

**RFI641**  
**Radio Frequency Interrogator**  
**(UHF)**



Firmware Version V2.0



**Software Versions**

Device version	Function	Version
RFI641-0422	Firmware	V 2.0

**Copyright**

Copyright © 2007- 2008  
SICK AG Waldkirch  
Auto Ident, Reute Plant  
Nimburger Strasse 11  
79276 Reute  
Germany

**Latest Telegram Listing Version**

For the latest version of this Telegram Listing (PDF), see [www.sick.com](http://www.sick.com).

## Contents

<b>1</b>	<b>Notes on this Document</b> .....	<b>7</b>
1.1	Purpose.....	7
1.2	Target Audience .....	7
1.3	Information Content.....	7
1.4	Symbols .....	7
<b>2</b>	<b>Safety Information</b> .....	<b>9</b>
2.1	Authorised Users .....	9
2.2	Intended Use .....	9
2.3	General Safety Instructions and Protection Measures.....	9
<b>3</b>	<b>Introduction</b> .....	<b>11</b>
3.1	Device components of an RFID system when using the RFI641.....	11
3.2	Functional details.....	12
3.3	Namespaces of the commands .....	12
3.4	Commando categories.....	12
3.5	Data types.....	13
3.6	Permissions and passwords.....	14
<b>4</b>	<b>Available commands</b> .....	<b>17</b>
4.1	READER commands .....	17
4.2	SETUP commands.....	31
4.3	INFO commands.....	37
4.4	COM commands.....	41
4.5	TAG Commands.....	51
4.6	DIO Commands .....	77
4.7	ANTENNAS Commands.....	81
4.8	MODEM Commands.....	91
4.9	USER Command .....	122
4.10	EVENT Commands .....	123
<b>5</b>	<b>Error messages</b> .....	<b>135</b>

**Tables**

Table 3-1: Data types in the commands ..... 13

Table 3-2: Initial passwords for the user levels ..... 14

Table 4-1: Overview: READER commands..... 17

Table 4-2: Overview: SETUP commands..... 31

Table 4-3: Overview: INFO commands..... 37

Table 4-4: Overview: COM commands..... 41

Table 4-5: Overview: TAG commands ..... 51

Table 4-6: Overview: DIO commands..... 77

Table 4-7: Overview: ANTENNAS commands ..... 81

Table 4-8: Overview: MODEM commands ..... 91

Table 4-9: Overview: EVENT commands..... 123

Table 5-1: Overview: error messages ..... 135

**Figures**

Fig. 3-1: Components of an RFID system when using the RFI641 ..... 11

## Abbreviations used

<b>ASK</b>	Amplitude Shift Keying
<b>CLI</b>	Command Line Interface
<b>CRC</b>	Cyclic-redundancy Check
<b>CT</b>	Configuration Tool
<b>CTS</b>	Clear to send
<b>CW</b>	Continous Wave
<b>dB</b>	Decibel
<b>DIO</b>	Digital In/Out
<b>DHCP</b>	Dynamic Host Configuration Portocol
<b>DNS</b>	Domain Name Server
<b>DSP</b>	Digital Signal Processor
<b>EAS</b>	Electronic Article Surveillance
<b>EPC</b>	Electronic Product Code
<b>EIRP</b>	Equivalent Isotropic Radiated Power
<b>ERP</b>	Enterprise Resource Planning
<b>ERP</b>	Effective Radiated Power
<b>ETSI</b>	European Telecommunications Standards Institute
<b>ETX</b>	End of Text
<b>FCC</b>	Federal Communications Commission
<b>Gen2</b>	EPC Class1 Gen2
<b>HW</b>	Hardware
<b>ID</b>	Identification
<b>I/O</b>	In/Out
<b>IP</b>	Internet Protocol
<b>ISO-B</b>	ISO-Standard 18000-6 B
<b>ISO-C</b>	ISO-Standard 18000-6 C
<b>LBT</b>	Listen Before Talk
<b>MAC</b>	Media Access Control
<b>NTP</b>	Network Time Protocol
<b>NXP</b>	Next Experience (formerly Philips Semiconductors)
<b>PC</b>	Personal Computer
<b>PC</b>	Protocol Control
<b>PID</b>	Process ID
<b>PLC</b>	Programmable Logic Controller
<b>RF</b>	Radio Frequency
<b>RFI</b>	Radio Frequency Interrogator
<b>RFT</b>	Radio Frequency Tag
<b>RN16</b>	16-bit random numer
<b>RP</b>	Reverse polarity
<b>RTS</b>	Request to send
<b>SCM</b>	Supply Chain Management
<b>SL</b>	Selected Flag
<b>STX</b>	Start of Text

---

<b>Supertag</b>	<u>Property protocol semipassive tag</u>
<b>SW</b>	Software
<b>TARI</b>	Type A Reference Interval
<b>TID</b>	Tag Identifier determines available commands, manufacturer and serial number of the tag
<b>UHF</b>	Ultra-High-Frequency
<b>VSWR</b>	Voltage Standing Wave Ratio

RFI641

# 1 Notes on this Document

## 1.1 Purpose

This document serves as a guide for configuration and programming the RFID Interrogator RFI641 in the following variants with a consolidated command language using so-called telegrams:

- RFI641-0422 Interrogator, IP 40
- RFI641-1522 Interrogator, installed in a cabinet, IP 65

This document contains information on:

- Device components of the RFI641
- Communication between user (host) and interrogator
- Commands/responses in the telegrams
- Error messages

**Important** From now on, the RFI641 interrogator (also called reader) with all the variants will simply be referred to as "RFI641".

The expressions "transponder" and "tag" are interchangeable. In this documentation "tag" is used.

## 1.2 Target Audience

This document is aimed at technicians and engineers.

## 1.3 Information Content

This document contains all the required information for the communication between the user and the RFI641 using telegram commands. This commands are grouped in sections regarding their usage.



For assembly, electrical installation, start-up and technical data of the RFI641 see *RFI641 Operating Instructions* (no. 8011867, English edition)

Further information on automatic identification when using the RFID is available from SICK AG, Auto Ident division. On the internet at [www.sick.com](http://www.sick.com).

## 1.4 Symbols

Some of the information in this document is marked specially so that you can access it quickly:



### WARNING

**Warning notice!**

Warnings are provided to prevent injury to operating personnel or serious damage to the RFI641.

- Always read warnings carefully and observe them all the times.

**Reference** Italics are used to refer to more detailed information elsewhere.

**Important** Notes indicate special features or characteristics.

**Explanation** Explanations provide background information on technical correlations.

**Recommendation** Recommendations help you to carry out certain procedures more effectively.

**Default settings** Marks a section containing the values of the factory default settings of the RFI641.



This symbol refers to additional technical documentation.

- This symbol provides instructions on a single action that you have to carry out. Multi-step instructions are provided in a numerical sequence.

#### Conventions used for the commands

**Bold Courier font with grey shading indicates code to be entered by the user.**

Courier font indicates answers of the RFI641.

**(values)** within parentheses indicate parameters.

*(values)* in italics indicate user defined variables.

<n> indicates a variable number used in a function that can apply to several different devices such as antennas or I/O ports.

## 2 Safety Information

This chapter deals with your and the operator's safety.

- Read this chapter carefully before using the RFI641.

### 2.1 Authorised Users

To ensure that the RFI641 works properly and safely, it must be parameterised and operated by sufficiently qualified personnel.

The following qualifications are required for commissioning and operation:

- Basic, practical training in electrical engineering
- Knowledge of the relevant safety guidelines
- Knowledge of the hardware and software environment for the relevant application
- Basic data transfer knowledge
- Basic programming knowledge

### 2.2 Intended Use

The RFI641 is used to read and write RFID-tags which are within the range of the antennas connected to the RFI641.

**Important** Any warranty claims vis-à-vis SICK AG will be rendered invalid if the device is used for any other purpose or if changes are made to the device, including any made during the installation and electrical connection procedures.

### 2.3 General Safety Instructions and Protection Measures

#### **NOTICE**

##### **Violation of national directives!**

This telegram listing only contains commands which enable operation of the RFI641 in accordance with national directives. Commands not listed in this document must only be used by SICK service, since these commands are able to cause the RFI641 to violate national directives.

- Only use commands described in this telegram listing.

#### **NOTICE**

##### **Risk of damages to the RFI641 due to activated and open antenna ports!**

Antenna ports of the RFI641 which are activated but not terminated may damage electronic components in the RFI641.

- Before powering up the RFI641 ensure that the cables of the used antennas are properly connected with the antenna ports.
- Do not disconnect the cables/terminating resistors during operation.

**Important** Execute the commands only when the RFI641 is ready-to-operate. It takes about 30 seconds to start up the RFI641 after turning on the power supply. During this time the red FAULT LED is on. The RFI641 is ready-to-operate when only the green LED „READY“ is on.

**Notes:**

### 3 Introduction

#### 3.1 Device components of an RFID system when using the RFI641

A RFID system normally consists of:

- RFI641 interrogator with external power supply
- 1 to 4 antennas (plus optional LBT antenna)
- 1 to 4 reading triggers (e. g. photoelectric reflex switch)
- Status signal of the 4 switching outputs
- Connection to a superior data processing system, where applicable using a fieldbus and a SICK connection module
- Temporary connection to a PC for configuration and diagnosis

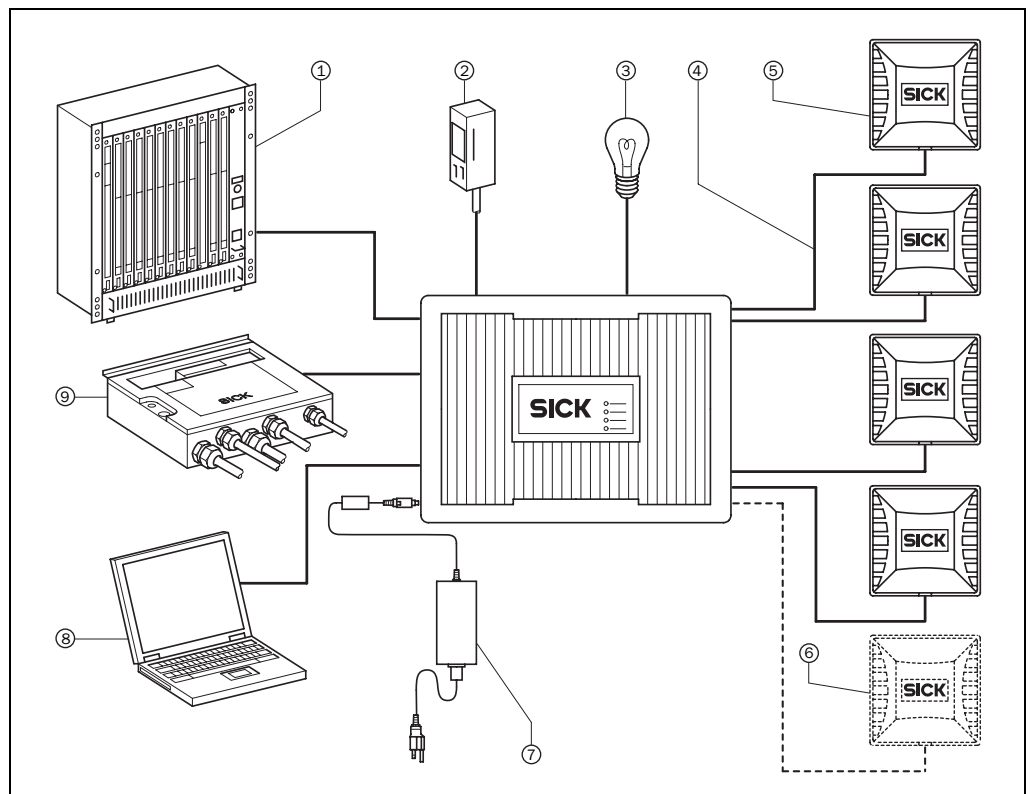


Fig. 3-1: Components of an RFID system when using the RFI641

- ① Middleware SCM/ERP-systems via Ethernet
- ② Four digital inputs
- ③ Four digital outputs
- ④ Antenna cable up to 10 m
- ⑤ RFA641 antennas
- ⑥ optional: additional LBT antenna
- ⑦ Power supply 100/240 V AC
- ⑧ Parameterisation and diagnosis via RS 232 or Ethernet
- ⑨ Profibus and DeviceNet via fieldbus boxes; optional: data exchange with PLC

## 3.2 Functional details

The RFI641 operates with an adjustable transmission frequency in the UHF-range (860 to 960 MHz). It is able to read and write the memory of appropriate tags within the range of up to four antennas connected to the RFI641. A fifth antenna for reading only (LBT) is optional obtainable.

## 3.3 Namespaces of the commands

The RFI641 reader is a highly versatile RFID system that can be configured to operate in almost any RFID application. As a result, all reader management functions and parameters can be easily configured by sending specific Command Line Interface (CLI) commands to the reader's command channel. These commands are organised using namespaces.

A namespace is a context for identifiers and is usually written in human readable form. A configuration namespace provides organisation and hierarchy to a set of configuration variables. Namespaces can be nested, with multiple namespaces existing under a higher namespace.

- **Reader** - Generic reader level functions
- **Setup** - Reader setup
- **Info** - Reader information
- **Com** - Communication level features
- **Tag** - Tag control features
- **DIO** - Digital input and output features
- **Antennas** - Antenna configuration
- **Modem** - Low-level modem control
- **User** - User defined variables
- **Events** - Events returned by reader
- **Errors** - Errors returned by reader

## 3.4 Commando categories

The RFI641 commands are categorised based on the following actions:

- Variable
- Function
- Error message

### 3.4.1 Variable

A variable can be `get` by a command or can be set by a command.

#### GET

A GET action retrieves the value of a specified reader configuration variable. The response of the RFI641 is either OK and a value or an error message.

The syntax of a GET command is as follows:

**variable\_name**

In the following example, the `com.serial.baudrate` variable is 57,600 bd:

```
com.serial.baudrate  
ok 57600
```

### SET

A SET action sets the value of the specified reader configuration variable. The response of the RFI641 is either OK or an error message.

The syntax of a SET command is as follows:

**variable\_name = value**

In the following example, the `modem.radio.lbt.enabled` variable is set to 1 (enabled):

```
modem.radio.lbt.enabled = 1
ok
```

### 3.4.2 Function

A function is a reader command which can be executed. The response of the RFI641 is either OK or an error message.

The syntax of a reader command is as follows:

**function\_name(name1=value1, name2=value2)**

In the following example, the `tag.write` command writes an EPC of `0x30112233445566778899aabb` and a kill code of `0x12345678` to a tag:

```
tag.write (tag_id = 0x30112233445566778899aabb, kill_pwd = 0x12345678)
ok
```

### 3.4.3 Error messages

If a command fails, an error is returned. For example, setting the value of the `tag.reporting.depart_time` to 1 will produce the `error.parser.value_out_of_range` response because the range of acceptable values for `tag.reporting.depart_time` is from 100 to 25,000 ms.

```
tag.reporting.depart_time = 1
error.parser.value_out_of_range
```

The errors returned by the reader are contained in the error namespace.

## 3.5 Data types

The RFI641 variable and function parameters use the following data types:

Data types	Description
Bool	True/false
String	Any string
Int	Numbers
Enum	Any one enum value
Enumlist	Multiple values from enum list
Array	Series of hex values
List	Array of integers

Table 3-1: Data types in the commands

## 3.6 Permissions and passwords

### 3.6.1 User level

The RF1641 supports an authentication scheme in which users must supply a valid login level and password to access the reader.

The **guest login level** provides read-only access to the reader. Clients that login in the guest level can read the settings of the reader and can access the tags that the reader has inventoried. Clients in this level cannot change the configuration of the reader.

The **admin login level** provides read-write access to the reader. Clients that login in the admin level can read and write the settings of the reader and can access the tags that the reader has inventoried.

To assign the login levels, each namespace command has one or more of the following permissions:

r = read

w = write

x = execute

- = not allowed

### 3.6.2 Initial Passwords

The reader is delivered with the following passwords for the login levels:

Login Level	Initial Password
guest	readerguest
admin	readeradmin

Table 3-2: Initial passwords for the user levels

### 3.6.3 Changing Passwords

The **reader.set\_pwd()** command is used to change the passwords. This command requires the following three parameters:

- login (level for password to be changed)
- pwd (existing password for the login level)
- new\_pwd (new password for the login level)

For example, to change the initial password for the guest login level to 19qht34 use the following command:

```
reader.set_pwd(guest,readerguest,19qht34)
```

ok

### 3.6.4 Setting the Default Login Level

In order to accommodate multiple access levels, the reader supports a default login level. The default login level is the level assigned to each client as it connects to the reader. The default login level can be set to:

- unauthenticated
- guest
- admin

#### unauthenticated

When the default login level is set to unauthenticated, all clients that connect to the reader must login (with the `reader.login()` command) before they can access any features of the reader. Only the `reader.login()` command is accepted when the default login level is set to unauthenticated. This allows the reader to be locked down and forces all connecting clients to login before accessing the reader.

#### guest

When the default login level is set to guest, all clients that connect to the reader are automatically logged in at the guest level at connection time. This means they can perform all guest permitted actions without issuing the `reader.login()` command.

#### admin

When the default login level is set to admin, all clients that connect to the reader are automatically logged in at the admin level at connection time. This means they can perform all admin permitted actions without issuing the `reader.login()` command.

**Default settings** The factory default login level is admin. This allows a quick turn-up and initial testing of the reader. When the reader is deployed, the default login level should be set to the level that best suits your security/accessibility needs. The default login level can be set using the `setup.default_login_level` configuration variable.

For example to set the default login level to guest use the following command:

```
setup.default_login_level = guest
ok
```

### 3.6.5 Embedded Application/Script Login

You can customise your reader by writing applications and scripts that execute on the reader. When an application or script executes, it connects as a client. When an application or script running on the reader connects, it is automatically logged in at the admin level. This allows applications and scripts to run on any reader without knowing the password.

**Notes:**

RFI641

## 4 Available commands

### 4.1 READER commands

Command	Description	See Chapter/Page
reader.check_status()	Checks and returns reader status.	<a href="#">4.1.1 / 18</a>
reader.check_service()	Checks and returns the status of a system service.	<a href="#">4.1.2 / 18</a>
reader.start_service	Starts a system service	<a href="#">4.1.3 / 19</a>
reader.stop_service	Stops a running system service	<a href="#">4.1.4 / 19</a>
reader.is_alive()	Verifies if reader is operational.	<a href="#">4.1.5 / 19</a>
reader.flash_led()	Flashes the reader LEDs.	<a href="#">4.1.6 / 20</a>
reader.events.bind()	Sets event channel for future calls to reader.events.register() function	<a href="#">4.1.7 / 20</a>
reader.events.list_registered()	Lists the channel id for each event that has been registered.	<a href="#">4.1.8 / 20</a>
reader.events.query_bind()	Queries the event channel ID for the current command session	<a href="#">4.1.9 / 21</a>
reader.events.register()	Registers event over an event channel	<a href="#">4.1.10 / 21</a>
reader.events.trigger()	Distributes event to all registered event channels	<a href="#">4.1.11 / 22</a>
reader.events.unregister()	Unregisters event over an event channel	<a href="#">4.1.12 / 22</a>
reader.login(login,pwd)	Allows reader login.	<a href="#">4.1.13 / 22</a>
reader.logout()	Allows reader logout.	<a href="#">4.1.14 / 23</a>
reader.reboot()	Reboots the reader.	<a href="#">4.1.15 / 23</a>
reader.rollback_firmware()	Rolls back the reader firmware.	<a href="#">4.1.16 / 23</a>
reader.set_pwd(login,pwd,new_pwd)	Changes reader passwords.	<a href="#">4.1.17 / 24</a>
reader.timestamp_all_events	Events that do not have finer-granularity timestamp control will contain a timestamp	<a href="#">4.1.18 / 24</a>
reader.upgrade_firmware(filename)	Upgrades reader firmware.	<a href="#">4.1.19 / 24</a>
reader.view_log(log_file)	Displays a log file.	<a href="#">4.1.20 / 24</a>
reader.who_am_i()	Reports the current login level.	<a href="#">4.1.21 / 25</a>
reader.apps.delete(filename)	Deletes a user application from reader.	<a href="#">4.1.22 / 25</a>
reader.apps.list()	Returns list of all user applications loaded on the reader.	<a href="#">4.1.23 / 25</a>
reader.apps.list_running()	Returns list of all user applications that are running on the reader.	<a href="#">4.1.24 / 26</a>
reader.apps.start_java(classname,classpath,jar,args, autostart)	Execute a user java application.	<a href="#">4.1.25 / 26</a>
reader.apps.start_python(filename,args,autostart)	Executes a user python script.	<a href="#">4.1.26 / 27</a>
reader.apps.stop(pid)	Stops running user script or java application.	<a href="#">4.1.27 / 27</a>
reader.apps.view_log(pid)	View an application log file.	<a href="#">4.1.28 / 28</a>
reader.profile.active	Contains the active profile for the reader.	<a href="#">4.1.29 / 28</a>
reader.profile.delete(filename)	Deletes the specified profile.	<a href="#">4.1.30 / 28</a>
reader.profile.list()	Returns list of all saved profiles.	<a href="#">4.1.31 / 29</a>
reader.profile.load(filename)	Loads specified reader profile.	<a href="#">4.1.32 / 29</a>
reader.profile.reset_factory_default()	Resets reader profile to the factory default.	<a href="#">4.1.33 / 29</a>

Table 4-1: Overview: READER commands

Command	Description	See Chapter/Page
reader.profile.save(filename)	Saves current reader configuration to specified profile.	<a href="#">4.1.34 / 30</a>
reader.profile.show_running_config	Shows the current running configuration of the reader	<a href="#">4.1.35 / 30</a>

Table 4-1: Overview: READER commands

#### 4.1.1 reader.check\_status()

This function causes the reader to perform some checks and returns the status of these checks. The information returned includes:

Type	function																				
<b>Permissions</b>	guest = x admin = x																				
<b>Return values</b>	<table border="1"> <tbody> <tr> <td>reader_uptime</td> <td>Number of seconds the reader has been running.</td> </tr> <tr> <td>in_use_memory</td> <td>Amount of memory in use.</td> </tr> <tr> <td>free_memory</td> <td>Amount of free memory available within the reader.</td> </tr> <tr> <td>cpu_load</td> <td>CPU load in percent.</td> </tr> <tr> <td>modem_alive</td> <td>Indication that the modem is up and running.</td> </tr> <tr> <td>modem_uptime</td> <td>Number of seconds the modem has been running.</td> </tr> <tr> <td>antenna_status</td> <td>Status of the antenna(s).</td> </tr> <tr> <td>tx_interlock</td> <td>Condition of regulatory interlock status.</td> </tr> <tr> <td>synth_locked</td> <td>Indication of the synthesiser lock status.</td> </tr> <tr> <td>ps_fault</td> <td>Power supply fault status.</td> </tr> </tbody> </table>	reader_uptime	Number of seconds the reader has been running.	in_use_memory	Amount of memory in use.	free_memory	Amount of free memory available within the reader.	cpu_load	CPU load in percent.	modem_alive	Indication that the modem is up and running.	modem_uptime	Number of seconds the modem has been running.	antenna_status	Status of the antenna(s).	tx_interlock	Condition of regulatory interlock status.	synth_locked	Indication of the synthesiser lock status.	ps_fault	Power supply fault status.
reader_uptime	Number of seconds the reader has been running.																				
in_use_memory	Amount of memory in use.																				
free_memory	Amount of free memory available within the reader.																				
cpu_load	CPU load in percent.																				
modem_alive	Indication that the modem is up and running.																				
modem_uptime	Number of seconds the modem has been running.																				
antenna_status	Status of the antenna(s).																				
tx_interlock	Condition of regulatory interlock status.																				
synth_locked	Indication of the synthesiser lock status.																				
ps_fault	Power supply fault status.																				

The following example gets the status of a reader:

```
reader.check_status()
ok
reader_uptime = 178,
in_use_memory = 43716608,
free_memory = 19787776,
cpu_load = 9,
modem_alive = true,
modem_uptime = 160,
antenna_status = ok,
tx_interlock = false,
synth_locked = true,
ps_fault = false
```

#### 4.1.2 reader.check\_service()

This function checks the status of a system service. The information returned could be one of the following:

Type	function				
<b>Permissions</b>	guest = x admin = x				
<b>Parameters</b>	program (enum definitions.enum.service_name)				
<b>Return values</b>	<table border="1"> <tbody> <tr> <td>is alive, pid=</td> <td>The service is running.</td> </tr> <tr> <td>is not running</td> <td>The service is not running</td> </tr> </tbody> </table>	is alive, pid=	The service is running.	is not running	The service is not running
is alive, pid=	The service is running.				
is not running	The service is not running				

RFI641

The following example gets the status of the discovery service:

```
reader.check_service(discovery)
```

```
ok
discovery is alive, pid=14573
```

#### 4.1.3 reader.start\_service()

This function starts a system service.

<b>Type</b>	function
<b>Permissions</b>	guest = - admin = x
<b>Parameters</b>	program

The following example starts the discovery service:

```
reader.start_service(discovery)
```

```
ok
```

#### 4.1.4 reader.stop\_service()

This function stops a running system service.

<b>Type</b>	function
<b>Permissions</b>	guest = - admin = x
<b>Parameters</b>	program

The following example stops the discovery service:

```
reader.stop_service(discovery)
```

```
ok
```

#### 4.1.5 reader.is\_alive()

This function verifies if the reader is operational.

<b>Type</b>	function
<b>Permissions</b>	guest = x admin = x

The following example shows how to use this command to verify the reader is alive and responsive:

```
reader.is_alive()
```

```
ok
```

**4.1.6 reader.flash\_led()**

This function flashes the reader LEDs. The led parameter is a bitmask. Time is the duty cycle on-off time in seconds. The LEDs will flash continuously until canceled. To cancel all flashing, set time=0, with any led parameter value. If no parameters are entered, all LEDs will flash for 5 seconds.

<b>Type</b>	function	
<b>Permissions</b>	guest = x admin = x	
<b>Parameters</b>	led (int)	1 = Power LED 2 = Fault LED 4 = Tx LED 8 = Sense LED
	time (int)	0 to 5 second

The following example continuously flashes both the power and sense LEDs for 1 second on, 1 second off:

```
reader.flash_led(led=9, time=1)
```

```
ok
```

**4.1.7 reader.events.bind()**

This function sets the event channel for future calls to the reader.events.register() function. Once connected to the event channel, a channel ID is provided. The bind function uses this channel ID to set the event channel so it does not need to be passed in on future calls to reader.events.register.

<b>Type</b>	function	
<b>Permissions</b>	guest = x admin = x	
<b>Parameters</b>	id (int)	

The following example binds future calls to reader.events.register() to channel ID 14 and registers for the "event.tag.report" on event channel 14:

```
reader.events.bind(id = 14)
```

```
ok
```

```
reader.events.register(name = event.tag.report)
```

```
ok
```

**4.1.8 reader.events.list\_registered()**

This command lists information indicating what event channels have registered for what events. An input for an event channel id can be provided so that only events registered on that specific event channel will be displayed.

<b>Type</b>	function	
<b>Permissions</b>	guest = - admin = x	
<b>Parameters</b>	id (int)	

RFI641

The following example shows using the `reader.events.list_registered` command to show what event channels are registered for what events:

```
reader.events.list_registered()
ok
(event = event.stats.report, fds = none)
(event = event.debug.trace, fds = 14 15)
(event = event.tag.report, fds = 15)
(event = event.tag.arrive, fds = 16)
(event = event.tag.depart, fds = 16)
(event = event.tag.scan_tags_complete, fds = 17)
(event = event.status, fds = 17)
```

If an input for event channel id is provided, it will filter the output as shown below:

```
reader.events.list_registered(16)
ok
(event = event.tag.arrive, fds = 16)
(event = event.tag.depart, fds = 16)
```

#### 4.1.9 `reader.events.query_bind()`

This function returns the event channel ID associated with the current command session.

<b>Type</b>	function
<b>Permissions</b>	guest = x admin = x

The following example returns the event channel ID associated with the current command channel:

```
reader.events.query_bind ()
ok
```

#### 4.1.10 `reader.events.register()`

This function registers to have events delivered over an event channel. Once connected to an event channel, a channel id is provided. This channel id, along with event name, are passed to this function to register for the specified event.

<b>Type</b>	function
<b>Permissions</b>	guest = x admin = x
<b>Parameters</b>	id (int) name (string)

The first example registers for "event.tag.report" on event channel 14.

The second and third examples register for "event.tag.arrive" and "event.tag.depart" on event channel 16.

The fourth example registers for all "event.tag.\*" events on event channel 17.

```
reader.events.register(14, event.tag.report)
```

ok

```
reader.events.register(16, event.tag.arrive)
```

ok

```
reader.events.register(16, event.tag.depart)
```

ok

```
reader.events.register(17, event.tag)
```

ok

#### 4.1.11 reader.events.trigger()

This function is used by a client to distribute an event to all registered event channels.

Type	function
Permissions	guest = x admin = x
Parameters	name (string)

The following example triggers "event.app.start\_business\_process" (sends it out to all registered event channels):

```
reader.events.trigger(name = "event.app.start_business_process ap_id = 23")
```

ok

#### 4.1.12 reader.events.unregister

This function unregisters events on a event channel.

Type	function
Permissions	guest = x admin = x
Parameters	id (int) name (string)

The following example un-registers for the "event.tag.depart" on event channel 16:

```
reader.events.unregister(id=16,name=event.tag.depart)
```

ok

#### 4.1.13 reader.login()

This function allows users to login to the reader.

Type	function
Permissions	guest = x admin = x
Parameters	login (string) pwd (string)

RFI641

The following example shows an administrator login:

```
reader.login(login = admin, pwd = 1Hjg7df45)
```

```
ok
```

#### 4.1.14 reader.logout()

This function allows users to logout of the reader. By default the login is changed to guest.

<b>Type</b>	function
<b>Permissions</b>	guest = x admin = x

The following example shows a typical user logout:

```
reader.logout()
```

```
ok
```

#### 4.1.15 reader.reboot()

This function reboots the reader.

<b>Type</b>	function
<b>Permissions</b>	guest = x admin = x

The following example forces the reader to reboot (function does not return any response):

```
reader.reboot()
```

#### 4.1.16 reader.rollback\_firmware()

This function rolls back the reader firmware to the previous version:

<b>Type</b>	function
<b>Permissions</b>	guest = - admin = x

The following example rolls back the reader firmware to the previous version:

```
reader.rollback_firmware()
```

```
ok
```

**4.1.17 reader.set\_pwd()**

This function changes the reader password to the password specified in the function parameter. The old and new password must be specified.

<b>Type</b>	function
<b>Permissions</b>	guest = x admin = x
<b>Parameters</b>	login (string) pwd (string) new_pwd (string)

The following example shows an administrator password change from 1Hjg7df45 to Tat1001:

```
reader.set_pwd (login = admin, pwd = 1Hjg7df45, new_pwd = Tat1001)
ok
```

**4.1.18 reader.timestamp\_all\_events**

When true, all events that do not have finer-granularity timestamp control will contain a timestamp. Certain events such as event.tag.report or event.tag.arrive have specific variables controlling whether or not they contain a timestamp, and are not affected by this variable.

<b>Type</b>	var
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	bool
<b>Default</b>	FALSE

**4.1.19 reader.upgrade\_firmware()**

This function upgrades the reader firmware with the file specified in the function parameter.

<b>Type</b>	function
<b>Permissions</b>	guest = - admin = x
<b>Parameters</b>	filename (string)

The following example upgrades the reader to the firmware contained in the "latest.sam" file:

```
reader.upgrade_firmware (filename=latest.sam)
ok
```

**4.1.20 reader.view\_log()**

This function displays the selected log file.

<b>Type</b>	function
<b>Permissions</b>	guest = x admin = x
<b>Parameters</b>	log_file (enum definitions.enum.reader.log_files)

RFI641

The following example displays the reader error log:

```
reader.view_log(READER_ERROR_LOG)
```

#### 4.1.21 reader.who\_am\_i()

This function reports the user's login level.

<b>Type</b>	function
<b>Permissions</b>	guest = x admin = x

The following example returns a response indicating that the user is a guest:

```
reader.who_am_i()  
ok guest
```

#### 4.1.22 reader.apps.delete()

This function deletes a user application that is on the reader.

<b>Type</b>	function
<b>Permissions</b>	guest = - admin = x
<b>Parameters</b>	filename (string)

The following example deletes an application called test.py:

```
reader.apps.delete(filename=test.py)  
ok
```

#### 4.1.23 reader.apps.list()

This function returns a list of all user applications loaded on the reader. Applications can be started with the reader.apps.start\_python or reader.apps.start\_java() function, and can be stopped with the reader.apps.stop() function. The list of currently running applications is obtained with the reader.apps.list\_running() function.

<b>Type</b>	function
<b>Permissions</b>	guest = x admin = x

The following example lists all user applications loaded on the reader:

```
reader.apps.list()  
ok apps = dio.py epc_app.jar test.py
```

**4.1.24 reader.apps.list\_running()**

This function returns a list of all user applications running on the reader. Running applications are stopped with the `reader.apps.stop()` function.

<b>Type</b>	function
<b>Permissions</b>	guest = x admin = x

The following example lists all user applications running on the reader:

```
reader.apps.list_running()
```

```
ok
(pid = 8537, config = auto, status = running, command = dio.py 1)
(pid = 0820, config = manual, status = ended, command = epc_app.jar)
```

**4.1.25 reader.apps.start\_java()**

This function executes a user java application. The application can be a class contained in a class/jar file or the application can be run from a standalone jar file. Runtime arguments can be passed to the java application.

If the application is a class contained in a jar file, the jar filename must be specified in the classpath parameter. If the application is to be run from a standalone jar file, the jar filename must be specified in the jar parameter.

Multiple jar filenames can be specified in the jar parameter by separating the names with the ":" character. If the autostart parameter is set to true, the application will immediately execute and also be set to run at the next system boot.

If the auto start parameter is set to false (the default), the application will run immediately, but will not run again at the next system boot.

The process ID (PID) associated with the application is returned. The status of running applications can be displayed with the `reader.apps.list_running()` command. If the application sends information to stdout or stderr, it can be viewed with the `reader.apps.view_log()` command, passing the PID of the application.

<b>Type</b>	function
<b>Permissions</b>	guest = - admin = x
<b>Parameters</b>	classname (string) classpath (string) jar (string) args (string) autostart (bool)

The following example shows the use of this command to start a java application named "epc\_app.jar":

```
reader.apps.start_java(epc_app.jar)
```

```
ok pid = 0820
```

RFI641

**4.1.26 reader.apps.start\_python()**

This function executes the user python script specified by the filename parameter (required). Runtime arguments (optional) can be passed to the script. If the autostart parameter is set to true, the script executes immediately and is set to run at the next system boot. If the auto start parameter is set to false (the default), the script will run immediately, but will not run again at the next system boot. The process ID (PID) associated with the function is returned.

The status of the running apps can be displayed with the `reader.apps.list_running()` command. If the script sends information to stdout or stderr, it can be viewed with the `reader.apps.view_log()` command, passing the PID of the script.

<b>Type</b>	function
<b>Permissions</b>	guest = - admin = x
<b>Parameters</b>	filename (string) args (string) autostart (bool)

The following example starts a python script named "dio.py" passing it the argument "1" and making the script autostartable:

```
reader.apps.start_python(dio.py, "1", true)
ok pid = 8537
```

**4.1.27 reader.apps.stop()**

This function stops the user script or java application previously started with the `reader.apps.start_python()` or `reader.apps.start_java()` function.

<b>Type</b>	function
<b>Permissions</b>	guest = - admin = x
<b>Parameters</b>	pid (int)

The following example stops a previously started application with a PID of 8537:

```
reader.apps.stop(8537)
ok
```

**4.1.28 reader.apps.view\_log()**

This function displays an application log file. If the file is greater than 10 KB in size, the last 10 KB of the file are displayed.

<b>Type</b>	function
<b>Permissions</b>	guest = - admin = x
<b>Parameters</b>	pid (int)

The following example views an application's log file:

```
reader.apps.view_log(pid = 8537)
ok
dio 1 went high, starting scan for tags
dio 1 went low, stop scan for tags
scan found 196 tags
```

**4.1.29 reader.profile.active**

This variable contains the name of the profile that is the current active profile in the reader.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = r
<b>Data Type</b>	string
<b>Default</b>	"factory"

The following example gets the name of the active profile in use on the reader:

```
reader.profile.active
ok isoc_portal
```

**4.1.30 reader.profile.delete()**

This function deletes the specified profile from the reader.

**NOTICE****Caution!****Data loss due to wrong operation.**

Once a profile is deleted, it cannot be recovered.

- Only use this function if you are sure that you won't need the specified profile any more.

<b>Type</b>	function
<b>Permissions</b>	guest = - admin = x
<b>Parameters</b>	filename (string)

The following example deletes test\_profile from the reader:

```
reader.profile.delete(test_profile)
ok
```

RFI641

**4.1.31 reader.profile.list()**

This function returns a list of all previously saved reader profiles.

<b>Type</b>	function
<b>Permissions</b>	guest = - admin = x

The following example lists all profiles on the reader:

```
reader.profile.list()
ok profiles = isoc_portal isoc_conveyor isob_portal test_profile
```

**4.1.32 reader.profile.load()**

This function loads the specified profile of reader settings. Once a profile is loaded, it is the active profile (see `reader.profile.active`) and it is the profile loaded at reader startup.

<b>Type</b>	function
<b>Permissions</b>	guest = - admin = x
<b>Parameters</b>	filename (string)

The following example shows the use of this command to load the reader profile named "isoc\_portal". Once the "isoc\_portal" profile is loaded, it is the active profile (see `reader.profile.active`) and it is the profile that will be loaded at reader startup.

```
reader.profile.load(isoc_portal)
ok
```

**4.1.33 reader.profile.reset\_factory\_default()**

This function resets the reader profile to the factory default profile.

<b>Type</b>	function
<b>Permissions</b>	guest = - admin = x

The following example resets the reader to its factory defaults:

```
reader.profile.reset_factory_default()
ok
```

**4.1.34 reader.profile.save()**

This function saves the reader's current configuration to the specified profile. The saved profile is the active profile (see `reader.profile.active`) and is the profile loaded at reader startup.

Profile names must consist of the characters A - Z, a - z, 0 - 9, "-" or "\_" and must be between 1 and 32 characters in length. The reader can store up to 8 different profiles.

<b>Type</b>	function
<b>Permissions</b>	guest = - admin = x
<b>Parameters</b>	filename (string)

The following example saves the reader's current settings to a profile named "isoc\_conveyor". Once the "isoc\_conveyor" profile is saved, it is the active profile (see `reader.profile.active`) and is the profile loaded at reader startup.

```
reader.profile.save(isoc_conveyor)
ok
```

**4.1.35 reader.profile.show\_running\_config()**

This function shows the current configuration of the reader.

<b>Type</b>	function
<b>Permissions</b>	guest = - admin = x

The following shows the readers current running configuration:

```
reader.profile.show_running_config()
ok setup.install_type portal setup.operating_mode standby
setup.protocols isoc setup.region fcc...
```

RFI641

## 4.2 SETUP commands

Command	Description	See Chapter/Page
setup.default_login_level	Sets the default login level used for command sessions.	<a href="#">4.2.1 / 31</a>
setup.install_type	Configures reader installation type.	<a href="#">4.2.2 / 32</a>
setup.operating_mode	Configures reader operating mode.	<a href="#">4.2.3 / 32</a>
setup.protocols	Configures tag protocols.	<a href="#">4.2.4 / 33</a>
setup.region	Configures reader for geographic region.	<a href="#">4.2.5 / 33</a>
setup.sub_region	Configures reader for a sub-region within geographic region.	<a href="#">4.2.6 / 34</a>
setup.sub_region_class	Indicates what class the reader is in (sub region EN302208).	<a href="#">4.2.7 / 35</a>
setup.tag_volume	Configures reader estimated number of tags in field.	<a href="#">4.2.8 / 35</a>
setup.advanced.preferred_frequencies	Sets preferred frequencies to use.	<a href="#">4.2.9 / 36</a>

Table 4-2: Overview: SETUP commands

### 4.2.1 setup.default\_login\_level

This variable sets the login level assigned to all new command sessions.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	enum
<b>Default</b>	GUEST
<b>Enum</b>	GUEST ADMIN UNAUTHENTICATED

The following example sets the default login level for the reader:

```
setup.default_login_level = admin
```

```
ok
```

### 4.2.2 `setup.install_type`

This variable configures the reader for the type of installation. Distance, Power and Modulation Depth are set in the reader based on the `install_type`.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	enum
<b>Default</b>	PORTAL
<b>Enum</b>	PORTAL CONVEYOR_BELT POINT_OF_SALE LABEL_APPLICATOR VEHICLE

The following example configures the reader for portal operation:

```
setup.install_type = portal  
ok
```

### 4.2.3 `setup.operating_mode`

This variable configures the reader to read continuously (Active mode), or to wait for individual tag read commands (Standby mode).

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	enum
<b>Default</b>	Active
<b>Enum</b>	Standby Active

The following example configures the reader to run continuously:

```
setup.operating_mode = active  
ok
```

RFI641

#### 4.2.4 setup.protocols

This variable configures the list of tag protocols supported by the reader.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	enumlist
<b>Default</b>	ISOC
<b>Enum</b>	ISOB ISOC Supertag EASALARM

The following example configures the reader to inventory ISO-B and ISO-C tags:

```
setup.protocols = ISOB ISOC
```

```
ok
```

#### 4.2.5 setup.region

This variable configures the reader for a specific geographic region.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Default</b>	ETSI
<b>Parameters</b>	australia brazil china etsi fcc hongkong india israel japan korea malaysia newzealand singapore southafrica taiwan thailand

**Important** Change of region is only possible for SICK AG Support.

The following example configures the reader for the ETSI regions:

```
setup.region = ETSI
```

```
ok
```

#### 4.2.6 setup.sub\_region

This variable configures the reader for a sub-region within the geographic region specified in "setup.region".

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Datatype</b>	Enum
<b>Default</b>	en302208
<b>Enum</b>	australia australia_dense brazil_band_1 brazil_band_2 brazil_band_1_dense brazil_band_2_dense china china_dense en300220 en302208 en302208_no_lbt en302208_dense fcc_a fcc_b fcc_c fcc_dense fcc_part90 hongkong_band_1 hongkong_band_2 hongkong_band_1_dense hongkong_band_2_dense india india_dense israel israel_dense japan japan_dense korea_mode_1 korea_mode_3 korea_mode_1_dense korea_mode_3_dense malaysia malaysia_dense newzealand newzealand_dense singapore_band_1 singapore_band_2 singapore_band_1_dense singapore_band_2_dense southafrica_band_1 southafrica_band_2 taiwan taiwan_dense thailand thailand_dense

RFI641

The following example configures the reader for a sub-region covered by EN302208 regulations:

```
setup.sub_region = EN302208
```

```
ok
```

#### 4.2.7 setup.sub\_region\_class

This variable indicates the reader's sub region class. It is valid only for sub region EN302208 and must be class 11, 12 or 13.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = r
<b>Data Type</b>	enum
<b>Default</b>	NA
<b>Enum</b>	NA 11 12 13

The following example returns the class when a reader has been configured for sub-region EN302208:

```
setup.sub_region_class
```

```
ok 13
```

#### 4.2.8 setup.tag\_volume

This variable configures the reader for an estimated number of tags in the field.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	enum
<b>Default</b>	1_4
<b>Enum</b>	1 1_4 4_8 8_16 16_32 32_64 64_128 128_256 256_512 512_1024

The following example configures the reader for 200 tags in the field:

```
setup.tag_volume = 128_256
```

```
ok
```

#### 4.2.9 `setup.advanced.preferred_frequencies`

This variable contains a list of the preferred frequencies (in kHz) that the reader should use when the `setup.region` is set to ETSI and the `setup.sub_region` is set to EN 302208.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	list
<b>Default</b>	0
<b>Parameters</b>	865700 865900 866100 866300 866500 866700 866900 867100 867300 867500

If `setup.sub_region` is set to EN 302208\_dense only the parameters 865700, 866300, 866900, 867500 may be selected.

The following example sets the preferred frequencies to 865900, 866300 and 866700 for reader:

```
setup.advanced.preferred_frequencies = 865900 866300 866700
```

```
ok
```

RFI641

### 4.3 INFO commands

Command	Description	See Chapter/Page
info.description	Set the reader's description.	<a href="#">4.3.1 / 37</a>
info.list_time_zones	Lists available time zone strings.	<a href="#">4.3.2 / 37</a>
info.time_zone	Set the actual reader time zone.	<a href="#">4.3.3 / 38</a>
info.location	Sets the reader's location.	<a href="#">4.3.4 / 39</a>
info.name	Sets the reader name (hostname).	<a href="#">4.3.5 / 39</a>
info.serial_number	Returns the reader's serial number.	<a href="#">4.3.6 / 39</a>
info.time	Returns the reader's current local time	<a href="#">4.3.7 / 40</a>
info.time_reporting	Set the time zone the reader timestamps are reported in.	<a href="#">4.3.8 / 40</a>
info.zone	Set the reader's zone.	<a href="#">4.3.9 / 40</a>

Table 4-3: Overview: INFO commands

#### 4.3.1 info.description

This variable contains the assigned description of the reader.

<b>Type</b>	variable
<b>Permissions</b>	guest = rw admin = rw
<b>Data Type</b>	string
<b>Default</b>	"unknown"

The following example sets the description of the reader to "SICK RFI641 set for portal processing":

```
info.description = SICK RFI641 set for portal processing
ok
```

#### 4.3.2 info.list\_time\_zones

This function lists the selectable time zones that can be used in the "info.time\_zone" variable. Output is provided as a directory listing. Those output strings within square brackets "[ ... ]" can be thought of as directories and used as an input into this function for further listings. Those outputs without the square brackets can be used directly in the "info.time\_zone" variable using its relative pathname (e.g. "US/Eastern").

<b>Type</b>	function
<b>Permissions</b>	guest = x admin = x
<b>Parameters</b>	dir (string)

The following example shows a time zone listing:

```
info.list_time_zones()
```

```
ok
CET CST6CDT Cuba EET EST EST5EDT Egypt Eire Factory GB GB-Eire GMT GMT+0
GMT-0 GMT0 Greenwich HST Hongkong Iceland Iran Israel Jamaica Japan
Kwajalein Libya MET MST MST7MDT NZ NZ-CHAT Navajo PRC PST8PDT Poland
Portugal ROC ROK Singapore Turkey UCT UTC Universal W-SU WET Zulu
[Africa] [America] [Antarctica] [Arctic] [Asia] [Atlantic] [Australia]
[Brazil] [Canada] [Chile] [Etc] [Europe] [Indian] [Mexico] [Mideast]
[Pacific] [US]
```

```
info.list_time_zones(US)
```

```
ok
Alaska Aleutian Arizona Central East-Indiana Eastern Hawaii Indiana-
Starke Michigan Mountain Pacific Samoa
```

To set the "info.time\_zone" variable with one of these values, specify the string like you would a directory path to the time zone you want to use. For example, to select the "Eastern" time zone underneath the "US" directory, use "US/Eastern" as the name to specify in the info.time\_zone variable.

```
info.time_zone=US/Eastern
```

```
ok
```

### 4.3.3 info.time\_zone

The variable contains the actual reader time zone (e.g. US/Eastern). This value is used by the info.time\_reporting variable as the LOCAL time zone. When info.time\_reporting is set to LOCAL, the reader reports all times in the time zone specified in this variable. Use info.list\_time\_zones to get a list of valid values to use for this variable.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	string
<b>Default</b>	"GMT"

The following example sets the reporting time to LOCAL and then changes the LOCAL time zone using the info.time\_zone variable to see the effects that change has on the reported time:

```
info.time_reporting = LOCAL
```

```
ok
```

```
info.time_zone = US/Eastern
```

```
ok
```

```
info.time
```

```
ok 2006-06-21T14:11:09.015
```

```
info.time_zone = US/Central
```

```
ok
```

```
info.time
```

```
ok 2006-06-21T13:11:09.015
```

RFI641

#### 4.3.4 info.location

This variable contains the assigned location of the reader.

<b>Type</b>	variable
<b>Permissions</b>	guest = rw admin = rw
<b>Data Type</b>	string
<b>Default</b>	"unknown"

The following example sets the reader location to "Warehouse #3":

```
info.location = Warehouse #3
```

```
ok
```

#### 4.3.5 info.name

This variable contains the assigned name or hostname of the reader.

<b>Type</b>	variable
<b>Permissions</b>	guest = rw admin = rw
<b>Data Type</b>	string
<b>Default</b>	"unknown"

The following example sets the reader name to "Reader #32":

```
info.name = Reader #32
```

```
ok
```

#### 4.3.6 info.serial\_number

This variable contains the reader serial number.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = r
<b>Data Type</b>	string
<b>Default</b>	"Unknown"

The following example returns the reader serial number:

```
info.serial_number
```

```
ok 0F17D4AB93062F56
```

#### 4.3.7 info.time

This variable contains the current time of the reader.

Type	variable
Permissions	guest = rw admin = rw
Data Type	string
Default	"2006-01-01T01:01:01"

The following example returns the reader's current local time:

```
info.time  
ok 2006-06-21T14:06:21.853
```

#### 4.3.8 info.time\_reporting

This variable contains the reader time zones. When set to LOCAL, the reader reports all times in the local time zone. When set to GMT, all times are reported in Greenwich Mean Time (winter time).

Type	variable
Permissions	guest = r, admin = rw
Data Type	enum
Default	LOCAL
Enum	LOCAL GMT

The following example sets the time reporting to GMT and then to LOCAL:

```
info.time_reporting = GMT  
ok  
info.time  
ok 2006-06-21T18:11:02.853Z  
info.time_reporting = LOCAL  
ok  
info.time  
ok 2006-06-21T14:11:09.015
```

#### 4.3.9 info.zone

This variable contains the assigned zone of the reader.

Type	variable
Permissions	guest = rw, admin = rw
Data Type	string
Default	"unknown"

The following example sets the reader zone to "Dock Door #21":

```
info.zone = Dock Door #21  
ok
```

RFI641

## 4.4 COM commands

Command	Description	See Chapter/Page
com.event.overflow_backoff_time	Sets the time in seconds in which the reader will back off (stop sending events) to a flooded client.	<a href="#">4.4.1 / 42</a>
com.network.dns_servers	Sets the DNS servers to be used.	<a href="#">4.4.2 / 42</a>
com.network.discovery_request_address	Sets the Multicast address used for discovery requests	<a href="#">4.4.3 / 42</a>
com.network.discovery_response_address	Sets the Multicast address used for discovery responses	<a href="#">4.4.4 / 43</a>
com.network.domain_list	Sets domain names to search.	<a href="#">4.4.3 / 42</a>
com.network.domainname	Returns the domainname of the reader.	<a href="#">4.4.6 / 43</a>
com.network.hostname	Returns the hostname of the reader.	<a href="#">4.4.7 / 44</a>
com.network.ntp_servers	Specifies NTP servers to be used to synchronise time.	<a href="#">4.4.8 / 44</a>
com.network.ping(ip_address)	Causes the reader to ping another machine.	<a href="#">4.4.9 / 44</a>
com.network.1.default_gateway	Returns default gateway used for first network interface on reader.	<a href="#">4.4.10 / 45</a>
com.network.1.ip_address	Returns the IP address of first network interface on reader.	<a href="#">4.4.11 / 45</a>
com.network.1.mac_address	Returns the MAC address of first network interface on reader.	<a href="#">4.4.12 / 45</a>
com.network.1.method	Returns the method used to acquire IP address.	<a href="#">4.4.13 / 46</a>
com.network.1.set(method,ip_address,subnet_mask,default_gateway)	Specifies the IP acquisition method, IP address, subnet mask and default gateway.	<a href="#">4.4.14 / 46</a>
com.network.1.settings	Sets all configuration data for the first network interface so it can be saved and restored with one variable.	<a href="#">4.4.15 / 47</a>
com.network.1.subnet_mask	Returns the subnet mask used for the first network interface on the reader.	<a href="#">4.4.16 / 47</a>
com.serial.baudrate	Returns the baudrate for serial communication.	<a href="#">4.4.17 / 47</a>
com.serial.console	Sets the program that will run on the serial console	<a href="#">4.4.18 / 48</a>
com.serial.databits	Returns the databits for serial communication.	<a href="#">4.4.19 / 48</a>
com.serial.echo	Returns the echo setting for serial communication.	<a href="#">4.4.20 / 48</a>
com.serial.parity	Returns the parity for serial communication.	<a href="#">4.4.21 / 49</a>
com.serial.rawmode	Sets the mode for serial communication.	<a href="#">4.4.22 / 49</a>
com.serial.set(baudrate,databits, stopbits,parity,echo)	Specifies the reader's serial port settings.	<a href="#">4.4.23 / 49</a>
com.serial.settings	Sets all serial port configuration data so it can be saved and restored with one variable.	<a href="#">4.4.24 / 50</a>
com.serial.stopbits	Returns the number of stopbits for serial communication.	<a href="#">4.4.25 / 50</a>

Table 4-4: Overview: COM commands

#### 4.4.1 com.event.overflow\_backoff\_time

This variable contains the time (in seconds) that the reader will back off or stop sending events to an event channel that cannot handle the current events. After the backoff time has expired, the reader will restart sending events.

Type	variable
Permissions	guest = r admin = rw
Data Type	int
Default	3
Min	0
Max	65536

The following example sets the overflow backoff time to 2 seconds:

```
com.event.overflow_backoff_time = 2
ok
```

#### 4.4.2 com.network.dns\_servers

This variable contains the setting used to set multiple domain name servers on the network for domain name resolution.

Type	variable
Permissions	guest = rw admin = rw
Data Type	string
Default	""

The following example sets two domain servers:

```
com.network.dns_servers = 10.1.1.1 10.1.1.50
ok
```

#### 4.4.3 com.network.discovery\_request\_address

This variable contains the multicast address used for discovery requests. The reader will listen to this address for incoming multicast discovery requests.

**Important** This setting will not take effect until the discovery process is restarted. This can be done by rebooting the reader or by executing "reader.stop\_service(discovery)" followed by "reader.start\_service(discovery)".

Type	variable
Permissions	guest = r admin = rw
Data Type	string
Default	"239.192.1.100"

RFI641

#### 4.4.4 com.network.discovery\_response\_address

This variable contains the multicast address used for discovery responses. The reader will send discovery replies to this multicast address.

**Important** This setting will not take effect until the discovery process is restarted. This can be done by rebooting the reader or by executing `reader.stop_service(discovery)` followed by `reader.start_service(discovery)`.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	string
<b>Default</b>	"239.192.1.101"

#### 4.4.5 com.network.domain\_list

This variable contains multiple domain names/addresses to search.

<b>Type</b>	variable
<b>Permissions</b>	guest = rw admin = rw
<b>Data Type</b>	string
<b>Default</b>	""

The following example sets two domain addresses:

```
com.network.domain_list = 10.1.1.1 10.1.1.50
ok
```

#### 4.4.6 com.network.domainname

This variable contains the reader's domain name.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	string
<b>Default</b>	"Unknown"

The following example returns the domain name of the reader:

```
com.network.domainname
ok Sick.com
```

#### 4.4.7 com.network.hostname

This variable contains the reader's host name.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	string
<b>Default</b>	"Unknown"

The following example returns the host name of the reader:

```
com.network.hostname
```

```
ok Dock_Door_01
```

#### 4.4.8 com.network.ntp\_servers

This variable contains the setting used to synchronise reader time with network time servers. You can enter multiple server addresses separated by a space.

<b>Type</b>	variable
<b>Permissions</b>	guest = rw admin = rw
<b>Data Type</b>	string
<b>Default</b>	""

The following example specifies three server addresses:

```
com.network.ntp_servers=10.1.1.1 10.1.1.2 192.10.34.1
```

```
ok
```

#### 4.4.9 com.network.ping()

This function causes the reader to ping another machine. It is typically used as a diagnostic tool to determine if the reader can connect to another machine.

<b>Type</b>	function
<b>Permissions</b>	guest = - admin = x
<b>Parameters</b>	ip_address (string)

The following example shows a reader can reach a machine at address 10.1.1.244, but cannot reach a machine at address 10.0.0.21:

```
com.network.ping(ip_address = 10.1.1.244)
```

```
ok address is reachable
```

```
com.network.ping(ip_address = 10.0.0.21)
```

```
error.network.address_not_reachable
```

RFI641

**4.4.10 com.network.1.default\_gateway**

This variable contains the default gateway used for the first network interface on the reader.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = r
<b>Data Type</b>	string
<b>Default</b>	""

The following example returns the default gateway of the first network interface on the reader. To set the default gateway, use the `com.network.1.set()` function.

```
com.network.1.default_gateway
```

```
ok 10.1.1.1
```

**4.4.11 com.network.1.ip\_address**

This variable contains the IP address of the first network interface on the reader.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = r
<b>Data Type</b>	string
<b>Default</b>	""

The following example gets the IP address of the first network interface on the reader. To set the IP address, use the `com.network.1.set()` function.

```
com.network.1.ip_address
```

```
ok 192.168.0.1
```

**4.4.12 com.network.1.mac\_address**

This variable contains the MAC address of the first network interface on the reader.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = r
<b>Data Type</b>	string
<b>Default</b>	"00:00:00:00:00:00"

The following example returns the MAC address of the reader:

```
com.network.1.mac_address
```

```
ok 83:F2:01:71:AD:0B
```

**4.4.13 com.network.1.method**

This variable contains the method used to acquire an IP address. This can be static, where the IP address is statically defined for the reader, or DHCP where the IP address is acquired from a DHCP server.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = r
<b>Data Type</b>	enum
<b>Default</b>	STATIC
<b>Enum</b>	DHCP STATIC

The following example returns the method used by the reader to acquire its IP address. To set the method used to acquire an IP address, use the `com.network.1.set()` function.

```
com.network.1.method
```

```
ok static
```

**4.4.14 com.network.1.set()**

This function specifies the IP acquisition method, IP address, subnet mask, and default gateway for the first network interface on the reader. The IP acquisition method can be either static or DHCP.

- If the method is static, the IP address, subnet mask, and default gateway must also be specified in the function call.
- If the IP acquisition method is DHCP, no IP address, subnet mask, or default gateway should be specified in the function call.

<b>Type</b>	function
<b>Permissions</b>	guest = - admin = x
<b>Parameters</b>	method (enum definitions.enum.network.ip_acq_methods) ip_address (string) subnet_mask (string) default_gateway (string)

The following example sets the reader to obtain its IP address via DHCP. The second example sets a static IP address, subnet mask and default gateway.

```
com.network.1.set(method=dhcp)
```

```
ok
```

```
com.network.1.set(method=static, ip_address=192.168.0.1,  
subnet_mask=255.255.255.0, default_gateway=10.1.1.1)
```

```
ok
```

RFI641

**4.4.15 com.network.1.settings**

This variable contains all configuration data used for the first network interface so it can be saved and restored with one variable.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = r
<b>Data Type</b>	string
<b>Default</b>	""

**4.4.16 com.network.1.subnet\_mask**

This variable contains the subnet mask used for the first network interface on the reader.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = r
<b>Data Type</b>	string
<b>Default</b>	""

The following example returns the subnet mask of the first network interface on the reader. To set the subnet mask, use the `com.network.1.set()` function.

```
com.network.1.subnet_mask
```

```
ok 255.255.255.0
```

**4.4.17 com.serial.baudrate**

This variable contains the baud rate for serial communication.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = r
<b>Data Type</b>	enum
<b>Default</b>	115200
<b>Enum</b>	115200 57600 38400 19200 9600 4800 2400 1200

The following example returns the serial port's baud rate. To set the baud rate, use the `com.serial.set()` function.

```
com.serial.baudrate
```

```
ok 115200
```

**4.4.18 com.serial.console**

This function sets the program that will run on the serial console.

Call this function with the NONE parameter to disable running a program on the serial console. Note that if the console is disabled by this command, an ok response is not issued on the serial console.

<b>Type</b>	function
<b>Permissions</b>	guest = - admin = x
<b>Parameters</b>	program

The following example disables running a program on the serial port:

```
com.serial.console(program=none)  
ok
```

**4.4.19 com.serial.databits**

This variable contains the number of databits used for serial communication.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = r
<b>Data Type</b>	enum
<b>Default</b>	8
<b>Enum</b>	7 8

The following example returns the number of data bits used by the serial port. To set the number of data bits, use the com.serial.set() function.

```
com.serial.databits  
ok 8
```

**4.4.20 com.serial.echo**

This variable contains the echo setting used for serial communication.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = r
<b>Data Type</b>	bool
<b>Default</b>	true

The following example returns the echo setting for the serial port. To enable or disable the echo, use the com.serial.set() function.

```
com.serial.echo  
ok true
```

RFI641

#### 4.4.21 com.serial.parity

This variable contains the parity for serial communication.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = r
<b>Data Type</b>	enum
<b>Default</b>	NONE
<b>Enum</b>	NONE EVEN ODD

The following example returns the parity setting for the serial port. To set the parity, use the `com.serial.set()` function.

```
com.serial.parity
```

```
ok none
```

#### 4.4.22 com.serial.rawmode

This variable contains the behaviour mode for the serial console.

- When set to true, a machine interface is presented on the serial console. In this mode no prompts are sent out of the serial port.
- When set to false, a human interface is presented. In this mode, the reader prompt is sent out of the serial port at the end of each command.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	bool
<b>Default</b>	false

The following example sets the serial port to raw mode (machine interface):

```
com.serial.rawmode = true
```

```
ok
```

#### 4.4.23 com.serial.set

This function specifies the reader's serial port setting including: baud rate, data bits, stop bits, parity and echo.

<b>Type</b>	function
<b>Permissions</b>	guest = - admin = x
<b>Parameters</b>	baudrate (enum definitions.enum.serial.baudrates) databits (enum definitions.enum.serial.databits) stopbits (enum definitions.enum.serial.stopbits) parity (enum definitions.enum.serial.parity) echo (bool)

The following example sets the reader's serial port to 57600 baud, 7 data bits, 1 stop bit, even parity, and echo:

```
com.serial.set(baudrate=57600, databits=7, stopbits=1, parity=EVEN,
echo=TRUE)
```

ok

4.4.24 com.serial.settings

This variable contains all serial port configuration data so it can be saved and restored with one variable.

Type	variable
Permissions	guest = r admin = r
Data Type	string
Default	""

4.4.25 com.serial.stopbits

This variable contains the number of stop bits for serial communication.

Type	variable
Permissions	guest = r admin = r
Data Type	enum
Default	1
Enum	1 2

The following example returns the number of stop bits used by the serial port. To set the number of stop bits, use the com.serial.set() function.

```
com.serial.stopbits
```

ok 1

RF1641

## 4.5 TAG Commands

Command	Description	See Chapter/Page
tag.erase(tag_id,pwd,antenna)	Erases tag fields.	<a href="#">4.5.1 / 52</a>
tag.kill(tag_id,kill_pwd,antenna)	Kills a tag.	<a href="#">4.5.2 / 52</a>
tag.lock(lock_fields,lock_type,tag_id, pwd,antenna)	Locks any subset of tag fields.	<a href="#">4.5.3 / 53</a>
tag.lock_access_pwd(lock_type, tag_id,pwd,antenna)	Locks tag access password.	<a href="#">4.5.4 / 53</a>
tag.lock_id(lock_type,tag_id,pwd, antenna)	Locks tag id.	<a href="#">4.5.5 / 54</a>
tag.lock_kill_pwd(lock_type,tag_id, pwd,antenna)	Locks tag kill password.	<a href="#">4.5.6 / 54</a>
tag.lock_tid(lock_type,tag_id,pwd, antenna)	Locks tag TID segment.	<a href="#">4.5.7 / 55</a>
tag.lock_user_data(lock_type,tag_id, pwd,antenna)	Locks tag user data.	<a href="#">4.5.8 / 55</a>
tag.read(report,tag_id,pwd,antenna)	Reads all tag fields.	<a href="#">4.5.9 / 56</a>
tag.read_access_pwd(tag_id,pwd, antenna)	Reads tag access password.	<a href="#">4.5.10 / 57</a>
tag.read_id(antenna)	Reads id of first tag.	<a href="#">4.5.11 / 57</a>
tag.read_kill_pwd(tag_id,pwd, antenna)	Reads tag kill password.	<a href="#">4.5.12 / 58</a>
tag.read_tid(tag_id,pwd,antenna)	Reads tag TID.	<a href="#">4.5.13 / 58</a>
tag.read_user_data(tag_id,pwd, antenna)	Reads tag user data.	<a href="#">4.5.14 / 59</a>
tag.unlock(unlock_fields,tag_id,pwd, antenna)	Unlocks tag fields.	<a href="#">4.5.15 / 59</a>
tag.write(new_tag_id,kill_pwd,access_pwd,tid,user_data,lock_fields, lock_type,tag_id,pwd,antenna)	Writes and lock any subset of tag fields.	<a href="#">4.5.16 / 60</a>
tag.write_access_pwd(access_pwd, lock_type,tag_id,pwd,antenna)	Writes tag access password.	<a href="#">4.5.17 / 61</a>
tag.write_id(new_tag_id,lock_type, tag_id,pwd,antenna)	Writes tag id.	<a href="#">4.5.18 / 62</a>
tag.write_kill_pwd(kill_pwd,lock_type, tag_id,pwd,antenna)	Writes tag kill password.	<a href="#">4.5.19 / 63</a>
tag.write_tid(tid,lock_type,tag_id,pwd,antenna)	Writes tag TID.	<a href="#">4.5.20 / 64</a>
tag.write_user_data(user_data,lock_type,tag_id,pwd, antenna)	Writes tag user data.	<a href="#">4.5.21 / 65</a>
tag.db.get()	Returns all tags in database.	<a href="#">4.5.22 / 66</a>
tag.db.get_and_purge()	Returns all tags and purges the database.	<a href="#">4.5.23 / 66</a>
tag.db.max_count	Sets the number of tags stored in tag database.	<a href="#">4.5.24 / 67</a>
tag.db.purge()	Purges tag database.	<a href="#">4.5.25 / 67</a>
tag.db.scan_tags(ms,antenna)	Returns all tags in the field.	<a href="#">4.5.26 / 68</a>
tag.filter.<n>.enabled	Enables specified filter.<n> = 1 to 8.	<a href="#">4.5.27 / 69</a>
tag.filter. <n>.inclusive	Includes either tags that match or don't match. <n> = 1 to 8.	<a href="#">4.5.28 / 69</a>
tag.filter. <n>.mask	Sets the mask for tag filter. <n> = 1 to 8.	<a href="#">4.5.29 / 69</a>
tag.filter. <n>.name	Sets the name for tag filter. <n> = 1 to 8.	<a href="#">4.5.30 / 70</a>
tag.filter. <n>.pattern	Sets the pattern for tag filter. <n> = 1 to 8.	<a href="#">4.5.31 / 70</a>
tag.reporting.arrive_fields	Sets tag fields reported in event.tag.arrive events.	<a href="#">4.5.32 / 71</a>
tag.reporting.depart_fields	Sets tag fields reported in event.tag.depart events.	<a href="#">4.5.33 / 72</a>
tag.reporting.depart_time	Sets tag detection delay.	<a href="#">4.5.34 / 73</a>
tag.reporting.report_fields	Stets tag fields reported in event.tag.report events.	<a href="#">4.5.35 / 73</a>

Table 4-5: Overview: TAG commands

Command	Description	See Chapter/Page
tag.reporting.report_write_verify	Enables report of verify data in response of tag write command	<a href="#">4.5.36 / 74</a>
tag.reporting.taglist_fields	Sets the tag fields reported in tag list responses.	<a href="#">4.5.37 / 75</a>
tag.reporting.raw_tag_data	Causes the reader to send event.tag.raw events.	<a href="#">4.5.38 / 76</a>

Table 4-5: Overview: TAG commands (contd.)

#### 4.5.1 tag.erase()

This function erases the fields of a tag. All memory locations supported for a particular tag protocol are erased. The response to this function is a response name.

<b>Type</b>	function	
<b>Permissions</b>	guest = x admin = x	
<b>Parameters</b>	tag_id (array)	Indicates the id of the tag to erase. If this parameter is not specified, the first tag found in the field will be the tag that is erased. (optional)
	pwd (array)	Indicates the access password required to operate on a locked tag. (optional; Gen 2 only)
	antenna (list)	Indicates the list of antennas on which the function executes. If no antenna is specified, all antennas will be tried until the function succeeds or no more antennas are available. (optional)

The following example erases a tag:

```
tag.erase(tag_id = 0x306800095EFDDF80000987A5)
```

```
ok
```

#### 4.5.2 tag.kill()

This function kills a tag. The response to this function is a response name.

<b>Type</b>	function	
<b>Permissions</b>	guest = x admin = x	
<b>Parameters</b>	tag_id (array)	Indicates the id of the tag to kill.
	kill_pwd (array)	Indicates the kill password required to kill the tag.
	antenna (list)	Indicates the list of antennas on which the function executes. If no antenna is specified, all antennas are tried until the function succeeds or no more antennas are available. (optional)

The following example kills a tag:

```
tag.kill(tag_id = 0x306800095EFDDF80000987A5, kill_pwd=0x28F5ACD8)
```

```
ok
```

RF1641

### 4.5.3 tag.lock()

This function locks any subset of fields on a tag. The response to this function is a response name.

<b>Type</b>	function	
<b>Permissions</b>	guest = x admin = x	
<b>Parameters</b>	lock_fields (string)	Indicates the set of tag fields to lock.
	lock_type (enum definitions.enum .tag.lock.types)	Indicates the type of lock to apply to the tag's fields.
	tag_id (array)	Indicates the id of the tag to apply the lock. If this parameter is not specified, the first tag found in the field is locked. (optional)
	pwd (array)	Indicates the password required to access a locked tag (optional; Gen 2 only).
	antenna (list)	Indicates the list of antennas on which the function executes. If no antenna is specified, all antennas are tried until the function succeeds or no more antennas are available. (optional)

The following example locks the kill password, access password, and user data of the first tag found in the field:

```
tag.lock(lock_fields = kau, lock_type=secured)
```

```
ok
```

### 4.5.4 tag.lock\_access\_pwd()

This function locks a tag's access password. The response to this function is a response name.

<b>Type</b>	function	
<b>Permissions</b>	guest = x admin = x	
<b>Parameters</b>	lock_type (enum definitions.enum .tag.lock.types)	Indicates the type of lock to apply to the tag's id.
	tag_id (array)	Indicates the id of the tag to apply the lock. If this parameter is not specified, the first tag found in the field is locked. (optional)
	pwd (array)	Indicates the access password required to operate on a locked tag. (optional; Gen 2 only)
	antenna (list)	Indicates the list of antennas on which the function executes. If no antenna is specified, all antennas will be tried until the function succeeds or no more antennas are available. (optional)

The following example locks the access password of a tag  
0x306800095EFDDF80000987A5:

```
tag.lock_access_pwd(tag_id=0x306800095EFDDF80000987A5,  
lock_type=secured)
```

```
ok
```

#### 4.5.5 tag.lock\_id()

This function locks a tag's id. The response to this function is a response name.

<b>Type</b>	function	
<b>Permissions</b>	guest = x admin = x	
<b>Parameters</b>	lock_type (enum definitions.enum .tag.lock.types)	Indicates the type of lock to apply to the tag's id.
	tag_id (array)	Indicates the id of the tag to apply the lock. If this parameter is not specified, the first tag found in the field is locked. (optional)
	pwd (array)	Indicates the access password required to operate on a locked tag. (optional; Gen 2 only)
	antenna (list)	Indicates the list of antennas on which the function executes. If no antenna is specified, all antennas will be tried until the function succeeds or no more antennas are available. (optional)

The following example permanently locks the id of tag 0x306800095EFDDF80000987A5:

```
tag.lock_id(tag_id=0x306800095EFDDF80000987A5,
lock_type=perma_locked)
ok
```

#### 4.5.6 tag.lock\_kill\_pwd()

This function locks a tag's kill password. The response to this function is a response name.

<b>Type</b>	function	
<b>Permissions</b>	guest = x admin = x	
<b>Parameters</b>	lock_type (enum definitions.enum .tag.lock.types)	Indicates the type of lock to apply to the tag's id.
	tag_id (array)	Indicates the id of the tag to apply the lock. If this parameter is not specified, the first tag found in the field is locked. (optional)
	pwd (array)	Indicates the access password required to operate on a locked tag. (optional; Gen 2 only)
	antenna (list)	Indicates the list of antennas on which the function executes. If no antenna is specified, all antennas will be tried until the function succeeds or no more antennas are available. (optional)

The following example secures the kill password of a tag with id of 0x306800095EFDDF80000987A5 and an access password of 0x892D9F0:

```
tag.lock_kill_pwd(tag_id=0x306800095EFDDF80000987A5, pwd=0x892D9F0,
lock_type=secured)
ok
```

RFI641

#### 4.5.7 tag.lock\_tid()

This function locks a tag's TID data. The response to this function is a response name.

<b>Type</b>	function	
<b>Permissions</b>	guest = x admin = x	
<b>Parameters</b>	lock_type (enum definitions.enum .tag.lock.types)	Indicates the type of lock to apply to the tag's id.
	tag_id (array)	Indicates the id of the tag to apply the lock. If this parameter is not specified, the first tag found in the field is locked. (optional)
	pwd (array)	Indicates the access password required to operate on a locked tag. (optional; Gen 2 only)
	antenna (list)	Indicates the list of antennas on which the function executes. If no antenna is specified, all antennas will be tried until the function succeeds or no more antennas are available. (optional)

The following example locks the TID data of the first tag found in the field:

```
tag.lock_tid(lock_type=secured)
```

```
ok
```

#### 4.5.8 tag.lock\_user\_data()

This function locks a tag's user data. The response to this function is a response name.

<b>Type</b>	function	
<b>Permissions</b>	guest = x admin = x	
<b>Parameters</b>	lock_type (enum definitions.enum .tag.lock.types)	Indicates the type of lock to apply to the tag's id.
	tag_id (array)	Indicates the id of the tag to apply the lock. If this parameter is not specified, the first tag found in the field is locked. (optional)
	pwd (array)	Indicates the access password required to operate on a locked tag. (optional; Gen 2 only)
	antenna (list)	Indicates the list of antennas on which the function executes. If no antenna is specified, all antennas will be tried until the function succeeds or no more antennas are available. (optional)

The following example shows the use of this command to permanently unlock the user data of a tag with id of 0x306800095EFDDF80000987A5:

```
tag.lock_user_data(tag_id=0x306800095EFDDF80000987A5,  
lock_type=perma_unsecured)
```

```
ok
```

#### 4.5.9 tag.read()

This function is a generic read function. This can be used to read all fields of the tag. The response to this function is a response name followed by the tag's data. If the requested fields could not be read, an error is returned.

<b>Type</b>	function	
<b>Permissions</b>	guest = x admin = x	
<b>Parameters</b>	report (enumlist definitions.enum .tag.read_fields)	List of fields to read from the tag.
	tag_id (array)	Indicates the id of the tag from which to read data. If this parameter is not specified, the first tag found in the field is read. (optional)
	pwd (array)	Indicates the access password required to read a locked tag. (optional; Gen 2 only)
	antenna (list)	Indicates the list of antennas on which the function executes. If no antenna is specified, all antennas will be tried until the function succeeds or no more antennas are available. (optional)
<b>Return Values</b>	kill_pwd(string) access_pwd(string) tag_id(string) tid(string) user_data(string)	

The following example reads the kill password, access password, and user data from tag 0x306800095EFDDF80000987A5:

```
tag.read (tag_id=0x306800095EFDDF80000987A5, report = kill_pwd
access_pwd user_data)
```

```
ok kill_pwd=0x0D19F572, access_pwd=0xA72975B4,
user_data=0x00FD9619028026000A87A5
```

RFI641

#### 4.5.10 tag.read\_access\_pwd()

This function reads a tag's access password. The response to this function is a response name followed by the tag access password. If the access password is not available in a specific tag type, an error is returned.

<b>Type</b>	function	
<b>Permissions</b>	guest = x admin = x	
<b>Parameters</b>	tag_id (array)	Indicates the id of the tag from which to read data.
	pwd (array)	Indicates the access password required to read a locked tag. (optional; Gen 2 only)
	antenna (list)	Indicates the list of antennas on which the function executes. If no antenna is specified, all antennas will be tried until the function succeeds or no more antennas are available. (optional)

The following example reads the access password from tag 0x306800095EFDDF80000987A5:

```
tag.read_access_pwd(tag_id=0x306800095EFDDF80000987A5)
```

```
ok access_pwd=0xA72975B4
```

#### 4.5.11 tag.read\_id()

This function reads the id of the first tag in the field. The response to this function is a response name followed by the tag id.

<b>Type</b>	function	
<b>Permissions</b>	guest = x admin = x	
<b>Parameters</b>	antenna (list)	Indicates the list of antennas on which the function executes. If no antenna is specified, all antennas will be tried until the function succeeds or no more antennas are available. (optional)

The following example reads the tag identifier of the first tag in the reader's field using antennas 1 and 2:

```
tag.read_id(antenna = 1 2)
```

```
ok tag_id=0x306800095EFDDF80000987A5
```

**4.5.12 tag.read\_kill\_pwd()**

This function reads a tag kill password. The response to this function is a response name followed by the tag's kill password. If the kill password is not available in a specific tag type, an error is returned.

<b>Type</b>	function	
<b>Permissions</b>	guest = x admin = x	
<b>Parameters</b>	tag_id (array)	Indicates the id of the tag from which to read the data.
	pwd (array)	Indicates the access password required to read a locked tag (optional; Gen 2 only).
	antenna (list)	Indicates the list of antennas on which the function executes. If no antenna is specified, all antennas will be tried until the function succeeds or no more antennas are available. (optional)

The following example reads the kill password from tag 0x306800095EFDDF80000987A5:

```
tag.read_kill_pwd(tag_id=0x306800095EFDDF80000987A5)
ok kill_pwd=0x0D19F572
```

**4.5.13 tag.read\_tid()**

This function reads a tag TID. The response to this function is a response name followed by the tag TID data. If the TID data is not available in a specific tag type, an error is returned.

<b>Type</b>	function	
<b>Permissions</b>	guest = x admin = x	
<b>Parameters</b>	tag_id (array)	Indicates the id of the tag from which to read the data.
	pwd (array)	Indicates the access password required to read a locked tag (optional; Gen 2 only).
	antenna (list)	Indicates the list of antennas on which the function executes. If no antenna is specified, all antennas will be tried until the function succeeds or no more antennas are available. (optional)

The following example reads the TID data from tag 0x306800095EFDDF80000987A5:

```
tag.read_tid(tag_id=0x306800095EFDDF80000987A5)
ok tid=0xE2001040
```

RFI641

#### 4.5.14 tag.read\_user\_data()

This function reads a tag's user data. The response to this function is a response name followed by the tag's user data. If the user data segment is not available in a specific tag type, an error is returned.

<b>Type</b>	function	
<b>Permissions</b>	guest = x admin = x	
<b>Parameters</b>	tag_id (array)	Indicates the id of the tag from which to read the data.
	pwd (array)	Indicates the access password required to read a locked tag (optional; Gen 2 only).
	antenna (list)	Indicates the list of antennas on which the function executes. If no antenna is specified, all antennas will be tried until the function succeeds or no more antennas are available. (optional)

The following example shows the use of this command to read the user data from a tag with id of 0x306800095EFDDF80000987A5:

```
tag.read_user_data(tag_id=0x306800095EFDDF80000987A5)
ok user_data=0x00FD9619028026000A87A5
```

#### 4.5.15 tag.unlock()

This function unlocks any subset of tag fields. The response to this function is a response name.

<b>Type</b>	function	
<b>Permissions</b>	guest = x admin = x	
<b>Parameters</b>	unlock_fields(string)	Indicates the set of tag fields to unlock.
	tag_id (array)	Indicates the id of the tag to unlock. If this parameter is not specified, the first tag found in the field will be unlocked. (optional)
	pwd (array)	Indicates the access password required to operate on a locked tag (optional; Gen 2 only).
	antenna (list)	Indicates the list of antennas on which the function executes. If no antenna is specified, all antennas will be tried until the function succeeds or no more antennas are available. (optional)

The following example unlocks the kill password and access password of the first tag found in the field:

```
tag.unlock(unlock_fields = ka)
ok
```

**4.5.16 tag.write()**

This function is a generic write function that writes and optionally locks any subset of fields on a tag. This function includes a verify operation and an "ok" response indicates the data written has been verified. The response to this function is a response name.

<b>Type</b>	function	
<b>Permissions</b>	guest = x admin = x	
<b>Parameters</b>	new_tag_id (array)	Indicates a new id to write to the tag. (optional)
	kill_pwd (array)	Indicates the kill password to write to a tag. (optional)
	access_pwd (array)	Indicates access password to write a tag. (optional)
	tid (array)	Indicates the TID data to write to a tag. (optional)
	user_data (array)	Indicates the user data to write to a tag. (optional)
	lock_fields (string)	Indicates which tag fields should be locked after the data is written. (optional) K= kill_pwd memory A= access_pwd memory U= user_data memory E= EPC memory T= TID memory
	lock_type (enum definitions.enum .tag.lock.types)	Indicates lock type to apply to the lock_fields. (optional)
	tag_id (array)	Indicates id of the tag to write the data. If this parameter is not specified, the first tag found in the field is written. (optional)
	pwd (array)	Indicates the access password required to write a locked tag. (optional; Gen 2 only)
	antenna (list)	Indicates the list of antennas on which the function executes. If no antenna is specified, all antennas are tried until the function succeeds or no more antennas are available. (optional)
<b>Return values</b>	verify_tag_id(string) verify_tid(string) verify_user_data(string) verify_access_pwd(string) verify_kill_pwd(string)	

The following example writes a kill password of 0x12345678, an access password of 0x88776655, and user data of 0x12345678AABBCCDD to a tag 0x306800095EFDDF8000000002.

It also permanently locks the kill password and the access password.

```
tag.write(tag_id=0x306800095EFDDF8000000002, kill_pwd=0x12345678,
access_pwd=0x88776655, user_data=0x12345678AABBCCDD, lock_fields = ka,
lock_type=perma_locked)
```

```
ok
```

RF1641

**4.5.17 tag.write\_access\_pwd()**

This function writes a tag access password. It includes a verify operation and an "ok" response indicates the data written has been verified. The response to this function is a response name.

<b>Type</b>	function	
<b>Permissions</b>	guest = x admin = x	
<b>Parameters</b>	access_pwd (array)	Indicates access password to write a tag.
	lock_type (enum definitions.enum .tag.lock.types)	Indicates lock type to apply to the lock_fields. (optional)
	tag_id (array)	Indicates id of the tag to write the data. If this parameter is not specified, the first tag found in the field is written. (optional)
	pwd (array)	Indicates the access password required to write a locked tag. (optional; Gen 2 only)
	antenna (list)	Indicates the list of antennas on which the function executes. If no antenna is specified, all antennas are tried until the function succeeds or no more antennas are available. (optional)
<b>Return values</b>	verify_access_pwd(string)	

The following example writes an access password of 0x1F8DEA90 to the first tag found on antennas 1 and 3:

```
tag.write_access_pwd(access_pwd = 0x1F8DEA90, antenna = 1 3)
```

```
ok
```

**4.5.18 tag.write\_id()**

This function writes a tag id. This function includes a verify operation, so an "ok" response indicates the tag id has been verified. The response to this function is a response name.

<b>Type</b>	function	
<b>Permissions</b>	guest = x admin = x	
<b>Parameters</b>	new_tag_id (array)	Indicates a new id to write to the tag.
	lock_type (enum definitions.enum .tag.lock.types)	Indicates lock type to apply to the lock_fields. (optional)
	tag_id (array)	Indicates id of the tag to write the data. If this parameter is not specified, the first tag found in the field is written. (optional)
	pwd (array)	Indicates the access password required to write a locked tag (optional; Gen 2 only).
	antenna (list)	Indicates the list of antennas on which the function executes. If no antenna is specified, all antennas are tried until the function succeeds or no more antennas are available. (optional)
<b>Return values</b>	verify_tag_id(string)	

The following example writes a new tag\_id of 0x306800095EFDDF80002AB003 to the first tag found in the field:

```
tag.write_id(new_tag_id = 0x306800095EFDDF80002AB003)
ok
```

RF1641

**4.5.19 tag.write\_kill\_pwd()**

This function writes a tag kill password. This function includes a verify operation and an "ok" response indicates the data written has been verified. The response to this function is a response name.

<b>Type</b>	function	
<b>Permissions</b>	guest = x admin = x	
<b>Parameters</b>	kill_pwd (array)	Indicates the kill password to write to a tag.
	lock_type (enum definitions.enum .tag.lock.types)	Indicates lock type to apply to the lock_fields. (optional)
	tag_id (array)	Indicates id of the tag to write the data. If this parameter is not specified, the first tag found in the field is written. (optional)
	pwd (array)	Indicates the access password required to write a locked tag (optional; Gen 2 only).
	antenna (list)	Indicates the list of antennas on which the function executes. If no antenna is specified, all antennas are tried until the function succeeds or no more antennas are available. (optional)
<b>Return values</b>	verify_kill_pwd(string)	

The following example writes kill password 0x42871904 to tag 0x306800095EFDDF8000000002, using the access pwd 0x75297C1F to lock the kill password:

```
tag.write_kill_pwd(tag_id = 0x306800095EFDDF8000000002, kill_pwd = 0x42871904, access_pwd = 0x75297C1F, lock_type = secured)
```

```
ok
```

**4.5.20 tag.write\_tid()**

This function writes tag TID data. This function includes a verify operation, so an "ok" response indicates the data written has been verified. The response to this function is a response name.

<b>Type</b>	function	
<b>Permissions</b>	guest = x admin = x	
<b>Parameters</b>	tid (array)	Indicates the TID data to write to a tag.
	lock_type (enum definitions.enum .tag.lock.types)	Indicates lock type to apply to the lock_fields. (optional)
	tag_id (array)	Indicates id of the tag to write the data. If this parameter is not specified, the first tag found in the field is written. (optional)
	pwd (array)	Indicates the access password required to write a locked tag. (optional; Gen 2 only)
	antenna (list)	Indicates the list of antennas on which the function executes. If no antenna is specified, all antennas are tried until the function succeeds or no more antennas are available. (optional)
<b>Return values</b>	verify_user_data(string)	

The following example writes TID data 0xE2001040 to tag 0x306800095EFDDF8000000002:

```
tag.write_tid(tag_id = 0x306800095EFDDF8000000002, tid = 0xE2001040)
```

```
ok
```

RF1641

**4.5.21 tag.write\_user\_data()**

This function writes tag user data. This function includes a verify operation and an "ok" response indicates the data written has been verified. The response to this function is a response name.

<b>Type</b>	function	
<b>Permissions</b>	guest = x admin = x	
<b>Parameters</b>	user_data (array)	Indicates the user data to write to a tag.
	lock_type (enum definitions.enum .tag.lock.types)	Indicates lock type to apply to the lock_fields. (optional)
	tag_id (array)	Indicates id of the tag to write the data. If this parameter is not specified, the first tag found in the field is written. (optional)
	pwd (array)	Indicates the access password required to write a locked tag (optional; Gen 2 only).
	antenna (list)	Indicates the list of antennas on which the function executes. If no antenna is specified, all antennas are tried until the function succeeds or no more antennas are available. (optional)
<b>Return values</b>	verify_user_data(string)	

The following example writes user data 0x1234567789a to tag 0x306800095EFDDF8000000002:

```
tag.write_user_data(tag_id = 0x306800095EFDDF8000000002, user_data = 0x1234567789a)
```

```
ok
```

**4.5.22 tag.db.get()**

This function returns all tags stored in the tag database.

<b>Type</b>	function
<b>Permissions</b>	guest = x admin = x
<b>Return Values</b>	tag_id(string) user_data(string) tid(string) first(string) last(string) antenna(int) repeat(int) type(string) max (int)

The following example returns all tags from the tag database in the reader:

**tag.db.get ( )**

```
ok
(tag_id=0x306800095EFDDF8000000002)
(tag_id=0x306800095EFDDF80000040A1)
(tag_id=0x306800095EFDDF80003F7421)
(tag_id=0x306800095EFDDF80000987A5)
```

Various fields can be reported for each tag and are defined with the configuration variable tag.reporting.taglist\_fields.

**4.5.23 tag.db.get\_and\_purge()**

This function returns all the tags stored in the database and then purges the database.

<b>Type</b>	function
<b>Permissions</b>	guest = x admin = x
<b>Return Values</b>	tag_id(string) user_data(string) tid(string) first(string) last(string) antenna(int) repeat(int) type(string)

The following example returns and purges all tags from the tag database:

**tag.db.get\_and\_purge ( )**

```
ok
(tag_id=0x306800095EFDDF8000000002)
(tag_id=0x306800095EFDDF80000040A1)
(tag_id=0x306800095EFDDF80003F7421)
(tag_id=0x306800095EFDDF80000987A5)
```

Various fields can be reported for each tag and are defined with the configuration variable tag.reporting.taglist\_fields.

RFI641

**4.5.24 tag.db.max\_count**

This variable is the maximum number of unique tags stored in tag database.

<b>Type</b>	variable
<b>Permissions</b>	guest = rw admin = rw
<b>Data Type</b>	int
<b>Default</b>	10000
<b>Min</b>	100
<b>Max</b>	10000

The following example sets the reader's tag database to hold 2000 unique tags:

```
tag.db.max_count = 2000
```

```
ok
```

**4.5.25 tag.db.purge()**

This function purges the tag database.

<b>Type</b>	function
<b>Permissions</b>	guest = x admin = x

The following example purges all tags from the reader's tag database:

```
tag.db.purge ( )
```

```
ok
```

#### 4.5.26 tag.db.scan\_tags()

When used in blocking mode (default), this function inventories all tags in the field for the specified number of milliseconds and returns all tags that were found in the field. When used in non-blocking mode, this function inventories all tags in the field (sending event.tag.report events for each tag as it is singulated) for the specified number of milliseconds. At the end of the time interval specified inventoring stops.

When complete, this function generates the event.tag.scan\_tags\_complete event.

<b>Type</b>	function
<b>Permissions</b>	guest = x admin = x
<b>Parameters</b>	ms (int) block (bool) antenna (list)
<b>Return Values</b>	tag_id(string) user_data(string) tid(string) first(string) last(string) antenna(int) repeat(int)

The following example scans for tags in the field for 6 seconds (6000 milliseconds): while blocking (i. e. waiting for the 6 seconds to complete):

```
tag.db.scan_tags()
```

```
ok
(tag_id=0x306800095EFDDF8000000002)
(tag_id=0x306800095EFDDF80000040A1)
(tag_id=0x306800095EFDDF80003F7421)
(tag_id=0x306800095EFDDF80000987A5)
```

The following example scans for tags in the field for 6 seconds (6000 milliseconds) without blocking.

It sees the event.tag.scan\_tags\_complete that is sent at the end of the command and then uses the tag.db.get() command to get the list of tags that were inventoried during the 6 seconds.

```
tag.db.scan_tags(6000, false)
```

```
ok
```

After 6 seconds, the event.tag.scan\_tags\_complete event is seen on an event channel.

```
tag.db.get()
```

```
(tag_id=0x306800095EFDDF8000000002)
(tag_id=0x306800095EFDDF80000040A1)
(tag_id=0x306800095EFDDF80003F7421)
(tag_id=0x306800095EFDDF80000987A5)
```

More fields can be reported for each tag. The fields to be reported with each tag are defined with the configuration variable tag.reporting.taglist\_fields.

RFI641

**4.5.27 tag.filter.<n>.enabled**

This variable enables this filter. <n> = 1 to 8.

<b>Type</b>	variable
<b>Permissions</b>	guest = rw admin = rw
<b>Data Type</b>	bool
<b>Default</b>	FALSE

The following example enables tag filter 1:

```
tag.filter.1.enabled = true
```

```
ok
```

**4.5.28 tag.filter.<n>.inclusive**

This variable indicates to either include tags that match (inclusive) or include tags that do not match (exclusive) the tag filter. <n> = 1 to 8.

<b>Type</b>	variable
<b>Permissions</b>	guest = rw admin = rw
<b>Data Type</b>	bool
<b>Default</b>	TRUE

The following example sets tag filter 1 to include tags that match the tag filter:

```
tag.filter.1.inclusive = true
```

```
ok
```

**4.5.29 tag.filter.<n>.mask**

This variable is the mask (as an array of hex bytes) for the tag filter. <n> = 1 to 8.

<b>Type</b>	variable
<b>Permissions</b>	guest = rw admin = rw
<b>Data Type</b>	array
<b>Default</b>	0x00

The following example sets the mask for tag filter 1 to 0x30FF:

```
tag.filter.1.mask = 0x30FF
```

```
ok
```

**4.5.30 tag.filter.<n>.name**

This variable is the name given to the tag filter. <n> = 1 to 8.

<b>Type</b>	variable
<b>Permissions</b>	guest = rw admin = rw
<b>Data Type</b>	string
<b>Default</b>	""

The following example sets the name for tag filter 1:

```
tag.filter.1.name = pallet filter
```

```
ok
```

**4.5.31 tag.filter.<n>.pattern**

This variable is the pattern (as an array of hex bytes) for the tag filter. <n> = 1 to 8.

<b>Type</b>	variable
<b>Permissions</b>	guest = rw admin = rw
<b>Data Type</b>	array
<b>Default</b>	0x00

The following example sets the pattern for tag filter 1 to 0x3017:

```
tag.filter.1.pattern = 0x3017
```

```
ok
```

RFI641

#### 4.5.32 tag.reporting.arrive\_fields

The reader supports three tag events:

- event.tag.report
- event.tag.arrive
- event.tag.depart

This variable contains the fields to be reported with event.tag.arrive events. Supported fields are:

- tag\_id
- tid
- user\_data
- type
- time
- antenna

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	enumlist
<b>Default</b>	tag_id
<b>Enum</b>	tag_id tid user_data type time antenna rssi

The following example includes the tag\_id, time, and antenna with all event.tag.arrive events:

```
tag.reporting.arrive_fields = tag_id time antenna
```

```
ok
```

After setting the tag.reporting.arrive\_fields to tag\_id time antenna, the event.tag.arrive events will appear as follows:

```
event.tag.arrive tag_id=0x306800095EFDDF8000000002, first=2006-06-22T09:54:34.050, antenna=1
event.tag.arrive tag_id=0x306800095EFDDF80000040A1, first=2006-06-22T09:54:34.063, antenna=2
event.tag.arrive tag_id=0x306800095EFDDF80003F7421, first=2006-06-22T09:54:34.077, antenna=3
event.tag.arrive tag_id=0x306800095EFDDF80000987A5, first=2006-06-22T09:54:34.093, antenna=4
```

**Important** The time field is labeled with the key "first" because this is the first time the reader inventoried this tag.

### 4.5.33 tag.reporting.depart\_fields

The reader supports three tag events:

- event.tag.report
- event.tag.arrive
- event.tag.depart

This variable contains the fields to be reported with event.tag.depart events. Supported fields are:

- tag\_id
- tid
- user\_data
- type
- time
- antenna
- repeat

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	enumlist
<b>Default</b>	tag_id
<b>Enum</b>	tag_id tid user_data type time antenna repeat

The following example includes the tag\_id, time, repeat count, and antenna with all event.tag.depart events:

```
tag.reporting.depart_fields = tag_id time repeat antenna
ok
```

After setting the tag.reporting.depart\_fields to tag\_id time repeat antenna, the event.tag.depart events will appear as follows:

```
event.tag.depart tag_id=0x306800095EFDDF8000000002, last=2006-06-22T09:58:33.638, antenna=2, repeat=14
event.tag.depart tag_id=0x306800095EFDDF80000040A1, last=2006-06-22T09:58:33.674, antenna=3, repeat=14
event.tag.depart tag_id=0x306800095EFDDF80003F7421, last=2006-06-22T09:58:33.686, antenna=4, repeat=14
event.tag.depart tag_id=0x306800095EFDDF80000987A5, last=2006-06-22T09:58:33.731, antenna=1, repeat=15
```

**Important** The time field is labeled with the key "last" because this is the last time the reader inventoried this tag. The repeat field indicates the number of times the reader inventoried this tag.

RFI641

#### 4.5.34 tag.reporting.depart\_time

This variable is the number of milliseconds of not detecting a previously detected tag in the field before generating the event.tag.depart event.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	int
<b>Default</b>	1000
<b>Min</b>	100
<b>Max</b>	25000

The following example sets the time for not detecting a tag to 3 seconds (3,000 milliseconds) before declaring the tag has departed the reader field and sending the event.tag.depart event:

```
tag.reporting.depart_time = 3000
ok
```

#### 4.5.35 tag.reporting.report\_fields

The reader supports 3 tag events:

- event.tag.report
- event.tag.arrive
- event.tag.depart

This variable contains the fields to be reported with event.tag.report events. Supported fields are:

- tag\_id
- tid
- user\_data
- type
- time
- antenna
- freq
- rssi
- tx\_power
- init\_clock\_offset
- ending\_clock\_offset
- quality\_metric

The reported fields are the union of the fields set in:

- the tag.reporting.report\_fields
- the tag.reporting.arrive\_fields
- the tag.reporting.depart\_fields
- the tag.reporting.taglist\_fields.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	enumlist
<b>Default</b>	tag_id
<b>Enum</b>	tag_id tid user_data type time antenna freq rssi tx_power init_clock_offset ending_clock_offset quality_metric

The following example includes the tag\_id, time, tag type and antenna with all event.tag.report events:

```
tag.reporting.report_fields = tag_id time type antenna
ok
```

After setting the tag.reporting.report\_fields to tag\_id time type antenna, the event.tag.report events will appear as follows:

```
event.tag.report tag_id=0x306800095EFDDF8000000002, type=ISOC,
antenna=1, time=2006-06-22T09:47:52.350
event.tag.report tag_id=0x306800095EFDDF80000040A1, type=ISOC,
antenna=2, time=2006-06-22T09:47:52.373
event.tag.report tag_id=0x306800095EFDDF80003F7421, type=ISOC,
antenna=3, time=2006-06-22T09:47:52.407
event.tag.report tag_id=0x306800095EFDDF80000987A5, type=ISOC,
antenna=4, time=2006-06-22T09:47:52.420
```

#### 4.5.36 tag.reporting.report\_write\_verify

This variable controls whether the read back verification results are reported as part of a "write" command result. If set to false, only "ok" or error are reported for the write result. Write commands will typically perform a "read" operation to verify that the data was written correctly to the tag. This variable will control whether the result of the "read" is reported back as part of the "write" command result.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	bool
<b>Default</b>	false

RF1641

The following example enables reporting of the verification read data after a write command:

```
tag.reporting.report_write_verify = true
ok
tag.write(kill_pwd=12345678, access_pwd=12345678)
ok
verify_kill_pwd=0x12345678,verify_access_pwd=0x12345678
```

#### 4.5.37 tag.reporting.taglist\_fields

This variable contains the fields reported for each tag in the response to tag list processing commands. Supported fields are:

- tag\_id
- tid
- user\_data
- type
- time
- antenna
- repeat

The tag list processing commands include:

- tag.db.get()
- tag.db.get\_and\_purge()
- tag.db.scan\_tags()

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	enumlist
<b>Default</b>	tag_id
<b>Enum</b>	tag_id tid user_data type time antenna repeat

The following example includes the tag\_id, time, repeat count, and antenna fields for all tags reported in the response to tag list processing commands:

```
tag.reporting.taglist_fields = tag_id time repeat antenna
ok
```

After setting the tag.reporting.taglist\_fields to tag\_id time repeat antenna, a call to tag.db.get() will return the tag list formatted as follows:

```
tag.db.get ( )
```

```
ok
(tag_id=0x306800095EFDDF8000000002, first=2006-06-22T10:07:21.592,
last=2006-06-22T10:07:32.096, antenna=1, repeat=97)
(tag_id=0x306800095EFDDF80000040A1, first=2006-06-22T10:07:21.653,
last=2006-06-22T10:07:32.041, antenna=4, repeat=96)
(tag_id=0x306800095EFDDF80003F7421, first=2006-06-22T10:07:21.640,
last=2006-06-22T10:07:32.039, antenna=3, repeat=96)
(tag_id=0x306800095EFDDF80000987A5, first=2006-06-22T10:07:21.604,
last=2006-06-22T10:07:32.002, antenna=2, repeat=96)
```

**Important** The field labeled "first" indicates the time at which the reader first inventoried the tag. The field labeled "last" indicates the time at which the reader last inventoried the tag. The field labeled "repeat" indicates the number of times the reader inventoried the tag.

#### 4.5.38 tag.reporting.raw\_tag\_data

When set to true, this variable will cause the reader to send event.tag.raw events. This event is generated each time the reader awaits a tag response during inventory, whether successful or not, and contains any data received by the reader, and statistical data associated with the received data. Note that enabling event.tag.raw events will decrease the read rate performance of the reader.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	bool
<b>Default</b>	FALSE

The following example enables event.tag.raw events from the reader:

```
tag.reporting.raw_tag_data = true
```

```
ok
```

After registering to receive event.tag.raw events, the event.tag.raw events will appear as follows:

```
event.tag.raw raw_data=0x300833B2DDD9014035050007,
type=ISOC, antenna=1, ending_clock_offset=0x1FEC9857,
frequency=910750, initial_clock_offset=0x0,
rssi=49836, demod_result=0, quality_metric=995
```

RF1641

## 4.6 DIO Commands

Command	Description	See Chapter/Page
dio.debounce.<n>	Sets the debounce time (ms) for digital input pin.	<a href="#">4.6.1 / 77</a>
dio.in.<n>	Sets digital IO input pin value. <n> = 1 to 4.	<a href="#">4.6.2 / 77</a>
dio.in.all	Sets all digital IO input pin values.	<a href="#">4.6.3 / 78</a>
dio.out.<n>	Sets digital IO output pin value. <n> = 1 to 4.	<a href="#">4.6.4 / 78</a>
dio.out.all	Sets all digital IO output pin values.	<a href="#">4.6.5 / 79</a>
dio.in.alarm.logic_level.<n>	Determines if an alarm (in the form of an event) should be generated.	<a href="#">4.6.6 / 79</a>
dio.in.alarm.timeout.<n>	Determines timeout for monitoring digital input.	<a href="#">4.6.7 / 80</a>

Table 4-6: Overview: DIO commands

### 4.6.1 dio.debounce.<n>

This variable contains the debounce time (in milliseconds) for the digital input pin. <n> = 1 to 4.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	int
<b>Default</b>	30
<b>Min</b>	0
<b>Max</b>	60000

The following example sets the debounce time for digital input pin 1 to 50 ms:

```
dio.debounce.1 = 50
```

```
ok
```

### 4.6.2 dio.in.<n>

This variable contains the digital IO input pin value. This variable is writeable, but will not actually change the value of the input. Writing this variable triggers event.dio.in.\*.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	int
<b>Default</b>	0
<b>Min</b>	0
<b>Max</b>	1

The following example sets digital input pin 1 low, generating event.dio.in.1 value=0:

```
dio.in.1 = 0
```

```
ok
```

### 4.6.3 dio.in.all

This variable contains all digital IO input pin values. This variable is writeable, but will not actually change the value of the input. Writing this variable triggers event.dio.in.\*.

Type	variable
Permissions	guest = r admin = rw
Data Type	int
Default	0
Min	0
Max	15

The following example sets digital input pins 1 and 3 to high, pins 2 and 4 to low. This will generate event.dio.in.n value=x for any pin that changes value and generate event.dio.all in=0x5 out=0x0.

```
dio.in.all = 0x5
```

```
ok
```

### 4.6.4 dio.out.<n>

This variable contains the digital IO output pin value. <n> = 1 to 4.

Type	variable
Permissions	guest = r admin = rw
Data Type	int
Default	0
Min	0
Max	1

The following example sets digital output pin 1 to high, generating event.dio.out.1 value=1:

```
dio.out.1 = 1
```

```
ok
```

RFI641

#### 4.6.5 dio.out.all

This variable contains all the digital IO output pin values.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	int
<b>Default</b>	0
<b>Min</b>	0
<b>Max</b>	15

The following example sets digital output pins 1, 2 and 3 to high, pin 4 to low. This generates event.dio.out.n value=x for any pin that changes value and generates event.dio.all in=0x5 out=0x07.

```
dio.out.all = 0x7
```

```
ok
```

#### 4.6.6 dio.in.alarm.logic\_level.<n>

This variable is used along with the corresponding input pin timeout value to determine if an alarm (in the form of an event) should be generated. If a timeout value is set, the input pin is monitored. If the input pin value does not change during the timeout period and the input pin value matches the alarm logic level, the event "event.dio.in.alarm.timeout.n" (where n is the pin number) is generated. This alarm event generation can be helpful in alerting to the loss of digital inputs to the reader.

<b>Type</b>	variable
<b>Permissions</b>	guest = - admin = rw
<b>Data Type</b>	int
<b>Default</b>	1
<b>Min</b>	0
<b>Max</b>	1

```
dio.in.alarm.logic_level.1 = 1
```

```
ok
```

**4.6.7 dio.in.alarm.timeout.<n>**

This variable is used along with the corresponding input pin logic level to determine if an alarm (in the form of an event) should be generated. If a timeout value is set, the input pin is monitored. If the input pin value does not change during the timeout period and the input pin value matches the alarm logic level, the event "event.dio.in.alarm.timeout.n" (where n is the pin number) is generated. This alarm event generation can be helpful in alerting to the loss of digital inputs to the reader. The maximum timeout value is 600 seconds. A timeout value of 0 disables the alarm event generation.

<b>Type</b>	variable
<b>Permissions</b>	guest= - admin = rw
<b>Data Type</b>	int
<b>Default</b>	0
<b>Min</b>	0
<b>Max</b>	600

```
dio.in.alarm.timeout.1 = 30
```

```
ok
```

RFI641

## 4.7 ANTENNAS Commands

Command	Description	See Chapter/Page
antennas.detected	Returns the list of antenna ports with antennas connected.	<a href="#">4.7.1 / 81</a>
antennas.mux_sequence	Specifies a list of antenna ports.	<a href="#">4.7.2 / 82</a>
antennas.<n>.conducted_power	Transmits power for antenna port.	<a href="#">4.7.3 / 82</a>
antennas.<n>.advanced.attenuation	Sets the attenuation level of the selected port.	<a href="#">4.7.4 / 83</a>
antennas.<n>.advanced.cable_loss	Sets the Cable loss on the selected port.	<a href="#">4.7.5 / 83</a>
antennas.<n>.advanced.computed_conducted_power	Sets computed conducted power on the selected port.	<a href="#">4.7.6 / 84</a>
antennas.<n>.advanced.gain	Sets the gain provided by the antenna attached to the selected port.	<a href="#">4.7.7 / 85</a>
antennas.<n>.advanced.gain_units	Set the units that the gain of the antenna is specified in.	<a href="#">4.7.8 / 85</a>
antennas.check.time	Sets the antenna checks interval.	<a href="#">4.7.9 / 86</a>
antennas.check.type	Determines when antenna checks are performed.	<a href="#">4.7.10 / 86</a>
antennas.lbt.listen_port	Selects antenna to use for LBT listening.	<a href="#">4.7.11 / 87</a>
antennas.lbt.advanced.cable_loss	Sets the cable loss on the specification antenna port.	<a href="#">4.7.12 / 87</a>
antennas.lbt.advanced.gain	Sets the gain provided by the specification antenna.	<a href="#">4.7.13 / 88</a>
antennas.lbt.advanced.gain_units	Sets the units that the gain of the specification antenna is specified in.	<a href="#">4.7.14 / 88</a>
antennas.port_count	Returns the number of antenna ports on the reader.	<a href="#">4.7.15 / 89</a>

Table 4-7: Overview: ANTENNAS commands

### 4.7.1 antennas.detected

This variable contains a list of port numbers. The ports in this list are presumed to have antennas connected, based on the reflected power.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = r
<b>Data Type</b>	list
<b>Default</b>	0

The following example returns the list of antenna ports that have antennas connected. In this example, antennas are connected to ports 1 and 2.

```
antennas.detected
```

```
ok 1 2
```

#### 4.7.2 antennas.mux\_sequence

This variable is a list of antennas used every time an operation is performed on all antennas. The list is used to rearrange the order in which antennas are serviced (3, 4, 1, 2). This list also specifies a subset of antenna ports to use (1, 3) or to increase the frequency that certain antennas are serviced (1, 2, 1, 3, 1, 4). Having a single antenna in this list will result in only that antenna being used for most operations.

Type	variable
Permissions	guest = rw admin = rw
Data Type	list
Default	1

The following example sets the antenna multiplexing sequence to antennas 1, 3 and 2:

```
antennas.mux_sequence = 1 3 2
```

```
ok
```

#### 4.7.3 antennas.<n>.conducted\_power

This variable is the antenna port transmit power during CW (in tenths of dBm). If this value is 0, the reader automatically calculates the transmit power based on the advanced settings for the antenna. <n> = 1 to 4.

Type	variable
Permissions	guest = r admin = rw
Data Type	int
Default	0
Min	0
Max	300
Units	ddBm

The following example sets the transmit power for antenna number 1 to 290 ddBm:

```
antennas.1.conducted_power = 290
```

```
ok
```

RFI641

#### 4.7.4 `antennas.<n>.advanced.attenuation`

This variable is the attenuation level (in tenths of dB) used to lower the transmit power level on a port. `<n>` = 1 to 4.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	int
<b>Default</b>	0
<b>Min</b>	0
<b>Max</b>	400
<b>Units</b>	ddB

The following example sets the attenuation level of antenna number 1 to 75 ddB:

```
antennas.1.advanced.attenuation = 75
```

```
ok
```

#### 4.7.5 `antennas.<n>.advanced.cable_loss`

This variable is the cable loss (in tenths of dB) for the cable(s) connected to this port. `<n>` = 1 to 4

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	int
<b>Default</b>	10
<b>Min</b>	0
<b>Max</b>	100
<b>Units</b>	ddB

The following example specifies the cable loss for antenna 1 as 15 ddB:

```
antennas.1.advanced.cable_loss = 15
```

```
ok
```

**4.7.6 antennas.<n>.advanced.computed\_conducted\_power**

This variable is the computed conducted power (decimal dB) on a port.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = r
<b>Data Type</b>	int
<b>Default</b>	0
<b>Min</b>	0
<b>Max</b>	330 dbic 310 dbdc 280 dbd 301 dbi
<b>Units</b>	ddB

The following example shows the advanced method for specifying the antenna data needed to have the reader set the conducted power for antenna number 1. The last command in the example reads the reader computed conducted power.

```
antennas.1.conducted_power = 0
ok
antennas.1.advanced.gain_units = dBdC
ok
antennas.1.advanced.gain = 50
ok
antennas.1.advanced.cable_loss = 15
ok
antennas.1.advanced.attenuation = 75
ok
antennas.1.advanced.computed_conducted_power
ok 259 <n> = 1 to 4.
```

RFI641

#### 4.7.7 `antennas.<n>.advanced.gain`

This variable is the gain provided by the antenna attached to this port. The units for the gain are specified by the `antennas.X.advanced.gain_units` variable. `<n>` = 1 to 4.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	int
<b>Default</b>	60
<b>Min</b>	-200
<b>Max</b>	200
<b>Units</b>	ddB

The following example specifies the gain for antenna 1 (a circular antenna) as 50 ddB:

```
antennas.1.advanced.gain_units = dBdC
```

```
ok
```

```
antennas.1.advanced.gain = 50
```

```
ok
```

#### 4.7.8 `antennas.<n>.advanced.gain_units`

This variable indicates the units antenna gain is specified. `<n>` = 1 to 4.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	enum
<b>Default</b>	dBdC
<b>Enum</b>	dBdC dBiC dBd dBi

The following example specifies gain units in dBdC for antenna 1:

```
antennas.1.advanced.gain_units = dBdC
```

```
ok
```

#### 4.7.9 `antennas.check.time`

This variable is the interval (in ms), for timed antenna checks. If the `antennas.check.type` is set to `TIMED`, this parameter determines the time interval. If a new antenna port is selected and it is checked in this amount of time, an antenna check is performed.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	int
<b>Default</b>	1000
<b>Min</b>	0
<b>Max</b>	65536
<b>Units</b>	ms

The following example sets the antenna checking interval to 2,000 ms:

```
antennas.check.type = timed
```

```
ok
```

```
antennas.check.time = 2000
```

```
ok
```

#### 4.7.10 `antennas.check.type`

This variable determines when the reader checks for an antenna:

- Whenever the antenna port is changed (`ALWAYS`)
- First time a port is used (`FIRST_TIME_ONLY`)
- When a port is selected and hasn't been checked for a given period of time (`TIMED`)

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	enum
<b>Default</b>	<code>ALWAYS</code>
<b>Enum</b>	<code>FIRST_TIME_ONLY</code> <code>TIMED</code> <code>ALWAYS</code>

The following example sets the antenna checking to the first time an antenna port is used:

```
antennas.check.type = first_time_only
```

```
ok
```

RFI641

**4.7.11 antennas.lbt.listen\_port**

This variable selects the antenna to use for LBT listening. LBT is only used in the ETSI region.

<b>Type</b>	variable	
<b>Permissions</b>	guest = r admin = rw	
<b>Data Type</b>	enum	
<b>Default</b>	LBT_ANTENNA_ALL	
<b>Enum</b>	LBT_ANTENNA1	Uses specified antenna for listening.
	LBT_ANTENNA2	
	LBT_ANTENNA3	
	LBT_ANTENNA4	
	LBT_ANTENNA_ALL	Uses current active (transmit) antenna for listening.
	LBT_DEDICATED_ANTENNA	Uses LBT antenna for listening.

The following example specifies antenna 4 as the LBT antenna:

```
antennas.lbt.listen_port = LBT_ANTENNA4
ok
```

**4.7.12 antennas.lbt.advanced.cable\_loss**

This variable is the loss (in decimal dB) for cable(s) connected to the LBT antenna port.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	int
<b>Default</b>	10
<b>Min</b>	0
<b>Max</b>	100
<b>Units</b>	ddB

The following example specifies the cable loss for the LBT antenna as 12 ddB:

```
antennas.lbt.advanced.cable_loss = 12
ok
```

#### 4.7.13 `antennas.lbt.advanced.gain`

This variable is the gain provided by the LBT antenna. The units for the gain are specified by the `antennas.lbt.advanced.gain_units` variable.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	int
<b>Default</b>	60
<b>Min</b>	-200
<b>Max</b>	200
<b>Units</b>	ddB

The following example specifies the gain for the LBT antenna (a circular antenna) as 40 ddB:

```
antennas.lbt.advanced.gain_units = dBdC
ok
antennas.lbt.advanced.gain = 40
ok
```

#### 4.7.14 `antennas.lbt.advanced.gain_units`

This variable indicates the gain for the LBT antenna.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	enum
<b>Default</b>	dBdC
<b>Enum</b>	dBdC dBiC dBd dBi

The following example specifies LBT antenna gain units in dBdC:

```
antennas.lbt.advanced.gain_units = dBdC
ok
```

RFI641

#### 4.7.15 antennas.port\_count

This variable is the number of antennas that can be attached to the reader. This number does not include the dedicated listening antenna (LBT) that may be present on some ETSI targeted platforms.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = r
<b>Data Type</b>	int
<b>Default</b>	4
<b>Min</b>	1
<b>Max</b>	8

The following example returns the number of antenna ports on the reader:

```
antennas.port_count  
ok 4
```

Notes:

RFI641

## 4.8 MODEM Commands

Command	Description	See Chapter/Page
modem.antennas.perform_check()	Checks all 4 antenna ports, to determine, if an antenna is connected.	<a href="#">4.8.1 / 93</a>
modem.control.inventory.perform_rounds(rounds, block)	Performs the specified number of inventory rounds.	<a href="#">4.8.2 / 93</a>
modem.protocol.cmd_retries	Returns number of command retries for failed command.	<a href="#">4.8.3 / 94</a>
modem.protocol.isob.lock(tag_id, address, antenna)	Locks specified byte of an ISO-B tag.	<a href="#">4.8.4 / 94</a>
modem.protocol.isob.query_lock(tag_id, address, antenna)	Determines if specified byte of an ISO-B tag is locked.	<a href="#">4.8.5 / 95</a>
modem.protocol.isob.read(tag_id, address, antenna)	Reads specified data from an ISO-B tag.	<a href="#">4.8.6 / 95</a>
modem.protocol.isob.write(tag_id, address, data, antenna)	Writes specified byte to an ISO-B tag.	<a href="#">4.8.7 / 96</a>
modem.protocol.isob.control.auto_quiet	Enables auto quiet feature for ISO-B tags.	<a href="#">4.8.8 / 96</a>
modem.protocol.isob.control.cmd_retries	Sets the number of command retries for failed command.	<a href="#">4.8.9 / 96</a>
modem.protocol.isob.control.unicode_epc_mode.enable	Enables EPC format support on UCODE ISO-B tag.	<a href="#">4.8.10 / 97</a>
modem.protocol.isob.control.unicode_epc_mode.length	Sets EPC length for EPC enabled UCODE tags.	<a href="#">4.8.11 / 97</a>
modem.protocol.isob.filter.<n>.address	Writes start address for Select memory comparison.	<a href="#">4.8.12 / 97</a>
modem.protocol.isob.filter.<n>.data	Writes data bytes to compare during Select.	<a href="#">4.8.13 / 97</a>
modem.protocol.isob.filter.<n>.enabled	Enables the ISO-B mask filter.	<a href="#">4.8.14 / 98</a>
modem.protocol.isob.filter.<n>.mask	Writes mask of which bytes to compare during Select.	<a href="#">4.8.15 / 98</a>
modem.protocol.isob.filter.<n>.opcode	Selects the command used to inventory tags.	<a href="#">4.8.16 / 100</a>
modem.protocol.isob.physical.modulation_depth	Specifies the percent modulation depth for ISO-B.	<a href="#">4.8.17 / 100</a>
modem.protocol.isob.physical.return_link_freq	Returns link frequency from tag to reader.	<a href="#">4.8.18 / 100</a>
modem.protocol.isoc.block_erase(tag_id, password, mem_bank, word_ptr, word_count, antenna)	Erases specified section of an ISO-C tag.	<a href="#">4.8.19 / 101</a>
modem.protocol.isoc.conf_test(command, antenna)	Generates the appropriate ISO-C conformance test commands.	<a href="#">4.8.20 / 101</a>
modem.protocol.isoc.control.crc_retries	Returns number of command retries for failed command.	<a href="#">4.8.21 / 102</a>
modem.protocol.isoc.control.enable_per_antenna	Enables per antenna settings for ISO-C.	<a href="#">4.8.22 / 102</a>
modem.protocol.isoc.control.verify_write	Enables read verify of ISO-C write commands.	<a href="#">4.8.23 / 102</a>
modem.protocol.isoc.control.antenna.<n>.session_id	Sets the ISO-C control settings for antenna <n>.	<a href="#">4.8.24 / 103</a>
modem.protocol.isoc.read(tag_id, pwd, mem_bank, word_ptr, word_count, antenna)	Reads specified data from an ISO-C tag.	<a href="#">4.8.25 / 103</a>
modem.protocol.isoc.write(tag_id, pwd, mem_bank, word_ptr, data, antenna)	Writes specified data to an ISO-C tag.	<a href="#">4.8.26 / 104</a>
modem.protocol.isoc.control.cmd_retries	Sets the number of command retries for failed command.	<a href="#">4.8.27 / 104</a>
modem.protocol.isoc.control.display_tag_crc	Enables the display of tag CRC.	<a href="#">4.8.28 / 105</a>
modem.protocol.isoc.control.include_alllock_bits	Includes all lock bits in lock procedure.	<a href="#">4.8.29 / 105</a>
modem.protocol.isoc.control.include_permaunlock_bit	Includes the permaunlock bit in lock procedure.	<a href="#">4.8.30 / 105</a>
modem.protocol.isoc.control.inventory_antenna_switch	Allows antenna switching during inventory.	<a href="#">4.8.31 / 105</a>
modem.protocol.isoc.control.inventory_both_targets	Enables inventory of tags for both inventoried flag targets.	<a href="#">4.8.32 / 106</a>

Table 4-8: Overview: MODEM commands

Command	Description	See Chapter/Page
modem.protocol.isoc.control.max_incr_slots_q	Sets Max Q = number_slots_Q + max_incr_slots_Q.	<a href="#">4.8.33 / 106</a>
modem.protocol.isoc.control.mem_bank_for_selection	Sets the memory bank to which select command applies.	<a href="#">4.8.34 / 106</a>
modem.protocol.isoc.control.number_slots_q	Sets the Initial number of slots in inventory round.	<a href="#">4.8.35 / 107</a>
modem.protocol.isoc.control.query_sel	Sets the ISO-C sel for the query command.	<a href="#">4.8.36 / 107</a>
modem.protocol.isoc.control.query_session	Sets the ISO-C session for the query command.	<a href="#">4.8.37 / 108</a>
modem.protocol.isoc.control.query_target	Sets the ISO-C target for the query command.	<a href="#">4.8.38 / 108</a>
modem.protocol.isoc.control.select_cmd_period	Sets the period of sending select command at start of inventory.	<a href="#">4.8.39 / 109</a>
modem.protocol.isoc.control.session_id	Sets the ISO-C Session to use for tag communication.	<a href="#">4.8.40 / 109</a>
modem.protocol.isoc.control.slots_restart	Sets the number of slots before a round is restarted.	<a href="#">4.8.41 / 109</a>
modem.protocol.isoc.control.use_block_write	Enables ISO-C block write capability.	<a href="#">4.8.42 / 110</a>
modem.protocol.isoc.filter.<n>.action	Sets action a tag takes based on filter match.	<a href="#">4.8.43 / 110</a>
modem.protocol.isoc.filter.<n>.enabled	Enables the ISO-C mask filter.	<a href="#">4.8.44 / 111</a>
modem.protocol.isoc.filter.<n>.length	Specifies the bit length of the ISO-C mask filter.	<a href="#">4.8.45 / 111</a>
modem.protocol.isoc.filter.<n>.mask	Specifies the bit mask for the ISO-C mask filter.	<a href="#">4.8.46 / 111</a>
modem.protocol.isoc.filter.<n>.mem_bank	Specifies the memory bank for the ISO-C mask filter.	<a href="#">4.8.47 / 112</a>
modem.protocol.isoc.filter.<n>.offset	Specifies the bit offset for the ISO-C mask filter.	<a href="#">4.8.48 / 112</a>
modem.protocol.isoc.filtering.enabled	Enables the ISO-C mask filter.	<a href="#">4.8.49 / 112</a>
modem.protocol.isoc.filtering.truncated_epc_response	Turns on ISO-C truncation.	<a href="#">4.8.50 / 112</a>
modem.protocol.isoc.filtering.truncated_tag_epc_length	Indicates the expected tag EPC length for truncation.	<a href="#">4.8.51 / 113</a>
modem.protocol.isoc.filtering.use_session	Targets the tag session inventoried flag for ISO-C filter.	<a href="#">4.8.52 / 113</a>
modem.protocol.isoc.nxp.calibrate	Performs NXP custom command, Calibrate.	<a href="#">4.8.53 / 114</a>
modem.protocol.isoc.nxp.change_eas	Performs NXP custom command, ChangeEAS.	<a href="#">4.8.54 / 114</a>
modem.protocol.isoc.nxp.eas_alarm	Performs NXP custom command, EASAlarm.	<a href="#">4.8.55 / 114</a>
modem.protocol.isoc.nxp.read_protect	Performs NXP custom command, ReadProtect.	<a href="#">4.8.56 / 115</a>
modem.protocol.isoc.nxp.reset_read_protect	Performs NXP custom command, ResetReadProtect.	<a href="#">4.8.57 / 115</a>
modem.protocol.isoc.physical.data_1_length	Specifies the length of a Data 1 symbol for ISO-C.	<a href="#">4.8.58 / 115</a>
modem.protocol.isoc.physical.interrogator_mode	Sets the reader mode for ISO-C.	<a href="#">4.8.59 / 116</a>
modem.protocol.isoc.physical.modulation_depth	Specifies the modulation depth percentage for ISO-C.	<a href="#">4.8.60 / 116</a>
modem.protocol.isoc.physical.pilot_tone	Turns on ISO-C pilot_tone from tag.	<a href="#">4.8.61 / 116</a>
modem.protocol.isoc.physical.return_link_freq	Returns link frequency from tag to reader.	<a href="#">4.8.62 / 117</a>
modem.protocol.isoc.physical.rt_modulation	Sets the reader-to-tag modulation type.	<a href="#">4.8.63 / 117</a>
modem.protocol.isoc.physical.set(tari,data_1_length,return_link_freq,interrogator_mod,pilot_tone,tr_encoding,rt_modulation,modulation_depth)	Sets ISO-C physical link parameters.	<a href="#">4.8.64 / 118</a>
modem.protocol.isoc.physical.settings	Sets ISO-C physical layer settings so it can be saved and restored with one variable.	<a href="#">4.8.65 / 118</a>
modem.protocol.isoc.physical.tari	Specifies the ISO-C Tari value.	<a href="#">4.8.66 / 118</a>
modem.protocol.isoc.physical.tr_encoding	Setes the tag-to-reader encoding format.	<a href="#">4.8.67 / 119</a>
modem.protocol.supertag.control.display_tag_crc	Enables display of tag CRC.	<a href="#">4.8.68 / 119</a>

Table 4-8: Overview: MODEM commands (contd.)

RFI641

Command	Description	See Chapter/Page
modem.protocol.supertag.control.dwell_time	Sets the time (milliseconds) per round that the reader searches for Supertag tags.	<a href="#">4.8.69 / 119</a>
modem.protocol.supertag.control.enable_crc_check	Enables CRC checking of tag response.	<a href="#">4.8.70 / 119</a>
modem.protocol.supertag.physical.return_link_freq	Sets the return link frequency from tag to reader.	<a href="#">4.8.71 / 120</a>
modem.radio.idle_cw	Sets the reader to transmits CW during idle time.	<a href="#">4.8.72 / 120</a>
modem.radio.rf_survey(rbw,span,time,coherent,center_freq)	Performs a spectrum analysis to survey RF environment.	<a href="#">4.8.75 / 121</a>

Table 4-8: Overview: MODEM commands (contd.)

#### 4.8.1 modem.antennas.perform\_check

This function performs a check on all 4 antenna ports, to determine if an antenna is connected. It returns a list of ports with antennas connected. This function will temporarily disrupt any current RFID transactions.

<b>Type</b>	function
<b>Permissions</b>	guest = - admin = x

#### 4.8.2 modem.control.inventory.perform\_rounds()

This function is used to perform the number of inventory rounds specified by the "rounds" parameter. If "block" is true or unspecified, the function blocks until the rounds are complete before returning. If "block" is false, the function returns immediately and does not block. In either case, an "event.status.inventory\_rounds\_complete" event is generated at the completion of the inventory rounds.

<b>Type</b>	function
<b>Permissions</b>	guest = - admin = x
<b>Parameters</b>	rounds (int) block (bool)
<b>Default</b>	block
<b>Min</b>	1
<b>Max</b>	65536

### 4.8.3 modem.protocol.cmd\_retries

This variable sets the number of command retries for a failed command. When a command sequence fails, it can be retried for the number of times indicated by this parameter. This is transparent to the user. Each protocol may also have a cmd\_retries variable for retrying at the air interface command level.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	int
<b>Default</b>	3
<b>Min</b>	0
<b>Max</b>	8

### 4.8.4 modem.protocol.isob.lock()

This function locks the specified byte of an ISO-B tag in the field. The response to this function is a response name.

<b>Type</b>	function	
<b>Permissions</b>	guest = x admin = x	
<b>Parameters</b>	tag_id (array)	Indicates the id of the tag to read. The tag is first singularized and put in a state for a read operation.
	address (int)	Specifies the address (byte offset) of the byte to be locked on the tag.
	antenna (int)	Indicates antenna on which to execute the function. If no antenna is specified, all antennas will be tried until the function succeeds or no more antennas are available. (optional)

The following example locks the byte at address 202 on a tag with id = 0x301122334455667788:

```
modem.protocol.isob.lock (tag_id = 0x301122334455667788, address = 202)
```

```
ok
```

RFI641

#### 4.8.5 modem.protocol.isob.query\_lock()

This function determines if the specified byte of an ISO-B tag in the field is locked. The response to this function is a response name followed by the lock status of the byte.

<b>Type</b>	function	
<b>Permissions</b>	guest = x admin = x	
<b>Parameters</b>	tag_id (array)	Indicates the id of the tag to read. The tag is first singulated and put in a state for a read operation.
	address (int)	Specifies the address (byte offset) of the byte to be locked on the tag.
	antenna (int)	Indicates antenna on which to execute the function. If no antenna is specified, all antennas will be tried until the function succeeds or no more antennas are available. (optional)

The following example gets the lock status of the byte at address 202 on tag 0x301122334455667788:

```
modem.protocol.isob.query_lock (tag_id = 0x301122334455, address = 202)
```

```
ok byte = locked
```

#### 4.8.6 modem.protocol.isob.read()

This function causes the modem to read specified data from an ISO-B tag in the field. This function reads 8 bytes, starting at the address specified from the tag. The response to this function is a response name, followed by the data read from the tag.

<b>Type</b>	function	
<b>Permissions</b>	guest = x admin = x	
<b>Parameters</b>	tag_id (array)	Indicates the id of the tag to be read. The tag is first singulated and put in a state for a read operation.
	address (int)	Specifies the start address of the data to read from the tag.
	antenna (int)	Indicates antenna on which to execute the function. If no antenna is specified, all antennas will be tried until the function succeeds or no more antennas are available. (optional)

The following example reads eight bytes of data at address 2 from a tag with ID = 0x301122334455667788:

```
modem.protocol.isob.read (tag_id = 0x301122334455667788, address = 2)
```

```
ok data = 0x9988776655443322
```

**4.8.7 modem.protocol.isob.write()**

This function writes a specified byte to an ISO-B tag in the field. It writes a single byte, to the specified address. The response to this function is a response name. If the byte is locked, the function will fail with a `error.tag.tag_not_writable` response.

<b>Type</b>	function	
<b>Permissions</b>	guest = x admin = x	
<b>Parameters</b>	tag_id (array)	Indicates the ID of the tag to read. The tag is first singularized and put in a state for a read operation.
	address (int)	Specifies the address (byte offset) of the byte to write to tag.
	data (int)	Specifies the byte of data to be written to the tag.
	antenna (int)	Indicates antenna on which to execute the function. If no antenna is specified, all antennas will be tried until the function succeeds or no more antennas are available. (optional)

The following example writes the byte 0xe4 at address 23 on a tag with ID = 0x301122334455667788:

```
modem.protocol.isob.write (tag_id = 0x30112233445566, data = 0xe4,
address = 23)
```

ok

**4.8.8 modem.protocol.isob.control.auto\_quiet()**

This variable enables the auto quiet feature for ISO-B tags. When set to True, tags that have been inventoried will remain quiet after having been identified. When set to False, tags are reset for every inventory cycle.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	bool
<b>Default</b>	False

**4.8.9 modem.protocol.isob.control.cmd\_retries**

This variable sets the number of command retries for a failed command. When a command sequence fails, it can be retried for the number of times indicated by this parameter. This is transparent to the user.

<b>Type</b>	variable
<b>Permissions</b>	guest = - admin = rw
<b>Data Type</b>	int
<b>Default</b>	3
<b>Min</b>	0
<b>Max</b>	8

RFI641

**4.8.10 modem.protocol.isob.control.unicode\_epc\_mode.enable**

This variable enables support for the EPC format on UCODE ISO-B tags. Philips UCODE ISO-B tags have a memory configuration that allows for compatibility with EPC 64 and 96 bit formats. When enabled, this variable displays the tag memory in the EPC format. When not enabled, an inventory of the tag will display the memory as stored on the tag.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	bool
<b>Default</b>	False

**4.8.11 modem.protocol.isob.control.unicode\_epc\_mode.length**

This variable sets EPC length (64 or 96 bit) for UCODE tags enabled for EPC mode. The EPC length will determine what tag ID is displayed on inventory.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	enum
<b>Default</b>	UCODE_LEN_64
<b>Enum</b>	UCODE_LEN_64 UCODE_LEN_96

**4.8.12 modem.protocol.isob.filter.<n>.address**

This variable sets the Start address for Select memory comparison. When a SELECT\_EQ, SELECT\_NE, SELECT\_GT, or SELECT\_LT command is used, this variable contains the address on the tag for where the comparison should start. Not used when FLAGS are compared. Up to 8 bytes starting at address will be compared. N = 1 to 8.

<b>Type</b>	variable
<b>Permissions</b>	guest = - admin = rw
<b>Data Type</b>	int
<b>Default</b>	0
<b>Min</b>	0
<b>Max</b>	255

**4.8.13 modem.protocol.isob.filter.<n>.data**

This variable contains the data bytes used in comparison for Select command. These are the data bytes that a tag compares against its memory location that begins at address. Only tags that match the bytes will respond in an inventory round. A maximum of 8 bytes can

be used in the comparison. A byte comparison is enabled by setting the corresponding bit in mask. For FLAGS Select, the first data byte (MSByte) is used for comparison against the flag data. N = 1 to 8.

<b>Type</b>	variable
<b>Permissions</b>	guest = - admin = rw
<b>Data Type</b>	array
<b>Default</b>	0x00

#### 4.8.14 modem.protocol.isob.filter.<n>.enabled

This variable enables a specific ISO-B mask filter. When set to true, only those ISO-B tags that match the filter mask will respond. The filter mask may be a bit pattern which starts at a specific address in the tag. N = 1 to 8.

<b>Type</b>	variable
<b>Permissions</b>	guest = - admin = rw
<b>Data Type</b>	bool
<b>Default</b>	false

#### 4.8.15 modem.protocol.isob.filter.<n>.mask

This variable selects 0 to 8 bytes to be compared during the Select process. When a SELECT\_EQ, SELECT\_NE, SELECT\_GT, or SELECT\_LT command is used, this parameter is a bit mask that determines which bytes are compared. N = 1 to 8.

A set bit means compare the byte in filter.x.data with tag memory. If a FLAGS Select is used, then a set bit means compare the bit from the first byte of mask with the corresponding FLAG.

Some usage for data comparison:

mask = 0x00 compare no data

mask = 0x80 compare MSByte of the eight bytes from filter.1.data

mask = 0x40 compare next MSByte of the eight bytes from filter.1.data

mask = 0x81 compare MSByte and LSByte of the eight bytes from filter.1.data

mask = 0xff compare all of the eight bytes from filter.1.data

Some usage for flag comparison:

mask = 0x00 compare no flags

mask = 0x80 compare msb of tags FLAG byte with msb of filter.1.data MSByte

mask = 0x40 compare next msb of tags FLAG byte with next msb of filter.1.data MSByte

mask = 0xff compare all bits of tags FLAG byte with the MSByte of filter.1.data

<b>Type</b>	variable
<b>Permissions</b>	guest = - admin = rw
<b>Data Type</b>	int
<b>Default</b>	0
<b>Min</b>	0
<b>Max</b>	255

RFI641

The following example inventories only tags who have a value of 0x3344 at offset 0x20 in tag memory. The mask of 0xC0 indicates to only compare the 2 MSBytes of the data at address.

```
modem.protocol.isob.filter.1.opcode = SELCET_EQ
ok
modem.protocol.isob.filter.1.address = 0x20
ok
modem.protocol.isob.filter.1.mask = 0xC0
ok
modem.protocol.isob.filter.1.data = 0x3344000000000000
ok
modem.protocol.isob.filtering.enabled = true
ok
```

The following example inventories only tags whose DE\_SB Flag bit is set:

```
modem.protocol.isob.filter.1.opcode = SELCET_EQ_FLAGS
ok
modem.protocol.isob.filter.1.address = 0x00
ok
modem.protocol.isob.filter.1.mask = 0x01
ok
modem.protocol.isob.filter.1.data = 0x01
ok
modem.protocol.isob.filtering.enabled = true
ok
```

**4.8.16 modem.protocol.isob.filter.<n>.opcode**

This variable selects the command to use for tag inventory. The ISO-B protocol uses various Select commands to determine which tags will respond during an Inventory. Tags whose memory matches the data pattern (EQ), does not equal (NE), is greater than (GT) or less than (LT) can be selected. Or the FLAGS field of the tag can be used for comparison (EQ\_FLAGS, NE\_FLAGS). N = 1 to 8.

<b>Type</b>	variable
<b>Permissions</b>	guest = - admin = rw
<b>Data Type</b>	enum
<b>Default</b>	SELCET_EQ_FLAGS
<b>Enum</b>	SELECT_EQ SELECT_NE SELECT_GT SELECT_LT UNSELECT_EQ UNSELECT_NE UNSELECT_GT UNSELECT_LT SELECT_EQ_FLAGS SELECT_NE_FLAGS UNSELECT_EQ_FLAGS UNSELECT_NE_FLAGS

**4.8.17 modem.protocol.isob.physical.modulation\_depth**

This variable specifies the percent modulation depth for ISO-B transmissions.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	int
<b>Default</b>	20
<b>Min</b>	0
<b>Max</b>	100

**4.8.18 modem.protocol.isob.physical.return\_link\_freq**

This variable sets the return link frequency for the tag to reader communication.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	enum
<b>Default</b>	LF160
<b>Enum</b>	LF160

RF1641

**4.8.19 modem.protocol.isoc.block\_erase**

This function erases the specified section of an ISO-C tag in the field. The response to this function is a response name.

<b>Type</b>	function	
<b>Permissions</b>	guest = x admin = x	
<b>Parameters</b>	tag_id (array)	Indicates the ID of the tag to erase. If the tag_id is not supplied, the first tag found will be erased. The tag is first singulated and put in a state for an erase operation. (optional)
	password (array)	Password used to erase locked portions of a tag. (optional)
	mem_bank (int)	Specifies which tag memory bank is to be erased (0 = Reserved, 1 = EPC, 2 = TID, 3 = User memory).
	word_ptr (int)	Specifies the starting word (16 bit) address in the memory bank for the erase.
	word_count (int)	Specifies the number of 16-bit words to erase.
	antenna (int)	Indicates antenna on which to execute the function. If no antenna is specified, all antennas will be tried until the function succeeds or no more antennas are available. (optional)

The following example erases one word at offset 4 from the epc memory section of a tag ID = 0x30112233445566778899aabb:

```
modem.protocol.isoc.block_erase (tag_id = 0x301122aabb, mem_bank = 1,
word_ptr = 4, word_count = 1)
ok
```

**4.8.20 modem.protocol.isoc.conf\_test**

Generates the appropriate ISO-C conformance test commands.

<b>Type</b>	function	
<b>Permissions</b>	guest = - admin = x	
<b>Parameters</b>	command (enum definitions.enum.protocol.isoc.test_commands) antenna (int)	

```
modem.protocol.isoc.conf_test (command=QUERY)
```

**4.8.21 modem.protocol.isoc.control.crc\_retries**

This variable sets how many times a tag ACKnowledge will be sent on a bad EPC CRC before moving on in the inventory cycle. If set to zero, the tag will not have it's inventory flag toggled. If the retries are set to one or more and the tag continues to give a bad CRC, the tag will have it's inventory state toggled before reporting the bad CRC.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	int
<b>Default</b>	3
<b>Min</b>	0
<b>Max</b>	8

**4.8.22 modem.protocol.isoc.control.enable\_per\_antenna**

This variable enables control of some ISO-C settings on a per antenna basis. When set to TRUE the modem.protocol.isoc.control.antenna.xxx variables will control what settings are used per antenna. When set to FALSE, all antennas will uses the same ISO-C configuration.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	bool
<b>Default</b>	false

**4.8.23 modem.protocol.isoc.control.verify\_write**

This variable enables read back verification of ISO-C writes. When set to true, a read command is issued following a write to verify the write success. When set to false, no read back is performed.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	bool
<b>Default</b>	false

RFI641

**4.8.24 modem.protocol.isoc.control.antenna.<n>.session\_id**

This variable contains the session\_id setting for ISO-C commands used on this antenna, when enable\_per\_antenna is set to TRUE.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	Enum
<b>Default</b>	SESSION_1
<b>Enum</b>	SESSION_0 SESSION_1 SESSION_2 SESSION_3

**4.8.25 modem.protocol.isoc.read**

This function reads specified data from an ISO-C tag in the field. The response to this function is a response name, followed by the data read from the tag.

<b>Type</b>	function	
<b>Permissions</b>	guest = x admin = x	
<b>Parameters</b>	tag_id (array)	Indicates the ID of the tag to read. If the tag_id is not supplied, the first tag found is read. The tag is first singulated and put in a state for a read operation. (optional)
	pwd (array)	Password used to read locked portions of a tag. (optional)
	mem_bank (int)	Specifies which tag memory bank is to read (0 = Reserved, 1 = EPC, 2 = TID, 3 = User memory).
	word_ptr (int)	Specifies the starting word (16 bit) address in the memory bank for the read.
	word_count (int)	Specifies the number of 16-bit words to read. If the word_count = 0 the entire memory bank is read.
	antenna (int)	Indicates antenna on which to execute the function. If no antenna is specified, all antennas will be tried until the function succeeds or no more antennas are available. (optional)

The following example reads four words at offset 2 from the user memory section of a tag with ID = 0x30112233445566778899aabb:

```
modem.protocol.isoc.read (tag_id = 0x30112233445566aabb, mem_bank = 3,
word_ptr = 2, word_count = 4)
ok data = 0x9988776655443322
```

#### 4.8.26 modem.protocol.isoc.write

This function writes specified data to an ISO-C tag in the field. The response to this function is a response name.

<b>Type</b>	function	
<b>Permissions</b>	guest = x admin = x	
<b>Parameters</b>	tag_id (array)	Indicates the ID of the tag to write. If the tag_id is not supplied, the first tag found will be written. The tag is first singulated and put in a state for a write operation. (optional)
	pwd (array)	Password used to write locked portions of a tag. (optional)
	mem_bank (int)	Specifies which tag memory bank to write (0 = Reserved, 1 = EPC, 2 = TID, 3 = User memory).
	word_ptr (int)	Specifies the starting word (16 bit) address in the memory bank for the write.
	data (array)	Specifies the data to write. If the data is not on word boundaries, it is padded with zeros to make a full word.
	antenna (int)	Indicates antenna on which to execute the function. If no antenna is specified, all antennas will be tried until the function succeeds or no more antennas are available. (optional)

The following example writes 0x12345678 at offset 3 from the EPC memory section of a tag with ID = 0x30112233445566778899aabb:

```
modem.protocol.isoc.write (tag_id = 0x301122334455aabb, mem_bank = 1,
word_ptr = 3, data = 0x12345678)
```

ok

#### 4.8.27 modem.protocol.isoc.control.cmd\_retries

This variable sets the number of command retries for failed command. When a command sequence fails, it can be retried for the number of times indicated by this parameter. This is transparent to the user.

<b>Type</b>	variable
<b>Permissions</b>	guest = - admin = rw
<b>Data Type</b>	int
<b>Default</b>	3
<b>Min</b>	0
<b>Max</b>	8

RF1641

**4.8.28 modem.protocol.isoc.control.display\_tag\_crc**

This variable enables display of the tag CRC. When set to true, entire tag data contents, including CRC, is displayed. When set to false, only the EPC data bits are displayed.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	bool
<b>Default</b>	False

**4.8.29 modem.protocol.isoc.control.include\_alllock\_bits**

This variable controls whether both lock bits of a ISO-C lock field are used in the lock procedure. The default setting is to use both lock bits, pwd-read/write and permalock, when locking a memory space. When set to false, the variable `modem.protocol.isoc.control.include_permalock_bit` gives finer control over the lock bits used.

<b>Type</b>	variable
<b>Permissions</b>	guest = - admin = rw
<b>Data Type</b>	bool
<b>Default</b>	true

**4.8.30 modem.protocol.isoc.control.include\_permalock\_bit**

This variable controls whether the permalock bit is included in the lock procedure when a lock command has been issued. If false, only the pwd-read/write bit is used. This variable is only used when `modem.protocol.isoc.control.include_alllock_bits` is false.

<b>Type</b>	variable
<b>Permissions</b>	guest = - admin = rw
<b>Data Type</b>	bool
<b>Default</b>	true

**4.8.31 modem.protocol.isoc.control.inventory\_antenna\_switch**

This variable allows antenna switching during inventory. When `slots_restart` is greater than 0, then `inventory_antenna_switch` allows the antenna to be switched before the new query command is sent.

To switch antennas during inventory, set `modem.protocol.isoc.control.slots_restart` to a value greater than 0, then set `inventory_antenna_switch` to true.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	bool
<b>Default</b>	false

**4.8.32 modem.protocol.isoc.control.inventory\_both\_targets**

This variable enables inventory of tags from A to B, and B back to A. When set to true, tags will be inventoried from the B state back to the A state. When false, only tags in the A state are inventoried to the B state.

<b>Type</b>	variable
<b>Permissions</b>	guest = rw admin = rw
<b>Data Type</b>	bool
<b>Default</b>	True

**4.8.33 modem.protocol.isoc.control.max\_incr\_slots\_q**

This variable controls how large the Q value from the ISO\_C specification can grow. The Q value is capped at the initial Q value (number\_slots\_Q) plus this max\_incr\_slots\_Q value.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	int
<b>Default</b>	15
<b>Min</b>	0
<b>Max</b>	15

**4.8.34 modem.protocol.isoc.control.mem\_bank\_for\_selection**

This variable is the memory bank that select command applies to perform filtering.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	enum
<b>Default</b>	MEMBANK_EPC
<b>Enum</b>	MEMBANK_RSVD MEMBANK_EPC MEMBANK_TID MEMBANK_USER

RFI641

**4.8.35 modem.protocol.isoc.control.number\_slots\_q**

This variable sets the initial number of slots in inventory round. The initial number of slots is the  $q$  value from the ISO-C specification, where the number of slots is equal to 2 to the power of  $q$  ( $=2^q$ ).

To set the number of slots to 1, set `number_slots_q = 0`. To set the number of slots to 2, set `number_slots_q = 1`.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	int
<b>Default</b>	4
<b>Min</b>	0
<b>Max</b>	15

**4.8.36 modem.protocol.isoc.control.query\_sel**

This variable sets the ISO-C sel used by the initial query command. If set to 0, the value used in the query is predetermined. For finer control this variable allows for specific sel values to be sent.

<b>Type</b>	variable
<b>Permissions</b>	guest = rw admin = rw
<b>Data Type</b>	int
<b>Values</b>	0 Value used in the query is predetermined.
	1 All tags respond to query.
	2 All tags respond to query.
	3 Tags with negated SL flag respond to query.
	4 Tags with asserted SL flag respond to query.
<b>Default</b>	0
<b>Min</b>	0
<b>Max</b>	4

**4.8.37 modem.protocol.isoc.control.query\_session**

This variable sets the ISO-C session used by the initial query command. For finer control this variable allows for specific session values to be sent.

<b>Type</b>	variable	
<b>Permissions</b>	guest = rw admin = rw	
<b>Data Type</b>	int	
<b>Values</b>	0	The value used in the query is modem.protocol.isoc.control.session_id.
	1	Session 0
	2	Session 1
	3	Session 2
	4	Session 3
<b>Default</b>	0	
<b>Min</b>	0	
<b>Max</b>	4	

**4.8.38 modem.protocol.isoc.control.query\_target**

This variable sets the ISO-C target used by the initial query command. ISO-C tags have an inventoried flag for each session. This flag will change state from TGT\_A to TGT\_B or vice versa when a tag has been singulated.

<b>Type</b>	variable	
<b>Permissions</b>	guest = rw admin = rw	
<b>Data Type</b>	enum	
<b>Default</b>	TGT_A	
<b>Enum</b>	TGT_A	
	TGT_B	

RFI641

**4.8.39 modem.protocol.isoc.control.select\_cmd\_period**

This variable controls how often a select command is sent, when ISO-C filtering is not enabled. The select command is sent at the start of every N inventory rounds.

<b>Type</b>	variable
<b>Permissions</b>	guest = rw admin = rw
<b>Data Type</b>	int
<b>Default</b>	0
<b>Min</b>	0
<b>Max</b>	65536

The following examples set to send a select command every 1, 5, and 0 (do not send) inventory rounds:

```
modem.protocol.isoc.control.select_cmd_period=1
modem.protocol.isoc.control.select_cmd_period=5
modem.protocol.isoc.control.select_cmd_period=0
```

**4.8.40 modem.protocol.isoc.control.session\_id**

This variable sets the ISO-C session to use for tag communication. The ISO-C specification allows for up to four sessions to be used concurrently. A tag stores state information for each session, so that a Reader in one Session does not disrupt the tag state for another Reader using a different Session.

<b>Type</b>	variable
<b>Permissions</b>	guest = rw admin = rw
<b>Data Type</b>	enum
<b>Default</b>	SESSION_1
<b>Enum</b>	SESSION_0 SESSION_1 SESSION_2 SESSION_3

**4.8.41 modem.protocol.isoc.control.slots\_restart**

This variable sets the number of slots before a round is restarted. An inventory round is started with a new query command after every slots\_restart, until the adaptive Q reaches 0. Tags that have been discovered in the previous round will not respond in the new round. To start a new inventory round every 16 slots, set slots\_restart = 16.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	int
<b>Default</b>	0
<b>Min</b>	0
<b>Max</b>	1024

**4.8.42 modem.protocol.isoc.control.use\_block\_write**

This variable enables ISO-C block write capability. This variable must be supported by tag. When set to true, the reader issues the optional ISO-C command BlockWrite for write operations. When set to false, the standard 16-bit write command is issued.

The write command is mandatory and must be supported by all tags.

<b>Type</b>	variable
<b>Permissions</b>	guest = rw admin = rw
<b>Data Type</b>	bool
<b>Default</b>	False

**4.8.43 modem.protocol.isoc.filter.<n>.action**

An ISO-C tag takes action based on whether it matches/unmatches the filter mask. This variable is the tag action taken based on that match. The action applies to the tag's SL flag if modem.protocol.isoc.filtering.use\_session is false.

If modem.protocol.isoc.filtering.use\_session is true, the action applies to the tags inventoried flag for the session specified in modem.protocol.isoc.control.session\_id.

The enum values indicate the action taken. The name before the underscore is the action for a match. The name after the underscore indicates the action taken for an unmatch. SL flag actions are asserted, deasserted, negated or not acted on.

The session inventoried flag will be set to A (asserted), B (deasserted), flipped A to B or B to A (negated) or not acted on.

<b>Type</b>	variable
<b>Permissions</b>	guest = - admin = rw
<b>Data Type</b>	enum
<b>Default</b>	ASSERT_DEASSERT
<b>Enum</b>	ASSERT_DEASSERT ASSERT_NOTHING NOTHING_DEASSERT NEGATE_NOTHING DEASSERT_ASSERT DEASSERT_NOTHING NOTHING_ASSERT NOTHING_NEGATE

For example, to inventory only tags that have value 0x22 at offset 0x30 OR a value 0x44 at offset 0x30, set up the filter variables for 2 filters, and then use action ASSERT\_DEASSERT for the first filter and action ASSERT\_NOTHING for the second filter.

To inventory only tags that have value 0x22 at offset 0x30 AND a value 0x44 at offset 0x40, set up the filter variables for 2 filters, and then use action ASSERT\_DEASSERT for the first filter and action NOTHING\_DEASSERT for the second filter.

To inventory only tags that have value 0x22 at offset 0x30 AND exclude tags with value 0x40 at offset 0x40, set up the filter variables for 2 filters, and then use action ASSERT\_DEASSERT for the first filter and action DEASSERT\_NOTHING for the second filter.

RFI641

**4.8.44 modem.protocol.isoc.filter.<n>.enabled**

This variable enables the ISO-C mask filter. When set to true, only those ISO-C tags that match the filter mask will respond. The filter mask is a bit pattern which starts at a specific bit offset in the tag and has a specific length.

<b>Type</b>	variable
<b>Permissions</b>	guest = - admin = rw
<b>Data Type</b>	bool
<b>Default</b>	False

**4.8.45 modem.protocol.isoc.filter.<n>.length**

This variable is the bit length of the ISO-C mask filter. This variable specifies the bit length of the ISO-C mask filter.

<b>Type</b>	variable
<b>Permissions</b>	guest = - admin = rw
<b>Data Type</b>	int
<b>Default</b>	0
<b>Min</b>	0
<b>Max</b>	255

**4.8.46 modem.protocol.isoc.filter.<n>.mask**

This variable is the bit mask for the ISO-C mask filter. This is the mask that a tag compares against its memory location that begins at offset and ends length bits later. If filtering is enabled, only tags that match the mask will respond in a round.

<b>Type</b>	variable
<b>Permissions</b>	guest = - admin = rw
<b>Data Type</b>	array
<b>Default</b>	0x00

**4.8.47 modem.protocol.isoc.filter.<n>.mem\_bank**

This variable is the memory bank for the ISO-C mask filter. This is the memory bank that a tag uses when comparing against its memory location that begins at offset and ends length bits later. Only tags that match the mask will respond in a round if filtering is enabled.

If set to NOT\_USED, then the variable `modem.protocol.isoc.control.mem_bank_for_selection` will control which memory bank is used for this filter.

<b>Type</b>	variable
<b>Permissions</b>	guest = - admin = rw
<b>Data Type</b>	enum
<b>Default</b>	NOT_USED
<b>Enum</b>	MEMBANK_EPC MEMBANK_TID MEMBANK_USER NOT_USED

**4.8.48 modem.protocol.isoc.filter.<n>.offset**

This variable specifies the bit offset for the ISO-C mask filter. It is the bit address where the tag will start the mask comparison.

<b>Type</b>	variable
<b>Permissions</b>	guest = - admin = rw
<b>Data Type</b>	int
<b>Default</b>	0
<b>Min</b>	0
<b>Max</b>	32767

**4.8.49 modem.protocol.isoc.filtering.enabled**

This variable enables the ISO-C mask filter. When set to true, only those ISO-C tags that match the filter mask will respond. The filter mask is a bit pattern which starts at a specific bit offset in the tag and has a specific length.

<b>Type</b>	variable
<b>Permissions</b>	guest = - admin = rw
<b>Data Type</b>	bool
<b>Default</b>	False

**4.8.50 modem.protocol.isoc.filtering.truncated\_epc\_response**

This variable turns on ISO-C truncation. When truncation is true and filtering is enabled, the mask filter is used and tags that match the mask will truncate their EPC response to the bits that follow the mask. The mask must end in the EPC memory or truncation will not be enabled.

RFI641

For truncation to work, the filter mask must start at offset 16 in EPC memory, which is the start of the Protocol-control (PC) bits. The most significant 5 bits of the PC indicate the length of the tags EPC in 16-bit words, plus the 1 word of PC bits. Only 1 filter mask can be used with truncation.

<b>Type</b>	variable
<b>Permissions</b>	guest = - admin = rw
<b>Data Type</b>	bool
<b>Default</b>	False

The following example sets up a filter and uses truncation to find 96-bit EPC tags with tag\_id=301020304050xxxxxxxxxx, where x is an arbitrary value. The length is 64 bits where: 64 bits = 16 (PC bits) + 48 bits (EPC bits in mask). The offset is the offset to PC bits in EPC memory). The filter mask is 3000301020304050 where the first 16 bits are the PC bits from EPC memory).

```
modem.protocol.isoc.filter.1.length=64
modem.protocol.isoc.filter.1.offset=16
modem.protocol.isoc.filter.1.mask=3000301020304050
modem.protocol.isoc.filter.1.action=assert_deassert
modem.protocol.isoc.filter.1.enabled=1
modem.protocol.isoc.filtering.truncated_epc_response=1
modem.protocol.isoc.filtering.enabled=1
```

#### 4.8.51 modem.protocol.isoc.filtering.truncated\_tag\_epc\_length

This variable sets the expected tag EPC length for truncation. When set to 0, the EPC length is calculated from the PC word included in the filter mask. See modem.protocol.isoc.filtering.truncated\_epc\_response. Otherwise, this value is used.

<b>Type</b>	variable
<b>Permissions</b>	guest = - admin = rw
<b>Data Type</b>	int
<b>Default</b>	0
<b>Min</b>	0
<b>Max</b>	512

#### 4.8.52 modem.protocol.isoc.filtering.use\_session

This variable causes the ISO-C filter to target the tags session inventoried flag instead of the tags SL flag. Any tag action based on match/mismatch of the filter mask will cause the tag to change its inventoried flag for the session, and not the tags SL flag.

<b>Type</b>	variable
<b>Permissions</b>	guest = - admin = rw
<b>Data Type</b>	bool
<b>Default</b>	False

**4.8.53 modem.protocol.isoc.nxp.calibrate()**

This function performs the NXP custom Calibrate command.

<b>Type</b>	function
<b>Permissions</b>	guest = x admin = x
<b>Parameters</b>	tag_id (array) pwd (array) antenna (int)

```
modem.protocol.isoc.nxp.calibrate()
```

```
ok data = 0x1122334455667788
```

**4.8.54 modem.protocol.isoc.nxp.change\_eas()**

This function performs the NXP custom ChangeEAS command.

<b>Type</b>	function
<b>Permissions</b>	guest = x admin = x
<b>Parameters</b>	tag_id (array) pwd (array) command (enum definitions.enum.protocol.isoc.nxp.change_eas_commands) antenna (int)

```
modem.protocol.isoc.nxp.change_eas(command=SET_EAS)
```

```
ok
```

**4.8.55 modem.protocol.isoc.nxp.eas\_alarm()**

This function performs the NXP custom EASAlarm command.

<b>Type</b>	function
<b>Permissions</b>	guest = x admin = x
<b>Parameters</b>	antenna (int)

```
modem.protocol.isoc.nxp.eas_alarm()
```

```
ok data = 0x690AEC7CD215D8F9
```

RF1641

**4.8.56 modem.protocol.isoc.nxp.read\_protect()**

This function performs the NXP custom read\_protect.

<b>Type</b>	function
<b>Permissions</b>	guest = x admin = x
<b>Parameters</b>	tag_id (array) pwd (array) antenna (int)

```
modem.protocol.isoc.nxp.read_protect ( )
```

```
ok
```

**4.8.57 modem.protocol.isoc.nxp.reset\_read\_protect()**

This function performs the NXP custom command reset\_read\_protect.

<b>Type</b>	function
<b>Permissions</b>	guest = x admin = x
<b>Parameters</b>	tag_id (array) pwd (array) antenna (int)

```
modem.protocol.isoc.nxp.reset_read_protect ( )
```

```
ok
```

**4.8.58 modem.protocol.isoc.physical.data\_1\_length**

This variable is the length of a Data 1 symbol for ISO-C. The ISO-C Data 1 symbol length can be set to either (Tari \* 1.5) or (Tari \* 2.0). To change, use the set function.

<b>Type</b>	variable
<b>Permissions</b>	guest = - admin = rw
<b>Data Type</b>	enum
<b>Default</b>	D1_LEN_20
<b>Enum</b>	D1_LEN_15 D1_LEN_20

**4.8.59 modem.protocol.isoc.physical.interrogator\_mode**

This variable sets the reader mode for ISO-C. The reader can be in 1 of 3 modes when ISO-C protocol is enabled: single interrogator, multiple interrogator, or dense interrogator. The selection depends on the environment and requirements.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	enum
<b>Default</b>	SINGLE
<b>Enum</b>	SINGLE MULTI DENSE MAINT1 MAINT2

**4.8.60 modem.protocol.isoc.physical.modulation\_depth**

This specifies the percent modulation depth for ISO-C transmissions. The ISO-C specification requires the depth to be between 80 % to 100 %.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	int
<b>Default</b>	90
<b>Min</b>	0
<b>Max</b>	100

**4.8.61 modem.protocol.isoc.physical.pilot\_tone**

This variable turns on the ISO-C pilot\_tone from the tag. When true, the pilot tone is on and the tag will prepend 12 leading zeros to its response preamble.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	bool
<b>Default</b>	True

RFI641

**4.8.62 modem.protocol.isoc.physical.return\_link\_freq**

This variable sets the return link frequency for tag-to-reader communication. The actual data rate depends on the encoding (such as FMO, Miller-2, Miller-4, or Miller-8). To change, use the set function.

Only the combination of return link frequency modulation and encoding allowed by the ISO 18000-6 C standard are supported.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	enum
<b>Default</b>	LF160
<b>Enum</b>	LF80 LF160 LF200 LF256 LF320 LF171 LF640 LFTEST12_50 LFTEST25_00

**4.8.63 modem.protocol.isoc.physical.rt\_modulation**

This variable is the reader-to-tag modulation type. The reader can transmit using 1 of 3 different modulation types: single sideband ASK, dual sideband ASK, or phase reversal ASK.

Only the combination of return link frequency modulation and encoding allowed by the ISO 18000-6 C standard are supported.

<b>Type</b>	variable
<b>Permissions</b>	guest = - admin = rw
<b>Data Type</b>	enum
<b>Default</b>	RT_MOD_DSB
<b>Enum</b>	RT_MOD_SSB RT_MOD_DSB RT_MOD_PR

**4.8.64 modem.protocol.isoc.physical.set**

This function sets ISO-C physical link parameters. You must use this function when you wish to change the TARI, data\_1\_length, or return\_link\_freq parameters. If any one of these parameters (TARI, data\_1\_length, return\_link\_freq) are set, they must all be set.

<b>Type</b>	function
<b>Permissions</b>	guest = - admin = x
<b>Parameters</b>	tari (enum definitions.enum.protocol.isoc.physical.tari) data_1_length (enum definitions.enum.protocol.isoc.physical.data_1_length) return_link_freq (enum definitions.enum.protocol.isoc.physical.return_link_freq) interrogator_mode (enum definitions.enum.protocol.isoc.physical.interrogator_mode) pilot_tone (int) tr_encoding (enum definitions.enum.protocol.isoc.physical.tr_encoding) rt_modulation (enum definitions.enum.protocol.isoc.physical.rt_modulation) modulation_depth (int)

**4.8.65 modem.protocol.isoc.physical.settings**

This variable contains ISO-C physical layer settings so they can be saved and restored with one variable.

<b>Type</b>	variable
<b>Permissions</b>	guest = - admin = rw
<b>Data Type</b>	string
<b>Default</b>	""

**4.8.66 modem.protocol.isoc.physical.tari**

This variable specifies the TARI (in microseconds) used for the ISO-C physical layer. To change, use the set function.

Only the combination of return link frequency modulation and encoding allowed by the ISO 18000-6 C standard are supported.

<b>Type</b>	variable
<b>Permissions</b>	guest = - admin = rw
<b>Data Type</b>	enum
<b>Default</b>	TARI_25_00
<b>Enum</b>	TARI_06_25 TARI_12_50 TARI_25_00

RF1641

**4.8.67 modem.protocol.isoc.physical.tr\_encoding**

This variable is the tag to reader encoding format. The reader instructs the tag to use 1 of 2 encoding formats: FMO baseband or Miller sub-carrier.

Only the combination of return link frequency modulation and encoding allowed by the ISO 18000-6 C standard are supported.

<b>Type</b>	variable
<b>Permissions</b>	guest = - admin = rw
<b>Data Type</b>	enum
<b>Default</b>	TR_ENC_FMO
<b>Enum</b>	TR_ENC_FMO TR_ENC_MILLER_2 TR_ENC_MILLER_4 TR_ENC_MILLER_8

**4.8.68 modem.protocol.supertag.control.display\_tag\_crc**

This variable enables display of the tag CRC. When set to true, entire tag data contents, including CRC, is displayed. When set to false, the tag data bits without the CRC are displayed.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	bool
<b>Default</b>	True

**4.8.69 modem.protocol.supertag.control.dwell\_time**

The time (millisecs) per round that the reader searches for Supertag tags.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	int
<b>Default</b>	50
<b>Min</b>	40
<b>Max</b>	1000

**4.8.70 modem.protocol.supertag.control.enable\_crc\_check**

This variable enables CRC checking of the tag response. When set to true, the received data is verified with the received CRC. When set to false, the CRC checking is disabled.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	bool
<b>Default</b>	True

**4.8.71 modem.protocol.supertag.physical.return\_link\_freq**

This variable sets the return link frequency for the tag to reader communication.

Type	variable
Permissions	guest = r admin = rw
Data Type	enum
Default	LF64
Enum	LF64 LF256

**4.8.72 modem.radio.idle\_cw**

When the read is idle and not talking to tags, the reader can either transmit CW or can turn RF off. If idle\_cw is set to True, the reader will transmit CW when it is idle.

Type	variable
Permissions	guest = - admin = rw
Data Type	bool
Default	False

**4.8.73 modem.radio.lbt.enabled**

This variable enables Listen Before Talk. Follows ETSI specification EN302208.

Type	variable
Permissions	guest = - admin = rw
Data Type	bool
Default	False

**4.8.74 modem.radio.lbt.listen\_antenna**

This variable selects the antenna to use for LBT listening, where:

- LBT\_ANTENNA[1-4]=use specified antenna for listening
- LBT\_DEDICATED\_ANTENNA=use LBT antenna for listening
- LBT\_ANTENNA\_ALL=use current active (transmit) antenna for listening

Type	variable
Permissions	guest = r admin = rw
Data Type	enum
Default	LBT_DEDICATED_ANTENNA
Enum	LBT_ANTENNA1 LBT_ANTENNA2 LBT_ANTENNA3 LBT_ANTENNA4 LBT_ANTENNA_ALL LBT_DEDICATED_ANTENNA

RFI641

**4.8.75 modem.radio.rf\_survey()**

This function performs a spectrum analysis at the current frequency. The results of this function are scaled to account for all system gains.

<b>Type</b>	function
<b>Permissions</b>	guest = - admin = x
<b>Parameters</b>	rbw (enum definitions.enum.resolution_bw) Specifies the resolution bandwidth.
	span (enum definitions.enum.freq_span) Specifies the frequency span.
	time (int) Specifies the dwell time before returning the results.
	coherent (bool) Enables coherent scaling
	center_freq (int) Specifies the geometric mean of the frequency span.
	output_freq (int) Specifies the output frequency. If not specified all frequencies will be returned.

## 4.9 USER Command

### 4.9.1 user.var<n>

This variable is for user access. It can be used to hold any users information or data. <n> is 1 to 64.

<b>Type</b>	variable
<b>Permissions</b>	guest = r admin = rw
<b>Data Type</b>	string
<b>Default</b>	""

The following example sets a tag identifier in the variable user.var1. The tag identifier stored in user.var1 could later be used in a tag.write\_id() function to have this tag identifier written to a tag.

```
user.var1 = 0x30112233445566778899aabb
```

```
ok
```

RFI641

## 4.10 EVENT Commands

Command	Description	See Chapter/Page
event.connection	Establishes new event channel.	<a href="#">4.10.1 / 124</a>
event.info	Enables informational event.	<a href="#">4.10.2 / 124</a>
event.configuration.change	Enables configuration change event.	<a href="#">4.10.3 / 124</a>
event.debug.trace.iop	Debugs tracing information sent from reader.	<a href="#">4.10.4 / 124</a>
event.debug.trace.modem	Sends debug tracing information from the modem.	<a href="#">4.10.5 / 125</a>
event.dio.all	Reports state of all digital inputs/outputs.	<a href="#">4.10.6 / 125</a>
event.dio.in.<n>	Enables digital IO event generated each time a digital input changes.	<a href="#">4.10.7 / 125</a>
event.dio.in.alarm.timeout.1	Enables digital IO event.	<a href="#">4.10.8 / 126</a>
event.dio.out.<n>	Enables digital IO event generated each time a digital output changes.	<a href="#">4.10.9 / 126</a>
event.error.antenna	Indicates that the reader has detected an antenna error.	<a href="#">4.10.10 / 126</a>
event.error.communication	Indicates that the reader has detected a communication error.	<a href="#">4.10.11 / 126</a>
event.error.configuration	Indicates that the reader has detected a configuration error.	<a href="#">4.10.12 / 126</a>
event.error.environmental	Indicates that the reader has detected an environmental error.	<a href="#">4.10.13 / 127</a>
event.error.file_handling	Indicates that the reader has detected an error while operating on a file.	<a href="#">4.10.14 / 127</a>
event.error.hw	Indicates that the reader has detected a hardware error.	<a href="#">4.10.15 / 127</a>
event.error.radio	Indicates that the reader has detected an error in the radio.	<a href="#">4.10.16 / 127</a>
event.error.sw	Indicates that the reader has detected a software error.	<a href="#">4.10.17 / 127</a>
event.status.antenna_transition	Indicates that the reader has transitioned to new antenna port.	<a href="#">4.10.18 / 128</a>
event.status.channel_power_report	Indicates a power report for a channel.	<a href="#">4.10.19 / 128</a>
event.status.channel_state	Reports a change in state for a channel.	<a href="#">4.10.20 / 128</a>
event.status.channel_transition	Indicates that the reader has transitioned to new channel.	<a href="#">4.10.21 / 128</a>
event.status.inventory_end	Indicates that the reader has completed inventory round.	<a href="#">4.10.22 / 129</a>
event.status.inventory_rounds_complete	Indicates that inventory rounds have completed.	<a href="#">4.10.23 / 129</a>
event.status.inventory_start	Indicates that the reader has started an inventory round.	<a href="#">4.10.24 / 129</a>
event.status.modem_halted	Indicates that the reader has detected unrecoverable error and halted.	<a href="#">4.10.25 / 129</a>
event.status.modem_ready	Indicates that the reader is ready to process commands.	<a href="#">4.10.26 / 129</a>
event.status.tag_collision	Indicates that the reader has detected collision between tags during inventory round.	<a href="#">4.10.27 / 129</a>
event.status.tag_loss_of_signal	Indicates that the reader has read tag but signal loss prevented CRC validation.	<a href="#">4.10.28 / 130</a>
event.status.tag_read_bad_crc	Indicates that the reader has read tag but CRC did not match.	<a href="#">4.10.29 / 130</a>
event.status.tx_active	Indicates that the reader's transmitter has been enabled/disabled.	<a href="#">4.10.31 / 131</a>
event.tag.alarm	Indicates that a tag has generated an alarm	<a href="#">4.10.30 / 130</a>
event.tag.arrive	Indicates that a tag has been detected for the first time.	<a href="#">4.10.32 / 131</a>
event.tag.depart	Indicates that a tag previously detected is no longer detected.	<a href="#">4.10.33 / 131</a>
event.tag.report	Indicates that a tag has been detected.	<a href="#">4.10.34 / 132</a>
event.tag.scan_tags_complete	Indicates that the tag.db.scan_tags() command has completed.	<a href="#">4.10.35 / 132</a>
event.status.tag_no_epc	Indicates that the reader ACK'd tag but no EPC was received	<a href="#">4.10.36 / 132</a>

Table 4-9: Overview: EVENT commands

Command	Description	See Chapter/Page
event.warning.antenna	Indicates that the reader has detected a warning condition on a specific antenna port.	<a href="#">4.10.37 / 132</a>
event.warning.radio	Indicates that the reader has detected a warning condition in the radio module.	<a href="#">4.10.38 / 133</a>
event.warning.hw	Indicates that the reader has detected a warning condition in hardware.	<a href="#">4.10.39 / 133</a>
event.warning.network	Indicates that the reader has detected network congestion forcing it to drop events.	<a href="#">4.10.40 / 134</a>

Table 4-9: Overview: EVENT commands (contd.)

#### 4.10.1 event.connection

This event is generated when a new event channel has been established with the reader. The new event channel is identified by the ID parameter passed back in this event. The ID can be used to register for events on this event channel with the bind and registerevent commands.

<b>Type</b>	event
<b>Permissions</b>	guest = rw admin = rw
<b>Return values</b>	id (int)

#### 4.10.2 event.info

These events go into the error/warning log but are informational only, and do not indicate an error or warning condition.

<b>Type</b>	event
<b>Permissions</b>	guest = rw admin = rw

#### 4.10.3 event.configuration.change

This event is generated each time a configuration is changed.

<b>Type</b>	event
<b>Permissions</b>	guest = rw admin = rw
<b>Return Values</b>	name (string) newvalue (string) oldvalue (string)

#### 4.10.4 event.debug.trace.iop

This event sends debug tracing information from the reader.

<b>Type</b>	event
<b>Permissions</b>	guest = rw admin = rw
<b>Return values</b>	file (string) line (int) time (string) data (string)

RFI641

The following example illustrates an `event.debug.trace.iop` event:

```
event.debug.trace.iop file=TagDatabase.c line=203 time=01/03/06-08:56:13 data='tag hashed to value 482'
```

#### 4.10.5 event.debug.trace.modem

This event sends debug tracing information from the modem. The `modem_ms` is the number of ms the modem has run since bootup (rough timestamp from modem DSP).

<b>Type</b>	event
<b>Permissions</b>	guest = rw admin = rw
<b>Return values</b>	file (string) line (int) modem_ms (int) data (string)

The following example illustrates an `event.debug.trace.modem` event:

```
event.debug.trace.modem file=mem.c line=804 modem_ms=827443 data='Free'ed a block which was not previously allocated!'
```

#### 4.10.6 event.dio.all

This event reports the state of all digital IOs and is generated each time a digital IO changes.

<b>Type</b>	event
<b>Permissions</b>	guest = rw admin = rw
<b>Return values</b>	in (int) out (int)

#### 4.10.7 event.dio.in.<n>

This event is generated each time a digital input changes. <n> = 1 to 4.

<b>Type</b>	event
<b>Permissions</b>	guest = rw admin = rw
<b>Return Values</b>	value (string)??? value (int)

**4.10.8 event.dio.in.alarm.timeout.1**

If a timeout value is set, the input pin is monitored. If the input pin value does not change during the timeout period AND the input pin value matches the alarm logic level, the event event.dio.in.alarm.timeout.n (where n is the pin number) is generated. This alarm event generation can be helpful in alerting to the loss of digital inputs to the reader.

<b>Type</b>	event
<b>Permissions</b>	guest = rw admin = rw
<b>Return Values</b>	value (int) logic_level (int) timeout (int)

**4.10.9 event.dio.out.<n>**

This event is generated each time a digital output changes. <n> = 1 to 4.

<b>Type</b>	event
<b>Permissions</b>	guest = rw admin = rw
<b>Return Values</b>	value (int)

**4.10.10 event.error.antenna**

This event indicates that the reader detected an error condition with the antennas.

<b>Type</b>	event
<b>Permissions</b>	guest = rw admin = rw
<b>Return Values</b>	id (int) time (string) text (string)

**4.10.11 event.error.communication**

This event indicates that the reader detected a communication error.

<b>Type</b>	event
<b>Permissions</b>	guest = rw admin = rw
<b>Return Values</b>	id (int) time (string) text (string)

**4.10.12 event.error.configuration**

This event indicates that the reader detected a configuration error.

<b>Type</b>	event
<b>Permissions</b>	guest = rw admin = rw
<b>Return Values</b>	id (int) time (string) text (string)

RFI641

**4.10.13 event.error.environmental**

This event indicates that the reader detected an error condition with the environmental condition around the reader.

<b>Type</b>	event
<b>Permissions</b>	guest = rw admin = rw
<b>Return Values</b>	id (int) time (string) text (string)

**4.10.14 event.error.file\_handling**

This event indicates that the reader detected an error condition while operating on a file.

<b>Type</b>	event
<b>Permissions</b>	guest = rw admin = rw
<b>Return Values</b>	id (int) time (string) text (string)

**4.10.15 event.error.hw**

This event indicates that the reader detected an error condition with the hardware.

<b>Type</b>	event
<b>Permissions</b>	guest = rw admin = rw
<b>Return Values</b>	id (int) time (string) text (string)

**4.10.16 event.error.radio**

This event indicates that the reader detected an error condition with the radio.

<b>Type</b>	event
<b>Permissions</b>	guest = rw admin = rw
<b>Return Values</b>	id (int) time (string) text (string)

**4.10.17 event.error.sw**

This event indicates that the reader detected an error condition with the software.

<b>Type</b>	event
<b>Permissions</b>	guest = rw admin = rw
<b>Return Values</b>	id (int) time (string) text (string)

**4.10.18 event.status.antenna\_transition**

This event indicates the reader has transitioned to a new antenna port. The new antenna port is identified by the `new_antenna` parameter. The previous antenna port is also provided, as well as the time (in ms) the previous antenna port was used prior to the transition.

<b>Type</b>	event
<b>Permissions</b>	guest = rw admin = rw
<b>Return Values</b>	prev_antenna (int) new_antenna (int) time (string)

**4.10.19 event.status.channel\_power\_report**

This event indicates a reader completed operation on a channel and is generated before it transitions to a new channel. The report contains the frequency the channel was using and the power used.

<b>Type</b>	event
<b>Permissions</b>	guest = rw admin = rw
<b>Return Values</b>	frequency (int) power (int) time (string)

**4.10.20 event.status.channel\_state**

This event indicates a channel changed state.

<b>Type</b>	event
<b>Permissions</b>	guest = r admin = rw
<b>Return Values</b>	new_frequency (int) old_state (enum) new_state (enum) time (int)

**4.10.21 event.status.channel\_transition**

This event indicates the reader has transitioned to a new channel. The channel is identified by the `new_frequency` parameter. The previous channel is also provided, as well as the time (in ms) the previous channel used prior to the transition.

<b>Type</b>	event
<b>Permissions</b>	guest = rw admin = rw
<b>Return Values</b>	prev_frequency (int) new_frequency (int) time (int)

RF1641

**4.10.22 event.status.inventory\_end**

This event indicates the reader has completed an inventory round.

<b>Type</b>	event
<b>Permissions</b>	guest = rw admin = rw
<b>Return Values</b>	tag_type (string) total_slots (int) empty_slots (int)

**4.10.23 event.status.inventory\_rounds\_complete**

The inventory rounds started with a modem.control.inventory.perform\_rounds function have completed.

<b>Type</b>	event
<b>Permissions</b>	guest = rw admin = rw

**4.10.24 event.status.inventory\_start**

This event indicates the reader has started an inventory round.

<b>Type</b>	event
<b>Permissions</b>	guest = rw admin = rw
<b>Return Values</b>	tag_type (string)

**4.10.25 event.status.modem\_halted**

This event indicates the modem has detected an unrecoverable error and halted. No further communication with the modem is possible after this event.

<b>Type</b>	event
<b>Permissions</b>	guest = rw admin = rw

**4.10.26 event.status.modem\_ready**

This event indicates the modem has finished initializing and is ready to process commands.

<b>Type</b>	event
<b>Permissions</b>	guest = rw admin = rw

**4.10.27 event.status.tag\_collision**

This event indicates the reader detected a collision between tags responding during an inventory round.

<b>Type</b>	event
<b>Permissions</b>	guest = rw admin = rw
<b>Return Values</b>	tag_type (string)

**4.10.28 event.status.tag\_loss\_of\_signal**

This event indicates a tag read but signal loss prevented validation of CRC.

<b>Type</b>	event
<b>Permissions</b>	guest = rw admin = rw
<b>Return Values</b>	raw_data (string) type (string) antenna (int) ending_clock_offset (int) frequency (int) initial_clock_offset (int) rssi (int) demod_result (int) tx_power (int) quality_metric (int)

**4.10.29 event.status.tag\_read\_bad\_crc**

This event indicates a tag read but the CRC on the tag data did not match the CRC received by reader.

<b>Type</b>	event
<b>Permissions</b>	guest = rw admin = rw
<b>Return Values</b>	raw_data (string) type (string) antenna (int) ending_clock_offset (int) frequency (int) initial_clock_offset (int) rssi (int) demod_result (int) tx_power (int) quality_metric (int)

**4.10.30 event.tag.alarm**

This event indicates a tag has generated an alarm. The current supported alarms are EasAlarm which is indicated by type=EASALARM, and BatteryLow Alarm, indicated by type=BATTERY\_LOW\_ALARM.

<b>Type</b>	event
<b>Permissions</b>	guest = rw admin = rw
<b>Return Values</b>	tag_id (string) type (string) time (string) antenna (int)

RFI641

**4.10.31 event.status.tx\_active**

This event indicates the when the reader's transmitter has been enabled or disabled.

<b>Type</b>	event
<b>Permissions</b>	guest = rw admin = rw
<b>Return Values</b>	active (int)

**4.10.32 event.tag.arrive**

This event is generated when a tag is detected for the first time. The amount of information present is controlled by tag.reporting.arrival.

<b>Type</b>	event
<b>Permissions</b>	guest = rw admin = rw
<b>Return Values</b>	tag_id (string) type (string) antenna (string) tid (string) user_data (string) time (string)

**4.10.33 event.tag.depart**

This event is generated when a previously detected tag is no longer detected. The information provided is controlled by tag.reporting.depart.

<b>Type</b>	event
<b>Permissions</b>	guest = rw admin = rw
<b>Return Values</b>	tag_id (string) type (string) antenna (string) tid (string) user_data (string) time (string) repeat (string)

**4.10.34 event.tag.report**

This event indicates a tag has been detected. The amount of information presented is controlled by tag.reporting.report.

<b>Type</b>	event
<b>Permissions</b>	guest = rw admin = rw
<b>Return Values</b>	tag_id (string) tid (string) user_data (string) type (string) time (string) antenna (int) freq (int) rssi (int) tx_power (int) init_clock_offset (int) ending_clock_offset (int) quality_metric (int)

**4.10.35 event.tag.scan\_tags\_complete**

This event indicated that a tag.db.scan\_tags() command has completed.

<b>Type</b>	event
<b>Permissions</b>	guest = rw admin = rw

**4.10.36 event.status.tag\_no\_epc**

This event indicates a tag's RN16 was ACK'd but no EPC was received from the tag.

<b>Type</b>	event
<b>Permissions</b>	guest = rw admin = rw
<b>Return Values</b>	raw_data (string) type (int) antenna (int) ending_clock_offset (int) frequency (int) initial_clock_offset (int) rssi (int) demod_result (int) tx_power (int) quality_metric (int)

**4.10.37 event.warning.antenna**

This event indicates that the reader detected a warning condition on a specific antenna port.

<b>Type</b>	event
<b>Permissions</b>	guest = rw admin = rw
<b>Return Values</b>	id (int) time (string) text (string)

RFI641

The following subevents can be generated:

Warning	Description
MDM_REFLECTED_POWER	An antenna's return loss is lower than the warning threshold, but not low enough to cross the fault threshold.

#### 4.10.38 event.warning.radio

This event indicates that the reader detected a warning condition in the radio module.

<b>Type</b>	event
<b>Permissions</b>	guest = rw admin = rw
<b>Return Values</b>	id (int) time (string) text (string)

Warning	Description
MDM_NO_AVAILABLE_FREQUENCIES	There are no frequencies available for transmission which meet the LBT and/or minimum-off-time requirements.
MDM_PREFERRED_FREQUENCIES_NOT_APPLICABLE	There are no frequencies available for transmission which meet the LBT and/or minimum-off-time requirements.

#### 4.10.39 event.warning.hw

This event indicates that the reader detected a warning condition during hardware health monitoring.

<b>Type</b>	event
<b>Permissions</b>	guest = rw admin = rw
<b>Return Values</b>	id (int) time (string) text (string)

The following subevents can be generated:

Warning	Description
MDM_OVER_TEMP	The measured temperature is higher than the warning threshold, but not high enough to cross fault threshold.
IOP_EVENTS_DROPPED	Because of network congestion (or slow client software), some events were dropped from the event channel on which this event is received.

**4.10.40 event.warning.network**

This event indicates that the reader has detected network congestion forcing it to drop events.

<b>Type</b>	event
<b>Permissions</b>	guest = rw admin = rw
<b>Return Values</b>	id (int) time (string) text (string)

RFI641

## 5 Error messages

The error namespace is a namespace tree. Each nested namespace provides more information about the specific error. For example the errors in the error.file namespace indicate there was an error processing a file. Errors in the error.parser namespace indicate there was an error parsing the command due to a mistake with the command's syntax.

Error message	Description	Causing command
error.app.delete_failure	Attempt to delete an application failed.	reader.apps.delete(filename)
error.app.list_failure	Attempt to list running apps failed.	reader.apps.list()
error.app.not_running	Attempt to stop application not running.	reader.kill_app()
error.app.start_failure	Attempt to start an application failed.	reader.exec_app() reader.exec_jvm()
error.app.stop_failure	Attempt to stop an application failed.	reader.kill_app()
error.file.close_failure	Closing a file on the reader failed.	reader.load_config() reader.save_config()
error.file.delete_failure	Deleting a file from the reader failed.	reader.delete_config()
error.file.open_failure	Opening a file on the reader failed.	reader.list_apps() reader.list_configs() reader.load_config() reader.save_config()
error.file.too_many_profiles	Maximum number of profiles exceeded.	reader.profile.save(filename)
error.flash.read_failure	Reader could not read flash memory.	Any get command
error.flash.write_failure	Reader could not write flash memory.	Any set command
error.fwupdate.processing_error	Error occurred during firmware change.	reader.upgrade_firmware() reader.rollback_firmware()
error.internal.out_of_memory	Operation failed due to lack of memory.	Any command
error.internal.processing_error	Operation failed due to internal processing error.	Any command
error.mode.command_in_invalid_mode	Command rejected because incorrect reader mode.	tag.db.get() tag.db.get_and_purge() tag.tb.purge() tag.db.scan_tags()
error.modem.calibration_failed	Calibration has failed.	modem.radio.tx.calibration.next() modem.radio.tx_calibration.lqoffset()
error.modem.invalid_frequency_table	Invalid frequency hop table specified.	modem.radio.freq_mgmt. Hop_table.set()
error.modem.invalid_response	Invalid modem response.	Any tag.* command
error.modem.radio_control_failure	Modemradio control problem.	modem.radio.tx.llrfon() modem.radio.rx.llrffoff() Any modem.tag.* command
error.modem.rf_memory_failure	Memory on RF board has failed.	Any modem.* command
error.modem.timeout	No modem response.	Any command
error.network.address_not_reachable	Reader could not ping network address.	com.network.ping()
error.network.failed_to_configure	Command to configure the network settings failed.	com.network.1.set()
error.parser.command_too_long	Command too long to be processed.	Any command
error.parser.illegal_parameter	Set of supplied parameters is illegal.	Any function
error.parser.illegal_parameter_type	Command uses invalid parameter type.	Any function

Table 5-1: Overview: error messages

Error message	Description	Causing command
error.parser.illegal_value	Illegal value in command.	Any command
error.parser.malformed_command	Command not in correct format.	Any command
error.parser.missing_parameter	Command parameter missing.	Any function
error.parser.permission_failure	Insufficient permissions for operation.	Any command
error.parser.processing_error	Error encountered during processing.	Any command
error.parser.response_too_long	Not enough memory to return full response.	Any command
error.parser.set_failure	The 'namespace.set()' function failed.	Any namespace.set()*
error.parser.unknown_command	Command was not recognised.	Any command
error.parser.unknown_parameter	Command contained an unknown parameter.	Any function
error.parser.unknown_variable	Command contained unknown variable.	Any get command Any set command
error.parser.value_out_of_range	Command value out of range.	Any set command
error.serial_port.failed_to_configure	Command to configure serial port failed.	com.serial.set()
error.tag.access_denied	Tag operation not allowed because of tag locks.	Any tag.* command
error.tag.buffer_overflow	Tag operation failed because tag response to a command is too large for the receiver buffer.	Any tag.* command
error.tag.no_tag_in_field	Tag operation failed because no tag was found.	Any tag.* command
error.tag.not_responding	Tag operation failed because tag not responding.	Any tag.* command
error.tag.tag_not_writable	Tag operation failed because tag not writeable.	Any tag.* command
error.tag.unsupported_command	Tag operation failed because the command is not supported by the tag.	Any tag.* command
error.tag.verify_failed	Tag verify operation failed.	Any tag.* command
error.tag.protocol.isoc.insufficient_power	ISO-C tag operation failed because of insufficient power.	Any tag.* command
error.tag.protocol.isoc.memory_locked	ISO-C tag operation failed because memory is locked.	Any tag.* command
error.tag.protocol.isoc.memory_overrun	ISO-C tag operation failed due to no memory.	Any tag.* command
error.tag.protocol.isoc.non_specific	ISO-C tag does not support error-specific codes.	Any tag.* command
error.tag.protocol.isoc.other_error	ISO-C tag operation failed.	Any tag.* command

Table 5-1: Overview: error messages (contd.)

RFI641

**Notes:**

**Australia**

Phone +61 3 9497 4100  
1800 33 48 02 - tollfree  
E-Mail sales@sick.com.au

**Belgium/Luxembourg**

Phone +32 (0)2 466 55 66  
E-Mail info@sick.be

**Brasil**

Phone +55 11 3215-4900  
E-Mail sac@sick.com.br

**Ceská Republika**

Phone +420 2 57 91 18 50  
E-Mail sick@sick.cz

**China**

Phone +852-2763 6966  
E-Mail ghk@sick.com.hk

**Danmark**

Phone +45 45 82 64 00  
E-Mail sick@sick.dk

**Deutschland**

Phone +49 211 5301-270  
E-Mail info@sick.de

**España**

Phone +34 93 480 31 00  
E-Mail info@sick.es

**France**

Phone +33 1 64 62 35 00  
E-Mail info@sick.fr

**Great Britain**

Phone +44 (0)1727 831121  
E-Mail info@sick.co.uk

**India**

Phone +91-22-4033 8333  
E-Mail info@sick-india.com

**Israel**

Phone +972-4-999-0590  
E-Mail info@sick-sensors.com

**Italia**

Phone +39 02 27 43 41  
E-Mail info@sick.it

**Japan**

Phone +81 (0)3 3358 1341  
E-Mail support@sick.jp

**Nederlands**

Phone +31 (0)30 229 25 44  
E-Mail info@sick.nl

**Norge**

Phone +47 67 81 50 00  
E-Mail austefjord@sick.no

**Österreich**

Phone +43 (0)22 36 62 28 8-0  
E-Mail office@sick.at

**Polska**

Phone +48 22 837 40 50  
E-Mail info@sick.pl

**Republic of Korea**

Phone +82-2 786 6321/4  
E-Mail kang@sickkorea.net

**Republika Slovenija**

Phone +386 (0)1-47 69 990  
E-Mail office@sick.si

**România**

Phone +40 356 171 120  
E-Mail office@sick.ro

**Russia**

Phone +7 495 775 05 34  
E-Mail info@sick-automation.ru

**Schweiz**

Phone +41 41 619 29 39  
E-Mail contact@sick.ch

**Singapore**

Phone +65 6744 3732  
E-Mail admin@sicksgp.com.sg

**Suomi**

Phone +358-9-25 15 800  
E-Mail sick@sick.fi

**Sverige**

Phone +46 10 110 10 00  
E-Mail info@sick.se

**Taiwan**

Phone +886 2 2365-6292  
E-Mail sickgrc@ms6.hinet.net

**Türkiye**

Phone +90 216 587 74 00  
E-Mail info@sick.com.tr

**USA/Canada/México**

Phone +1(952) 941-6780  
1 800-325-7425 - tollfree  
E-Mail info@sickusa.com

More representatives and agencies  
in all major industrial nations at  
[www.sick.com](http://www.sick.com)