
# RAIN Communication Interface Guideline

Version: 2018-09

**RAIN RFID Alliance Guideline**

V.1.0

September 2018

# **RAIN RFID Guideline:** RAIN Communication Interface (RCI)

The document is in part and in total owned and managed by the members of the RAIN RFID Alliance.

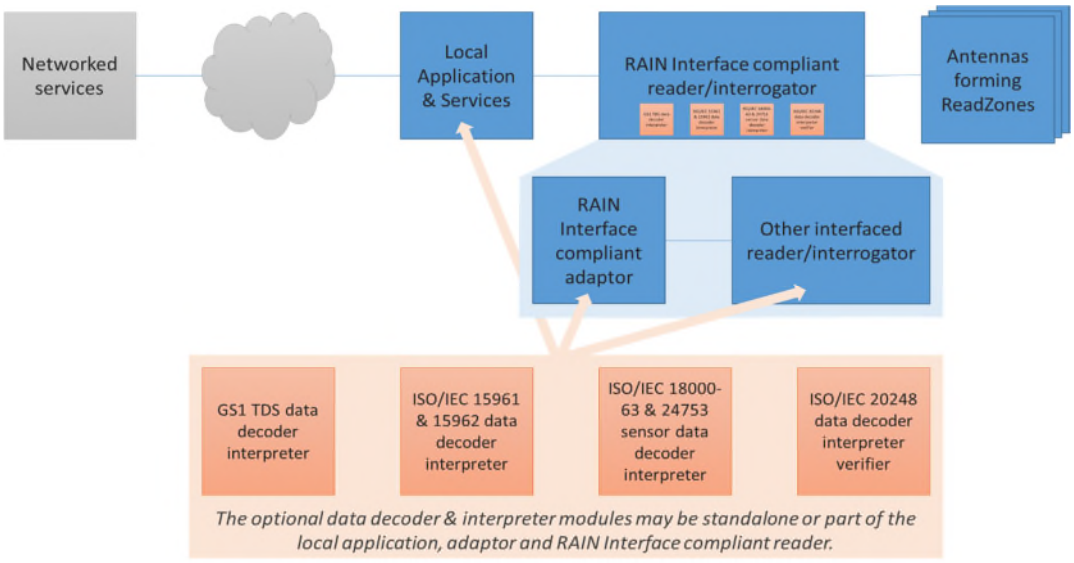
# Contents

# 1 Introduction

The draft proposes an application developer friendly interface between applications and RAIN RFID readers (interrogators). The interface supports readers integrated in automated systems ranging from ultra-lightweight readers to complex multi-antenna readers.

The mechanism of the interface is based on the principle that the application is designed to interact with a predetermined set of tags in a predetermined manner in a predetermined read zone. This interaction is reported to an application as a tag spot event (as in trainspotting). This method allows readers to intelligently use the air protocol to perform the task of tag inventory and access as well as tag data interpretation. Reader vendors are encouraged to optimise the reader for specific applications.

The top line of objects in the figure illustrates a full-function RCI (RAIN Communication Interface) compliant reader within a networked system. Such a reader may be implemented using an RCI adaptor.




Tag data decoding and interpretation forms a key part of the RCI. Code modules to decode and interpret tag data may be used by the application, the adaptor or the reader. Interfaces to such code modules will be developed in harmony with this interface.

An RCI compliant reader is used as follows:

1. Configure the general parameters of the reader ensuring its use complies with the local regulations.
2. Configure the antennas, and their power levels, that will be used to read tags in a specific read zone. This configuration is called a ReadZone.
3. Configure when each ReadZone polls for tags. This may be based on external triggers.
4. Instruct the reader which tags are of interest, when and how to access such tags, and how to report such a spot event. This is called a SpotProfile.
5. Activate the read zones upon which the reader will report the tag spot events.

The interface allows for command driven access of tags and other reader functions.

The interface allows for proprietary commands, command parameters and parameter values.

# 2 Terminology

A familiarity with the RAIN (both ISO/IEC and GS1) air protocol and associated data standards is assumed. The RAIN terminology is used. Other terms, if not defined, use the Oxford English definitions.

This guideline uses the following terminology:

1. **Access or interrogate**: The command/respond actions to communicate with a tag to select, read, write and/or configure the tag in an open or secured method.
2. **Known/wanted tag**: This interface uses the principle that a RAIN reader system is designed to identify specific RAIN tagged items. These tags are identifiable by means of the tag data structure and class data within the data structure. Examples of tags include, a GS1 GTIN tag or an ISO/20248 anti-counterfeit parts tag. Specific tag level identification is achieved using item specific information in the tag. An unknown tag may be a wanted tag, for example un-programmed tags in a production environment. The SpotProfiles makes provision to detect such tags.
3. **RAIN Communication Interface (RCI)**: This guideline.
4. **read**: The command/respond actions to obtain data from a tag.
5. **Reader or interrogator (Rdr)**: The device which communicates with the tag by a method of interrogation. "Reader" and "interrogator" are synonyms in this guideline. This document uses the term reader.
6. **ReadZone (RZ)**: The desired area where tags will be accessed. More than one antenna may be used within a ReadZone; in this case the reader will combine the data from all the ReadZone antennas for the spot report.
7. **Spot**: The report of a tag access, as directed by a SpotProfile, of a tag within a ReadZone; a Spot may be the result of many reads, writes, accesses, and supporting air protocol commands as implemented by the reader.
8. **SpotProfile (Prof)**: A set of parameters instructing a reader which tags to report upon; when, how, and how hard it should try to complete the instructions. A list of SpotProfiles is a *ToDo* list for the reader.
9. **write**: The command/respond actions to store data on a tag.

# 3 Interface description

## 3.1 Principles

The interface assumes that an application knows which tags it wants to access, how to access them and what it needs to know about the access; i.e. it is assumed that the integrator understands the desired outcome and use case of the tag access scenario(s). The application uses this knowledge to configure and task the reader to independently select, inventory and access tags. The result of such actions is reported unilaterally (event driven) by the reader.

The interface supports very simple single antenna readers (e.g. a serial port reader which reads a tag by pressing a button on the reader) up to a complex multi-antenna crypto reader system. The interface assists "out-of-the box" operability, i.e. no configuration required. As such, the interface aims to be useful without the need of an SDK; a simple serial terminal will be able to display reader output and instruct the reader.

The air protocol complexity is contained in the reader while the interface uses the language and thinking of the application/integration side. This is achieved by using fields (<field name>:<field

value>) to communicate between the application and the reader. The same fields are used to configure the reader parameters and for the reader to report command responses and events.

A key focus is placed on the outcome of an interrogation, making the reader responsible for optimizing and managing tag selection and access. This allows reader vendors to optimise readers for specific outcomes and compete based on reader intelligence and performance.

The reader vendor may choose the functional level of implementation. All commands and fields, except basic reader information, basic configuration and tag inventory, are optional. As such, a vendor may choose which fields to support. To ensure proper function all fields are assigned default values. The following actions are taken when a reader or application is confronted with:

1. A "field value" not set ➔ use the default value as specified in this guideline.
2. A "field value" not supported or known ➔ report it as not supported.

Compliance with the interface method is mandatory to ensure future interoperability and interface expansion. The interface makes provision for proprietary commands, fields (parameters) and field values.

The interface provides the following functions:

1. Reader
    a. Reader information and status
    b. Reader configuration
    c. Antenna configuration and grouping into read zones
    d. Expert air protocol configuration
    e. Reader heartbeat
    f. General purpose IO; simple binary switches and multi-values
        i. Output value settings
        ii. Input value report
        iii. Input trigger antenna activation
2. Tag access
    a. August 2018 release:
        i. Inventory (PC, XPC, UII/EPC and simple sensors in binary) with optional select
        ii. SpotProfile directed tag access
        iii. Command driven access using the ThisTag command
        iv. Inventory interpretation (ISO AFI recognition and GS1 EPC)
        v. Basic write
    b. Future releases:
        i. Tag memory locking
        ii. Passwords
        iii. Data interpretation (ISO/IEC 15961 & 15962, ISO/IEC 18000-63 & 24753, ISO/IEC 20248 and GS1 TDS)
        iv. Crypto
        v. BAP & sensors
3. Proprietary functions, fields and field values, see 8.

## 3.2 Data exchange

The following applies:

1. The application instructs a reader with a command and the reader shall report a response.
2. The reader reports events unilaterally.

Note: It is possible that a command is issued when the reader is processing an event. The completion of the event takes priority. It may result in an ignored command. The application should be resilient against such a scenario.

The interface commands are:

1. GetInfo
2. SaveFields, ReadFields, DefaultFields, Reboot, ActivateUpdateMode
3. GetCfg, SetCfg
4. GetRZ, SetRZ, AddRZ, DelRZ
5. GetGPIOs, SetGPIOs
6. GetProf, SetProf, AddProf, DelProf
7. StartRZ, GetActRZ, StopRZ
8. ThisTag, ThisTagStop

The interface reports are:

1. Command responses
2. Error
3. Heartbeat
4. Event – reader event
5. Spot – tag access report

## 3.3 Tag interrogation (inventory and access) method

### 3.3.1 Description

Tags are selected, inventoried and accessed according to the vendor reader configuration, application reader configuration, and the SpotProfiles.

A SpotProfile, see 6.6, tells the reader what additional interrogation actions are to be taken once the UII/EPC has been read during inventory and how the interrogation is to be reported to the application. The SpotProfile also indicates how hard the reader must try to complete the SpotProfile actions.

A SpotProfile is associated with specific tags by means of a selection mask. Tags inventoried but not matching a SpotProfile are counted and ignored.

Tags interrogated within a ReadZone, see 6.4, are reported as spots, see 7.4:

**FirstSeen**: The first time the tag is seen by the reader. FirstSeen reports the full interrogation as directed by the relevant SpotProfile.

**LastSeen** – optional: The tag has not been seen by the reader for a specified period of time (LastSeen timeout). The reader shall "forget" the tag when it reports it as LastSeen**.** A reader shall also report the tag as LastSeen when forced to remove it to make space in its memory for a more recently inventoried tag.

**Seen** – optional: After a periodic interval (Seen interval) the tag is still seen by the reader OR when any tag volatile data has changed since the previous report.  Sensor data is an example

of such volatile data. A sensor data change is indicated by the sensor alarm bit in the XPC word.

Note: A tag's UII/EPC (including the PC word, optional XPC words and optional Simple Sensor Data) are read when it enters the ReadZone and while it remains in the ReadZone during the inventory process. The inventory process allows for some selection of tags.

### 3.3.2 Event driven

Tags are read/interrogated according to a *ToDo* list, the list of SpotProfiles.

SpotProfiles have a mandatory tag select mask and a priority value. The reader shall use the matching SpotProfile with the highest priority to complete an inventoried tag interrogation and report such interrogations as spot(s).

A tag matches a SpotProfile's mask if: the tag's content is in the specified memory bank, from bit address MaskStart for MaskLength bits, has the same value as the leftmost MaskLength bits of the MaskValue. Vendors may decide the length and amount of memory bank masks to support.

Note: See Annex D for an example implementation.

The Application manages the reader's *ToDo* list of SpotProfiles.

The *ToDo* list is optional. When the *ToDo* list is not supported one of the following shall apply:

- If SpotProfiles are not supported, then the default SpotProfile shall be used.
- One SpotProfile is supported which may be configured according to this document.

Note: The number of profiles supported is a key reader performance parameter which should be reported by the reader as part of the reader specification.

The default SpotProfile reports all UII/EPCs (not interpreted; i.e. in binary) inventoried as FirstSeen.

### 3.3.3 Command driven

The application may require immediate access to a specific tag or set of tags. This is facilitated with the ThisTag command, see 6.8. This command contains a list of SpotProfiles to be executed immediately within the ThisTag timeout period.

## 3.4 Tag interrogation (inventory and access) specifics

### 3.4.1 Inventory (PC, XPC, UII/EPC in binary) with optional select

The tag Inventory process provides the UII/EPC in binary. This is the simplest form of reading RAIN tags.

The ability to inventory a tag, distinguish between ISO and GS1 and report it as FirstSeen is a mandatory function.

The reader shall decode the PC word to determine if the data is encoded as ISO or GS1 and report UII for ISO encoding and EPC for GS1 encoding.

In some cases, the UII/EPC may not be able to detect unique tags. The field Uniqueness, see 6.6.2, allows for the inclusion of the TID to determine unique tags.

### 3.4.2 Data interpretation

#### 3.4.2.1 General

The reader shall attempt to interpret the data read from the tag when InterpretData is set to true.

Interpreted data shall be provided as a JSON object with fields (<field name>:<field value>) according to the specification of the specific interpretation.

Data interpretation recommended implementation is on two levels:

1. Inventory interpretation
2. Advance data interpretation as specified by ISO/IEC 15961 & 15962, ISO/IEC 20248 and GS1 TDS. In all cases the latest specifications shall apply.

#### 3.4.2.2 Inventory interpretation (ISO AFI recognition and GS1 EPC)

When inventory data interpretation is supported the reader shall be able to detect and present:

1. ISO AFI value using the "AFI":<AFI in HexString> field

   Example: `":92":<ISO/IEC 20248 data>`

2. GS1 EPC code using the EPC type specified in EncodingType 6.6.1.

   Example: `"SGTIN":<SGTIN binary value>`

3. Simple sensor data when appended to the UII/EPC. This data shall be an object of simple sensor fields as specified in 3.4.4.

#### 3.4.2.3 Advanced data interpretation (ISO/IEC 15961 & 15962, ISO/IEC 20248, ISO/IEC 24753 and GS1 TDS)

For future releases.

### 3.4.3 Basic write

The interface allows two write scenarios:

1. A tag or set of tags is written by the application after it was spotted. This is achieved with the ThisTag command.
2. The application expects a tag or set of tags to be spotted; it pipe-lines the write operation with a SpotProfile.

Sets of tags are handled with the DwnCnt field in the SpotProfile, which facilitates multiple use of the SpotProfile.

The result of a write operation shall be reported as a FirstSeen.

### 3.4.4 BAP & Sensors

For future releases.

### 3.4.5 Crypto Tags

For future releases.

# 4 Interface method

## 4.1 Message format

The guideline uses ISO/IEC 21778 JSON as the interface language. See www.json.org for the specification and www.jsonlint.com JSON data syntax verification.

All data is represented as a JSON object pair (string:value) ➔ <field name>:<field value> called fields.

The interface message is a single JSON object containing a set of fields terminated with an EOL.

The interface message is by default unformatted JSON (no whitespaces, see the JSON format).

```
{<message>}EOL
```

The reader shall be able to handle all types of EOL |= <LF>, <CR>, <LF><CR> or <CR><LF>.

The reader shall ignore all formatting whitespaces (see JSON format rules) received.

Replies from the reader shall use EOL = <CR><LF>.

The application may optionally request the reader to send formatted JSON, see 6.3.2.

Note: communication buffer size limitations may necessitate multiple messages to complete a command or report.

## 4.2 Command message format

The command message has the following format:

```
{"Cmd":<command name>,<command parameter field>…}
```

Note 1: No meaning may be derived from nor inserted by the field order within the <command parameter fields>. The field order may change unpredictably along the data path.

Note 2: Only one command is allowed in a message.

## 4.3 Command response message format

The reader shall respond to each command with the command specific report, see 6. It shall have the following format:

```
{"Report":<command name>,
 "ErrID":<error number>,
 "ErrDesc":<error description>,
 "ErrInfo":<additional error information>
 <command response field>…}
```

The ErrDesc field is optional. It is by default not included in the message, see 6.3.2.

The ErrInfo field is only included when required.

## 4.4 Event report message format

An event report, see 7, message has the following format:

```
{"Report":<report name>,
 <event field>,…}
```

## 4.5 Binary data

Binary data is not natively supported by JSON. Binary data, for this interface, shall be a JSON string with one of the following formats:

1. **HexString**: The character 0 to 9 and A to F (not case sensitive) shall be used to represent binary values $0000_2$ to $1111_2$. The HexString is a continuum of 16-bit words (to align with the air protocol) presented a string of 4 characters precede with the space character (":"). A 16-bit word may be truncated on a 4-bit boundary.

   Example: `":0123:4567:89AB:CDEF:0123:4567:89AB:CDEF"` and `":92"`

2. **Base64String**: The IETF RFC 4648 Base 64 URL shall be used.

   Example: "ASNFZ4mrze8BI0VniavN7w=="

Binary data shall be represented in big-endian on 16-bit words. With big-endian the most significant byte (MSB) value is at the lowest address. The other bytes follow in decreasing order of significance.

The Application shall configure the reader for the report message binary format. HexString is the default.

Applications and readers shall auto distinguish between HexString and Base64String by using the presence of the colon (":") character in the JSON string.

# 5 Interface media

## 5.1 General

The interface communicates over a point to point serial stream carrier media, e.g. an RS-232, Bluetooth, TCP/IP connection, or similar connection.

The interface is particularly well suited for use over web sockets (IETF RFC 6455).

## 5.2 Serial readers

Serial media is directly controlled by the application. There is therefore no requirement for reader discovery and address resolution to be part of this guideline.

The default serial setting shall be: 115,200 bits per second, 8 bits, no parity, 1 stop bit and no flow control.

Serial connections may be unreliable in which case a cyclic redundancy check (CRC) may be optionally added to the message by setting the UseCRC to true with the SetCfg command.

The CRC used shall be the CRC specified by ISO/IEC 18000-63 which is the 16-bit CRC-CCITT International Standard, ITU Recommendation X.25 using the polynomial $x^{16} + x^{12} + x^5 + 1$.

The CRC shall be the last field pair in the message.

The CRC shall be calculated over the highlighted message characters (which includes the first "{" but excludes the ",").

```
{<message>,"CRC":<CRC as a JSON integer number>}
```

Note: The CRC shall be calculated over the transmitted message string. This implies that the CRC field has to be inserted without using JSON processing.

Example: `{"Cmd":"GetInfo","Fields":["ALL"],"CRC":366}`

## 5.3 IP Networked readers

TCP/IP shall be used when connected to an IP network. However, reader discovery and address resolution are beyond the scope of this guideline.

Note: Support for other types of networks is to be determined.

# 6 Commands

## 6.1 GetInfo

Reader information and status cannot be set by the application. The information can be obtained by the application using the following command, or by adding the desired information and status fields to the heartbeat.

```
{"Cmd":"GetInfo","Fields":<array of requested information fields>}
```

Example:

```
{"Cmd":"GetInfo","Fields":["RdrModel","Version"]}
```

The reserved field name "ALL" requests all information fields with their respective values:

```
{"Cmd":"GetInfo","Fields":["ALL"]}
```

The reader shall support GetInfo(All) as specified above. GetInfo for selective fields is optional.

On success the reader shall respond with:

```
{"Report":"GetInfo",<error fields>,<requested information field>…}
```

The following table lists the information fields. Optional fields are indicated by a †.

| Field name | Value type | Notes |
|---|---|---|
| AirProtSet | String | A description of the values and legal combinations for the air protocol settings in 6.3.7 of this specific reader. |
| BootCnt† | Number | This number increments with every reboot. It will wrap at some time. The size is vendor specific, but at least 8 bits long. |
| FreqRegSet | An array of Strings | The FreqRegulation settings available on this reader.<br>Example: ["AU9HA","AU9FA","EU8FA","EU8FB","EU9A"] |
| RdrBufSize | Number | The readers receive buffer size. Messages that are too large to fit in the receive buffer shall be ignored. The value of ReaderBufSize shall be at least 256 bytes. A value of zero means unlimited. |
| RdrModel | String | Vendor specific with hardware version |
| RdrSN | String | Vendor specific product serial number |
| RdrTemp† | Number | Reader temperature in Celsius. |
| RdrTempPA† | Number | Reader power amplifier temperature in Celsius. |
| ReadErrors† | Number | The number of read errors since the value was last read with a Config command or sent in a Heartbeat; e.g. partial reads, timeouts. |
| Reads† | Number | The number of reads executed since the value was last read with a Config command or sent in a Heartbeat. |
| Version | String | Vendor specific firmware version |

| Field name | Value type | Notes |
|---|---|---|
| WriteErrors† | Number | The number of write errors since the value was last read with a Config command or sent in a Heartbeat. |
| Writes† | Number | The number of writes executed since the value was last read with a Config command or sent in a Heartbeat. |

## 6.2 SaveFields, ReadFields, DefaultFields, Reboot, ActivateUpdateMode

It is recommended that a reader store the settings in non-volatile memory.

**SaveFields**: Saves the current field settings to non-volatile memory thereby obtaining assurance that the settings are available after power-off and power-on, and the reader will start as configured.

```
{"Cmd":"SaveFields"}
```

On success the reader shall respond with:

```
{"Report":"SaveFields",<error fields>}
```

**ReadFields:** Reads field settings from non-volatile memory thereby obtaining assurance that all the fields are set with the saved settings.

```
{"Cmd":"ReadFields"}
```

On success the reader shall respond with:

```
{"Report":"ReadFields",<error fields>}
```

**DefaultFields:** Sets all the fields to its default values, this include the deletion of all ReadZone and SpotProfile objects.

```
{"Cmd":"DefaultFields":""}
```

Note: this command shall NOT change the non-volatile memory, allowing the restoration of the saved settings.

On success the reader shall respond with:

```
{"Report":"DefaultFields",<error fields>}
```

**Reboot:** This command reboots the reader. The outcome shall be the same as when a reader is powered down and then powered up.

```
{"Cmd":"Reboot"}
```

The reader shall respond with:

```
{"Report":"Reboot",<error fields>}
```

after which the reader shall reboot.

**ActivateUpdateMode:** This command puts the reader in a firmware update mode. The issue of this command shall NOT change the firmware, but only readies the reader for a firmware update. It is recommended that the reader escapes this state by either receiving a firmware update or by timeout, where after the reader reboots.

Note: Some devices require a hardware intervention to switch to upgrade mode. This command will NOT work in those cases.

The method of firmware update is vendor specific.

```
{"Cmd":"ActivateUpdateMode"}
```

The reader shall respond with:

```
{"Report":"ActivateUpdateMode",<error fields>}
```

where after, if no errors, the reader goes into update mode.

# 6.3 GetCfg, SetCfg

## 6.3.1 Message format

The reader general functions are configured using the fields in the tables below. All these fields are optional. The default values shall apply where the field is not used in the specific RCI implementation or not set by the application.

These commands manage the reader global settings.

**GetCfg** has the same format and operation as **GetInfo** see 6.1.

**SetCfg** has the following format:

```
{"Cmd":"SetCfg",<configuration field (name and value) to be set>…}
```

On success the reader shall respond with:

```
{"Report":"SetCfg",<error fields>}
```

Note: The reader may set a value to an appropriate approximate when the exact configuration value is not supported by the reader. This shall be indicated in the error fields.

## 6.3.2 General reader configuration

The following fields are general reader settings:

| Field name | Value type | Default | Notes |
|---|---|---|---|
| AppBufSize | Number | 0 | The maximum message size (in bytes) for messages that the reader sends to the application. The value of AppBufSize shall be at least 256 bytes. Zero indicates unlimited. |

| Field name | Value type | Default | Notes |
|---|---|---|---|
| Binary | "HEX" or "BASE64" | "HEX" | Selects the binary format.<br>Note: some fields only use the HEX format. Those fields using binary format are explicitly noted so. |
| DateTime | String | - | Set the reader date and time.<br>Use ISO 8601 format:<br>YYYY-MM-DDThh:mm:ss.sssZ<br>No time zone means local time. |
| FormatReports | Boolean | False | When set to true JSON compliant whitespaces are added to improve the human readability of the report message. |
| HBFields | Array of strings | ["RdrName"] | Fields listed here shall be included in the Heartbeat, see 7.2. |
| HBGPIOs | Array of numbers | [] | Report the specified GPIOs within the heartbeat, see 6.5. |
| HBPeriod | Number | 0 | Set the heartbeat period in seconds.<br>Only the start beat is sent if set to zero. |
| RdrDesc | String | "" | Example: "Mouse trap reader – configuration controlled by the mouse trap manufacturing system" |
| RdrLocality | String | "" | Description of reader position.<br>Example: "Dock door 3" |
| RdrName | String | <Company name>-<br><six hex digit random number> | Defines a unique name for this reader. ReaderName should be used with reader discovery.<br>A networked reader should use the last six hex digits of the MAC address for the random number.<br>Example: company-2FEF88 |
| RdrStart | "ACTIVE" or "NOTACTIVE" | "NOTACTIVE" | The ACTIVE setting instructs the reader to executes, at startup, the equivalent of ReadFields and StartRZ(ALL).<br>The NONACTIVE setting instructs the reader shall wait for application commands. |
| ReportErrDesc | Boolean | False | When available, add the error description in the report. |

### 6.3.3 Serial reader specific configuration

The following field is used with serial readers:

| Field name | Value type | Default | Notes |
|---|---|---|---|
| UseCRC | Boolean | False | See 5.2. |

### 6.3.4 IP reader specific configuration

The following fields are used with IP readers:

| Field name | Value type | Default | Notes |
|---|---|---|---|
| DHCP | Boolean | True | If no IP is specified DHCP shall be set to true. |
| IPAddr | String | Configured by the vendor | See IP configuration. Note: Code objects can distinguish between IPv4 and Ipv6.<br>The vendor should label the reader with the default values. |
| IPGateway | String | As IPAddr | As IPAddr |
| IPMask | String | As IPAddr | As IPAddr |
| IPPort | Number | As IPAddr | As IPAddr |
| IPSpotPort | Number | 0 | Optional port on which only the tag events reports are sent.<br>Zero indicates the port is not configured. |

### 6.3.5 Spot report general configuration

The following fields specify the general parameters for Spot report creation, see 7.4:

| Field name | Value type | Default | Notes |
|---|---|---|---|
| LastSeenTO | Number | 1000 | Milliseconds. Determine when to trigger LastSeen. |
| SeenInterval | Number | 1000 | Milliseconds. Determine when to trigger Seen. |
| SpotAnt | Boolean | false | Report the Antenna number resulting in the Spot. |
| SpotDT | Boolean | false | Report the date-time timestamp of the Spot. |
| SpotInvCnt | Boolean | false | Report the inventory count of the Spot. |
| SpotPhase | Boolean | false | Report the phase of the tag access event resulting in the Spot.<br>FirstSeen: The phase of the inventory.<br>Seen & LastSeen: The phase of the last tag access. |
| SpotProf | Boolean | false | Report the SpotProfile number resulting in the Spot. |
| SpotRange | Boolean | false | Report the distance from the reader to the tag of the tag access event resulting in the Spot.<br>FirstSeen: Initial distance.<br>Seen: Current distance<br>LastSeen: Last distance. |
| SpotRSSI | Boolean | false | Report the RSSI of the tag access event resulting in the Spot.<br>FirstSeen: The RSSI of the inventory.<br>Seen & LastSeen: The RSSI of the last tag access. |
| SpotRZ | Boolean | false | Report the ReadZone number resulting in the Spot. |
| ThisTagTO | Number | 1000 | Milliseconds. The maximum duration of ThisTag. |

### 6.3.6 Air protocol general configuration

The following fields configure the air protocol:

| Field name | Value type | Default | Notes |
|---|---|---|---|
| Channel[1] | Number | 0 | Select the channel when a fixed frequency regulation method is selected.<br>Zero indicates the reader shall chose the channel. |
| Freq[1] | Number | 0 | Set the centre frequency when a fixed frequency regulation method is selected.<br>Zero indicates the reader shall chose the frequency. |
| FreqReg[1] | String | Configured by the vendor | See FreqRegSet in 6.1. |
| Mode | "AUTO", "DRM", "HDR" or "MONITOR" | "AUTO" | AUTO means the reader uses its default operating mode.<br>DRM means the reader uses an operating mode compliant to GS1 Dense Reader Mode spectral mask requirements. It is recommended when many readers are operating in the same area.<br>HDR means the reader uses an operating mode for High Data Rates for better performance with small tag populations and in environments with few readers.<br>MONITOR means the reader uses an operating mode to monitors slow changing populations of tags.<br>Note: Mode selection will result in a reader setting vendor defined default values for Session, Target, Q, Tari, BLF, Modulation, DataEncoding, and Preamble. |
| TargetTags | Array of string containing "ALL" or a combination of: "SIMPLE", "READ", "WRITE", "BAP", "SIMPLESENSOR", "SENSOR", "CRYPTO" | ["ALL"] | This field indicates the types of tags the application is interested in. This assists the reader intelligence.<br>SIMPLE: Tags are only inventoried.<br>READ: Tags inventoried also require data to be read from them.<br>WRITE: Tags inventoried also require data to be written to them.<br>BAP: Battery assisted tags may be in the ReadZone.<br>SIMPLESENSOR: Tags may provide simple sensor data.<br>SENSOR: Tags may provide sensor data.<br>CRYPTO: Crypto tags may be in the ReadZone. |

Note 1: It is important that the local frequency regulations are adhered to. This is the responsibility of the vendor, the integrator and the operator of the reader. This interface guideline and information available on the RAIN Alliance website provides guidance. The RAIN Alliance does not take any accountability in contravention of any regulations.

### 6.3.7 Air protocol expert configuration

These fields are used by experts who know and understand the air protocol well.

Note: Certain combinations of settings may contravene local regulations. The field AirProtSet in 6.1 provides guidance.

These settings are typically set by the vendor according to the Mode field, TargetTags field and ReadZone configurations, and intelligence in the reader.

| Field name | Value type | Default | Notes |
|---|---|---|---|
| BLF | Number | Set by Mode | Used to override the value for the reader set by Mode.<br>The value is in the range 40 to 640. The reader shall set and report the closest appropriate value. |
| DataEncoding | "FM0", "M2", "M4", "M8", "M16", "M32", or "M64" | Set by Mode | Used to override the value for the reader set by Mode.<br>The reader shall set and report the value.<br>Note: M16, M32, M64 implies the use of Flex Query instead of Query and only applies to BAP tags. |
| Modulation | "DSB-ASK", "SSB-ASK" or "PR-ASK" | Set by Mode | Used to override the value for the reader set by Mode.<br>The reader shall set and report the value. |
| Preamble | "SHORT" or "LONG" | Set by Mode | Used to override the value for the reader set by Mode.<br>The reader shall set and report the value. |
| Tari | Number | Set by Mode | Used to override the value for the reader set by Mode.<br>The value is in the range 6.25 to 25. The reader shall set and report the closest appropriate value. |

## 6.4 GetRZ, SetRZ, AddRZ, DelRZ

### 6.4.1 General

The following commands and fields configure the read zones. A read zone (ReadZone) is the desired space in which tags are inventoried and the desired (as specified by the SpotProfiles) tags are accessed. Tags accessed are reported as Spots within a ReadZone.

ReadZones are defined with one or more antennas. A ReadZone uses specific power settings and duty cycles to achieve the most efficient tag access results. These settings may apply to individual antennas within a ReadZone.

Antenna and ReadZone identification numbers shall be positive integers. The following shall apply:

- Antennas shall be numbered from one (1) by the vendor.
- It is recommended that ReadZones are numbered from one (1).

The default setting for read zones is one ReadZone (number 1) containing all the antennas of the reader.

Note: The ReadZone list may be implemented as a fixed or variable list. The command makes provision to manage both methods.

**GetRZ**: Obtains the fields and values of a ReadZone.

```
{"Cmd":"GetRZ",
 "ID":<the id of the requested ReadZone>}
```

On success the reader shall respond with:

```
{"Report":"GetRZ",<error fields>,
 <requested ReadZone fields, see 6.4.2 and 6.4.3>}
```

**SetRZ**: Sets the field values of a ReadZone. Fields not changing may be omitted.

```
{"Cmd":"SetRZ",
 "ID":<ReadZone ID>,
 <ReadZone fields and values to be set}
```

On success the reader shall respond with:

```
{"Report":"SetRZ",<error fields>}
```

**AddRZ**: Add a ReadZone record. Only non-default fields need to be included. The ReadZone ID may be omitted or set to zero, in which case the reader shall assign a number.

```
{"Cmd":"AddRZ",
 <ReadZone to be added; fields and values>}
```

On success the reader shall respond with:

```
{"Report":"AddRZ",<error fields>,
 "ID":<the ID of the added ReadZone>}
```

**DelRZ**: Delete ReadZone objects.

```
{"Cmd":"DelRZ",
 "RZ":<the ReadZone ID for deletion>]}
```

On success the reader shall respond with:

```
{"Report":"DelRZ",<error fields>}
```

### 6.4.2 ReadZone fields

The following table lists fields defining ReadZones.

| Field name | Value type | Default | Notes |
|---|---|---|---|
| ID | Positive number | - | The ReadZone ID. |
| Ants | An array of Positive numbers | [0] | Antennas may be in one or more ReadZone.<br>[0] indicates all antennas. |
| ReadPwr | Number | 0.0 | Value is in dBm correct to one tenth. |
| WritePwr | | | |
| DutyCycle | An array of 3 non-negative values | [0,0,0] | The values of the array specify the start delay, ON and OFF durations in milliseconds.<br>Note: vendors may adjust the timing on the fly to ensure proper completion of an air protocol communication. |
| ReadPwrAnt | An array of numbers | All the values in the array are 0.0 | The position of the value corresponds with the antenna list.<br>Value is in dBm correct to one tenth. |
| WritePwrAnt | | | |
| DutyCycleAnt | An array of arrays of 3 non-negative numbers | [[0,0,0]] | The position of the values corresponds with the antenna list.<br>The values of the array specify the start delay, ON and OFF durations in milliseconds.<br>Note: vendors may add a dwell to ensure proper completion of an air protocol communication. |
| StartTrigger | An array of trigger tuples | [[]] | See 6.4.3.<br>When no StartTrigger is defined then the ReadZone shall read tags from when the ReadZone is started, see 6.7.<br>When one or more StartTriggers are specified then the ReadZone shall start reading tags after the ReadZone was started and any of the StartTriggers detected. |
| StopTrigger | An array of trigger tuples | [[]] | See 6.4.3.<br>The ReadZone shall stop reading tags when the ReadZone is stopped, see 6.7.<br>When one or more StopTriggers are specified then the ReadZone shall stop reading tags on detection of any of the StopTriggers. The ReadZone shall remain in the active (started) state. |
| Q | Number | Set by vendor values for a specific Mode | Used to override the value for the reader set by Mode.<br>The integer value is in the range 0 to 15. |

| Field name | Value type | Default | Notes |
|---|---|---|---|
| Session | Number | 0 | Define the session to be used in this ReadZone. |
| Target | "NONE", "A", "B" or "AB" | "NONE" | Select tags using a target. |
| SelectFlag | "NONE", "SL" or "~SL" | "NONE" | Select tags using the selected flag. |

### 6.4.3 ReadZone triggers

A trigger is a GPIO input switch which is used to enable and disable the spotting of tags within an active ReadZone (see 6.7 for ReadZone activation). The ReadZone fields specifies how the tags will be spotted (duty cycles, power levels, antennas, etc).

Each trigger is specified with a tuple of 4 values:

Note: A tuple is an ordered list of elements/values.

| Field name | Value Type | Notes |
|---|---|---|
| InputID | Number | ID of the GPIO input to be used for the trigger.<br>An InputID of 0 indicates ALL inputs. |
| Delay | Number | The trigger execution delay in milliseconds. |
| RisingEdge | Boolean | true for rising edge, false for falling edge. |
| TriggerState | -1, 0, 1 | The trigger execution only takes place when, at the time of the execution:<br>1: the TriggerState is SPOTTING-STARTED<br>-1: the TriggerState is SPOTTING-STOPPED<br>0: the last trigger executed was either |

When ReadZone triggers are defined then the StartRZ command shall set the TriggerState to SPOTTING-STOPPED.

Note 1: Vendors should use appropriate dwell to ensure that air protocol commands are completed.

Note 2: Duty cycles are handled in the RZ specification and the RZ must be active for the triggers to have effect.

Note 3: Duty cycles (ReadZones and Antennas) synchronisation and antenna power matching is left for the vendor and must be encapsulated inside the reader.

Examples:

1. Spot tags while a button is held or an IR beam is broken (say Input 1).

   StartTrigger: [[1,0,true,0]]

   StopTrigger: [[1,0,false,0]]

2. Read for a 2 seconds duration after an IR beam was broken.

   StartTrigger: [[1,0,true,0]]

   StopTrigger: [[1,2000,true,0]]

3. Start spotting tags when the button is pressed, and then stop when it pressed again.

   StartTrigger: [[1,0,true,-1]]

   StopTrigger: [[1,0,true,1]]

4. Start spotting tags when IR beam 1 or 2 is broken, and then stop 1 second after IR beam 3 or 4 is broken.

   StartTrigger: [[1,0,true,-1],[2,0,true,-1]]

   StopTrigger: [[3,1000,true,1],[4,1000,true,1]]

   Depending on the physical configuration the following will also work:

   StartTrigger: [[1,0,true,0],[2,0,true,0]]

   StopTrigger: [[3,1000,true,0],[4,1000,true,0]]

## 6.5 GetGPIOs, SetGPIOs

GetGPIOs, SetGPIOs and RdrEvent is used to set, get, and report on GPIOs. Two types of outputs are catered for: switches and values (typically analogue to digital convertors or control registers).

Switches can be set to ON or OFF. ON means signal active or closed circuit. The default is OFF.

Each GPIO shall be identifiable by a vendor specified unique positive integer number.

GPIOs configurable for input and output shall be numbered independently for input and output. The interface shall NOT provide a method to configure the direction of a GPIO.

**GetGPIOs**: Obtains the values of the listed GPIOs in the order as the request list. The GPIO identifier 0 (ALL) will result in all the available GPIOs to be reported. Get GPIOs also configure when to report the values.

```
{"Cmd":"GetGPIOs","ReportNow":[<GPIO ID>…],
         "ReportEvent":[<GPIO ID>…],
         "ReportHB":[<GPIO ID>…]}
```

Note: any of the lists may be omitted or not supported. For RdrEvent see 7.3 and ReportHB see 7.2.

Example:

```
{"Cmd":"GetGPIOs","ReportNow":[1,3,4],"ReportEvent":[2]}
```

On success the reader shall respond with the ReportNow GPIO tuples:

```
{"Report":"GetGPIOs",<error fields>,
 "GPIOs":[[<GPIO ID>,<GPIO type>,<GPIO value>]…]}
```

With

| GPIO | GPIO type | GPIO value |
|---|---|---|
| Input | "IN" | Boolean – false → (OFF), true → (ON) |
| Output | "OUT" | Boolean – false → (OFF), true → (ON) |
| Analogue 2 Digital | "A2D" | HexString |
| Digital 2 Analogue | "D2A" | HexString |
| Register | "REG" | HexString |

Example:

```
{"Report":"GetGPIOs","ErrID":0,
 "GPIOs":[[1,"IN",true],[2,"A2D",":12"]]}
```

**SetGPIOs**: Sets the GPIO values. The command assumes the application knows the GPIO types by using the GetGPIOs command. As such a GPIO is set using a tuple of 3 values; the ID, the new value and a toggle duration in milliseconds. A zero toggle duration, the default value, shall mean: forever. When the toggle value is non-zero then the GPIO shall switch to the new value and return to the old value after the duration. If the current and new value is the same, then an error is reported.

```
{"Cmd":"SetGPIOs","GPIOs":[<GPIO set tuple>…]}
```

Example where GPIO 1 is a switch and 5 is a register:

```
{"Cmd":"SetGPIOs","GPIOs":[[1,true,500],[5,":A345",0]]}
```

The reader shall respond with:

```
{"Report":"SetGPIOs",<error fields>}
```

## 6.6 GetProf, SetProf, AddProf, DelProf

### 6.6.1 General

The GetProf, SetProf, AddProf and DelProf commands manage the SpotProfile list.

SpotProfiles shall be numbered with a positive integer. The SpotProfile number 0 shall mean ALL.

**GetProf**: Obtains the fields and values of a SpotProfile.

```
{"Cmd":"GetProf",
 "ID":<number of the requested SpotProfile>}
```

On success the reader shall respond with:

```
{"Report":"GetProf",<error fields>,
 <requested SpotProfile fields>}
```

See 6.6.2 and 6.6.3 for more information about SpotProfile fields.

**SetProf**: Sets the field values of a SpotProfile. Fields not changing may be omitted.

```
{"Cmd":"SetProf",
 "ID":<SpotProfile ID>,
 <SpotProfiles fields and values to be set}}
```

On success the reader shall respond with:

```
{"Report":"SetProf",<error fields>}
```

**AddProf**: Adds a new SpotProfile to the list. Only non-default fields need to be included. The SpotProfile number may be omitted or set to zero.

```
{"Cmd":"AddProf",
 <SpotProfiles fields and values to be added}
```

On success the reader shall respond with:

```
{"Report":"AddProf",<error fields>,
 "ID":<number of the added SpotProfile>}
```

**DelProf**: Deletes the listed SpotProfiles. In a static list all the values are defaulted. In a dynamically sized list the SpotProfile is returned to the available memory pool.

```
{"Cmd":"DelProf",
 "Profs":[<list of the SpotProfiles IDs for deletion>]}
```

On success the reader shall respond with:

```
{"Report":"DelProf",<error fields>}
```

### 6.6.2 SpotProfile fields

The following table lists the SpotProfile fields.

| Field name | Value type | Default | Notes |
|---|---|---|---|
| DwnCnt | Number | -1 | The number of times this SpotProfile shall be used to spot unique tags:<br>If DwnCnt is < 0 (typically -1), then this SpotProfile will be used infinite times.<br>If DwnCnt is zero, then do NOT use this SpotProfile.<br>If DwnCnt is > 0 then decrement DwnCnt for each unique tag the SpotProfile has been used for. |

| Field name | Value type | Default | Notes |
|---|---|---|---|
| EncodingType | "ALL", "GS1", "ISO", "SGTIN", "SSCC", "SGLN", "GRAI", "GIAI", "GSRN", "GSRNP", "GDTI", "CPI", "SGCN", "GINC", "GSIN", "GID", "USDOD", "ADI", "BIC" or AFIvalue | "ALL" | EncodingType is used to select tags during inventory.<br>ALL means ignore the encoding type.<br>GS1 report all GS1 encoded tags as indicated by the PC word.<br>ISO report all ISO encoded tags as indicated by the PC word.<br>AFIvalue is a HexString. The AFI is set for ISO tags and is part of the PC word.<br>All other values indicate a GS1 EPC which is part of the EPC encoding. |
| FirstSeen | Boolean | true | If true, the profile reports tags that are seen for the first time. |
| ID | Positive number | - | The SpotProfile ID |
| InterpretData | Boolean | false | If true, the reader shall attempt to interpret the data and report it as fields. |
| LastSeen | Boolean | false | If true, the profile reports tags that are not seen anymore. |
| MBMask | An array of mask tuples | [[]] | An array of masks to be used on a memory bank read. The values are:<br>Memory bank – number (0 to 3)<br>StartBit – number<br>Bit length of the mask – number<br>Bit mask – HexString padded with zero bits (head and tail) to ensure 16-bit word alignment.<br>Bit mask value – HexString padded with zero bits (head and tail) to ensure 16-bit word alignment.<br>Example (mask on additional data in MB01 and in MB11):<br>[[1,128,16,":FFFF",":1234"],<br>[3,0,8,":FF",":11"]] |
| Priority | Positive number | 0 | Priority of the profile. If a tag matches multiple profiles, the profile with the highest priority value will be used. |
| Read | An array of read tuples | [[]] | An array of read access instructions tuples. The values are:<br>Memory bank – number (0 to 3)<br>StartWord – number<br>Words to be read – number<br>Retry limit – number<br>Example (MB10-TID and MB11-User memory):<br>[[2,0,6,3],[23,0,16,1]] |

| Field name | Value type | Default | Notes |
|---|---|---|---|
| ReadZone | Array of numbers | [0] | ReadZone(s) for which this SpotProfile applies.<br>The ReadZone zero, the default, indicates that the SpotProfile applies to all the ReadZones. |
| ReportPC | Boolean | false | Set to true to report PC (including XPC) |
| ReportSensor | Boolean | false | Set to true to report tag sensor values based on the sensor alarm bit. |
| Seen | Boolean | false | If true, the profile reports tags that are seen repeatedly. |
| Write | An array of write tuples | [[]] | See 3.4.3 and 6.6.3 for details. |

### 6.6.3 SpotProfile write fields

The Write field in the SpotProfile, see 6.6.2, specifies the write methods and values using an array of write instruction tuples. The tuple has the following values, in the order of the list:

| Description | Type | Values |
|---|---|---|
| Memory bank to be written | Number | 0 to 3 |
| Start word | Number | - |
| Words to be written | Number | - |
| Retry limit | Number | The maximum number of retries for each write, check and lock operation. |
| Write method | Tuple of values | The write method has the following values:<br>Method – string: "VAL", "RND", "COPY", "DATE", "DT"<br>Method parameters as specified in the following table. |
| Check | Boolean | The default is false; no check.<br>When true the reader shall read the written data to confirm it was correctly written. |
| Lock method | To be finalised | To be finalised |

The following table provides the write method parameters:

| Write method | Write method value(s) | Description |
|---|---|---|
| COPY | [<source MB number>, <source 16-bit word start>] | Copy data from another part of the tag.<br>The default is [1, 0].<br>The MB is indicated with 0, 1, 2 & 3 representing MBs 00, 01, 10 & 11.<br>Example (copy TID to UII/EPC): ["COPY",2,0] |

| Write method | Write method value(s) | Description |
|---|---|---|
| DATE | String | Write a 16-bit word specifying the date. The value is a positive integer with zero indicating 1 January 1970. This provides dates until 2149-06-06. The date format is: "YYYY-MM-DD".<br>Example: "2016-04-18" results in 16909. |
| DT | String | Write the 32-bit Unix Time (epoch) value as specified by ISO 8601. The time instance used shall be taken at the time the write command is executed. The DT format is: "YYYY-MM-DDThh:mm:ss.sssZ". The string may be shortened as specified by ISO 8601, for example "YYYY-MM-DDThh:mm". |
| RND | - | A random number of the size specified will be written and changed each time.<br>Example: ["RND"] |
| VAL | BitString | The exact binary value is written. The binary string shall be padded with zero bits when too short and truncated when too long.<br>Example: ["VAL","kfjhksay9832rfk="] |

Note: Writing to a tag when other tags are also present in the ReadZone may be problematic. The use of an intelligent reader and/or a proper ReadZone design should be considered when writing tags.

# 6.7 StartRZ, GetActRZ, StopRZ

## 6.7.1 General

The ReadZone antenna configuration, power settings, duty cycles, triggers and SpotProfiles are set with the commands described in 6.4 and 6.6.

The reader functions shall be deactivated by default. The field RdrStart, see 6.3.2, configures the default start state of the reader.

When the reader is active the read zones, antennas and GPIO shall behave as configured in 6.4, 6.5 and 6.6.

## 6.7.2 ReadZone activation

**StartRZ:** Activates the listed ReadZones.

```
{"Cmd":"StartRZ","RZs":[<list of ReadZone IDs to be activated>]}
```

On success the reader shall respond with:

```
{"Report":"StartRZ",<error fields>}
```

**GetActRZ:** Lists the active ReadZones.

```
{"Cmd":"GetActRZ"}
```

On success the reader shall respond with:

```
{"Report":"GetActRZ","RZs":[<list of activate ReadZones>]}
```

**StopRZ:** Deactivates the listed ReadZones.

```
{"Cmd":"StopRZ","RZs":[<list of ReadZones to be deactivated>]}
```

On success the reader shall respond with:

```
{"Report":"StopRZ",<error fields>}
```

## 6.8 ThisTag, ThisTagStop

The ThisTag command format is as follows:

```
{"Cmd":"ThisTag","Prof":[<SpotProfile numbers>…]}
```

The SpotProfile numbers are as specified in 6.6

The ThisTag SpotProfile overrides all other SpotProfiles until the ThisTag is successfully completed or the ThisTagTimeout (see 6.3.5) occurs.

An error event shall be reported when ThisTag reaches its timeout without inventorising or spotting a tag.

```
{"Report":"ThisTag",<error fields>}
```

The ThisTag command shall activate the ReadZone(s), as specified by the ThisTag SpotProfile, for the execution of the ThisTag and return all ReadZone(s) to their pre-ThisTag state.

ThisTag shall ignore all triggers of the ReadZone during the execution of the ThisTag command.

A ThisTag execution may be stopped with the following command.

```
{"Cmd":"ThisTagStop"}
```

On success the reader shall respond with:

```
{"Report":"ThisTagStop",<error fields>}
```

# 7 Reports

## 7.1 Error event report

The Error report message is used to report reader event errors.

The format of the "Error" message shall be as follows:

```
{"Report":"Error",
 "ErrID":<error number>,
 "ErrDesc":<description>,
 "ErrInfo":<additional error information>}
```

## 7.2 Heartbeat

The heartbeat provides the status of the reader.

A heartbeat shall be sent on the connection once it has been established. Thereafter the heartbeat is sent every HBPeriod. No further heartbeats shall be sent if the HBPeriod is set to zero.

The format of the Heartbeat report is as follows:

```
{"ReportHB",<heartbeat field>…}
```

Example:

```
{"ReportHB","RdrName":"RAIN-123DEF","BootCnt":123456}
```

Any combination of appropriate reader information and configuration fields (clause 6) may be included in the heartbeat. The HBFields field in 6.3.2 is used to specify which fields to include.

When ReportHB, see 6.5, is set then the GPIO fields are included in the Heartbeat.

## 7.3 Reader event report

The reader may report events unsolicited and at any time. The reports use the relevant command respond formats, e.g. errors, see 7.1 and GPIO changes, see 6.5. In the case where such an event is not specified by a command response the following report shall be used:

```
{"Report":"RdrEvent",<reader event field>…}
```

The GPIO event report has the following format, see 6.5:

```
{"Report":"RdrEvent","GPIOs":[<GPIO tuple>…]}
```

There are currently no other reader events defined.

## 7.4 Tag spot report

The spot of a tag is reported by the reader with the following unsolicited message.

Note: the sequence is NOT important, and fields excluded have an empty or default value):

```
{"Report":"TagEvent",<error fields>,<TagEvent field>…}
```

The following TagEvent fields provides information about the spot. Optional fields are indicated by a † and †† and are excluded by default. These fields are included by using settings in 6.3.5† and 6.6.2††.

| Fieldname | Value type | Default | Notes |
|---|---|---|---|
| Ant† | Number | - | Antenna ID the tag was spotted with. |
| DT† | String | - | Use ISO 8601 format:<br>YYYY-MM-DDThh:mm:ss.sssZ<br>No time zone means local time. |
| DwnCnt | Number | -1 | Defined in 6.6.2. Report after decremented.<br>Only report if ≥ 0. |
| PC†† | HexString | - | PC, XPC_W1 & XPC_W2. |
| Phase†† | Number | - | Phase of the tag signal. |
| Prof† | Number | - | Spot profile ID used to read the tag. |
| Range†† | Number | - | Distance from tag to reader in cm. |
| RSSI†† | Number | - | Received signal strength in dBm. |
| RZ† | Number | - | ReadZone ID the tag was spotted in. |
| Spot | "FirstSeen", "Seen", "LastSeen", "ThisTag" | "FirstSeen" | Spot may be omitted for FirstSeen spots. |

The following TagEvent fields specifies tag data. Tag data errors (read, write and write check) shall be indicated by setting the field value to null.

| Fieldname | Value type | Default | Notes |
|---|---|---|---|
| EPC or UII | Binary | - | The field name for inventoried data:<br>EPC for tags with T=0 in PC word<br>UII for tags with T=1 in PC word |
| MB†† | Array of read tuples | [[]] | Memory bank data read shall be reported in the same sequence as specified in 6.6.2 (Read).<br>The read tuple contains the following values:<br>Memory bank – number (0 to 3)<br>StartWord – number<br>Data – bitstring<br>If a read error occurred the data value shall be set to null.<br>See below for examples. |
| SensorAlarm†† | Boolean | false | Sensor alarm is obtained via XPC word during inventory or can be read from bit 214h in XPC_W1.<br>Only report when true. |
| SSD†† | HexString | - | Simple sensor binary data.<br>Only report when present and selected. |

Examples: The default (and shortest) spot, which is a FirstSeen spot:

An ISO tag: `{"Report":"TagEvent","ErrID":0,"UII":":0123:4567:89AB:CDEF"}`

A GS1 tag: `{"Report":"TagEvent","ErrID":0,"EPC":":0123:4567:89AB:CDEF"}`

Example: Seen spot:

```
{"Report":"TagEvent","ErrID":0,"DT":"2017-09-11T13:06:01.000",
 "Spot":"SEEN","InvCnt":25,"PC":":3592",
 "UII":":0123:4567:89AB:CDEF:89AB:CDEF",
 "MB":[[3,0,":2323:2323:2323:2323"]]}
```

Example: Seen spot with an error (note the application knows the intended operation and as such knows the error type):

```
{"Report":"TagEvent","ErrID":34,"DateTime":"2017-09-11T13:06:01.000",
 "Spot":"SEEN","InvCnt":25,"PC":":3592",
 "UII":":0123:4567:89AB:CDEF:89AB:CDEF",
 "MB":[[2,0,null]]}
```

Note: in this example no XPCs were returned.

# 8 Proprietary functions

Proprietary functions are implemented with proprietary commands, fields and field values. Proprietary fields shall start with an underscore and may be used in all messages.

An example of a proprietary field in the "Config" message:

```
{"Cmd":"SetCfg","_Led":"Red"}
```

An example of a proprietary message:

```
{"Cmd":"_VendorCmd","_VendorValue":100}
```

Proprietary field values should be described in the vendor reader description.

# 9 Reader documentation

It is recommended that a vendor provides a reader specification which includes the information fields and its values, and a list of all the commands, fields and field values a reader supports.

# Annex A Abbreviations

| | |
|---|---|
| ,… | the previous one or more times |
| … | the previous zero or more times |
| Addr | Address |
| AirProt | Air Protocol |
| Ant(s) | Antenna(s) |
| Cfg | Configuration |
| Cmd | Command |
| Cnt | Count |
| CRC | Cyclic Redundancy Check |
| Date | Date and Time are the 32-bit Unix Time (epoch) value as specified by ISO 8601 unless otherwise specified. |
| Del | Delete |
| Desc | Description |
| DT | Date and time |
| DwnCnt | Down Count |
| Err | Error |
| FreqReg | Frequency Regulation(s) |
| HB | Heartbeat |
| ID | Identifier/Index/Number |
| Info | Information |
| Int | Interval |
| Inv | Inventory |
| IP | Internet protocol |
| Len | Length |
| MB | Memory bank |
| Msg | Message |
| Net | Network |
| PC | Protocol Control |
| Prof(s) | SpotProfile(s) |
| Pwr | Power |
| Rdr | Reader |
| RSSI | Read Signal Strength Indicator |
| RZ | ReadZone |
| SN | Serial number |
| Temp | Temperature |
| Time | See Date. |
| TO | Timeout |
| Val | Value |

# Annex B Error numbers and descriptions

The following table provides defined errors. Proprietary errors shall have a negative value.

| Number (ErrID) | Description (ErrDesc) | Optional information (ERRInfo) | Notes |
|---|---|---|---|
| 0 | No error(s) | - | No error on the command when in response to a command.<br>Error condition cleared when reported as an event. |
| 1 | Bad message | JSON string with the bad message | The JSON is not correct or the message is missing parts. |
| 2 | CRC error | JSON string with actual CRC calculated | - |
| 3 | Buffer full | JSON number with the receive buffer size | - |
| 4 | Response too big | JSON number with the transmit buffer size | This may happen when a reader uses a fixed size transmit buffer or run out of memory. |
| 5 | Memory overrun | JSON string: <which memory> | - |
| 6 | Reader too cold | JSON string: <which component> | This may result in inaccurate calibration/settings. |
| 7 | Reader hot | JSON string: <which component> | This does NOT result in a functional termination or malfunction. |
| 8 | Reader too hot | JSON string: <which component> | This will result in a functional termination or malfunction. |
| 20 | Command not supported | JSON string showing which command is not supported | - |
| 21 | Field not supported | Array of strings of not supported fields | - |
| 22 | Field value not supported | Array of strings of fields of which the value is not supported | - |
| 23 | Field value changed | Array of strings of fields of which the value is not supported | The reader may change requested field values to a more appropriate supported value. |
| 30 | GPIO toggle value the same | Array of numbers identifying the GPIO IDs with the problem | A toggle could not be performed. |
| 31 | GPIO not settable | Array of numbers identifying the GPIO IDs with the problem | The GPIO is not an output, D2A or register. |
| 25 | Trigger not an input switch | Array of number identifying the offending GPIO | - |

| Number (ErrID) | Description (ErrDesc) | Optional information (ERRInfo) | Notes |
|---|---|---|---|
| 30 | SpotProfiles full | - | - |
| 31 | SpotProfile error | Array with:<br>JSON number: SpotProfile number,<br>JSON string: <more info> | A spot profile resulted in an air protocol configuration error. |
| 32 | Illegal SpotProfile | Number array listing the illegal SpotProfiles | - |
| 33 | ThisTag timeout | String stating one of the following:<br>• "No tags inventoried."<br>• "No SpotProfile triggered." | No tags were spotted during the ThisTag duration. |
| 34 | Spot error | String describing the error | The spot event could not be completed. |
| 40 | ReadZones full | - | - |
| 41 | ReadZone start error | Array of which the first element is a string describing the start error, followed by numbers indicating the ReadZones with a start error | - |
| 42 | ReadZone definition error | An array listing the offending fields | - |
| ≥1000 | <Proprietary errors> | <Vendor specific> | <Vendor specific> |

# Annex C RAIN RFID 101

This annex provides a summary of air-protocol interrogation and the tag memory organisation.

## C.1 General tag types

The following general tag type may be present in the ReadZone:

**SIMPLE**: Access to these tags is limited the UII/EPC during inventory.

**READ**: More data than the UII/EPC will be read from the tags. It takes longer to complete a tag read interrogation. Often these actions reduce the read range of the tag.

**WRITE**: Data will be written to the tags. Tags need more electric power to store the data permanently. It takes longer to complete a tag write interrogation. Write actions reduce the read range of the tag.

**BAP**: Battery assisted tags may be in the ReadZone. The battery provides electric power for the tag intelligence (and sensors), in so ensuring the optimal read range for all types of interrogations. Battery assisted tags have dedicated commands and modulation to optimise the use of such tags.

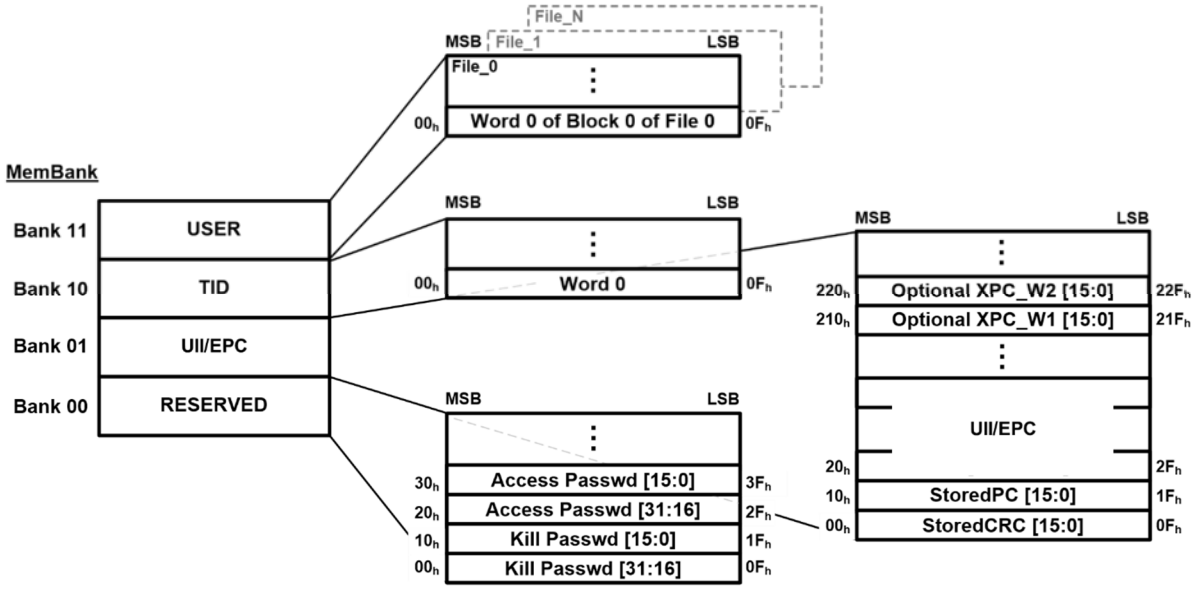
**SIMPLESENSOR**: Tags may provide simple sensor data. Simple sensor data is added to the UII/EPC during inventory as indicated by the protocol

**SENSOR**: Tags may provide sensor data.

**CRYPTO**: Crypto tags may be in the ReadZone. Crypto tags comply with ISO/IEC 29167. Crypto tags have the ability to perform cryptographic functions for tag security outcomes. Cryptographic functions requires more power and time to complete.

## C.2 Tag memory organisation

RAIN tags have the following memory map:




Note: MB and MemBank is an abbreviation of memory bank.

The Protocol Control (PC) word and eXtended PC (XPC) words 1 and 2 provide information on the data received and available on the tag. It also indicates the security access methods for the tag. The PC and optional XPC words are transmitted to the reader during inventory.

Memory Bank 11 (user memory) may be organised in separate data blocks called files. The default configuration is one file, called file 0. The Tag manufacturer chooses where a Tag stores its FileType and FileNum data. The Tag manufacturer also chooses the file-allocation block size (from one to 1024 words). User memory and the files in it may be encoded according to the GS1 EPC Tag Data Standard or to ISO/IEC 15961/15962 and ISO/IEC 20248.

GS1 tags and ISO tags have the same memory bank structure. They do differ in the content of banks 01 (EPC/UII) and 11 (User Memory). The following examples illustrate the differences.

Simple ISO tag with 128 bits UII

| MB-01 PC Bits | | | | | MB-01 UII |
|---|---|---|---|---|---|
| UII len | UserMem | XI | Standard | AFI | UII as specified by the AFI |
| 01000 | 0 | 0 | 1 (ISO) | 8 bits | 128 bits as per UII len |

Simple GS1 tag with 96 bits EPC

| MB-01 PC Bits | | | | | MB-01 UII |
|---|---|---|---|---|---|
| EPC len | UserMem | XI | Standard | RFU | EPC as specified by GS1 |
| 00110 | 0 | 0 | 0 (GS1) | 0x00 | 96 bits as per EPC len |

GS1 or ISO tag with ISO/IEC 15961 & 15962 defined User Memory data

| MB-01 PC Bits | | | | | MB-01 UII | MB-11 User Memory | |
|---|---|---|---|---|---|---|---|
| UII/EPC len | UserMem | XI | Standard ISO\|GS1 | AFI/ RFU | UII/ EPC | DSFID | Data fields according to ISO/IEC 15961 & 15962 |
| 00110 | 1 | 0 | 1 or 0 | 0x00 | *96 bits* | 8 bits | *≥ 0 bits* |

ISO tag with ISO/IEC 20248 defined User Memory data

| MB-01 PC Bits | | | | | MB-01 UII | | | MB-11 User Memory |
|---|---|---|---|---|---|---|---|---|
| UII Len | UserMem | XI | Standard | AFI | DAID | CID | Optional company assigned fields | signature, timestamp Optional company assigned fields |
| 00110 | 1 | 0 | 1 (ISO) | 0x92 | 32, 40 or 48 bits | 16 bits | 48 bits | ≥ 256 bits |

GS1 tag with ISO/IEC 20248 defined User Memory data

| MB-01 PC Bits | | | | | MB-01 UII | MB-11 User Memory | | | |
|---|---|---|---|---|---|---|---|---|---|
| EPC len | UserMem | XI | Standard | RFU | EPC | DSFID | DAID | CID | Signature, timestamp & optional company assigned fields |
| 00110 | 1 | 0 | 1 (GS1) | 0x00 | *96 bits* | 0x11 | *32 or 40 bits* | *16 bits* | *≥ 0 bits* |

ISO tag with a simple sensor

| MB-01 PC Bits | | | | | MB-01 UII | MB-01 Simple Sensor Data | XPC |
|---|---|---|---|---|---|---|---|
| UII len | UserMem | XI | Standard | AFI | UII as specified by the AFI | As specified by ISO | Simple sensor bit set |
| 01000 | 0 | 1 | 1 (ISO) | 8 bits | 128 bits as per UII len | 32 or 48 bits | 16 bits |

## C.3 Air-protocol summary

Tags are energised by the reader. Once energised a tag will listen for a command from the reader. If the command is intended for the tag, the tag will respond by modulating and reflecting the signal received from the reader. The commands form part of 3 basic operations:

1. **Select**. The operation of choosing a tag population for inventory and access. A Select command may be applied successively to select a particular tag population based on user-specified criteria.
2. **Inventory**. The operation of identifying tags. Inventory comprises multiple commands. The result is the PC/XPC word(s), UII/EPC, and CRC from the tag. The PC/XPC bits inform the reader on the availability and access methods of additional information (e.g. sensor and crypto tags).

   Crypto tags inform the reader that they have encrypted information and which crypto suite is to be used to access the information. The application must provide the keys and use the indicated crypto access method to gain access to the protected data and/or verify the tag and/or the data.

3. **Access**. The operation of communicating with (reading from and/or writing to) a tag. An individual tag must be uniquely identified prior to access and a tag access handle obtained. Access comprises multiple commands (using the tag access handle) with multiple results and directed by the application.

## C.4 Sessions and tag targets

The Sessions function is an expert air protocol method to add an additional layer of tag separation where readers are located close together. Other methods are RF shielding and frequency separation. Sessions may also be used to limit the number of times you read the same tag.

The Targets function is an expert air protocol method to silence tags already interrogated. This is very useful in preventing denial of service due to tag-flooding in the ReadZone. Tag-flooding is where there are too many tags for the reader to cope with. It is also very useful to control battery assisted tag responses, since battery assisted tags may have substantially longer read ranges.

## C.5 Air protocol parameters

The air protocol parameters are used to optimise the speed and reliability of the over-the-air communication channel from the reader to the tag and from the tag to the reader. The over-the-air communications are influenced by various environmental and use factors in the read scenario. It must be noted that tags have very little power available (since it harvested electric power from the reader radiation) and a small chip and is therefore limited in the speed and intelligence it can apply to decode reader radio messages. Readers, on the other hand can have all the electric power and intelligence they need to decode the tags' radio messages.

The following air protocol parameters are used

1. **Q**: The Q value is used to optimize the reading speed in relation to the number of tags simultaneously under the field. The higher Q values are good for large number of tags while lower values are good for small population of tags. Valid Q values are between 0 and 15 but typical values range between 3 and 7. Reader vendors typically implement an automatic Q value adjustment algorithm to adapt the reading speed dynamically to the tag population.
2. **Tari**: Tari is a factor used to determine the data link speed to the tag.
3. **BLF**: BLF is a factor used to determine the data link speed from the tag.
4. **Modulation**: The modulation specifies the method used to put data on the RF carrier. RF environments and use cases differ requiring different modulations methods.
5. **Data encoding**: Data encoding specifies the method for encoding the data onto the carrier (as specified by the modulation parameter).
6. **Preamble**: Long or Short; Long for noisy areas.
7. **RSSI**: Receive Signal Strength Indicator - An indicator of how well the reader has seen the tag. Should be interpreted with read count.
8. **Phase**: An RF indicator to assist in the optimisation of the read scenario.

Parameters 2 to 6 specifies the radio modulations. These parameters influence the air link speed and robustness against radio noise. Typically, a higher speed will result in a reduced robustness. Each read scenario should be evaluated carefully for the optimal settings. Successful reader vendor implementations tend to automate and perform this evaluation frequently.

## C.6 Tag passwords

For future releases.

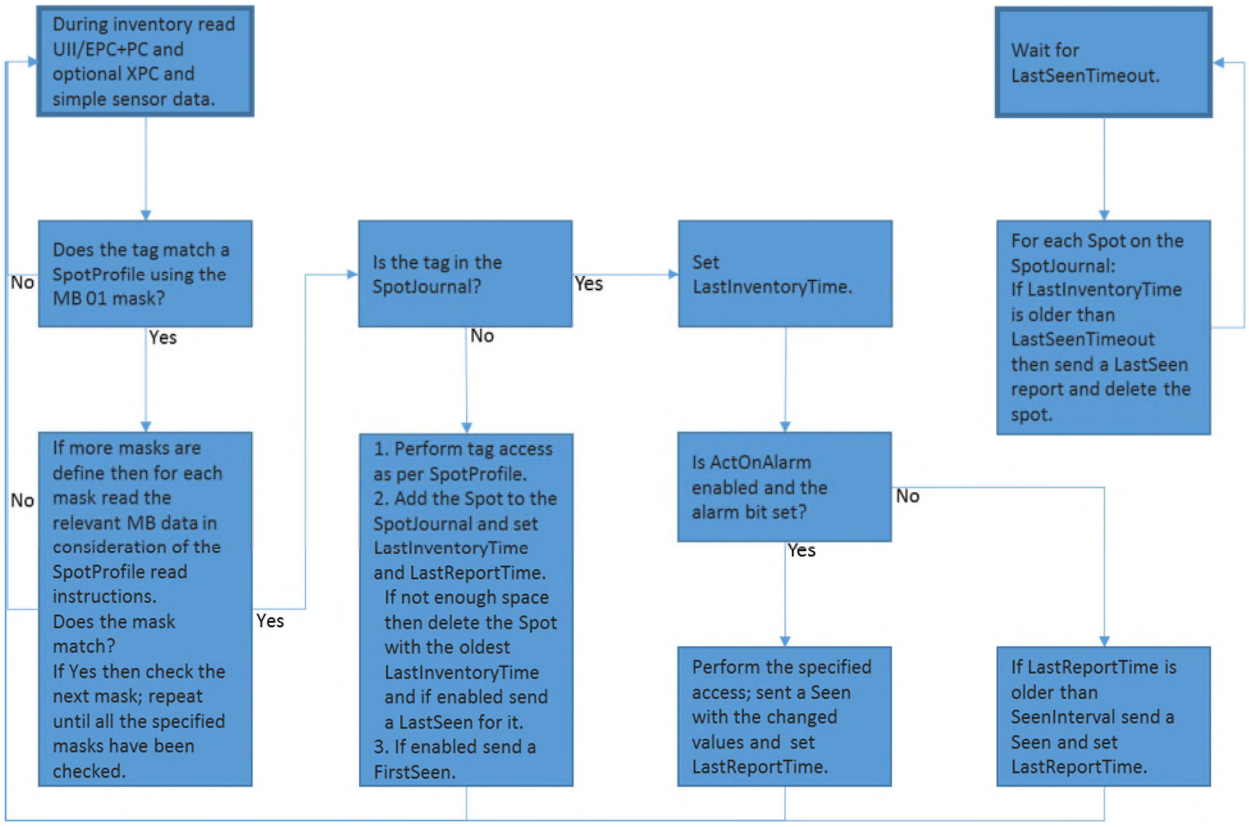
# Annex D Tag spot example implementation

This example implementation makes use of the following data structures:

1. The SpotProfileList ordered in decreasing priority. This can easily be achieved using linked lists.
2. A SpotJournal which contains all the Spots of interest; a Spot which matched a SpotProfile and has not triggered the LastSeenTimeout or has been deleted to make space for a newer spot. Each Spot in the SpotJournal has two timestamps; LastInventoryTime and LastReportTime.

Note: this is only one of many ways to organise the *ToDo* list of SpotProfiles and process an inventoried tag as specified by the relevant SpotProfile. Reader vendors are encouraged to seek and implement the best methods as a competitive edge.

For this example, adding SpotProfiles, processing commands and transmitting Spots and command responses are assumed to happen it their own time in separate processing streams.




The following suggestions are made for a full featured implementation:

1. A multicore control should be used where the cores are assigned to independently perform the following tasks:
    a. Drive the air-protocol.
    b. Perform the configuration and connectivity functions.
    c. Perform the LastSeen pruning and data decoding functions.
    d. Perform the crypto functions.
2. The SpotProfileList, SpotJournal and communications buffer are dual access memory locations controlled with semaphores.

3. The SpotProfileList and SpotJournal are dual-linked lists to assist list traveling optimisation. Fixed format tables are wasteful in memory and in processing resources.

Lean implementations should optimise features against the intended read scenario(s). Note that all the features, except for basic reading and FirstSeen, are optional. Defaults have been chosen to support lean implementations.

# Annex E Contributors

This guideline is developed and maintained by the RAIN Developers Workgroup with the following experts contributing:

| | |
|---|---|
| Bertus Pretorius (Author, Original Proposal, and editor, Co-Chair from 2018-04) | Tönnjes ISI Patent Holding GmbH |
| Bryan Berezdivin | Impinj, Inc. |
| Danny Haak | Nedap Retail |
| David Ciampi | Tönnjes ISI Patent Holding GmbH |
| Georg Michel | Kathrein |
| Harinath Reddy | ThingMagic |
| John T. Armstrong | MonsoonRF, Inc. |
| Lars Thuring (Chair) | Logopak Systeme GmbH & Co. KG |
| Matt Robshaw | Impinj, Inc. |
| Mattias Edlund | TagMaster AB |
| Rob Collins | Impinj, Inc. |
| Stefano Coluccini (Co-Chair up to 2018-04) | CAEN RFID srl |
| Thomas Frederick | Clairvoyant Technology LLC |

# ABOUT RAIN RFID ALLIANCE

The RAIN RFID Alliance is an organization supporting the universal adoption of RAIN UHF RFID technology. A wireless technology that connects billions of everyday items to the internet, enabling businesses and consumers to identify, locate, authenticate and engage each item. The technology is based on the EPC Gen2 UHF RFID specification, incorporated into the ISO/IEC 18000-63 standard. For more information, visit www.RAINRFID.org. The RAIN Alliance is part of AIM, Inc. AIM is the trusted worldwide industry association for the automatic identification industry, providing unbiased information, educational resources and standards for nearly half a century.



# RAIN RFID Alliance

One Landmark North
20399 Route 19
Cranberry Township, PA 16066

Visit the RAIN RFID website – RAINRFID.org. If you are interested in learning more about the RAIN RFID Alliance, contact us at info@rainrfid.org.