ID [31:16]															
Message ID [15:00]																Vendor Identifier [31:16]															
Vendor Identifier [15:00] = 161																Message Subtype = 12															

MOTO_UPDATE_RADIO_CONFIG_RESPONSE



0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Rsvd			Ver			Message Type = 1023										Message Length [31:16]															
Message Length [15:00]																Message ID [31:16]															
Message ID [15:00]																Vendor Identifier [31:16]															
Vendor Identifier [15:00] = 161																Message Subtype = 13															
LLRPStatus																															

MOTO_GET_RADIO_UPDATE_STATUS

0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Rsvd			Ver			Message Type = 1023										Message Length [31:16]															
Message Length [15:00]																Message ID [31:16]															
Message ID [15:00]																Vendor Identifier [31:16]															
Vendor Identifier [15:00] = 161																Message Subtype = 14															

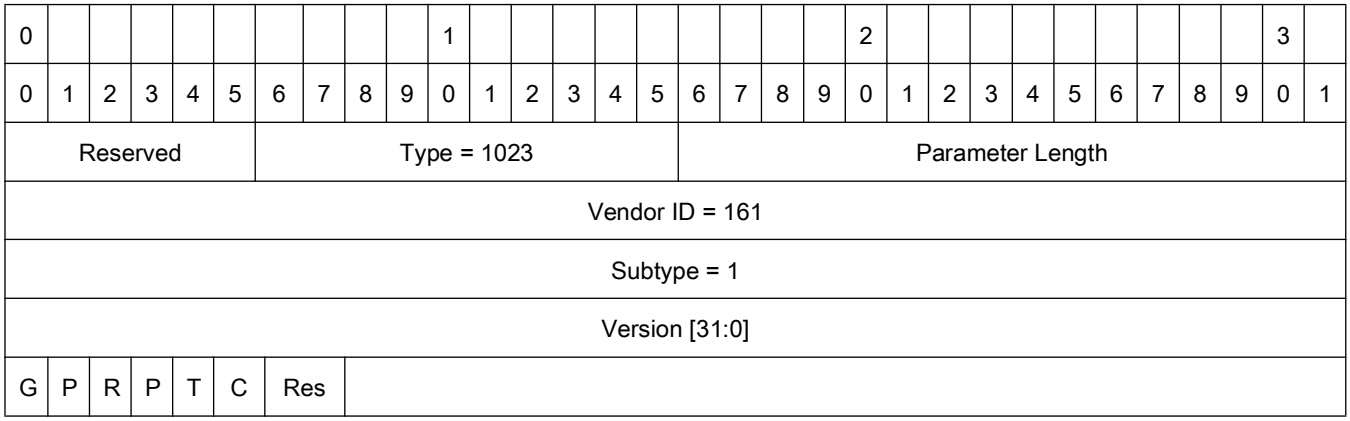
MOTO_GET_RADIO_UPDATE_STATUS_RESPONSE

0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Rsvd			Ver			Message Type = 1023										Message Length [31:16]															
Message Length [15:00]																Message ID [31:16]															
Message ID [15:00]																Vendor Identifier [31:16]															
Vendor Identifier [15:00] = 161																Message Subtype = 15															
UpdateStatusInfo																															

MotoGeneralRequestCapabilities

0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 50																															
RequestedData																															

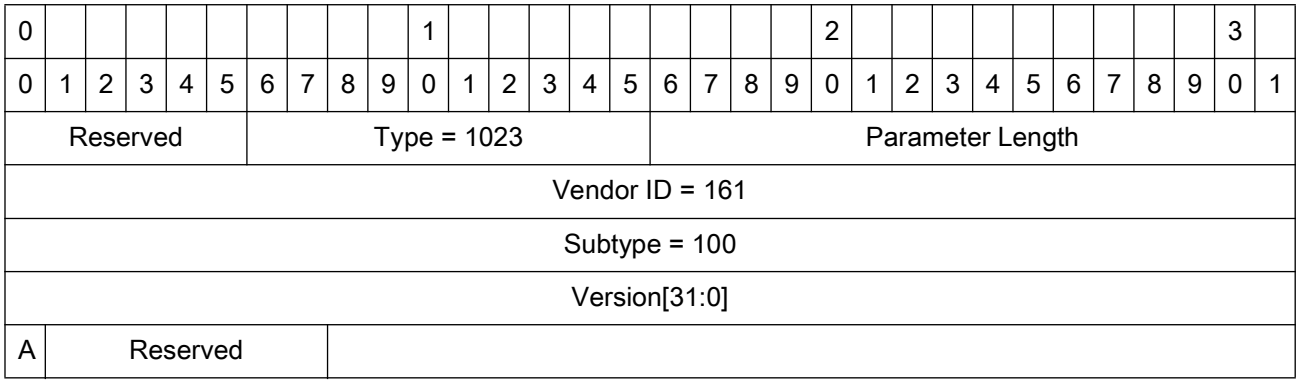
MotoGeneralCapabilities



Abbreviations:

- G - Can report general custom parameter
- P - Can report reader hardware part number
- R - Can report radio version
- P - Can support radio power state
- T - Can support radio transmit delay
- C - Can support custom trigger

MotoAutonomousCapabilities



Abbreviations:

- A - Can support autonomous mode

MotoTagEventsGenerationCapabilities

0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 120																															
Version[31:0]																															
E	F	M	Reserved																												

Abbreviations:

- E - Can select event type to monitor and report
- F - Can select reporting format
- M - Can select moving event type to be monitored and reported

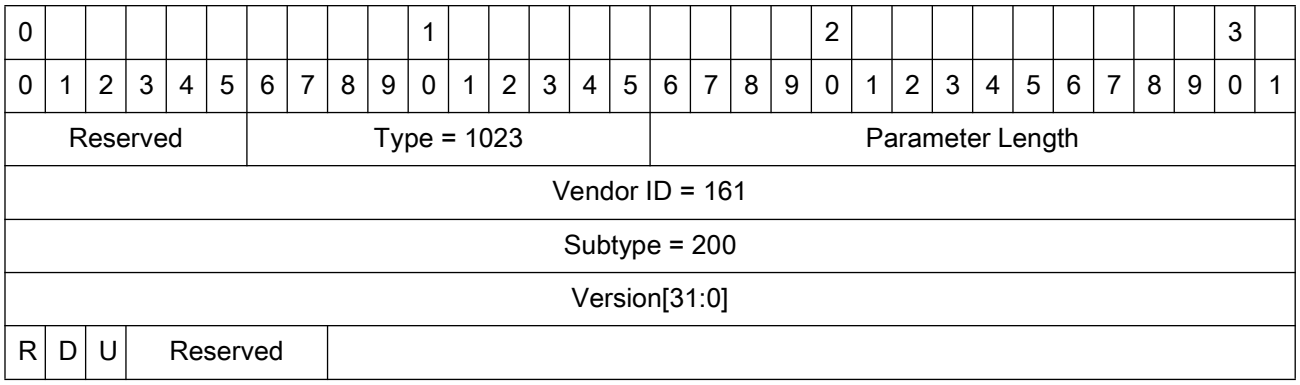
MotoLocationCapabilities

0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 130																															
Version[31:0]																															
F	Reserved															Reserved															

Abbreviations:

- F - Can support find specific item

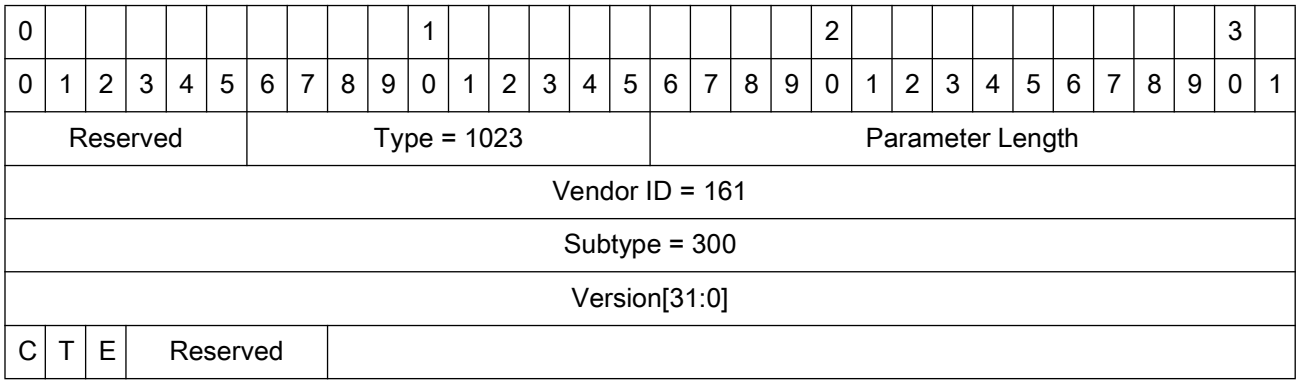
MotoFilterCapabilities



Abbreviations:

- R - Can filter tags based on RSSI
- D - Can filter tags based on time of the day
- U - Can filter tags based on the UTC timestamp of the tag read

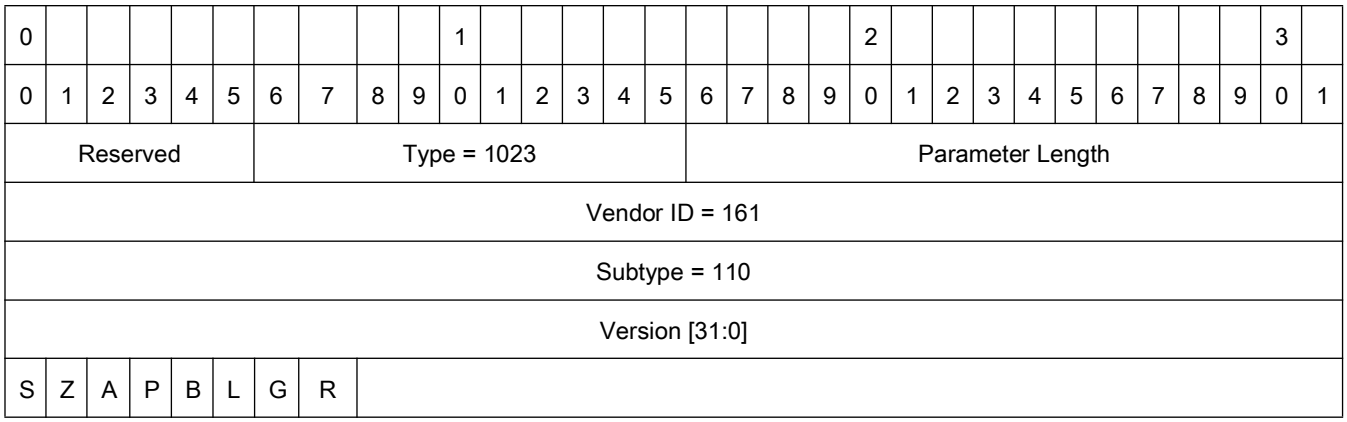
MotoPersistenceCapabilities



Abbreviations:

- C - Can save reader configuration
- T - Can save read tags
- E - Can save events associated with tags

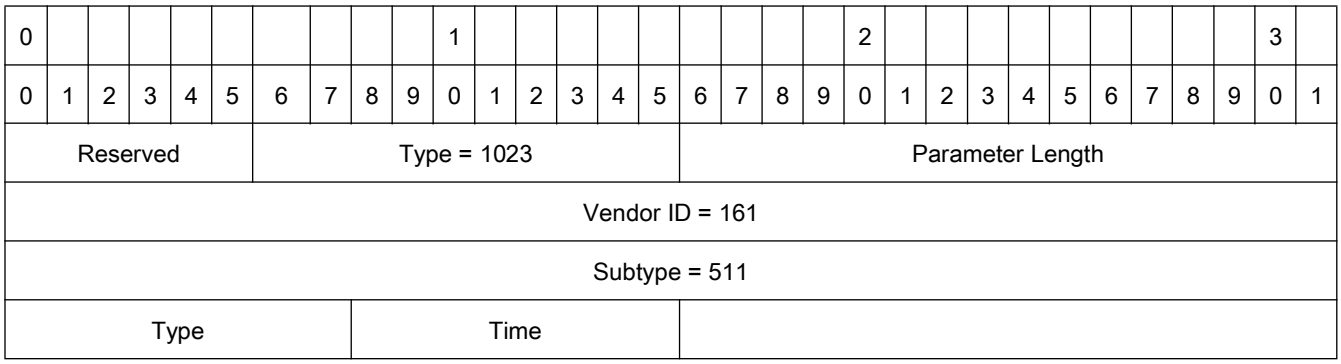
MotoAdvancedCapabilities



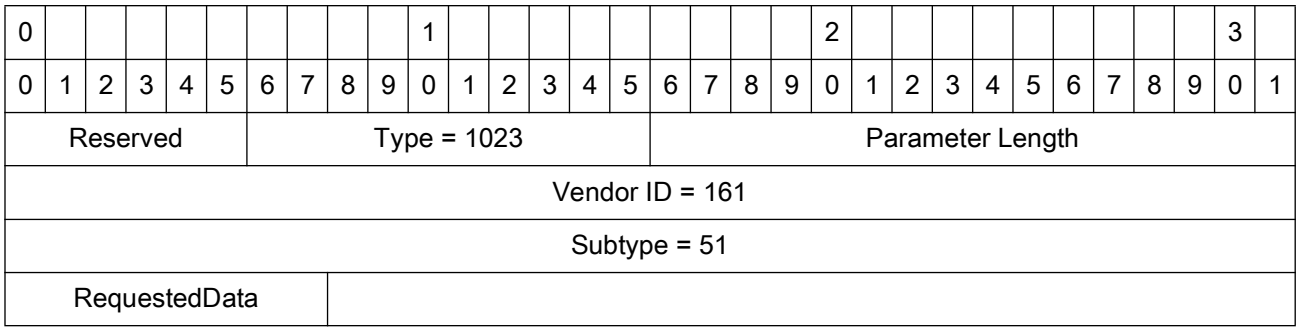
Abbreviations:

- S - Can report Phase as part of the tag data
- Z - Can support zone configuration and operation
- A - Can support RF Config extensions for Antenna
- P - Can Support Periodic Tag Report
- B - Can support Sled battery status
- L - Can support Logical Antenna
- G - Can report GPS co-ordinates (longitude, latitude and altitude) as part of the tag data
- R - Reserved

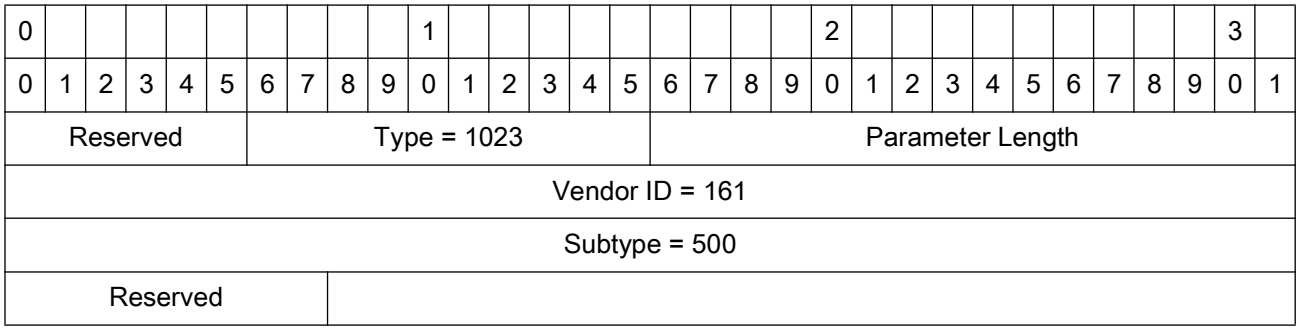
MotoRadioTransmitDelay



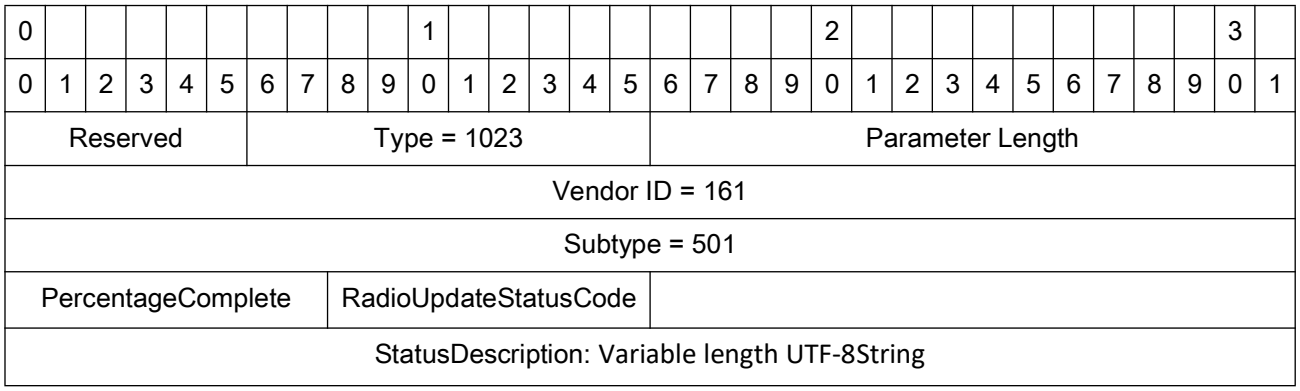
MotoGeneralGetParams



MotoRadioPowerState



MotoRadioUpdateStatusInfo



MotoRadioDutyCycle

0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 502																															
DutyCycleIndex																															

MotoRadioDutyCycleTable

0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 503																															
DutyCyclePercentage(1-N)																															

MotoVersion

0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 256																															
ModuleName: Variable length UTF-8String																															
ModuleVersion: Variable length UTF-8String																															

MotoVersion List

0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 504																															
MotoVersion (0-n)																															

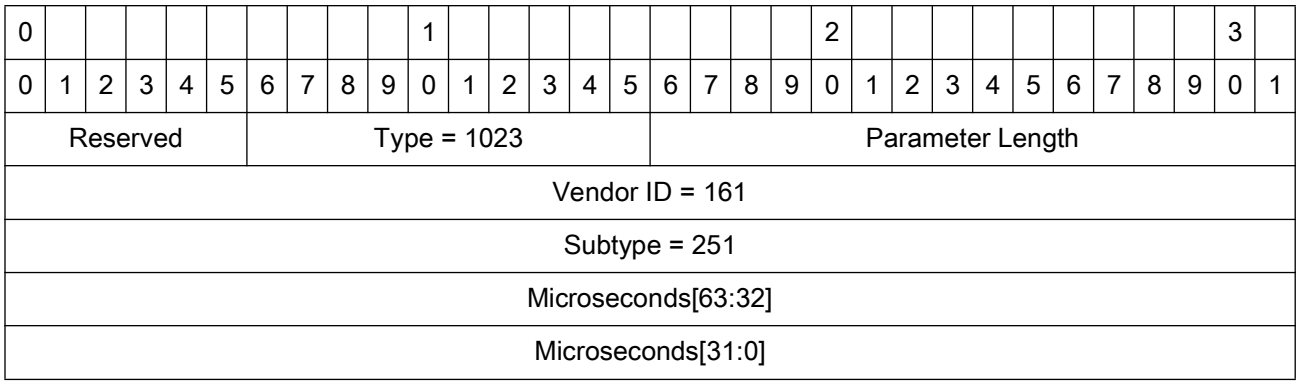
MotoSledBatteryStatus

0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 508																															
BatteryLevel[31:0]																															
Status																															

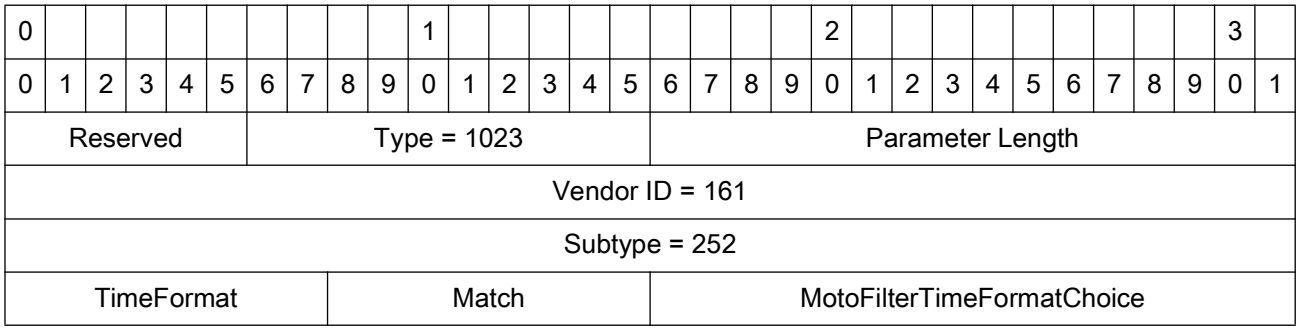
MotoFilterRule

0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 254																															
RuleType																															
MotoFilterRSSIRange (0-1)																															
MotoFilterTimeRange (0-1)																															
MotoFilterTagList (0-N)																															

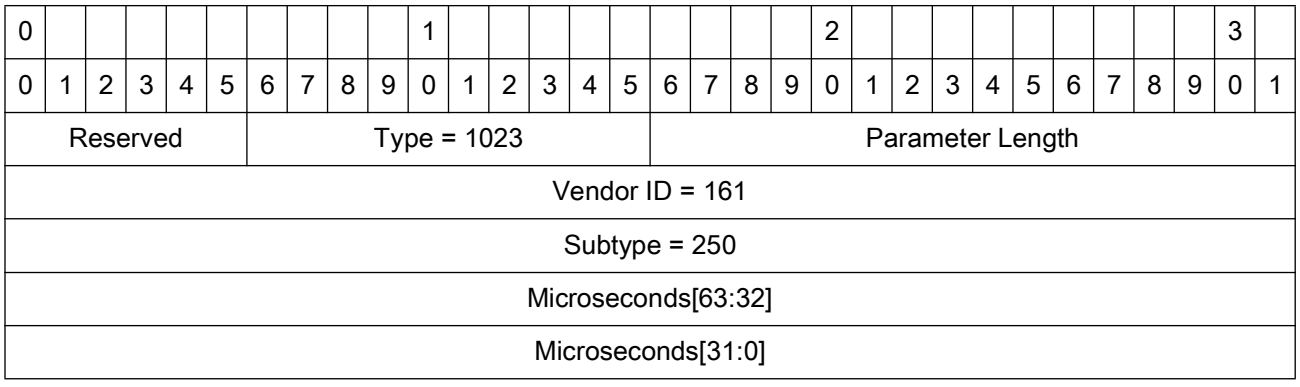
MotoFilterTimeOfDay



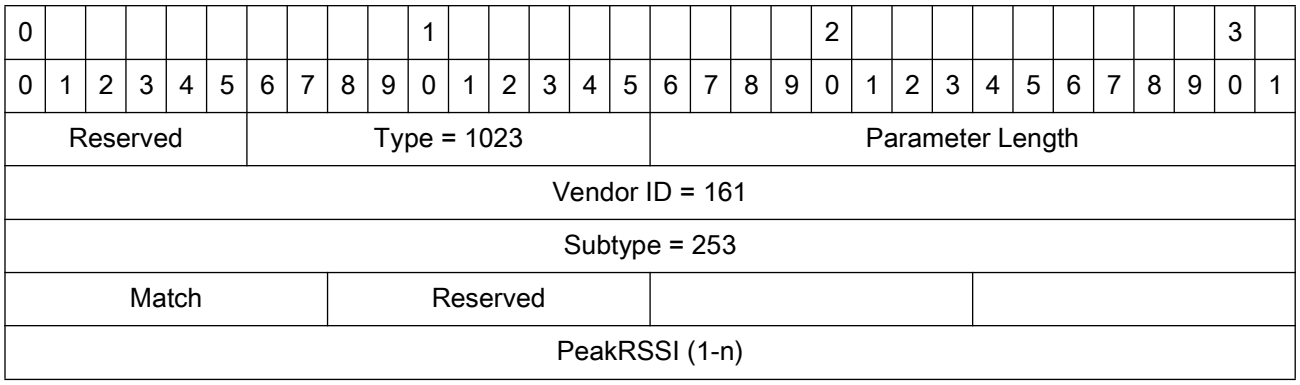
MotoFilterTimeRange



MotoUTCTimestamp

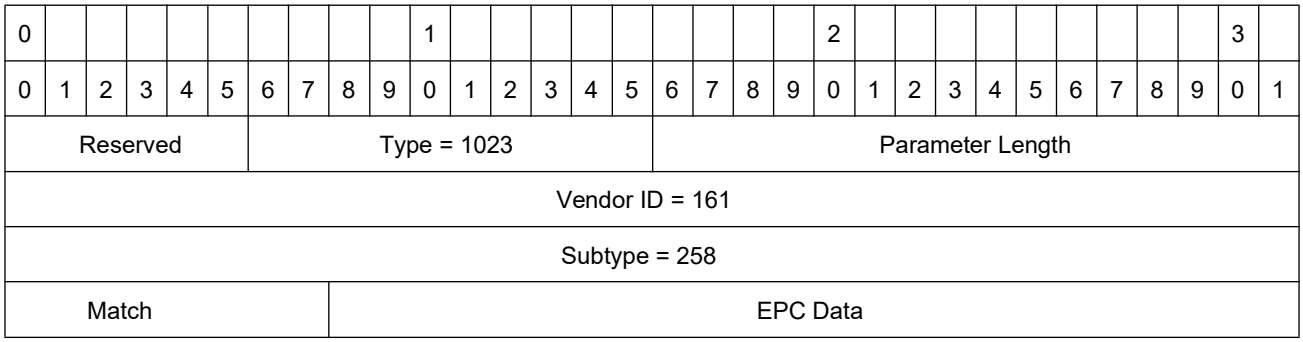


MotoFilterRSSIRange



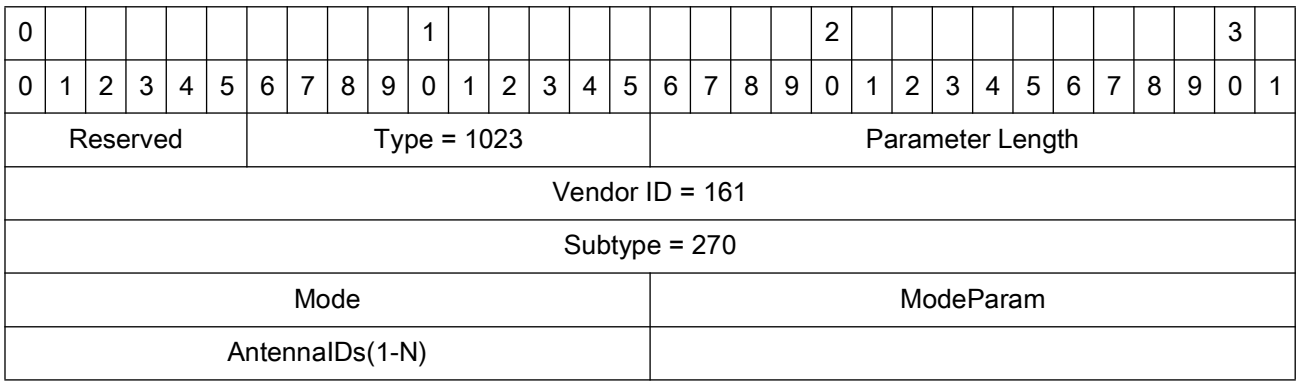
0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 253																															
Match								Reserved																							
PeakRSSI (1-n)																															

MotoFilterTagList



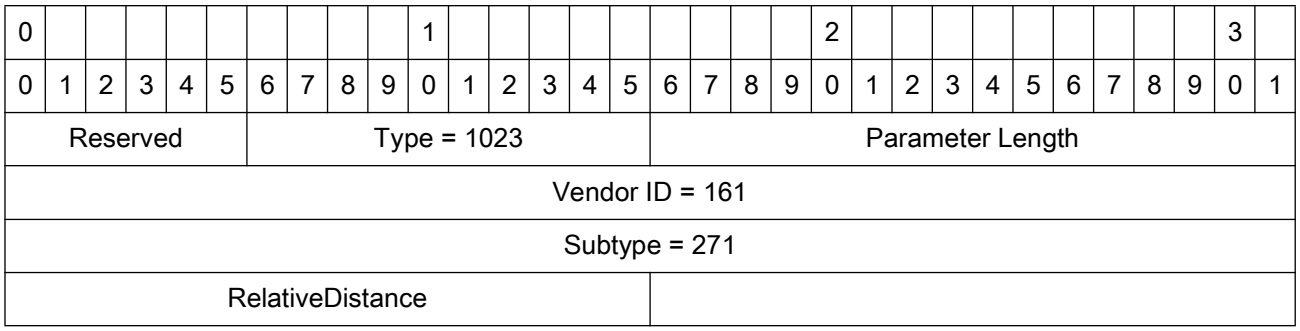
0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 258																															
Match										EPC Data																					

MotoFindItem

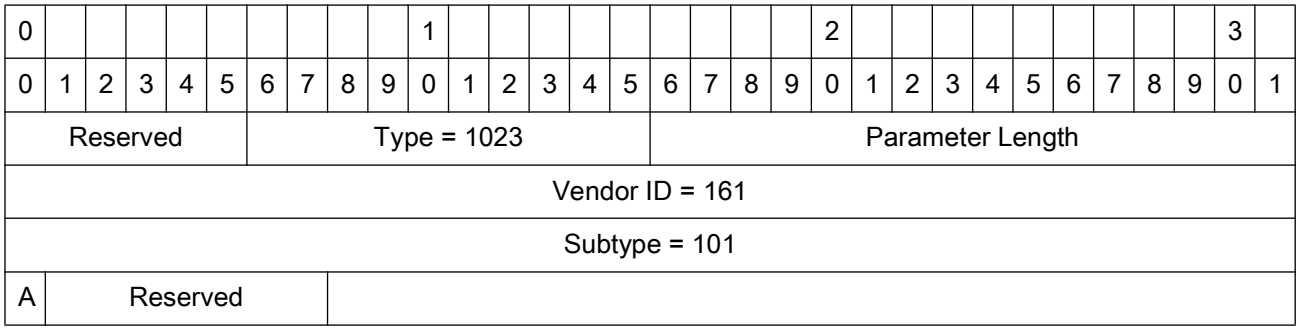


0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 270																															
Mode																ModeParam															
AntennaIDs(1-N)																															

MotoLocationResult



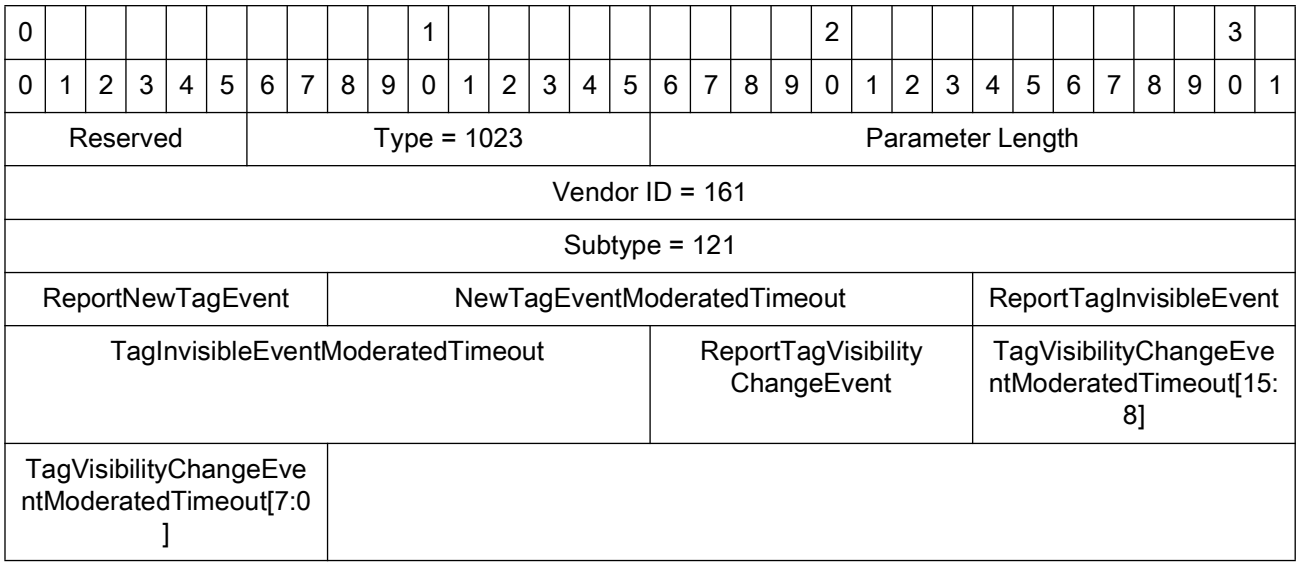
MotoAutonomousState



Abbreviations:

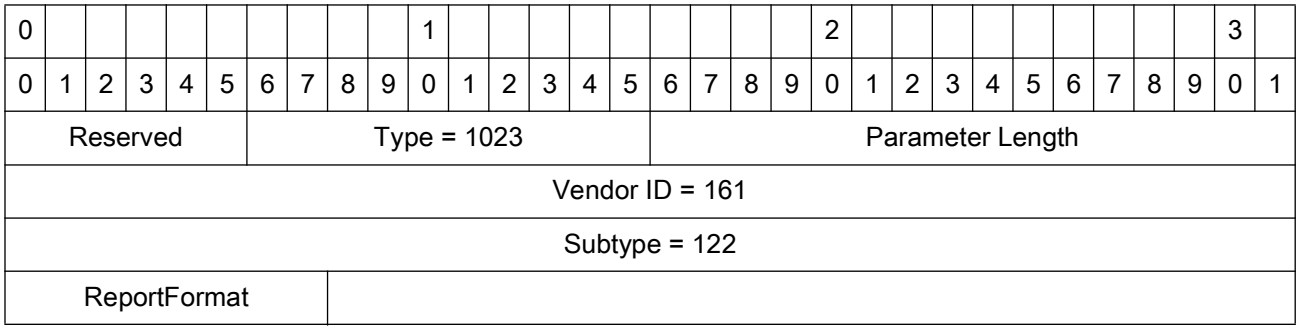
- A - Autonomous mode state

MotoTagEventSelector



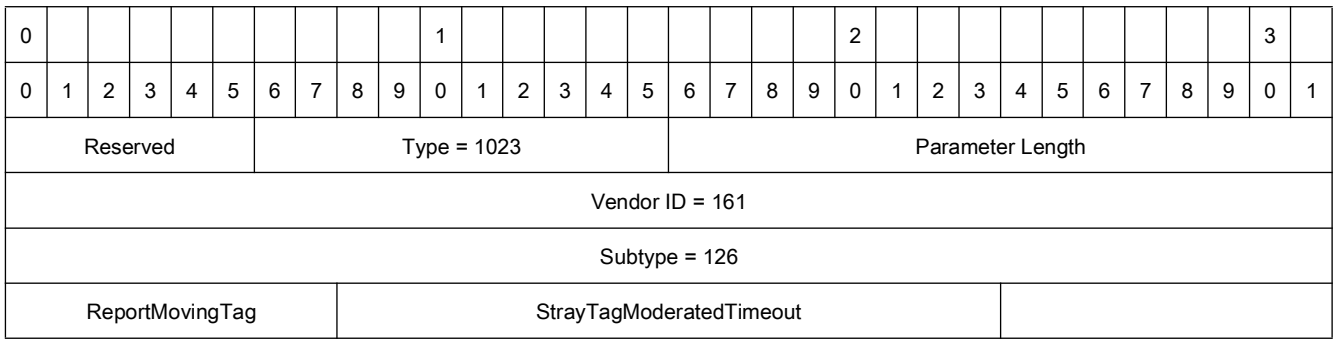
0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 121																															
ReportNewTagEvent										NewTagEventModeratedTimeout																				ReportTagInvisibleEvent	
TagInvisibleEventModeratedTimeout																				ReportTagVisibility ChangeEvent										TagVisibilityChangeEve ntModeratedTimeout[15: 8]	
TagVisibilityChangeEve ntModeratedTimeout[7:0]																															

MotoTagReportMode



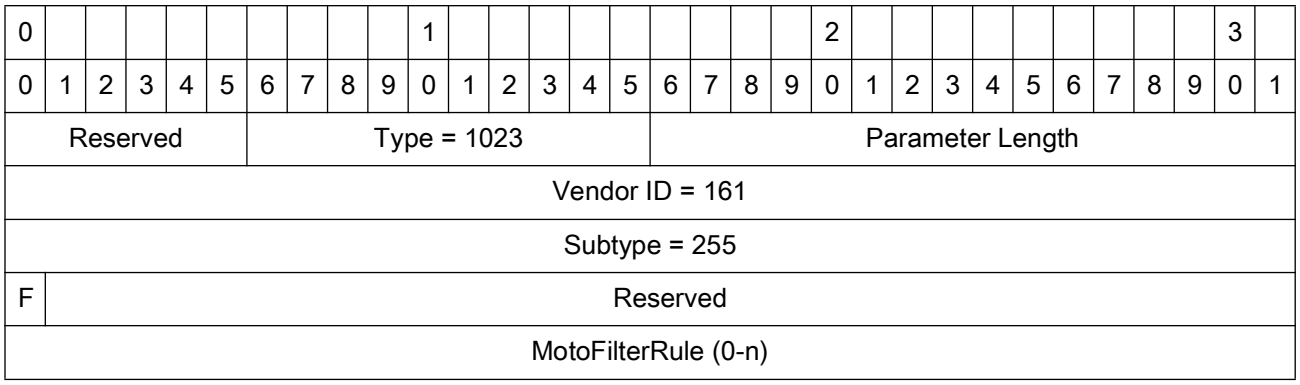
0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 122																															
ReportFormat																															

MovingStationaryTagReport



0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 126																															
ReportMovingTag										StrayTagModeratedTimeout																					

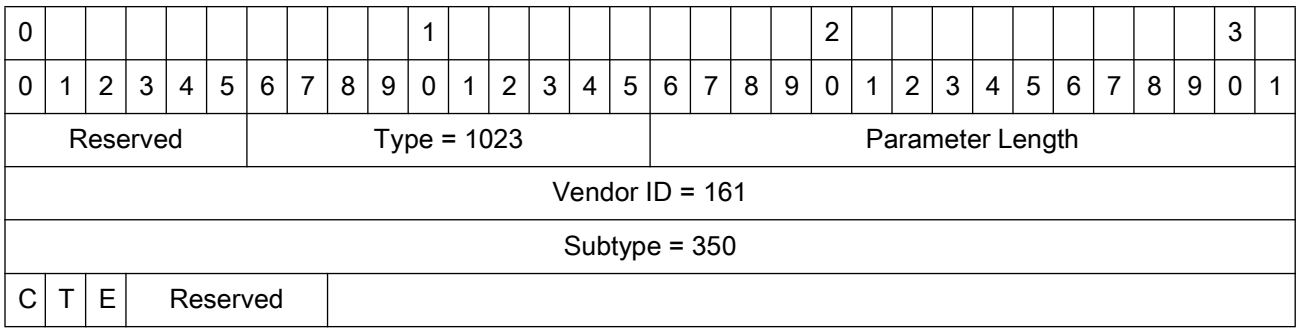
MotoFilterList



Abbreviations:

- F - Enable filtering

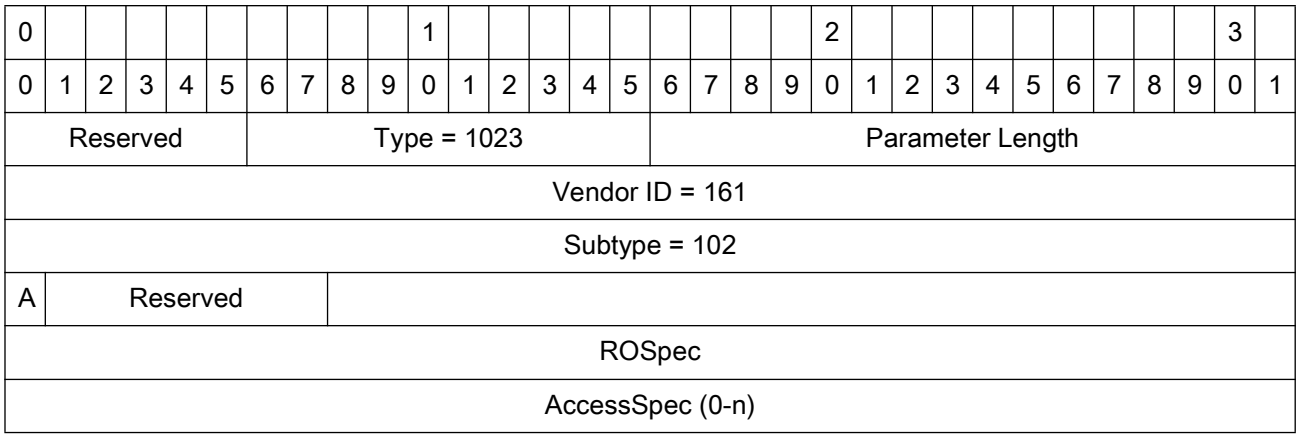
MotoPersistenceSaveParams



Abbreviations:

- C - Save configuration
- T - Save tag reports
- E - Save tag reports with event list

MotoDefaultSpec

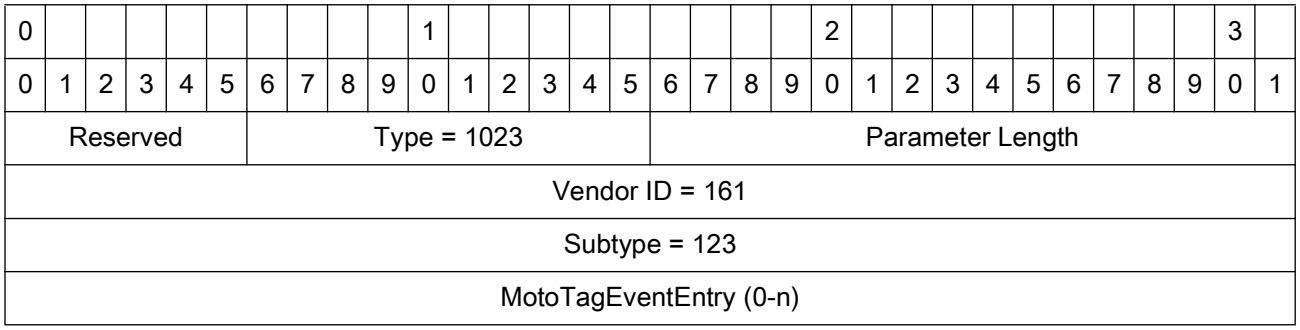


0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 102																															
A	Reserved																														
ROSpec																															
AccessSpec (0-n)																															

Abbreviations:

- A - Use default spec for autonomous mode

MotoTagEventList



0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 123																															
MotoTagEventEntry (0-n)																															

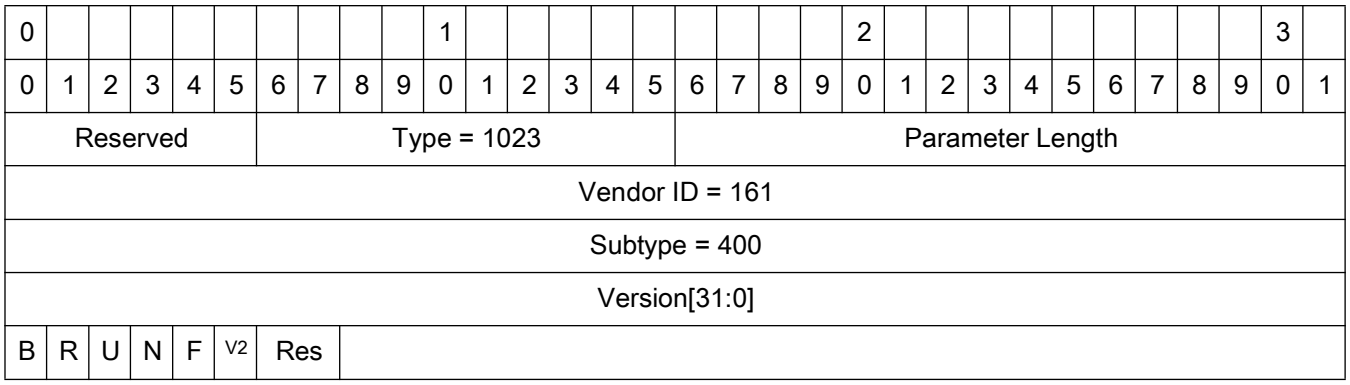
MotoTagEventEntry

0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 124																															
EventType										Microseconds[63:40]																					
Microseconds[39:8]																															
Microseconds[7:0]																															

MotoROReportTrigger

0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 125																															
MotoReportTrigger																															

MotoC1G2LLRPCapabilities

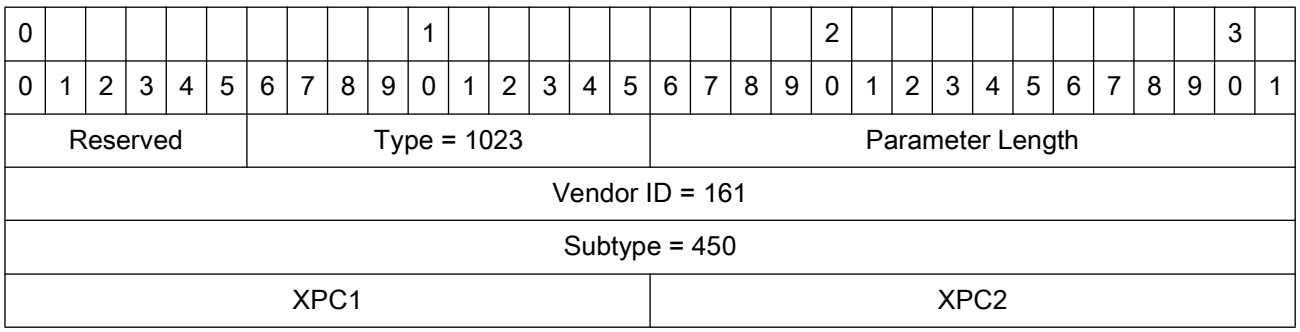


0										1										2									3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length														
Vendor ID = 161																														
Subtype = 400																														
Version[31:0]																														
B	R	U	N	F	V2	Res																								

Abbreviations:

- B - Can support **BlockPermaLock**
- R - Can support tag recommissioning
- U - Can support writing user memory indicator bit
- N - Can support NXP custom commands
- F - Can support Fujitsu custom commands
- V2 - Can Support G2V2 Commands

MotoC1G2ExtendedPC



0										1										2									3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length														
Vendor ID = 161																														
Subtype = 450																														
XPC1																XPC2														

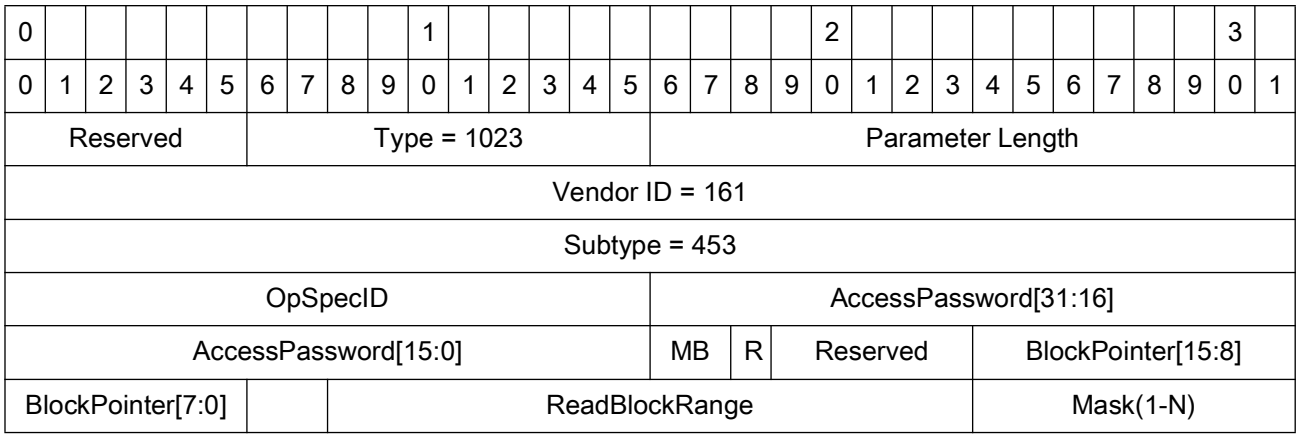
MotoC1G2Recommission

0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 451																															
OpSpecID																KillPassword[31:16]															
KillPassword[15:0]																Operation															

MotoC1G2RecommissionOpSpecResult

0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 452																															
Result										OpSpecID																					

MotoC1G2BlockPermalock

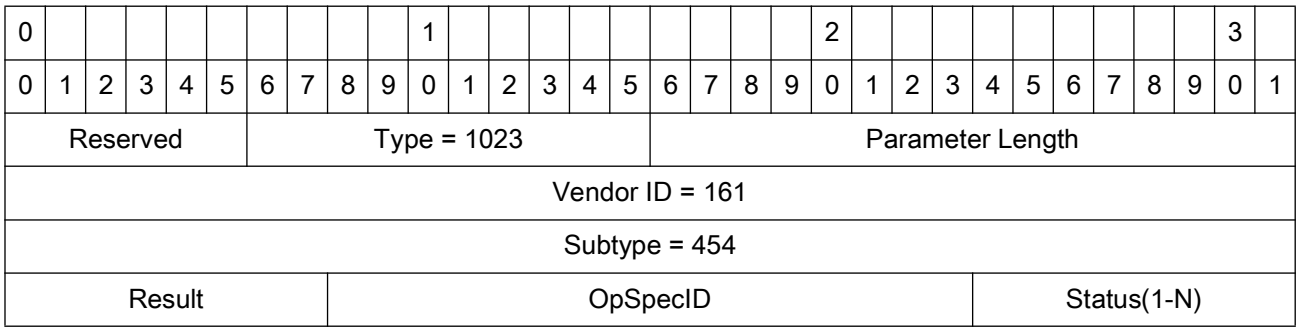


0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 453																															
OpSpecID																AccessPassword[31:16]															
AccessPassword[15:0]																MB		R		Reserved						BlockPointer[15:8]					
BlockPointer[7:0]										ReadBlockRange																Mask(1-N)					

Abbreviations:

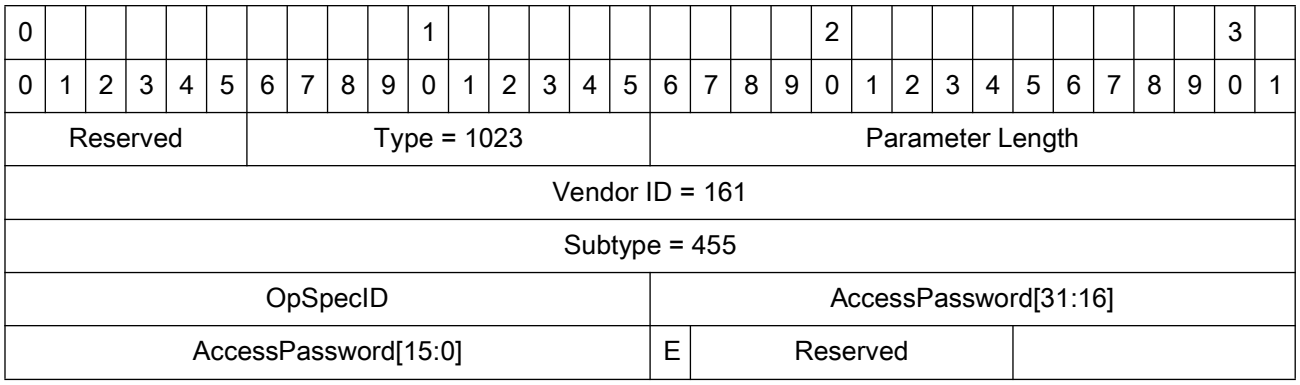
- R - Read lock
- MB - Memory bank

MotoC1G2BlockPermalockOpSpecResult



0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 454																															
Result										OpSpecID																Status(1-N)					

MotoNXPChangeEAS

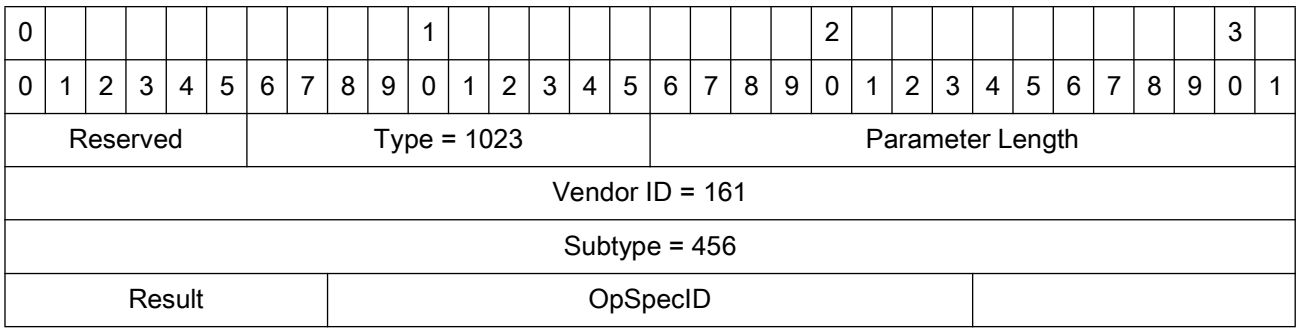


0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 455																															
OpSpecID																AccessPassword[31:16]															
AccessPassword[15:0]																E	Reserved														

Abbreviations:

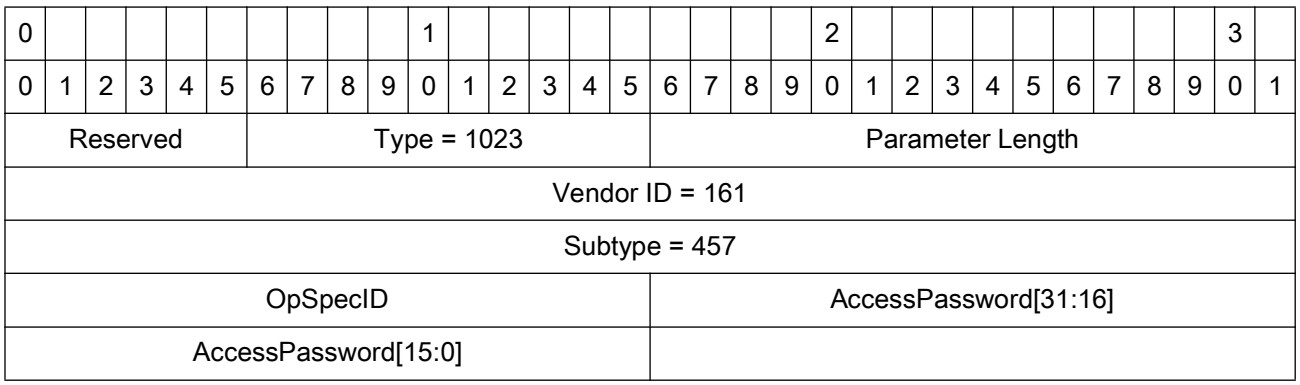
- E - EAS state to be set

MotoNXPChangeEASOpSpecResult



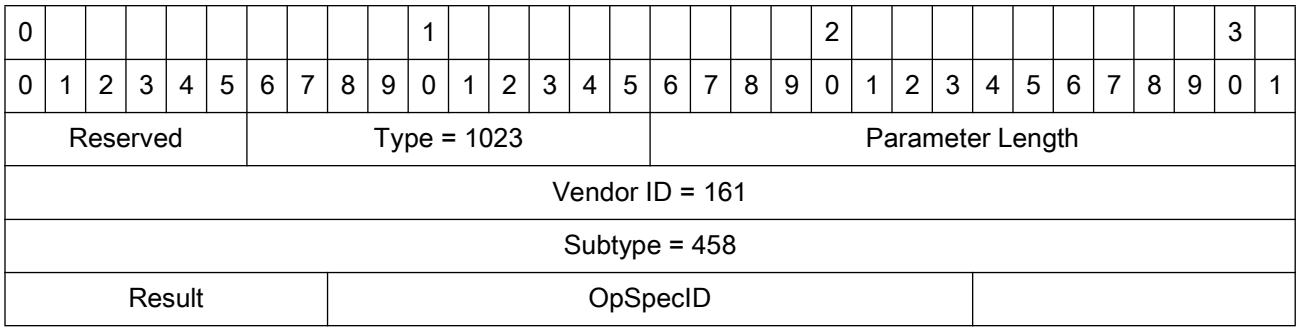
0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 456																															
Result						OpSpecID																									

MotoNXPSetQuiet



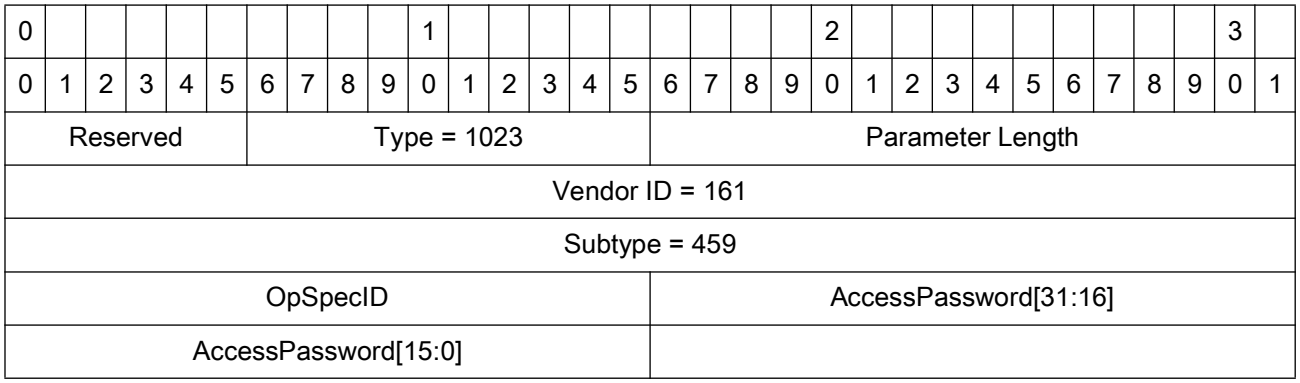
0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 457																															
OpSpecID																AccessPassword[31:16]															
AccessPassword[15:0]																															

MotoNXPSetQuietOpSpecResult



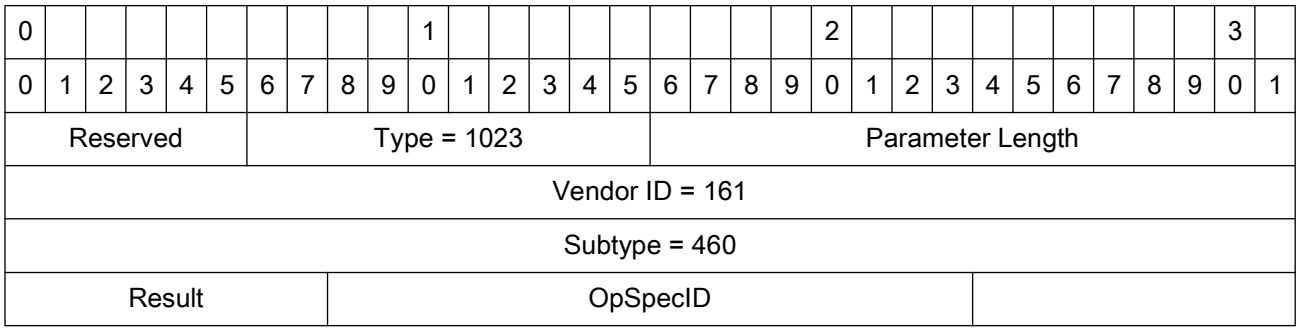
0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 458																															
Result										OpSpecID																					

MotoNXPResetQuiet



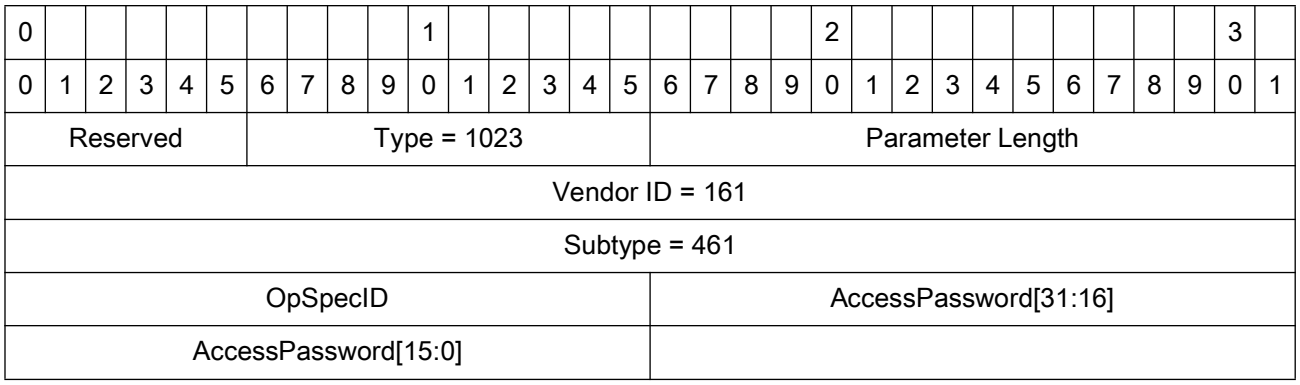
0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 459																															
OpSpecID																AccessPassword[31:16]															
AccessPassword[15:0]																															

MotoNXPResetQuietOpSpecResult



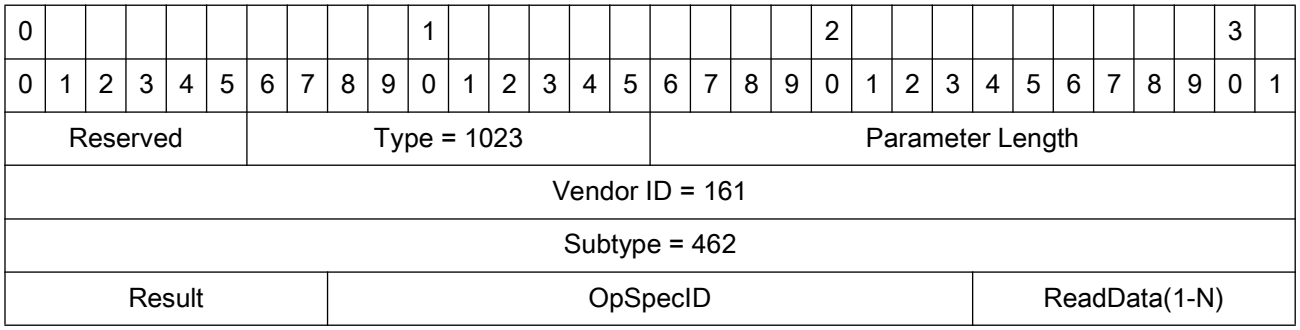
0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 460																															
Result										OpSpecID																					

MotoNXPCalibrate



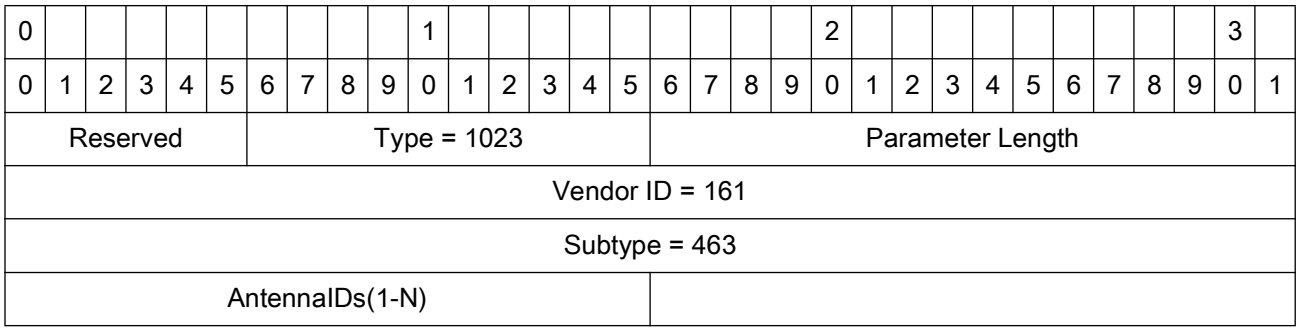
0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 461																															
OpSpecID																AccessPassword[31:16]															
AccessPassword[15:0]																															

MotoNXPCalibrateOpSpecResult



0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 462																															
Result										OpSpecID																					
ReadData(1-N)																															

MotoNXPEASAlarmSpec



0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 463																															
AntennaIDs(1-N)																															

MotoNXPEASAlarmNotification

0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 464																															
EASAlarmCode[63:32]																															
EASAlarmCode[31:0]																															
AntennaID (0-1)																															

MotoConnectionFailureReason

0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 465																															
ErrorCode Variable length UTF-8String																															

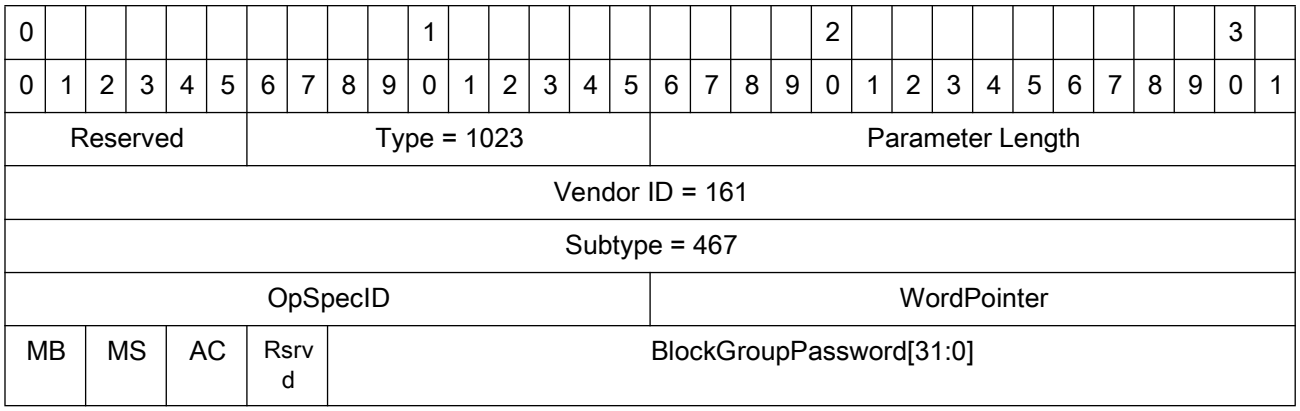
MotoCustomCommandOptions

0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 466																															
Q	Reserved																														

Abbreviations:

- Q - NXP Set Quiet and Reset Quiet Command enable option

MotoFujitsuChangeWordLock

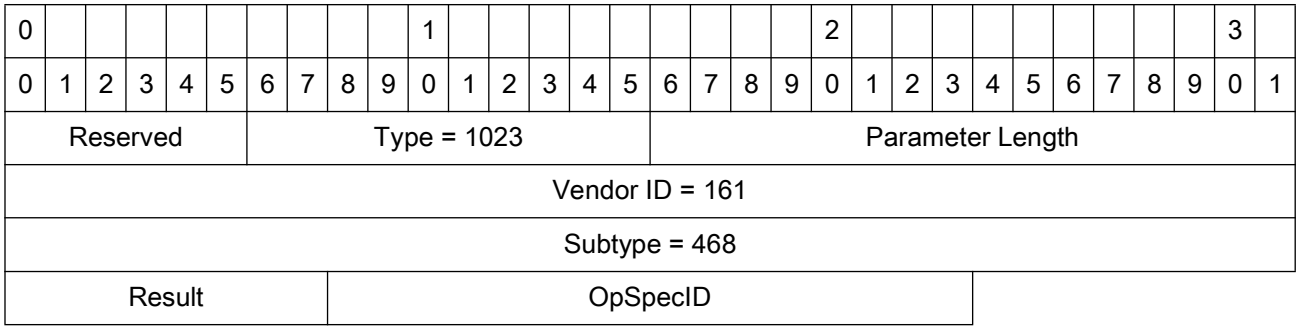


0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 467																															
OpSpecID																WordPointer															
MB		MS		AC		Rsrvd		BlockGroupPassword[31:0]																									

Abbreviations:

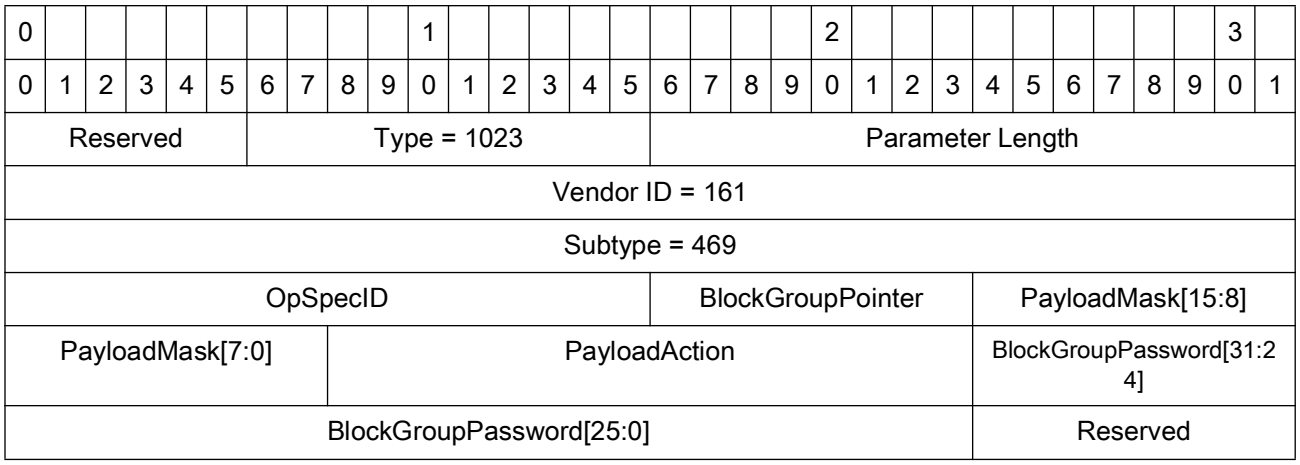
- MB - Memory bank
- MS - Change word lock pay load mask
- AC - Change word lock pay load action

MotoFujitsuChangeWordLockOpSpecResult



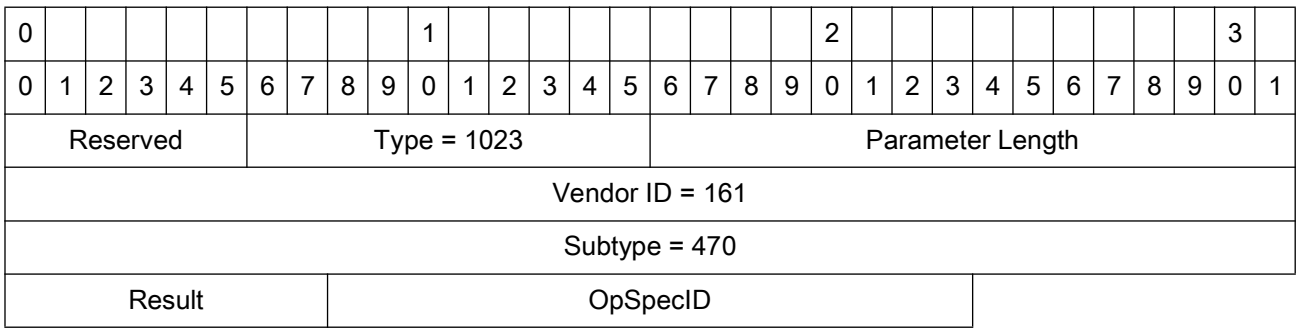
0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 468																															
Result																OpSpecID															

MotoFujitsuChangeBlockLock



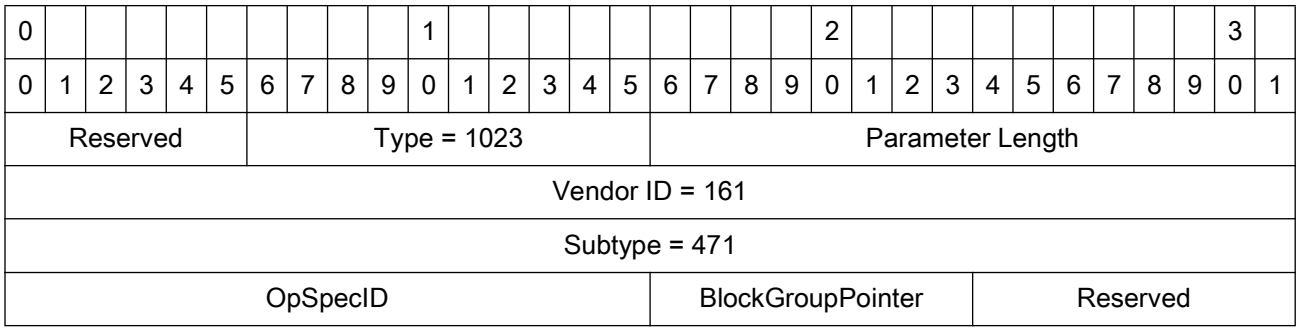
0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 469																															
OpSpecID																BlockGroupPointer															
PayloadMask[15:8]																															
PayloadMask[7:0]										PayloadAction																					
BlockGroupPassword[31:24]																															
BlockGroupPassword[25:0]																															
Reserved																															

MotoFujitsuChangeBlockLockOpSpecResult



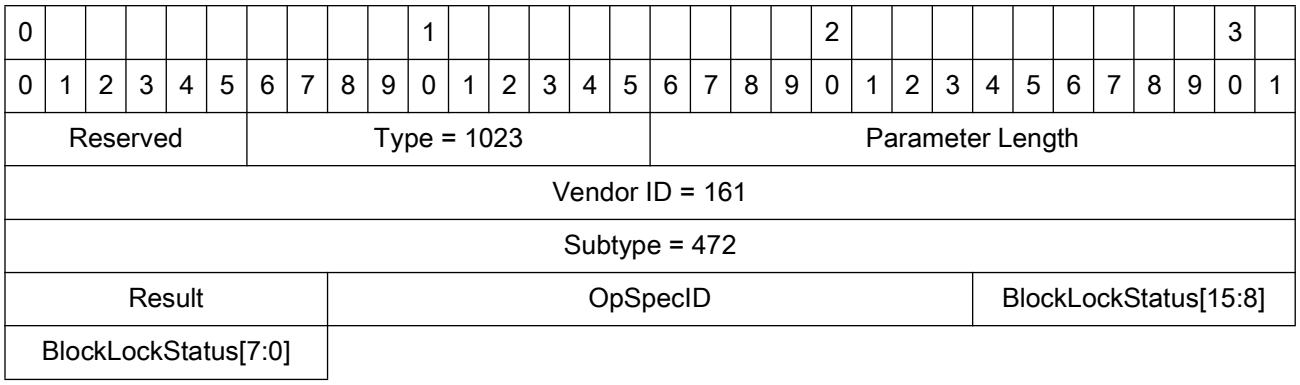
0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 470																															
Result						OpSpecID																									

MotoFujitsuReadBlockLock

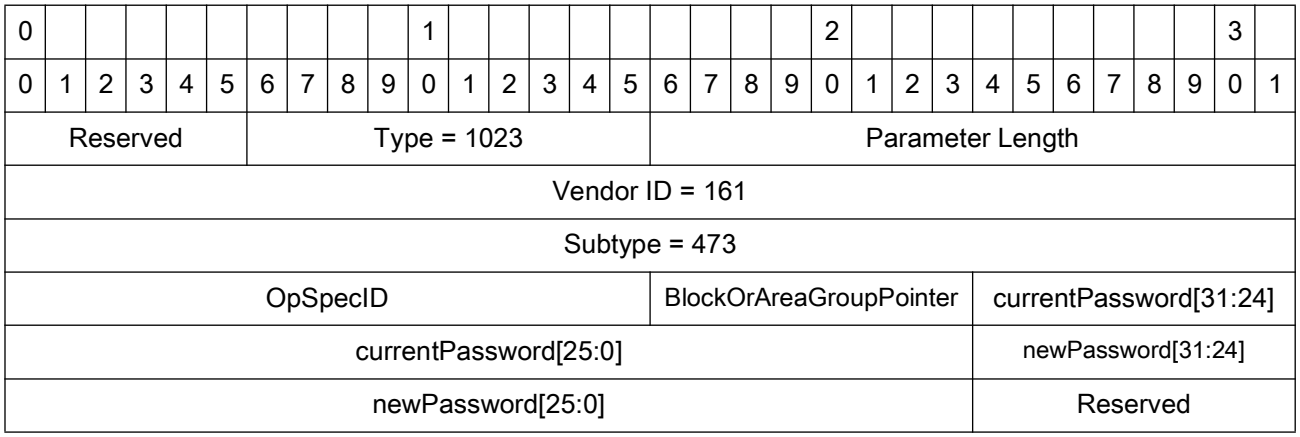


0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 471																															
OpSpecID																BlockGroupPointer															
Reserved																															

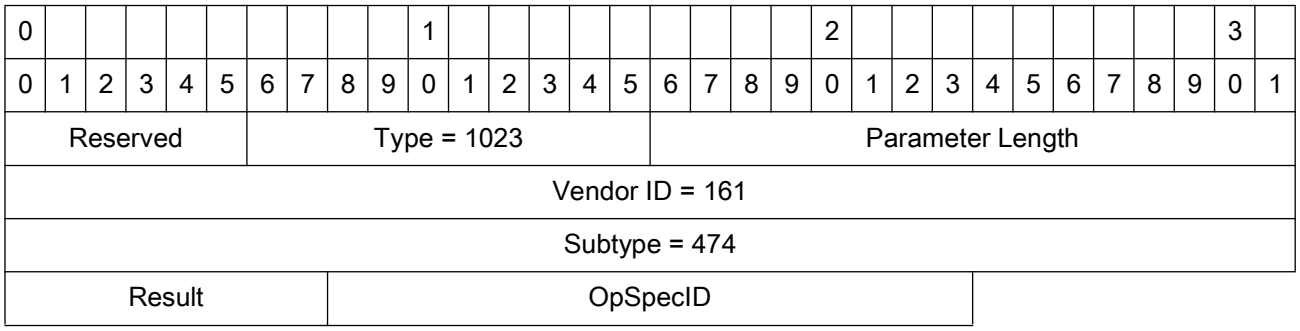
MotoFujitsuReadBlockLockOpSpecResult



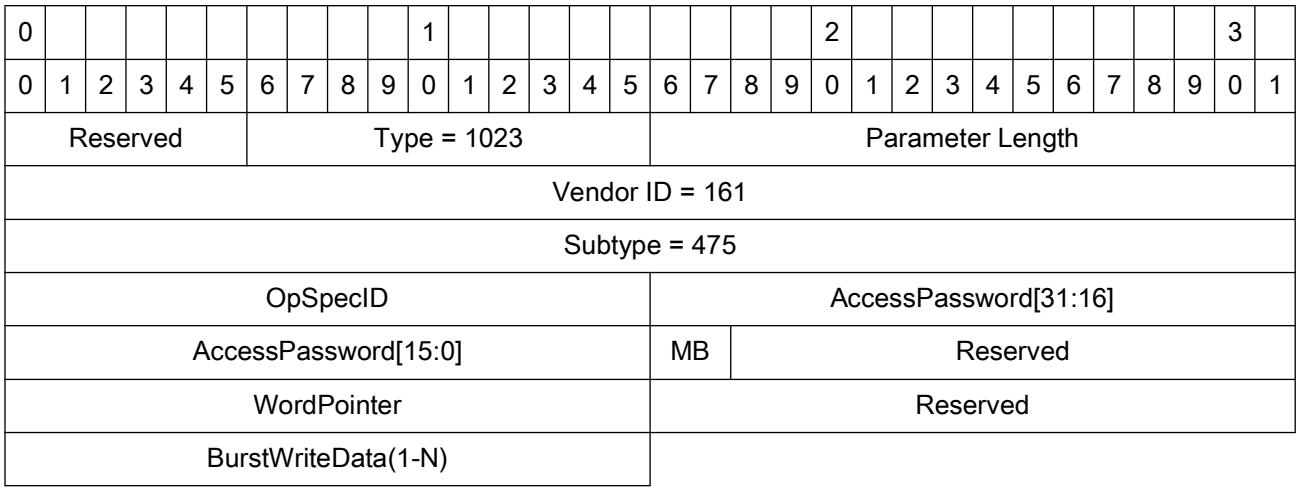
MotoFujitsuChangeBlockOrAreaGroupPassword



MotoFujitsuChangeBlockOrAreaGroupPasswordOpSpecResult



MotoFujitsuBurstWrite

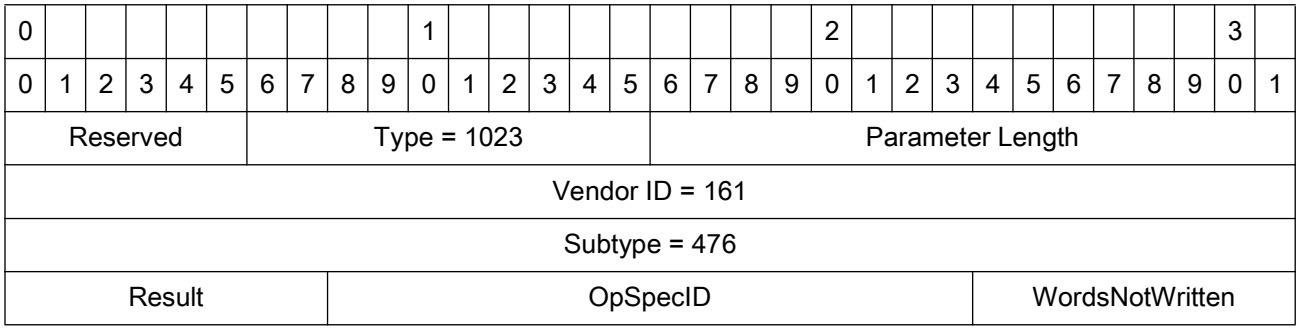


0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 475																															
OpSpecID																AccessPassword[31:16]															
AccessPassword[15:0]																MB		Reserved													
WordPointer																Reserved															
BurstWriteData(1-N)																															

Abbreviations:

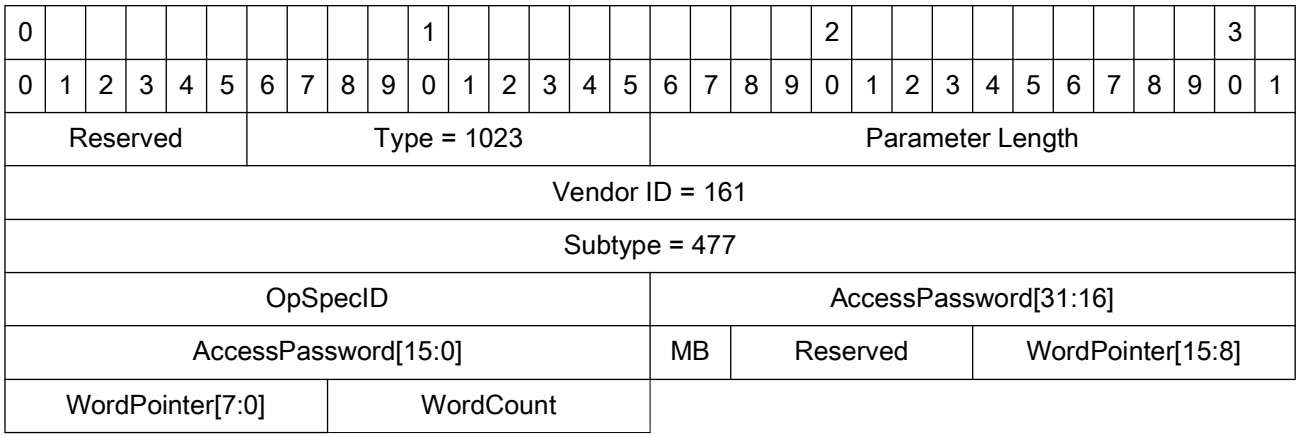
- MB - Memory bank

MotoFujitsuBurstWriteOpSpecResult



0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 476																															
Result						OpSpecID														WordsNotWritten											

MotoFujitsuBurstErase

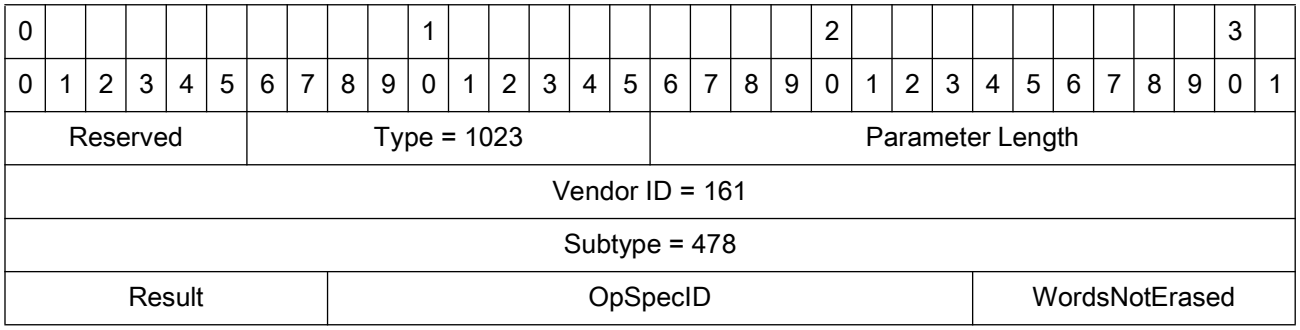


0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 477																															
OpSpecID																AccessPassword[31:16]															
AccessPassword[15:0]																MB		Reserved								WordPointer[15:8]					
WordPointer[7:0]										WordCount																					

Abbreviations:

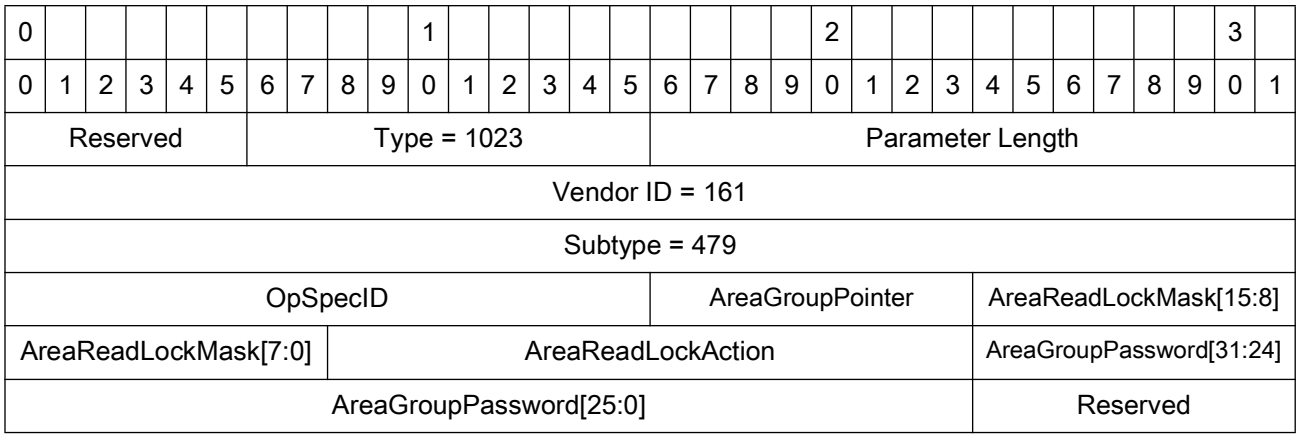
- MB - Memory bank

MotoFujitsuBurstEraseOpSpecResult

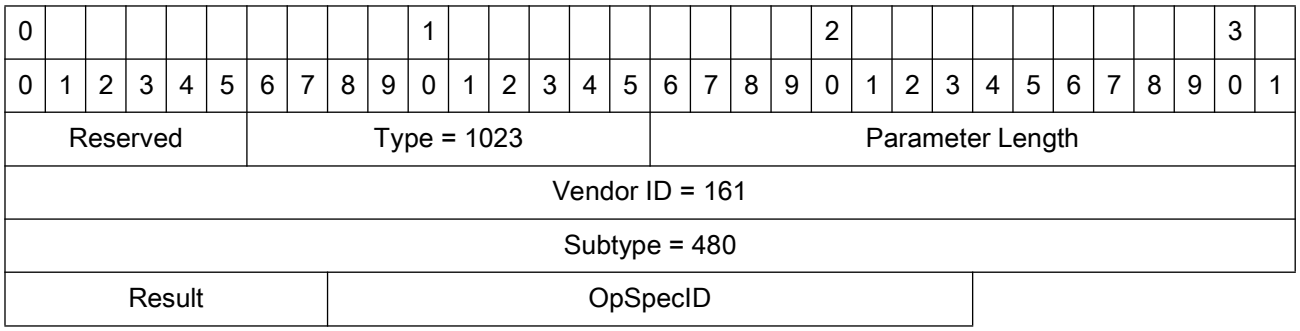


0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 478																															
Result										OpSpecID																				WordsNotErased	

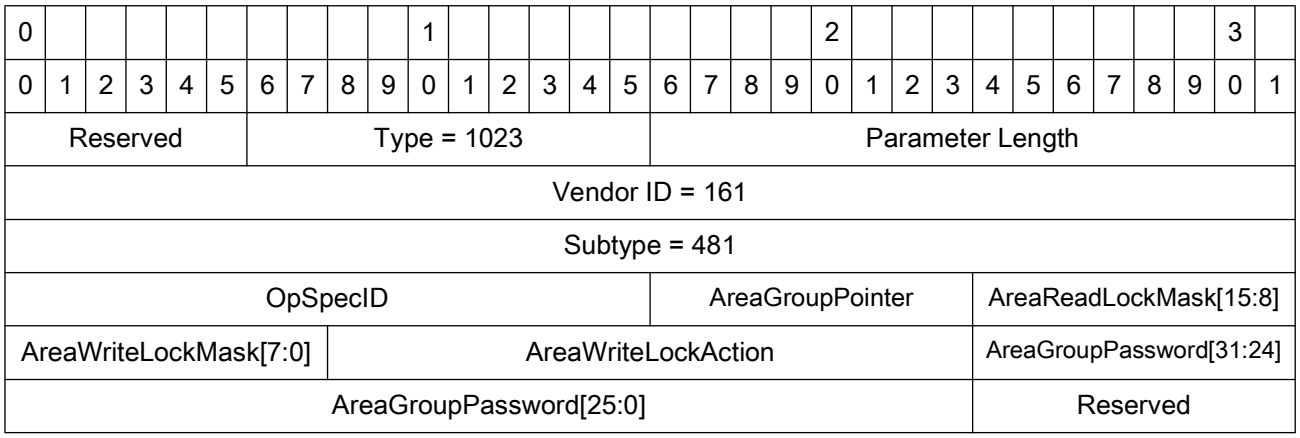
MotoFujitsuAreaReadLock



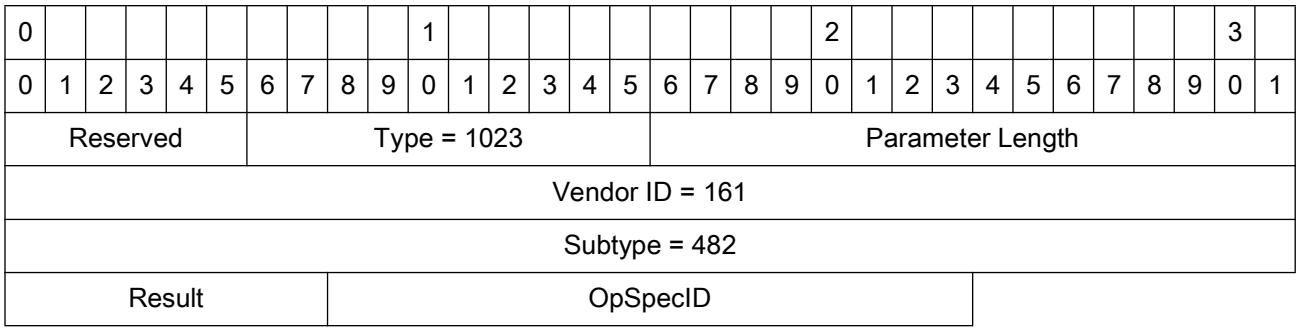
MotoFujitsuAreaReadLockOpSpecResult



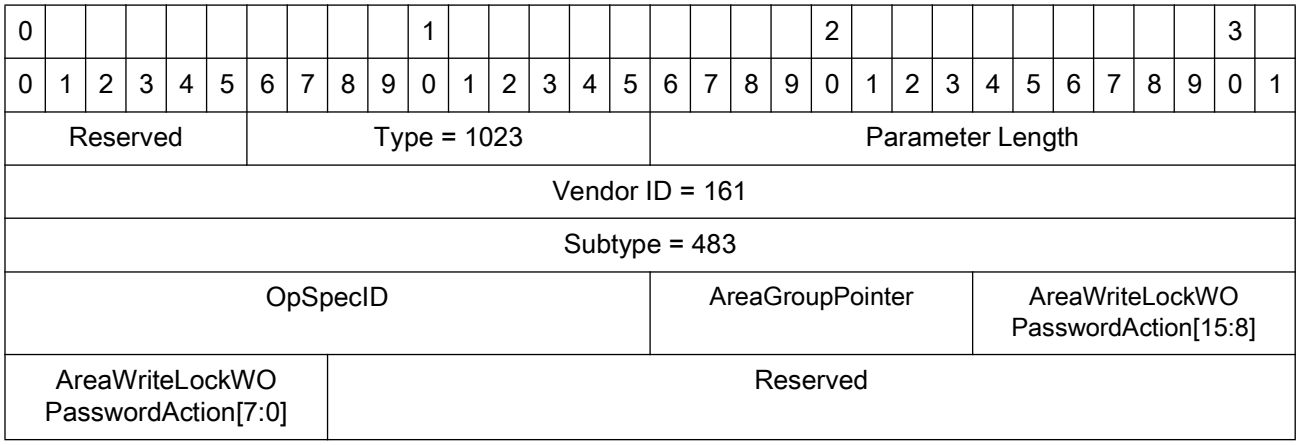
MotoFujitsuAreaWriteLock



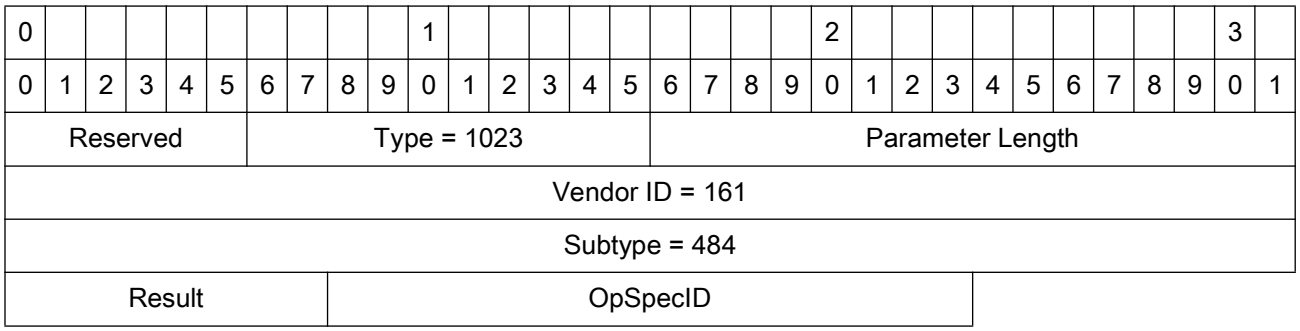
MotoFujitsuAreaWriteLockOpSpecResult



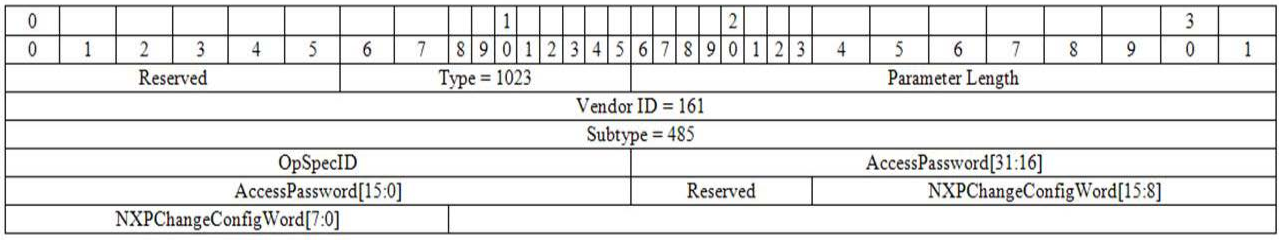
MotoFujitsuAreaWriteLockWOPassword



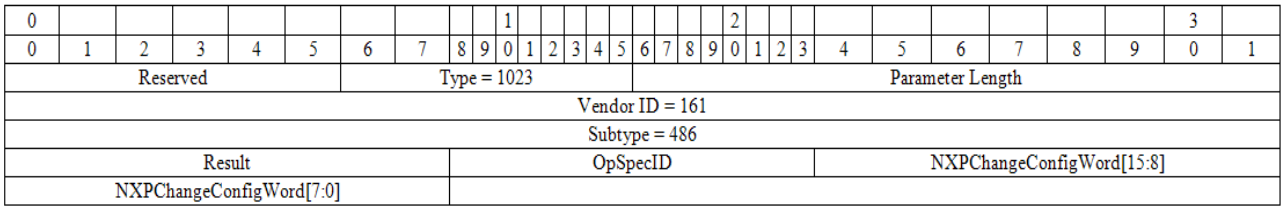
MotoFujitsuAreaWriteLockWOPasswordOpSpecResult



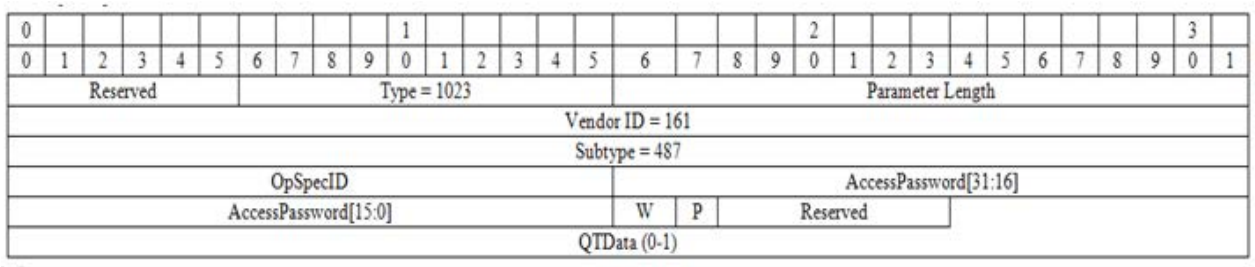
MotoNXPChangeConfig



MotoNXPChangeConfigOpSpecResult



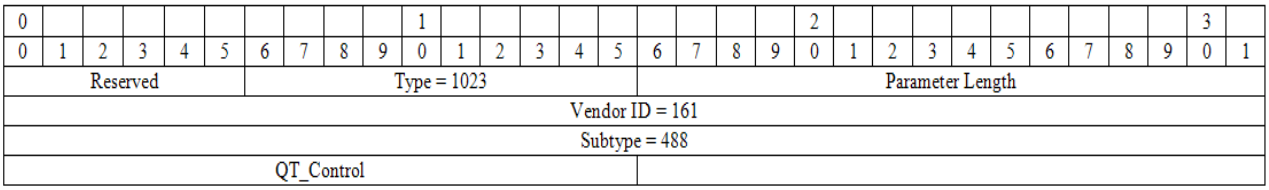
MotoImpinjQT



Abbreviations:

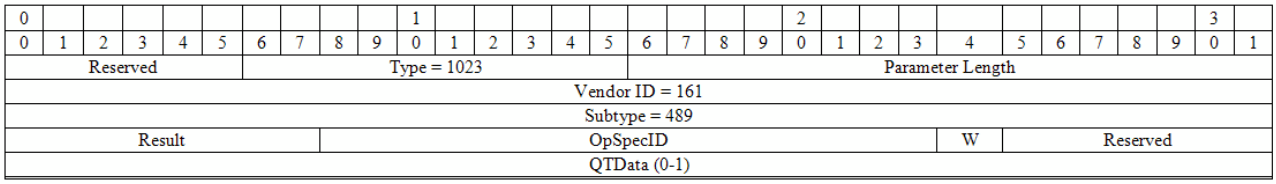
- W - QT_Write
- P - QR_Persist

QTData



0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 488																															
QT_Control																															

MotoImpinjQTOpSpecResult

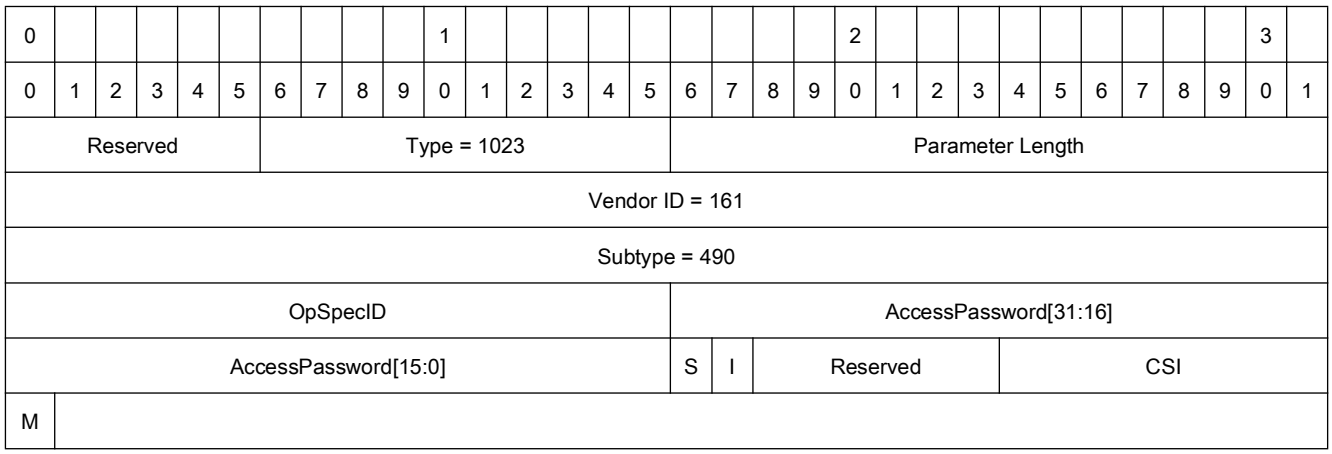


0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 489																															
Result						OpSpecID														W	Reserved									
QTData (0-1)																															

Abbreviation:

- W - QT_Write

MotoC1G2Authenticate

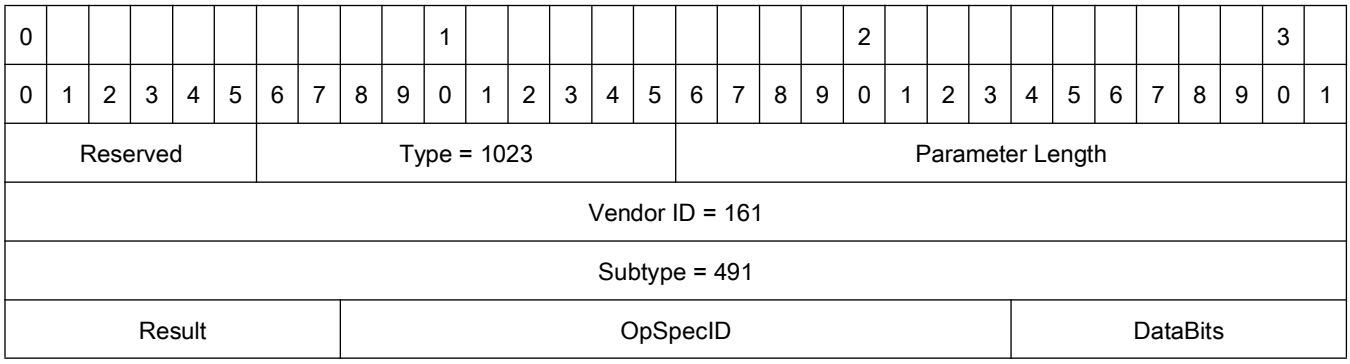


0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 490																															
OpSpecID																AccessPassword[31:16]															
AccessPassword[15:0]																S	I	Reserved										CSI			
M																															

Abbreviations:

- M - Message
- S - SenResp
- I - IncRespLen

MotoC1G2AuthenticateOpSpecResult



0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 491																															
Result										OpSpecID														DataBits							

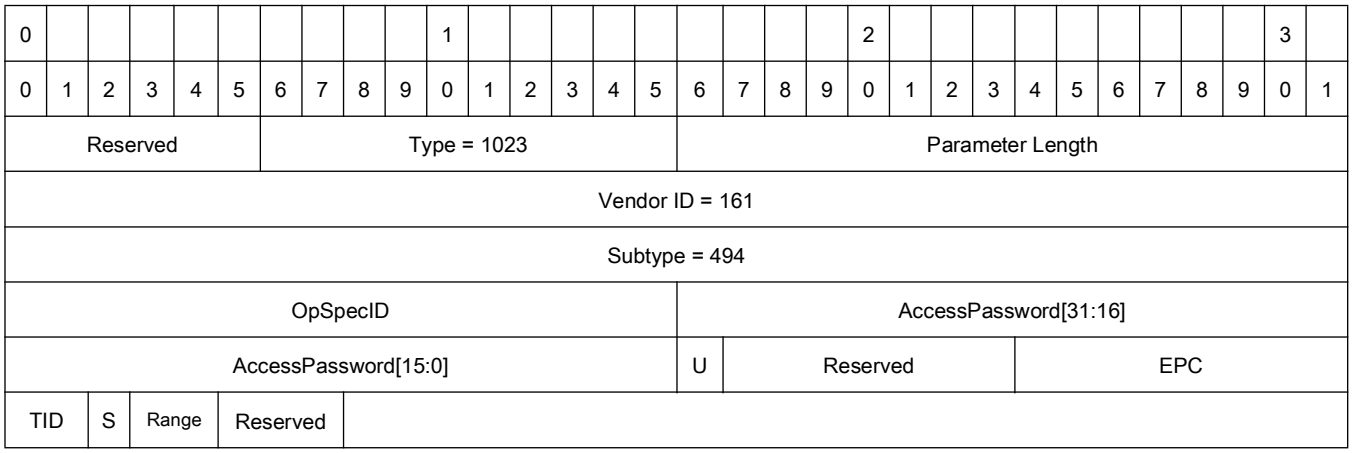
MotoC1G2ReadBuffer

0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 492																															
OpSpecID																AccessPassword[31:16]															
AccessPassword[15:0]																WordPtr															
BitCount																															

MotoC1G2ReadBufferOpSpecResult

0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 493																															
Result										OpSpecID														DataBits							

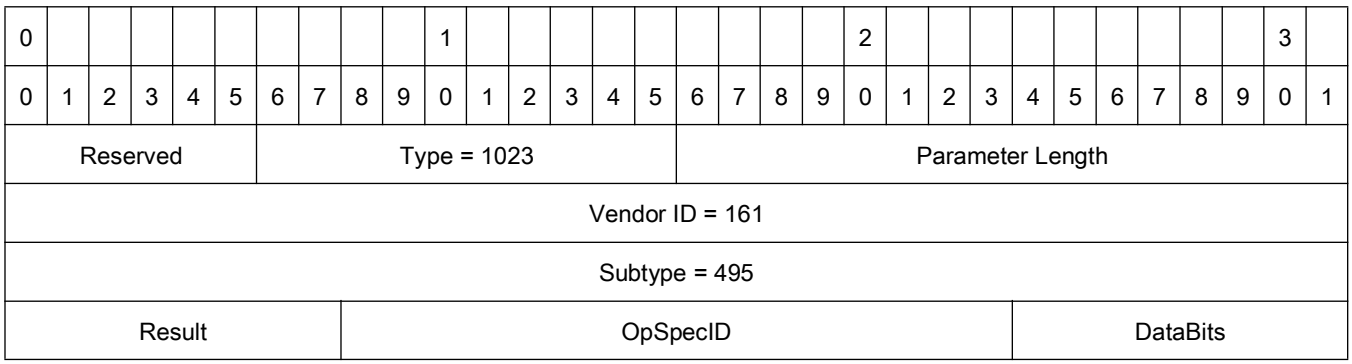
MotoC1G2Untraceable



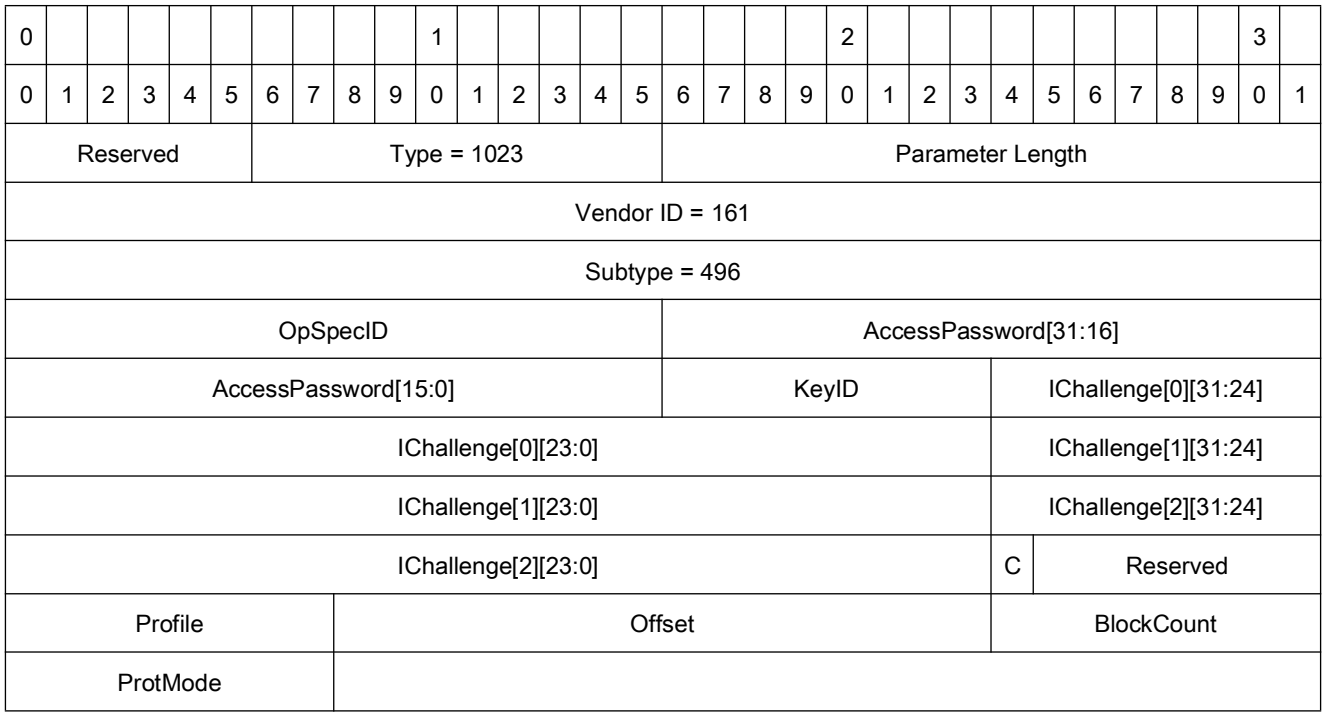
Abbreviation:

- S - User

MotoC1G2UntraceableOpSpecResult



MotoC1G2Crypto

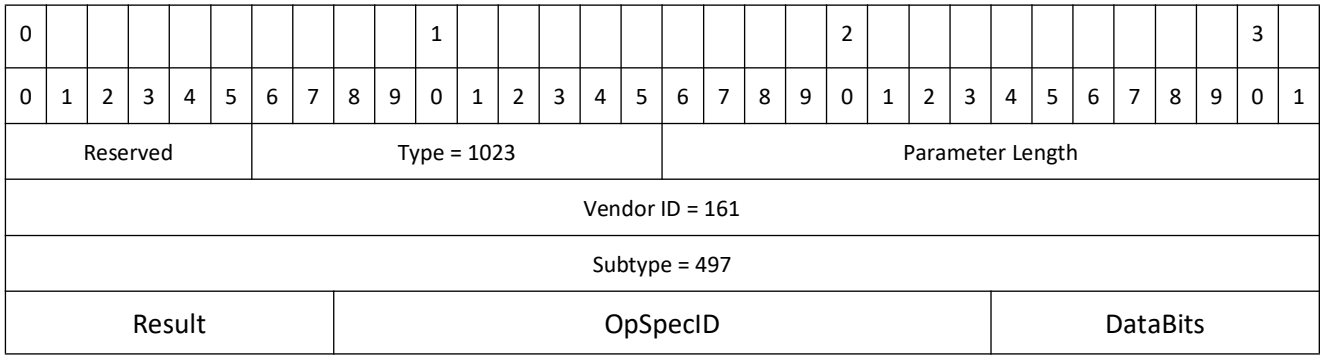


0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 496																															
OpSpecID																AccessPassword[31:16]															
AccessPassword[15:0]																KeyID				IChallenge[0][31:24]											
IChallenge[0][23:0]																				IChallenge[1][31:24]											
IChallenge[1][23:0]																				IChallenge[2][31:24]											
IChallenge[2][23:0]																				C	Reserved										
Profile						Offset														BlockCount											
ProtMode																															

Abbreviation:

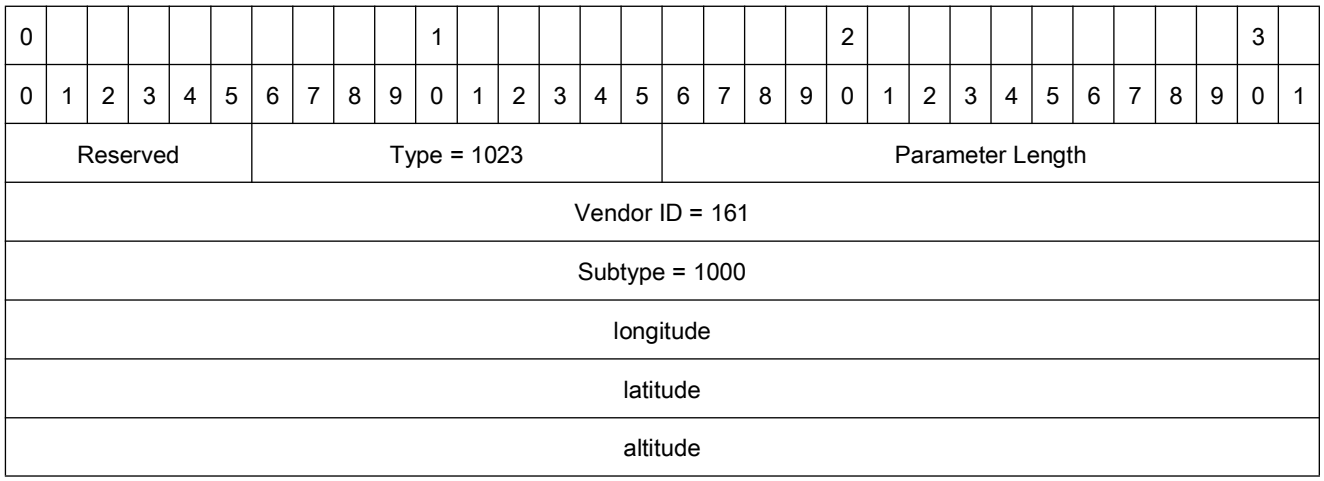
- C: CustomData

MotoC1G2CryptoOpSpecResult



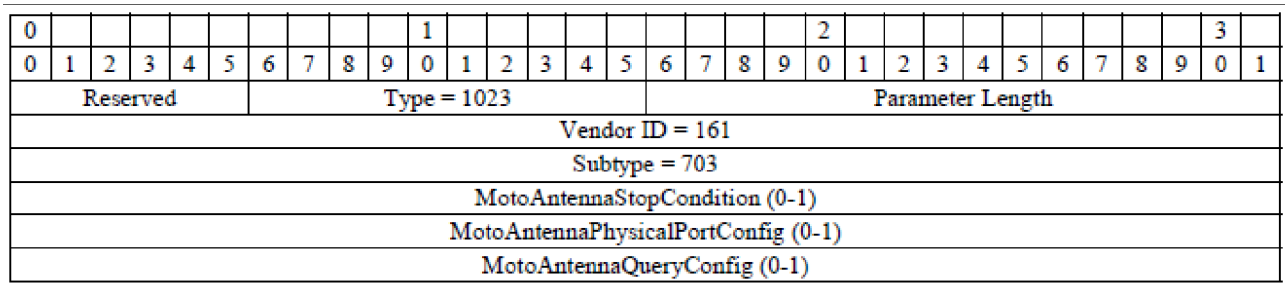
0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length															
Vendor ID = 161																															
Subtype = 497																															
Result						OpSpecID														DataBits											

MotoTagGPS



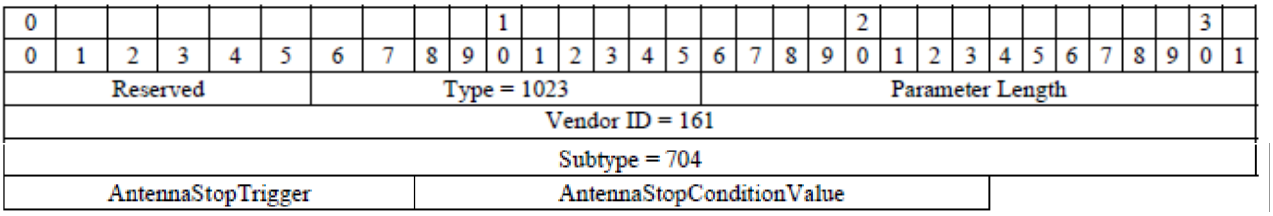
0										1									2									3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length														
Vendor ID = 161																															
Subtype = 1000																															
longitude																															
latitude																															
altitude																															

MotoAntennaConfig



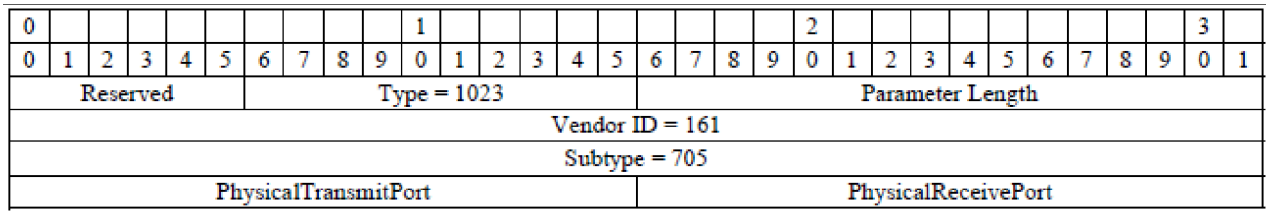
0										1									2									3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length														
Vendor ID = 161																															
Subtype = 703																															
MotoAntennaStopCondition (0-1)																															
MotoAntennaPhysicalPortConfig (0-1)																															
MotoAntennaQueryConfig (0-1)																															

MotoAntennaStopCondition

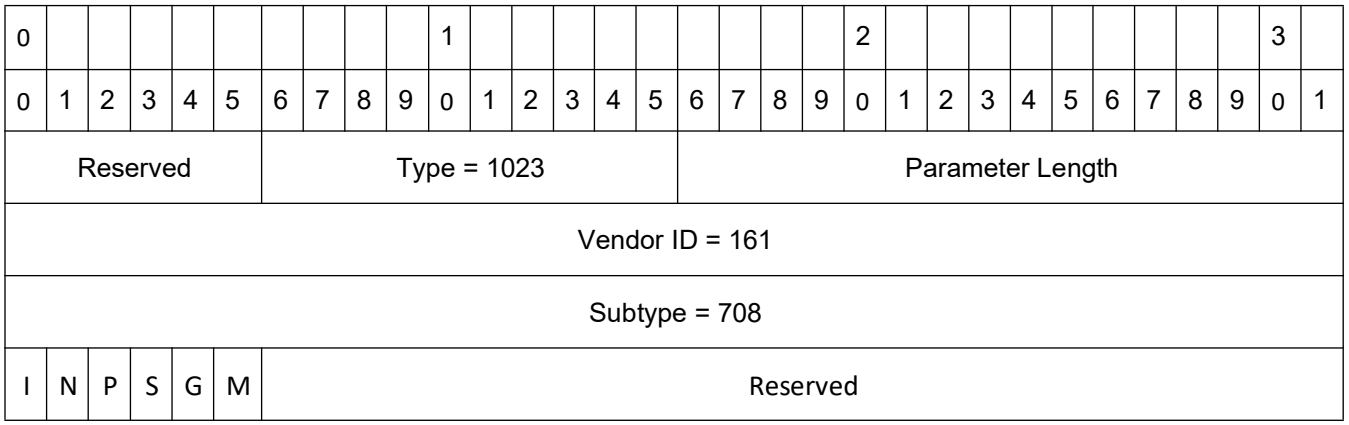


0										1									2									3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved						Type = 1023										Parameter Length														
Vendor ID = 161																															
Subtype = 704																															
AntennaStopTrigger																AntennaStopConditionValue														

MotoAntennaPhysicalPortConfig



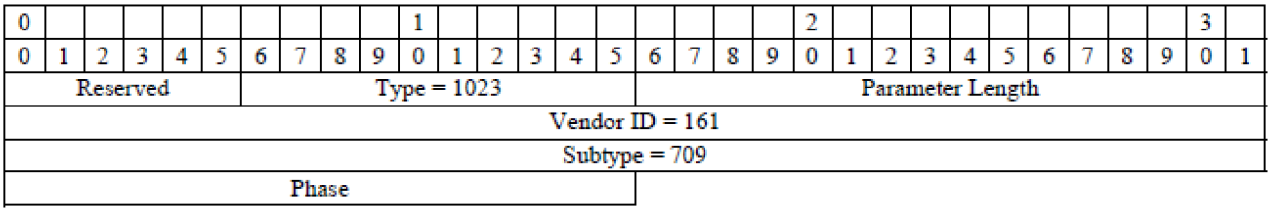
MotoTagReportContentSelector



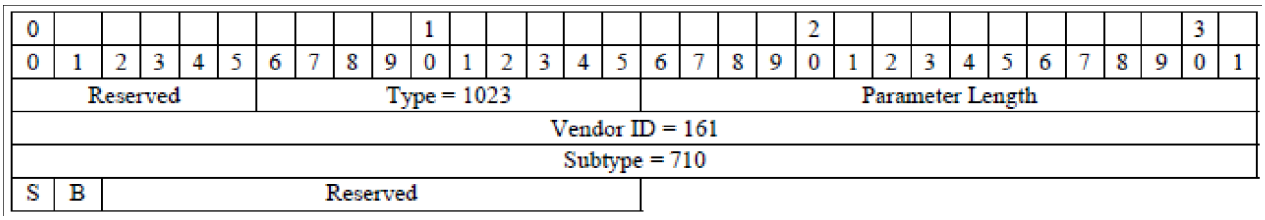
Abbreviations:

- I - Enable ZoneID in TagReport
- N - Enable ZoneName in TagReport
- P - Enable physical port in TagReport
- S - Enable phase in TagReport
- G - Enable GPS in TagReport
- M - Enable MLT algorithm report

MotoTagPhase



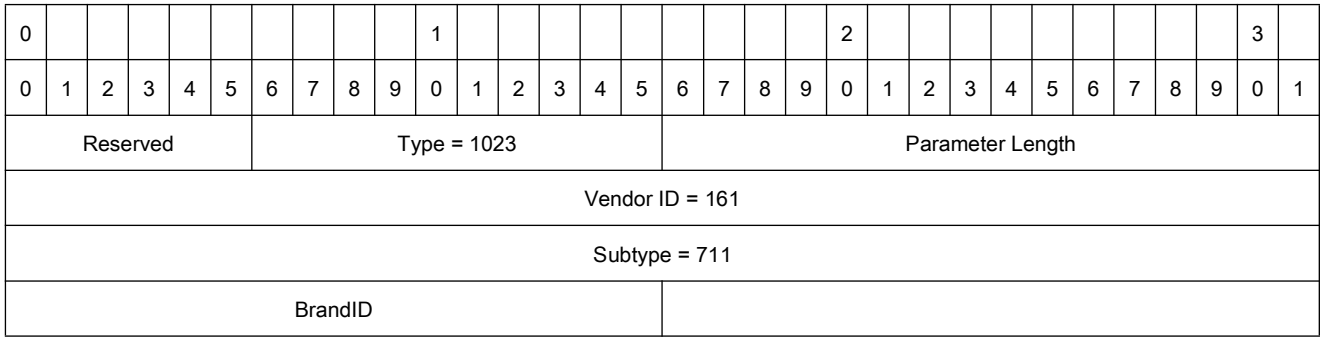
MotoAntennaQueryConfig



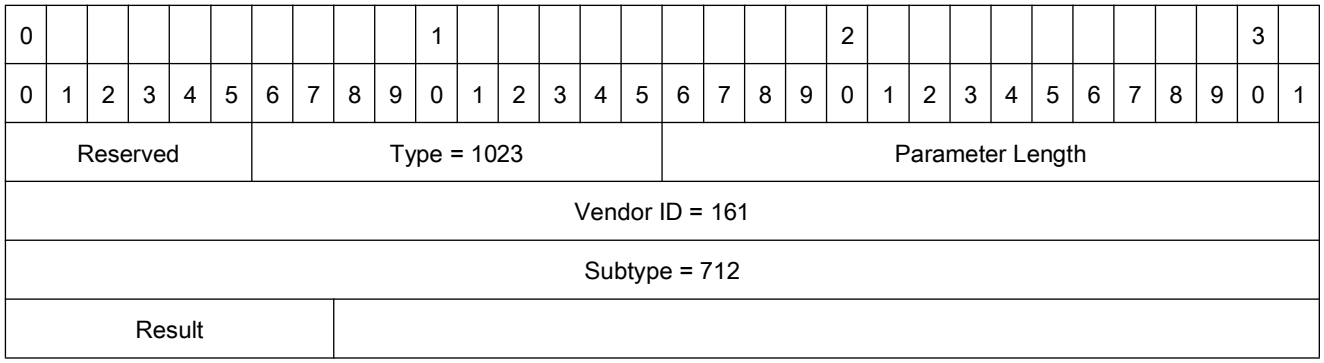
Abbreviations:

- S - Enable SL All
- B - Enable AB Flip

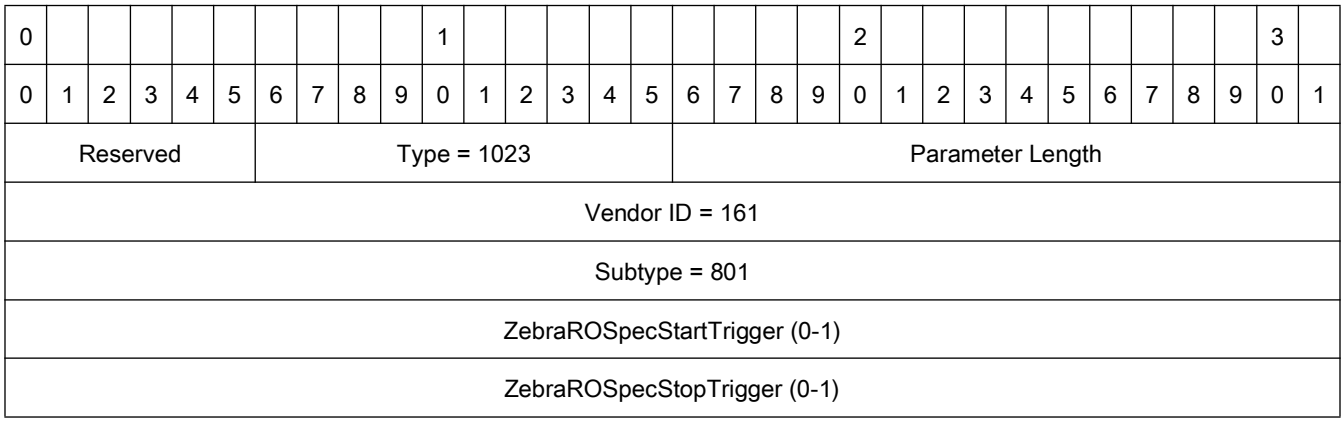
NXPBrandIDCheckConfig



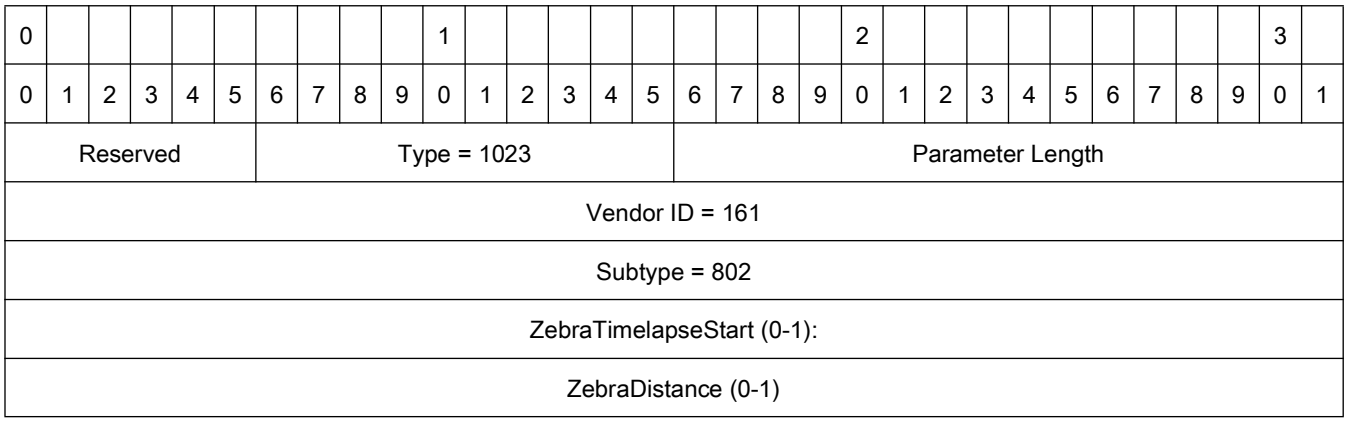
BrandIDCheckStatus



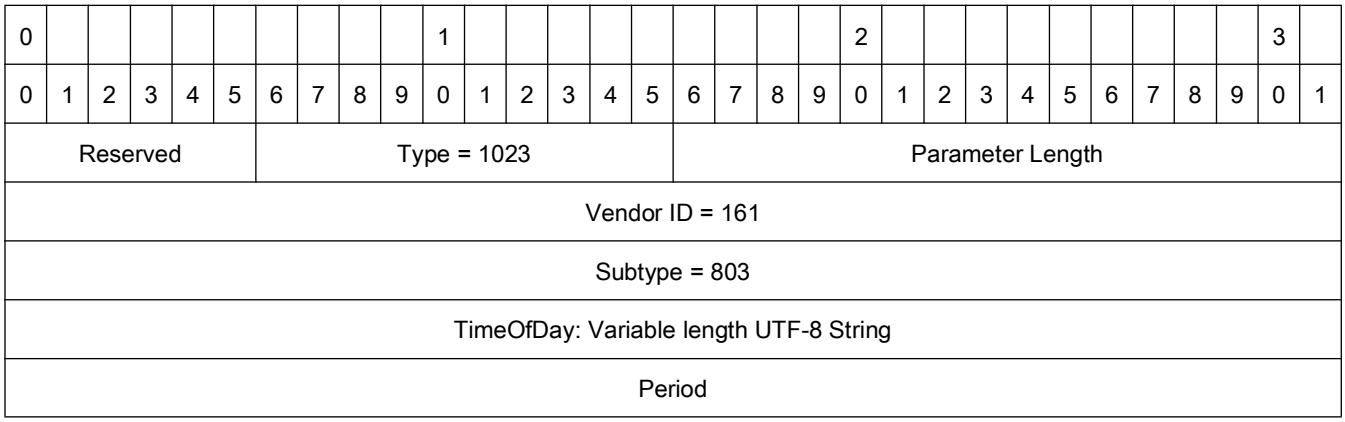
ZebraROTriggerSpec



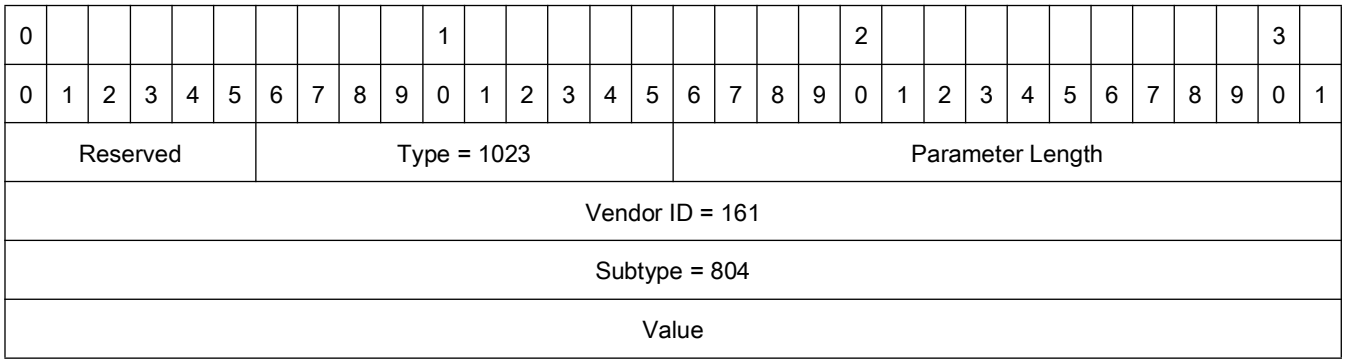
ZebraROSpecStartTrigger



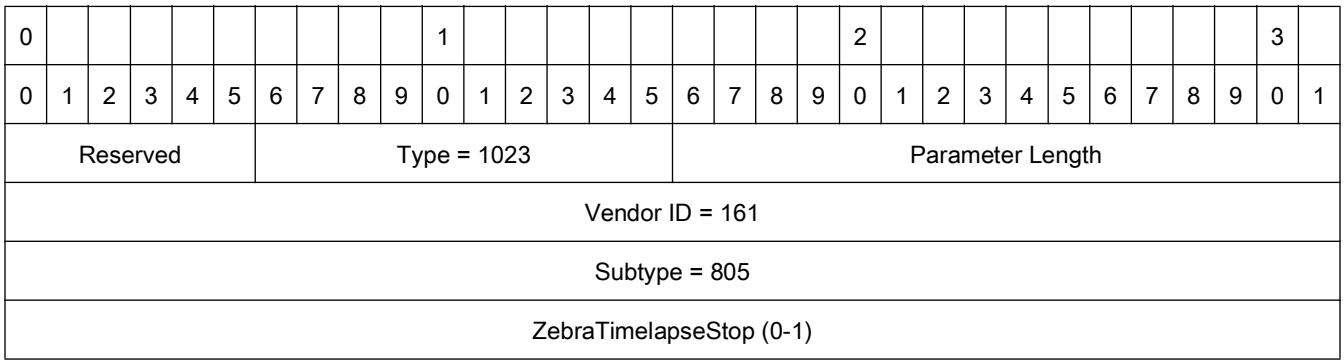
ZebraTimelapseStart



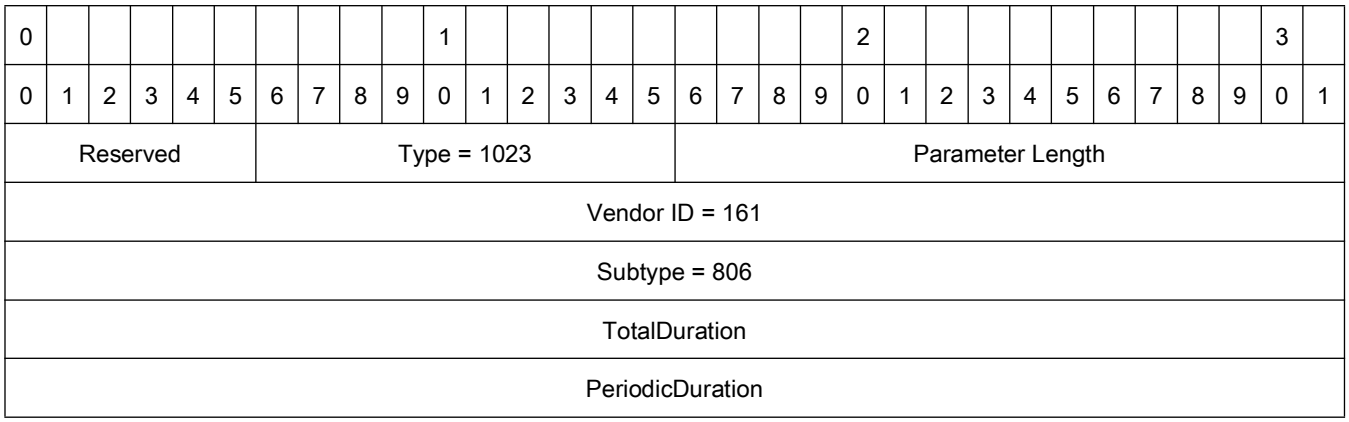
ZebraDistance



ZebraROSpecStopTrigger



ZebraTimelapseStop



SNMP

Introduction

Zebra FX7400, FX7500, FX9500,FX9600, and ATR7000 readers support RFC1213 (MIB for Network Management of TCP/IP-based internets: MIB-II). FX7400 and FX7500, FX9600, and ATR7000 readers also support GS1 standard RM MIB, and Zebra custom MIB.
Go to gs1.org/standards/epc-rfid/reader-management/1-0-1 for more details.

NOTE: MC Series readers do not support SNMP.

GS1 RM protocol MIB

Following are the contents of the GS1 RM protocol MIB:

```
-- ****************************************************************
-- Copyright (c) 2005-2007 EPCglobal Inc(tm), All Rights Reserved.
-- ****************************************************************
EPCGLOBAL-SMI-MIB DEFINITIONS ::= BEGIN

IMPORTS

    enterprises, 6974
    MODULE-IDENTITY 6975
    FROM SNMPv2-SMI;

epcglobal MODULE-IDENTITY

    LAST-UPDATED "200610040000Z" 6980
    ORGANIZATION "EPCglobal, Inc." 6981
    CONTACT-INFO
        "    EPCglobal MIB Administrator
             GS1/EPCglobal, Inc.
             Princeton Pike Corporate Center
             1009 Lenox Drive, Suite 202
             Lawrenceville, NJ 08648
             US

             Tel:   +1 609 620 0200
             Email: mibs@lists.epcglobalinc.org"
```

```
    DESCRIPTION
            "The EPCglobal central registration module, containing
            the top-level organization of the EPCglobal private
            enterprise namespace."
    REVISION     "200610040000Z"
    DESCRIPTION
            "Defined in conformance with the EPCglobal Reader Management and
             Reader Protocol specifications"

    ::= {enterprises 22695}  -- assigned by IANA

    --
    -- The EPCglobal private enterprise number assigned by
    -- the Internet Assigned Numbers Authority (IANA).
    --
    epcgSmiManagement         OBJECT IDENTIFIER ::= { epcglobal 1 }
    epcgSmiExperimental       OBJECT IDENTIFIER ::= { epcglobal 2 }

END
```

Zebra Custom MIB

Following are the contents of the Zebra custom MIB:

```
    --
    --    MOTOROLA-FX7400.mib
    --    MIB generated by MG-SOFT Visual MIB Builder Version 6.0 Build 88
    --    Tuesday, June 15, 2010 at 12:04:23
    --

    MOTOROLA-FX7400 DEFINITIONS ::= BEGIN

    IMPORTS
    enterprises, MODULE-IDENTITY, OBJECT-IDENTITY
    FROM SNMPv2-SMI;

    --    1.3.6.1.4.1.161
    motorolaMIB MODULE-IDENTITY
    LAST-UPDATED "201006151111Z" -- June 15, 2010 at 11:11 GMT
    ORGANIZATION



    DESCRIPTION
    "This MIB module contains custom MIB variables for Motorola FX7400 and FX7500 RFID readers."
    REVISION "201006151136Z" -- June 15, 2010 at 11:36 GMT
    DESCRIPTION
    "Revision on June 15 2010"
```

```
::= { enterprises 161 }

--
--    Node definitions
--

--    1.3.6.1.4.1.161.0
motorolaTrap OBJECT-IDENTITY
STATUS current
DESCRIPTION
"Sub-tree for registrations" ::=
{ motorolaMIB 0 }

-- 1.3.6.1.4.1.161.1
motorolaCustomCmds OBJECT-IDENTITY
STATUS current
DESCRIPTION
"Sub-tree for common object and event definitions"

::= { motorolaMIB 1 }

--    1.3.6.1.4.1.161.1.1
motorolaCommitCmd OBJECT-IDENTITY
STATUS current
DESCRIPTION
"Commit command saves the changes to flash."
::= { motorolaCustomCmds 1 }

--    1.3.6.1.4.1.161.1.2
motorolaSysObjId OBJECT-IDENTITY
STATUS current
DESCRIPTION
"Sub-tree for specific object and event definitions"
::= { motorolaCustomCmds 2 }

END

--
--    MOTOROLA-FX7400.mib
--
```

SNMP MIB

A new SNMP MIB custom commit command preserves SNMP data across reader reboots. This command is equivalent to the **Commit** command in the web console, so using either command preserves the changes made using either the console and SNMP. A commit operation saves the following elements:

- epcgReadPointOperStateNotifyEnable

- epcgReadPointOperNotifyFromState
- epcgReadPointOperNotifyToState
- epcgReadPointOperNotifyStateLevel
- epcgRdrDevOperNotifStateLevel.

SNMP only supports the custom **Commit** operation and does not support the **Discard** operation.

Global RM MIB

The Global RM MIB supports the following elements:

- epcgRdrDevDescription
- epcgRdrDevRole
- epcgRdrDev
- epcgRdrDevSerialNumber
- epcgRdrDevTimeUtc
- epcgRdrDevReboot
- epcgRdrDevResetStatistics
- epcgRdrDevResetTimestamp
- epcgRdrDevNormalizePowerLevel
- epcgRdrDevNormalizeNoiseLevel
- epcgRdrDevOperStatus
- epcgRdrDevOperStatusPrior
- epcgRdrDevOperStateEnable
- epcgRdrDevOperNotifFromState
- epcgRdrDevOperNotifToState
- epcgRdrDevOperNotifStateLevel
- epcgRdrDevOperStateSuppressInterval
- epcgRdrDevOperStateSuppressions
- epcgRdrDevFreeMemory
- epcgRdrDevFreeMemoryNotifEnable
- epcgRdrDevFreeMemoryNotifLevel
- epcgRdrDevFreeMemoryOnsetThreshold
- epcgRdrDevFreeMemoryAbateThreshold
- epcgRdrDevFreeMemoryStatus
- epcgRdrDevMemStateSuppressInterval
- epcgRdrDevMemStateSuppressions
- epcgReaderServerAddressType
- epcgReaderServerAddress
- epcgReaderServerRowStatus
- epcgReadPointName

- epcgReadPointDescription
- epcgReadPointAdminStatus
- epcgReadPointOperStatus
- epcgReadPointOperStateNotifyEnable
- epcgReadPointOperNotifyFromState
- epcgReadPointOperNotifyToState
- epcgReadPointOperNotifyStateLevel
- epcgReadPointOperStatusPrior
- epcgReadPointOperStateSuppressInterval
- epcgReadPointOperStateSuppressions
- epcgAntRdPntTagsIdentified
- epcgAntRdPntTagsNotIdentified
- epcgAntRdPntMemoryReadFailures
- epcgAntRdPntReadFailureNotifEnable
- epcgAntRdPntReadFailureNotifLevel
- epcgAntRdPntWriteOperations
- epcgAntRdPntWriteFailures
- epcgAntRdPntWriteFailuresNotifEnable
- epcgAntRdPntWriteFailuresNotifLevel
- epcgAntRdPntKillOperations
- epcgAntRdPntKillFailures
- epcgAntRdPntKillFailuresNotifEnable
- epcgAntRdPntKillFailuresNotifLevel
- epcgAntRdPntEraseOperations
- epcgAntRdPntEraseFailures
- epcgAntRdPntEraseFailuresNotifEnable
- epcgAntRdPntEraseFailuresNotifLevel
- epcgAntRdPntLockOperations
- epcgAntRdPntLockFailures
- epcgAntRdPntLockFailuresNotifEnable
- epcgAntRdPntLockFailuresNotifLevel
- epcgAntRdPntPowerLevel
- epcgAntRdPntNoiseLevel
- epcgAntRdPntTimeEnergized
- epcgAntRdPntMemoryReadOperations
- epcgAntRdPntReadFailureSuppressInterval
- epcgAntRdPntReadFailureSuppressions
- epcgAntRdPntWriteFailureSuppressInterval
- epcgAntRdPntWriteFailureSuppressions

- epcgSrcOperStateSuppressInterval
- epcgSrcOperStateSuppressions
- epcgRdPntSrcRowStatus
- epcgReaderDeviceOperationState
- epcgRdrDevMemoryState
- epcgReadPointOperationState
- epcgReaderAntennaReadFailure
- epcgReaderAntennaWriteFailure
- epcgReaderAntennaKillFailure
- epcgReaderAntennaEraseFailure
- epcgReaderAntennaLockFailure
- epcgReaderIoPortOperationState
- epcgReaderSourceOperationState
- epcgReaderNotificationChanOperState

TRAP Services

SNMP also supports TRAP services. Traps are sent in the following cases:

- Heartbeat
- Starting/stopping application
- Change of service (Telnet/SSH/FTP/FTPS) (FX7500 and FX9600 do not support Telnet)
- Firmware upgrade
- epcgReadPointOperationState (refer to RM MIB for details)
- epcgReaderDeviceOperationState (refer to RM MIB for details)

XML Schema for RM Extensions

Introduction

The XML schema that defines the RM extensions can be found on the support site for the FX7400, FX9500, FX9600, and ATR7000 at: zebra.com/support. The XML scheme enables a simple method of exercising the Reader Management commands described in Reader Management Custom Extensions.

Index

C

E

F

I

L

M

O

R

S

T

V