RAIN Reader Communication Interface Guideline

RAIN RFID Alliance
Guideline

V.4.0

August 2020

RAIN RFID Guideline: RAIN Reader Communication Interface (RCI)

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

Contents

Revision history

Version 3 to Version 4

1. Add on-chip crypto read support for TagAuth and private/untraceable data for current products which implement ISO/IEC 29167 parts 10 and/or 13.
2. Add GS1 EPC URI interpretation.
3. Add sensor support.
4. Add multi access port support.
5. Add support to report when tags with the same data are detected.
6. Add information on the proper handling of the XPC words.

Version 2 to Version 3

1. Non-technical editorial corrections.
2. Add a clarification of tuple value omission and defaults.
3. Add a description on the order of SpotProfile actions.
4. Add new write methods to create awareness of proper data standards use.
5. Add tag kill.
6. Add access password
7. To assist reading of normal text, bold all defined fields and values.
8. To assist in keeping the guide accessible move the detail of clause 3.3 to annexes.
9. Fix the GPIO error codes.
10. Ensure the defaults for the type of tag events supports the simplest reader which reports all inventories as FirstSeen: Correct and add words to clause 3.3.1 and set the **LastSeenTO** to zero.
11. Remove the GS1 EPCs BIC, GSIN and GINC since they have no meaning within RAIN.
12. Add a note for why two XPC Attribute flags are not supported.
13. Explain **TargetTags** better: 6.3.6 and C1.
14. Add the missing **InvCnt** to the SpotProfile in 7.4 and words in 6.3.5.
15. Add words to specify that the RCI is case sensitive in clause 4.1.
16. Add values and fields to deal with "not configured" (ISO) and "unprogrammed" (GS1) tags.

Version 1 to Version 2

1. Non-technical editorial corrections.
2. Correct the selection masking description in 3.3.2.
3. Add ISO/IEC 20248 Interpretation.
4. Add a length field for error detection over serial communications.
5. Add a method to change the serial communication parameters.
6. 6.3.6: Specify the frequency dimension to be kHz.
7. Add a timestamp which is the epoch value of **DT** (date time) to assist with sensor data fusion in IoT systems.
8. Make **BootCnt** setable by moving it from 6.1 to 6.3.2.
9. Allow **Mode** and **TargetTags** to be specified for a **ThisTag** command, so settings could be altered for that operation, then returned to the previous settings once **ThisTag** was complete.
10. Ensure consistency with the field names for the identifiers of read zones and **SpotProfiles**. All now uses the field **ID**.
11. Add an optional sequence number to the **HB** report.
12. Add an optional command/respond **ID** number.
13. Remove mixed type arrays from reader reports, specifically **MB** in the **Spot** report.
14. Changed copyright date in front page.

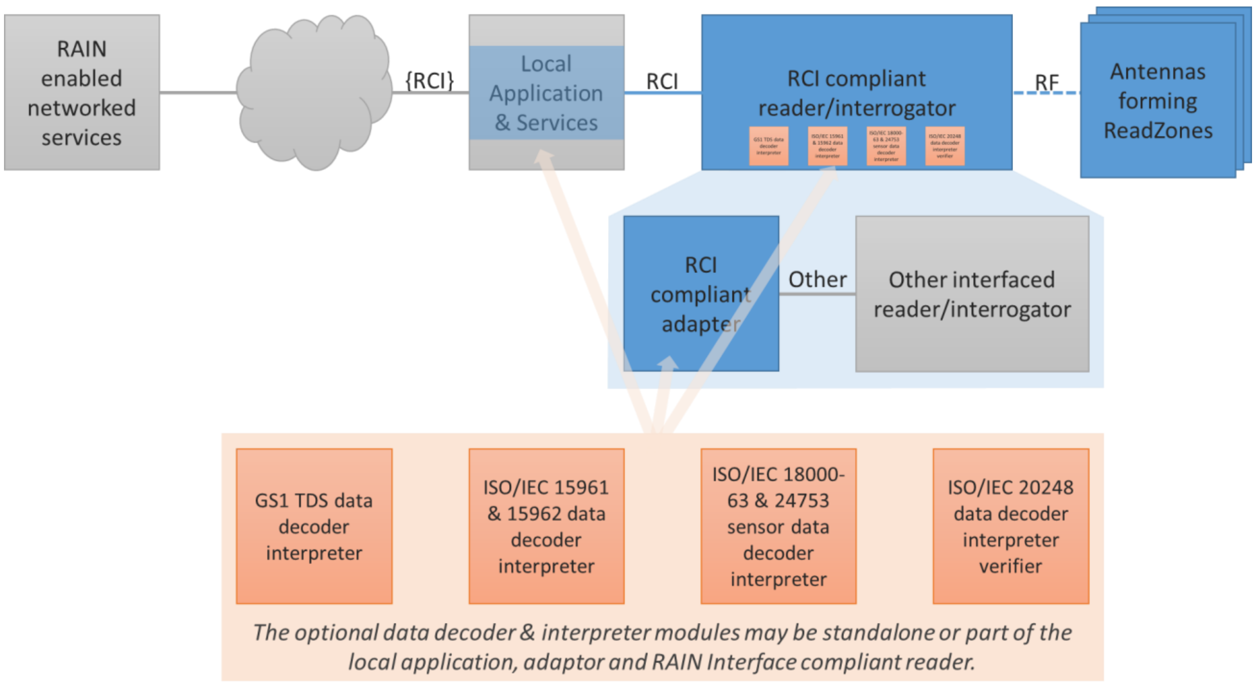
1 Introduction

This document describes an application developer friendly interface between applications and RAIN RFID readers (interrogators). The interface has been designed to accommodate all RAIN RFID reading devices regardless of processing resources and is therefore applicable for a variety of use cases.

The underlying assumption is that the application of the interface is within a deterministic system. It is based on the principle that the application is designed to interact with a predetermined set of tags, in a predetermined manner and within in a predetermined read zone. The interaction between the tag and the reader is reported to an application as an event known as a 'tag spot'. By applying this method readers can focus resources on application specific use-cases, and it allows the efficient use of the RAIN air protocol in performing the necessary tasks of tag inventory, access and data interpretation. This interface allows differentiation of readers per use case and as such reader vendors are encouraged to optimise the devices for specific applications.

Note: The RAIN air protocol is defined by the *ISO/IEC 18000-63* and *GS1 EPC Gen2 specifications*.

The figure below, shows a high-level architecture of the interface. The diagram depicts an RCI (Reader Communication Interface) compliant reader within a networked system.


RAIN enabled networked services
{RCI}
Local Application & Services
RCI
RCI compliant reader/interrogator
RF
Antennas forming ReadZones
RCI compliant adapter
Other
Other interfaced reader/interrogator
GS1 TDS data decoder interpreter
ISO/IEC 15961 & 15962 data decoder interpreter
ISO/IEC 18000-63 & 24753 sensor data decoder interpreter
ISO/IEC 20248 data decoder interpreter verifier
The optional data decoder & interpreter modules may be standalone or part of the local application, adaptor and RAIN Interface compliant reader.

The RCI is designed to be available with several levels of functionality. For example:

- A single reader may implement all the RCI functions as a single device.
- A local application may add RCI functionality (e.g. data interpretation). Such local application typically adds its own functions with RCI embedded within its interface.
- A separate application, an RCI adapter, may convert other reader interfaces to be RCI compliant.
- An RCI adapter may even be used to combine several RCI readers into a single controlled multi-antenna reader.

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

Readers compliant with the RAIN Reader Communication Interface are configured and activated in the following way:

1. Reader Configuration: Configure the general parameters of the reader including any regulatory considerations.
2. **ReadZone** Configuration: Configure individual antenna parameters and combine them into logical read zones (**ReadZone**).
3. Configure **ReadZone** Activation: Configure the conditions with which each **ReadZone** will poll for tags. Activation can be triggered via time-based algorithms or external sensors.
4. Configure **SpotProfile**: create the **SpotProfile** configuration that will include the tags of interest, when and how tags should be accessed and the expected reporting parameters.
5. **ReadZone** Activation: Once all settings are configured, the **ReadZones** are activated. Tag **Spot** reports will be returned following each tag **Spot** event.

The interface has been designed to allow command-driven access to tags and other reader functions including 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. See Annex C for information about RAIN tags such as memory organization and air protocol. Other terms, if not defined, use the Oxford English definitions. This guideline uses the following terminology:

Term	**Meaning**
access or interrogate	The command and response actions to communicate with a tag in order to select, read, write and configure the tag in an open or secured state.
binary	A stream of bits represented in **HexString** or **Base64String**. These binary data representations are specified in 4.5.
known tag or 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 the GS1 GTIN tag and the 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** make provision to detect such tags.
null	No value.
RAIN Reader Communication Interface (RCI)	This guideline.
read	The command/respond actions to obtain data from a tag.

Term	Meaning
reader (Rdr)	The device which communicates with the tag by a method of interrogation (also known as an 'Interrogator').
ReadZone (RZ)	The desired area where tags will be accessed. One or more antennas can be combined to create a **ReadZone**; in which case the reader will combine the data from all the **ReadZone** antennas for the **Spot** report.
SAMEs	A tag which appears to have been reported as another tag, see D.2.
Spot Spot report	The report of a tag interaction event recorded within a **ReadZone**. A **Spot**, as defined by a **SpotProfile**, may be the result of multiple reads, writes, accesses, and other commands as implemented by the reader.
SpotProfile (Prof)	A set of parameters instructing a reader which tags should be reported; it also includes instructions indicating timing, the method and the intenseness with which the instructions should be applied. A list of **SpotProfiles** is a 'to do' list for the reader.
tuple	A tuple is an ordered list of elements/values. Each element/value has a specific meaning. Values may be omitted from the end of the list. In RCI the omitted values shall assume the specified default value.
write	The command and response actions used to record or change data on a tag.
repetition	… indicates a repetition of the previous element one or more times. … indicates a repetition of the previous element zero or more times.
defined terms	To prevent confusion, terms defined by this guideline are **bolded** when used in text.

3 Interface description

3.1 Principles

As an underlying principle of the RCI, it is assumed that applications know which tags the reader should expect, how to perform access on those tags and the desired outcome following the access activity with regards to the specific use case and scenarios. The application uses this knowledge to configure and instruct the reader to independently select, inventory and access tags. The result of such actions is reported unilaterally (event-driven) by the reader.

RCI has been designed to be flexible in terms of the sophistication of systems it supports. It can be utilised with limited functionality readers intended for basic use cases (for example a single antenna serial-port interfaced reader used to read a tag when a physical button is pressed). RCI also supports complex readers with features like multiple antennas, data interpretation and cryptography. The interface assists "out-of-the box" operability as no configuration is 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 within the reader while the interface has been designed to use an intuitive application development language. This is achieved by using fields (<field name>:<field value>) to communicate between the application and the reader. The same field pairs are used for both configuration of the reader parameters and for the reader to report command responses and events. The operations are therefore schema based with a minimum command/response set to configure the operational schema. Specific commands and triggers activate/deactivate the schema. All commands and triggers are atomic and executed in sequence, even when received from multiple connections.

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

Hardware vendors have the flexibility to provide readers with varying levels of functionality as all commands and fields, except basic reader information, basic configuration, and tag inventory, are optional. Consequently, a vendor is free to choose which RCI fields to support. Consistency is ensured by specifying default values for all fields.

If the application is confronted with field values that are either not known or not supported, the following actions are taken:

1. A "field value" not set: The default value as specified in this guideline will be used.
2. A "field value" not supported or known: It will be reported 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
2. Tag access
    a. This release of the RCI guidelines:
        i. Inventory (PC, XPC, UII/EPC, and sensors) with optional select
        ii. **SpotProfile** directed tag access
        iii. Command-driven access using the **ThisTag** command
        iv. Inventory interpretation (ISO AFI UII recognition and GS1 EPC)
        v. Write supporting ISO AFI UII and GS1 EPC
        vi. Data interpretation: Tag use flags, sensors, EPC-URI and ISO/IEC 20248
        vii. Tag memory locking
        viii. Passwords
        ix. Crypto: read-only for ISO/IEC 29167 parts 10 and 13.
        x. **SAMEs** (different tags reported with the same UII/EPC)
    b. Considerations for future RCI guideline releases:
        i. Data interpretation: TID, ISO/IEC 15961 & 15962, and GS1 TDS
        ii. Password diversification.
        iii. Proprietary functions, fields and field values, see 8.

        iv. Crypto extensions

3. General purpose IO

    a. Simple binary switches and multi-values
    b. Output value settings
    c. Input value report

3.2 Data exchange

The following applies:

1. The application instructs a reader with a command; the reader shall issue a report responding to the command.

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 potentially resulting in an ignored command. Applications should be designed to be resilient in the event of 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**
9. **SetConnection**, **GetConnection**

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 (tag interrogation) under the direction of the vendor reader configuration, application reader configuration, and the **SpotProfiles**.

A **SpotProfile** tells the reader what additional interrogation actions are to be taken following the reading of the UII/EPC during inventory; it also describes how the interrogation is to be reported to the application. The **SpotProfile** indicates the effort with which the reader must attempt to complete the **SpotProfile** actions.

Note: RCI allows all reports to be enabled or disabled. Default values for the report and the reporting of it assists the implementation of RCI on restricted platforms.

Tags match a **SpotProfile** using the **SpotProfile** 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 - **LastSeenTO**). The reader shall "forget" the tag following a **LastSeen** event. A reader shall report the tag as **LastSeen** when forced to remove it to make space in its memory for a more recently inventoried tag.
>
> Note: When **LastSeenTO** = 0, the reader shall report all tags every time it has been inventoried with a **FirstSeen** report. As such, the **Seen** and **LastSeen** reports shall not be sent when **LastSeenTO** = 0.
>
> **Seen** – optional: The tag shall be reported after a periodic interval (**Seen** interval) if it is still seen by the reader OR when any tag volatile data has changed since the previous report, e.g. sensor data.

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

3.3.2 Event-driven

All the **SpotProfiles** are combined into a *ToDo* list and it is this list which determines when and how tags are inventoried and accessed.

Each **SpotProfile** contains a mandatory tag selection mask and a priority value. The reader shall use the SpotProfile with the highest priority to complete an inventoried tag interrogation and report such interrogations as **Spots**.

A tag matches a **SpotProfile**, see mask **MBmask** in 6.6.2, when

> all the **MaskValues** of the specified masks have the same value as
> the value derived by a logical AND of the specified tag content with the specified **Mask**.

Note: Vendors may decide the length and number of memory bank masks to support. See Annex D for an example *ToDo* list implementation.

The Application manages the reader's *ToDo* list of **SpotProfiles**; however, the *ToDo* list is optional. When the *ToDo* list is not supported one of the following shall apply:

- If the **SpotProfile** function is not supported, then the default **SpotProfile** values shall be used to inventory and access a tag and report the tag spot.
- If one **SpotProfile** is supported, it shall be configured according to this document.

Note: The number of **SpotProfiles** 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)

During the inventory process the tag transmit the PC word, the optional XPC words and UII/EPC in that order 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.

The reader shall interpret the AFI (ISO indicated) and the EPC header (GS1 indicated) to report the tag with the appropriate AFI, UII field name (**UII**, **UII-NOT-CONFIGURED** and **UII-PROPRIETARY**) and GS1 scheme (GS1 TDS specified schemed, **UNPROGRAMMED** and **TID**).

Note: **UNPROGRAMMED** and **TID** are not EPC schemes, they are special EPC header values. AFIs 0x01 to 0x07 are specified by ISO/IEC 15961 for closed loop systems, in other words, a proprietary system.

3.4.2 Additional tag access

Additional tag access is achieved by using **SpotProfiles** in an active **ReadZone** or with the **ThisTag** command.

3.4.3 Writing data to a tag

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 Data interpretation

Data interpretation is configured as part of a **SpotProfile**, see 6.6.2.

The RCI data interpretation levels:

1. Inventory interpretation deals with data which is provided during the inventory of the tag. This data is defined by the *ISO/IEC 18000-6*3 and *GS1 EPC Gen2 specifications*.

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

The reader shall attempt to interpret the data read from the tag when **InterpretData**, see 6.6.2, is set to a value it supports. The interpretation specific configuration shall be supplied using the interpretation identifier as part of the **SpotProfile**. The format is as follow:

```
<interpreted data identifier>:<data interpretation configuration>
```

Example: `"20248":{"DDDdataTagged":true,"Timezone":"+1000","Language":"en"}`

If a reader does not support the requested interpretation, then the reader shall respond to the applications with a field value error message listing the interpretations it supports.

Interpretation information and errors shall be provided within the **ResponseCode** value object. The response code number zero (0) shall indicate a successful data interpretation.

The guideline makes provision for the following data interpretations:

Identifier	**Specification**	**Notes**
"TAGUSE"	ISO/IEC 18000-63	See Annex E.2
"20248"	ISO/IEC 20248	See Annex F
"EPC-URI"	GS1 TDS	See Annex G
"SIMPLESENSOR"	ISO/IEC 18000-63	See Annex 0
"SNAPSHOTSENSOR"	ISO/IEC 18000-63	See Annex H.3

Interpreted data shall be inserted in the **Spot** report using the following format:

```
<interpreted data identifier>:{"ResponseCode":<response code>,
                              <interpreted data>}

"ResponseCode":{"Code":<Error code - Number>,
                "Desc":<Error description - String>}
```

Example:

```
"20248":{"ResponseCode":{"Code":0,"Desc":"OK"},"DDDdataTagged":{…}}
```

Each Interpretation shall implement the specific specification error codes as per its standard specification. Where such error codes are not specified the following error codes shall be used:

Code	**Desc**
0	"OK"
1	"Failed"

The Interpretation value shall be set to **null** when the interpretation module fails or become not available.

Example: `"20248":null`

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.

JSON is case sensitive, as such the RCI shall be case sensitive.

All data is represented as a JSON object pair, called a "field", which is depicted by a string and a corresponding value separated by a colon ':' in the following manner:

```
<field name>:<field value>
```

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 and therefore contains 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

An RCI reader operation is configured using atomic commands which is executed sequentially. Meaning, when multiple connections each send a command at the same time, they will be processed one at a time in sequence as received.

The command message has the following format:

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

CmdID is optional. The value is a number to uniquely identify a command instance. The reader shall, if capable, repeat **CmdID** and its value in the response message of the command instance.

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: Messages shall contain only one command.

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

The **CmdID** and **ErrInfo** fields are optional.

4.4 Event report message format

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

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

4.5 Binary data

Binary data is not natively supported by JSON. Therefore, within this interface, binary data shall be represented by a JSON string with one of the following two formats:

3. **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 as a string of 4 characters preceded by the colon (":"). A 16-bit word may be truncated on a 4-bit boundary.

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

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

   Example: "`ASNFZ4mrze8BI0VniavN7w==`"

Note: In this guide the term "binary" indicates a binary representation by configuration. The use of **HexString** or **Base64String** specifies that specific representation.

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.

The default serial setting shall be:

- 115,200 bits per second,
- 8 bits, no parity,
- 1 stop bit, and
- no flow controls.

As serial connections can sometimes be unreliable, a cyclic redundancy check (**CRC**) and/or length (**Len**) field may be optionally added to the message. This is completed by setting the fields **UseCRC** and **UseLen** to true using the **SetCfg** command, see 6.3.3.

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

Where a CRC is used, it shall be the CRC specified by ISO/IEC 18000-63. This 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 calculated starting with the first "{" to the end of the <message> excluding the ",".

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

with the CRC calculated over `{"Cmd":"GetInfo","Fields":["ALL"]`

The length shall be calculated over the full length of the message including the EOL, since it is easy to add the length of the length field and its value. This shall represent the number of bytes received from the serial port.

Example:

```
{"Report":"TagEvent","MB":[{"ID":2,"Start":0,"Data":":E280:1105:2000:3693:E0D8:00
12"},{"ID":3,"Start":0,"Data":":0531:57BA:5BDB:67B3:B28F:7DA9:4944:FCD1:C509:FDA0
:BBB7:D45A:1C49:C4B0:6A62:28A6"}],"UII":":0B4F:7500:6B12:199D:39C7:4104:7800","Ti
mestamp":1540208055.450914,"PC":":4592:C098","ErrID":0,"DT":"2018-10-
22T11:34:15.450914","Len":303}
```

5.3 IP Networked readers

TCP/IP shall be used when connected to an IP network.

Note: 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 a periodic **Heartbeat** report.

```
    {"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 of this specific reader, see 6.3.7.
FreqRegSet	An Array of Strings	The **FreqRegulation** settings available on this reader. 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 with a value of zero meaning an unlimited size.
RdrModel	String	Vendor specific with product 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 reported with 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 reported with a **Heartbeat**.

Field name	Value type	Notes
Version	String	Vendor specific firmware version.
WriteErrors†	Number	The number of write errors since the value was last read with a **Config** command or reported with a **Heartbeat**.
Writes†	Number	The number of writes executed since the value was last read with a **Config** command or reported with a **Heartbeat**.

6.2 SaveFields, ReadFields, DefaultFields, Reboot, ActivateUpdateMode

It is recommended that readers should store all 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 the default values. This includes 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, after which 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>}
```

if there are no errors, the reader shall then go into update mode.

6.3 GetCfg, SetCfg

6.3.1 Message format

A reader's 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. Note: Some fields only use the HEX format. Those fields using binary format are explicitly noted.
BootCnt	Number	-	This integer increments with every reboot. The size of this number is vendor specific (but should be at least 8 bits long); therefore, once the limit has been reached, the counter will restart.
DateTime	String	-	Set the reader date and time using the ISO 8601 format: YYYY-MM-DDThh:mm:ss.sssZ No time zone means local time. Note: A reader without a real-time clock will start at epoch-zero. Not setting the date-time will therefore indicate "up-time" of the reader, which is easily detectable due to its very low epoch value. Note: Fraction setting of date-time will ensure proper time synchronisation.
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. Only the start beat is sent if set to zero.
RdrDesc	String	""	The description of the reader. Example: "Mouse trap reader – configuration controlled by the mouse trap manufacturing system"
RdrLocality	String	""	Description of reader position. Example: "Dock door 3"
RdrName	String	<Company name>-<six hex digit random number>	Defines a unique name for this reader. **RdrName** should be used with reader discovery. A networked reader should use the last six hex digits of the MAC address for the random number. Example: company-2FEF88
RdrStart	"ACTIVE" or "NOTACTIVE"	"NOTACTIVE"	The ACTIVE setting instructs the reader to execute, at startup, the equivalent of **ReadFields** and **StartRZ**(ALL). The NONACTIVE setting instructs the reader to wait for application commands.

Field name	Value type	Default	Notes
ReportErrDesc	Boolean	False	When available, this will add the error description into 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.
UseLen	Boolean	False	See 5.2.
SerCfg	A tuple	[115200,8, "n",1,"n"]	The serial communication settings: BAUD, character bits, parity ("n": none, "o": odd and "n": even), stop bits, flow control ("n": none, "r": CTS/RTS and "x": XON/XOFF).

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, then DHCP shall be set to true.
IPAddr	String	Configured by the vendor	See IP configuration. Note: Code objects can distinguish between IPv4 and Ipv6. 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

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	0	Milliseconds. A **LastSeen** report is sent when a tag has not been seen for this duration.
SeenInterval	Number	1000	Milliseconds. The interval on which the **Seen** report is sent.
SpotAnt	Boolean	false	Report the antenna number resulting in the **Spot**.
SpotDT	Boolean	false	Report the date-time timestamp of the **Spot** in a human readable format using the field **DT**.

Field name	**Value type**	**Default**	**Notes**
SpotInvCnt	Boolean	false	Report the inventory count (the amount of times the tag was inventoried since the last report) of the **Spot**.
SpotPhase	Boolean	false	Report the phase of the tag access event resulting in the **Spot**. **FirstSeen**: The phase of the inventory. **Seen** & **LastSeen**: The phase of the last tag access.
SpotProf	Boolean	false	Report the **SpotProfile** ID 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. **FirstSeen**: Initial distance. **Seen**: Current distance **LastSeen**: Last distance.
SpotRSSI	Boolean	false	Report the Received Signal Strength Indicator (RSSI) of the tag access event resulting in the **Spot**. **FirstSeen**: The RSSI of the inventory. **Seen** & **LastSeen**: The RSSI of the last tag access.
SpotRZ	Boolean	false	Report the **ReadZone ID** resulting in the **Spot**.
SpotTS	Boolean	false	Report the date-time timestamp of the **Spot** as the epoch unsigned integer using the field **TimeStamp**.
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. Zero (0) indicates the reader shall choose the channel.
Freq[1]	Number	0	Set the centre frequency when a fixed frequency regulation method is selected. The frequency shall be set in kHz as an integer value. Zero (0) indicates the reader shall choose the frequency.
FreqReg[1]	String	Configured by the vendor	See **FreqRegSet** in 6.1.

Field name	Value type	Default	Notes
Mode	"AUTO", "DRM", "HDR" or "MONITOR"	"AUTO"	**AUTO** means the reader uses its default operating mode. **DRM** means the reader uses an operating mode compliant to the GS1 Dense Reader Mode spectral mask requirements. This is recommended when many readers are operating in the same area. **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. **MONITOR** means the reader uses an operating mode to monitor slow changing populations of tags. 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", "ALARMSENSOR", "SNAPSHOTSENSOR", "SIMPLESENSOR", "FULLSENSOR", "CRYPTO"	["ALL"]	This field indicates the tag types of interest to the application. This assists the reader intelligence, see C.1.

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 only. The RAIN Alliance does not take any responsibility in event of contravention of regulations.

6.3.7 Air protocol expert configuration

The following fields are intended to be used by experts who fully understand the air protocol.

Note: Certain combinations of settings may contravene local regulations. The field **AirProtSet**, see 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. The value is in the range 40 to 640. The reader shall set and report the closest appropriate value.

Field name	Value type	Default	Notes
DataEncoding	"FM0", "M2", "M4", "M8", "M16", "M32", or "M64"	Set by Mode	Used to override the value for the reader set by Mode. The reader shall set and report the value. Note: **M16**, **M32** and **M64** imply the use of Flex Query instead of Query and only apply 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. 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. The reader shall set and report the value.
Tari	Number	Set by Mode	Used to override the value for the reader set by Mode. 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 applies:

- **Antennas** shall be numbered from one (1) by the vendor.
- **ReadZones** shall be numbered from one (1).

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

Note 1: A reader may support no **ReadZones**; the above defaults apply. A reader may also support a singular **ReadZone** in which case only **GetRZ** and **SetRZ** can be used. In this case the **ID** field may be omitted and/or ignored since the default value of 1 for **ID** shall apply.

Note 2: 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 - the default is 0 meaning all>,
 <ReadZone fields and values to be set>}
```

Example: `{"Cmd":"SetRZ", "ReadPwr":20.0}` set all the **ReadZone**s' read power to 20.0 dBm.

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",
 "ID":[<list of the ReadZone IDs for deletion; 0 is not allowed>]}
```

The default **ReadZone ID** 0 is not allowed to prevent the accidental deletion of all the **ReadZones**.

On success the reader shall respond with:

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

6.4.2 ReadZone fields

The following table lists fields for defining a **ReadZone**.

Field name	**Value type**	**Default**	**Notes**
ID	Positive number	-	The **ReadZone** identification number.
Ants	An array of Positive numbers	[0]	Antennas may be in one or more **ReadZone**. [0] indicates all antennas.
ReadPwr	Number	0.0	Value is in dBm correct to one tenth (0.1 dBm).
WritePwr			

Field name	**Value type**	**Default**	**Notes**
DutyCycle	An array of 3 non-negative values	[0,0,0]	The values of the array specify: • the start delay (ms), • ON duration (ms), and • OFF duration (ms). Note: Vendors may dynamically adjust the timing to ensure proper completion of an air protocol communication.
ReadPwrAnt WritePwrAnt	An array of numbers	All the values in the array are 0.0	The position of the value corresponds with the antenna list. Value is in dBm correct to one tenth (0.1 dBm).
DutyCycleAnt	An array of arrays of 3 non-negative numbers	[[0,0,0]]	The position of the values corresponds with the antenna list. The values of the array specify: • the start delay (ms), • ON duration (ms), and • OFF duration (ms). 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. When no **StartTrigger** is defined, then the **ReadZone** shall read tags from when the **ReadZone** is started, see 6.7. When one or more **StartTriggers** are specified, then the **ReadZone** shall start reading tags after the **ReadZone** was started and any of the **StartTriggers** was detected.
StopTrigger	An array of trigger tuples	[[]]	See 6.4.3. The **ReadZone** shall stop reading tags when the **ReadZone** is stopped, see 6.7. 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. The integer value is in the range 0 to 15.
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 as in the following table:

Field name	**Value Type**	**Notes**
InputID	Number	The **ID** of the **GPIO** input to be used for the trigger. 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: 1: the **TriggerState** is SPOTTING-STARTED -1: the **TriggerState** is SPOTTING-STOPPED 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 cycle (**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 (for example using 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 either IR beam 1 or 2 is broken, and then stop 1 second after either 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 same order as the request list. The **GPIO** identifier 0 (**ALL**) will result in the values of all the available **GPIOs** to be reported. **GetGPIOs** 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. A **GPIO** is set using a tuple of 3 values: the **ID**, the new value, and a toggle duration in milliseconds. The default toggle duration value is zero (0) and is interpreted as indefinitely. 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: **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 zero (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 - the default is 0 meaning all>,
 <SpotProfiles fields and values to be set}}
```

Example: The following command may be used to set a field AAA to a value "bbb" in all the existing SpotProfiles:

```
{"Cmd":"SetProf", "AAA":"bbb"}
```

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**.

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

The **ID** default value ([0] meaning **ALL**) is not allowed to prevent accidental deletion of the Profiles.

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**
AccessPWD	[<32-bit password binary>]	[""]	When the value is a **null**, an empty tuple or the password binary is not 32 bits, then no access actions shall be attempted.
DwnCnt	Number	-1	The number of times this **SpotProfile** shall be used to spot unique tags: If **DwnCnt** is < 0 (typically -1), then this **SpotProfile** will be used infinite times. If **DwnCnt** is zero, then do NOT use this **SpotProfile**. 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", EPC Schemes: "SGTIN", "SSCC", "SGLN", "GRAI", "GIAI", "GSRN", "GSRNP", "GDTI", "CPI", "SGCN", "GID", "USDOD", "ADI" EPC header values: "TID", "RFU", "UNPROGRAMMED" or AFIvalue	"ALL"	**EncodingType** is used to select tags during inventory. **ALL**: Ignore the encoding type. **ISO**: Report all ISO encoded tags as indicated by the PC bit T=1. **GS1**: Report all GS1 encoded tags as indicated by the PC bit T=0. EPC Schemes and EPC header values: values as specified in GS1 TDS table 14-1. 0x00: **UNPROGRAMMED** 0x2C to 0x41: Specified EPC Schemes Note: GS1 TDS may expand this list. 0xE0, 0xE2: **TID**, values reserved to avoid confusion when the TID in MB10 is copied to MB01 starting at bit 0x020. All other values: **RFU** **AFIvalue** is an 8-bit **HexString**; the second byte of the PC word when PC bit T=1.
FirstSeen	Boolean	true	If true, the profile reports tags that are seen for the first time.
ID	Positive number	-	The **SpotProfile** ID
InterpretData	Array of Objects	[]	The array shall contain the interpretation objects as specified in 3.4.4.
KillPWD	[<32-bit password binary>]	[""]	When the value is a **null**, an empty tuple or the password binary is not 32 bits, then no kill action shall be attempted.
LastSeen	Boolean	false	If true, the profile reports a tag no longer in the **ReadZone**. After the report, that tag is forgotten. **LastSeen** shall not be reported when **LastSeenTO** = 0.
MBMask	An array of mask tuples	[[]]	An array of mask tuples to select which spots this profile applies to. The values are: Memory bank – number (0 to 3) StartBit – number Bit length of the mask – number Bit mask – **HexString** padded with zero bits (head and tail) to ensure 16-bit word alignment. Bit mask value – **HexString** padded with zero bits (head and tail) to ensure 16-bit word alignment. Example (mask on additional data in MB01 and in MB11): [[1,128,16,":FFFF",":1234"], [3,0,8,":FF",":11"]]

Field name	Value type	**Default**	**Notes**
Priority	Positive number	0	Priority of the profile. If a tag matches multiple profiles, the profile with the highest priority value will be used.
PrivateData	Object	{}	See Annex I.1.
Read	An array of read tuples	[[]]	An array of read access instructions tuples. The values are: Memory bank – number (0 to 3) StartWord – number Words to be read – number Retry limit – number Example (MB10-TID and MB11-User memory): [[2,0,6,3],[3,0,16,1]]
ReadZone	Array of numbers	[0]	**ReadZone**(s) for which this **SpotProfile** applies. The value zero (0), the default, indicates that the **SpotProfile** applies to all the **ReadZone**s.
ReportPC	Boolean	false	Set to true to report the PC and XPC words with the **PC** field.
ReportSAMEs	Boolean	false	Set to report detected **SAMEs**, see Annex D.2
ReportSensor	Boolean	false	Set to true to report tag sensor values.
Seen	Boolean	false	If true, the profile reports tags that are seen repeatedly. **Seen** shall not be reported when **LastSeenTO** = 0.
TagAuth	Object	{}	See Annex I.1.
Write	An array of write tuples	[[]]	See 6.6.3 for details.
WriteEPC	Tuple	[]	The PC numbering toggle bit (T) shall be set to T=0. See 6.6.3 for details.
WriteUII	Tuple	[]	The PC numbering toggle bit (T) shall be set to T=1. See 6.6.3 for details.
WriteUM	Tuple	[]	See 6.6.3 for details.
WriteAccessPWD	Tuple	[]	32-bit binary value. See 6.6.3 for details.
WriteKillPWD	Tuple	[]	32-bit binary value. See 6.6.3 for details.
WriteRetries	Number	3	The maximum write retries.

6.6.3 SpotProfile write field definitions

This clause specifies the write fields values and defaults. A **SpotProfile** may contain any combination of these write fields. The reader may execute the write fields in any order. **WriteAccessPWD** should be executed last.

Note 1: Some combinations of write fields are in conflict.

Note 2: Vendors are dissuaded from implementing only **Write**. **WriteUII**, **WriteEPC** and **WriteUM** should be the priority write methods.

Write field with tuple definition	**Default tuple values**
"Write":[[<MB>,<start>,[<method>],<check>,<lock>]…]	[2,0,["VAL",""],false,"NO-CHANGE"]
"WriteEPC":[<binary>,<check>,<lock>]	["",false,"NO-CHANGE"]
"WriteUII":[<binary>,[<AFI>,<DSFID>],<check>,<lock>]	["",[":01",""],false,"NO-CHANGE"] where a DSFID = "" means there is no DSFID.
"WriteUM":[<binary>,<DSFID>,<check>,<lock>]	["","",false,"NO-CHANGE"] where a DSFID = "" means there is no DSFID.
"WriteAccessPWD": [<binary>,<check>,<lock>]	["",false,"NO-CHANGE"]
"WriteKillPWD": [<binary>,<check>,<lock>]	["",false,"NO-CHANGE"]

The binary length shall be derived from the binary string length for **Write**, **WriteEPC**, **WriteUII** and **WriteUM**. The binary shall be 32 bits for **WriteAccessPWD** and **WriteKillPWD**.

A write error shall be indicated with a "Write error" message.

The following table defines the write methods' tuple vales:

Tuple value	**Type**	**Values**
AFI	Binary	8 bits
binary bitstring to be written	Binary	Any length. The reader shall pad the binary with zero bits to a 16-bit word boundary.
check	Boolean	An error shall be reported when the check fails. Note: A password locked as NO-ACCESS cannot be checked.
DSFID	Binary	Multiples of 8 bits.
lock	String	Data lock |= NO-CHANGE, OPEN, SECURED, PERMALOCKED, PERMAUNLOCKED with the default NO-CHANGE Password lock |= NO-CHANGE, OPEN, SECURED, PERMALOCKED, NO-ACCESS with the default NO-CHANGE
MB	Number	0 to 3
start word	Number	-
Write method	Tuple of values	The write method has the following values: Method – string: "VAL", "RND", "COPY", "DATE", "DT" Write method parameters as specified in the next table.

The following table provides the write method parameters:

Method	**Write method value(s)**	**Description**
COPY	[<source **MB** number>, <source 16-bit word start>]	Copy data from another part of the tag. The default is [1,0]. The **MB** is indicated with 0, 1, 2 & 3 representing MBs 00, 01, 10 & 11. Example: (copy TID to UII/EPC): ["COPY",2,0]
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". 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. Example: ["RND"]
VAL	Binary	The binary value to be written. The reader shall pad the binary with zero bits to a 16-bit word boundary. Examples: • ["VAL","kfjhksay9832rfk="] • ["VAL",":91F8:E192:C6B2:F7CD:F6AD:F900"]

Examples:

`"WriteEPC":[":3034:4567:abcd"]` the header, filter and partition included in the binary.

`"WriteUII":[":1234:4567:abcd"]` resulting in an AFI of 0x01, a proprietary UII.

`"WriteUII":[":1234:4567:abcd",[":92"]]` writes ISO/IEC 20248 data with no DSFID.

Note: Writing to a tag when other tags are also present in the **ReadZone** may be problematic. The use of an intelligent reader should be considered along with a well-designed **ReadZone** 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 **ReadZones**, **Antennas** and **GPIO** shall behave as configured in 6.4, 6.5 and 6.6.

6.7.2 ReadZone activation

StartRZ: Activates the listed **ReadZone**(s).

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

The default value for **ID** is [0] which means all. `{"Cmd":"StartRZ"}` is therefore valid. It will start all **ReadZones** at once.

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 active ReadZones>]}
```

StopRZ: Deactivates the listed **ReadZones**.

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

The default value for "ID" is [0] which means all. `{"Cmd":"StopRZ"}` is therefore valid. It will stop all **ReadZones** at once.

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 default value for **Prof** is [0] meaning all. `{"Cmd":"ThisTag"}` is therefore a valid command executing **ThisTag** with all the configured profiles.

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

The **ThisTag** command may include the fields Mode and **TargetTags**, see 6.3.6, which shall apply for the duration of the **ThisTag** execution.

An error event shall be reported when **ThisTag** reaches its timeout without completing an inventory on a tag 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 **ReadZones** 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>}
```

Note when using multiple connections: Only one **ThisTag** may be active at a given time. A **ThisTag** shall not be interfered with besides stopping it, see Annex B error 38.

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** report 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:

```
{"Report":"HB","Seq":<sequence number>,<heartbeat field>…}
```

The value of **Seq** is a positive integer which increments by one with every **Heartbeat**. The size of this number is vendor specific (but should be at least 8 bits long); once the limit has been reached, the counter will restart.

Seq is optional.

Example:

```
{"Report":"HB","RdrName":"RAIN-123DEF","BootCnt":123456,
 "TimeStamp":687652641.324}
```

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** is set, see 6.5, 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

A tag spot is reported by the reader with the following unsolicited message:

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

Note: The field sequence is NOT important, and fields excluded have an empty (**null**) or default value.

A tag spot is the result of a set tag access tasks and report timing and format according to the matching **SpotProfile**. The typical order of executing the **SpotProfile** tasks is:

1. Optional tag selection and challenge.
2. Inventory and matching to a **SpotProfile**
3. Additional **SpotProfile** matching
4. Additional access enablement (password and/or crypto)
5. Optional additional read access (data and sensors)
6. Optional kill with its outcome report
7. Optional read report (optional due the **FirstSeen**, **Seen** and **LastSeen** instructions).
8. Optional writing with its outcome report
9. Optional locking with its outcome report

Note 1: Additional access enablement may need to take place before additional **SpotProfile** matching.

Note 2: The outcome of a **SpotProfile** execution is combined into one tag spot report.

The following **TagEvent** fields provides information about the spot. Optional fields are indicated by a † and †† which are both excluded by default. These fields can be included by using settings in 6.3.5† (applicable to all **TagEvent** reports) and 6.6.2†† (applicable to a specific **SpotProfile**).

Fieldname	Value type	Default	Notes
Ant†	Number	-	**Antenna ID** the tag was spotted with.

Fieldname	Value type	Default	Notes
DT†	String	-	Use ISO 8601 format: YYYY-MM-DDThh:mm:ss.sssZ No time zone means local time.
DwnCnt	Number	-1	Defined in 6.6.2. Report after decremented. Only report when **DwnCnt** ≥ 0.
InvCnt†	Number	-	The amount of times the tag was inventoried since the last report.
Phase†	Number	-	Phase of the tag signal.
Prof†	Number	-	**SpotProfile ID** used to read the tag.
Range†	Number	-	Distance from tag to reader in millimetres.
RSSI†	Number	-	Received signal strength in dBm.
RZ†	Number	-	**ReadZone ID** the tag was spotted in.
Spot	"FirstSeen", "Seen", "LastSeen", "Killed", "Written"	"FirstSeen"	**Spot** may be omitted for **FirstSeen**.
TimeStamp†	Number	-	The epoch date-time value. Note: epoch is in seconds. A smaller fraction of time is indicated by the decimal fraction of the number, e.g. 2143235.234 indicates 234 ms after the epoch second 2143235.

The following **TagEvent** fields specify 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
20248††	Object	-	ISO/IEC 20248 interpreted data, see Annex F .
AFI	8-bit binary		This field shall be included when the PC bit T=1.
EPC-URI	Object		GS1 EPC Tag URI, see Annex G
Killed	Boolean	false	-
MB††	Array of read tuples	[]	Memory bank data read shall be reported as an array of objects. The objects shall contain the following fields: "ID":<memory bank number: 0 to 3> "Start":<start word of the data – number> "Data":<binary data> The data value shall be set to **null** when a read error occurred. See below for examples.

Fieldname	**Value type**	**Default**	**Notes**
PC††	HexString	-	PC, XPC_W1, and XPC_W2, see C.4.
SIMPLESENSOR††	Object	-	See Annex 0
SNAPSHOTSENSOR††	Object	-	See Annex H.3
UII, EPC, UII-NOT-CONFIGURED or UII-PROPRIETARY	Binary	-	The field name for inventoried data: For tags with PC bit T=1: **UII** for AFI>0x07. **UII-NOT-CONFIGURED** for AFI=0x00. **UII-PROPRIETARY** for AFI=0x01 to 0x07. **EPC** for tags with PC bit T=0. These fields are compulsory for all RCI implementations.
Scheme	String	-	EPC numbering schemes as specified by GS1 TDS, **RFU** and **TID**, see **EncodingType** in 6.6.2. This field shall be included when the PC bit T=0.
TagIndicator	Array of Strings	[]	XPC tag indicator interpretation values, see Annex E.2.

Examples:

ISO: T=1

```
{"Report":"TagEvent","AFI":":92","UII":":0123:4567:89AB:CDEF"}
{"Report":"TagEvent","AFI":":00","UII-NOT-CONFIGURED":":0123:456…"}
```

For AFI=0x01 to 0x07

```
{"Report":"TagEvent","AFI":":03","UII-PROPRIETARY":":0123:4567…"}
```

GS1 T=0; see TDS table 14-1

Header=0x00

```
{"Report":"TagEvent","Scheme":"UNPROGRAMMED","EPC":":0022:1234…"}
```

Header=0x2C to 0x41

```
{"Report":"TagEvent","Scheme":"SGTIN","EPC":":3003:4567:89AB…"}
```

Header=0xE0 and 0xE2

```
{"Report":"TagEvent","Scheme":"TID","EPC":":E203:4567:ABCD…"}
```

Note: **TID** is NOT a GS1 Scheme. It is PERMANENTLY RESERVED to avoid confusion with the first eight bits of TID memory when the TID has been copied to UII/EPC.

All other header values shall be reported as **RFU** (reserved for future use).

```
{"Report":"TagEvent","ErrID":0,"Scheme":"RFU","EPC":":0103:4567:89…"}
```

Example: A **Seen Spot** with XPC and user memory

```
{"Report":"TagEvent","DT":"2017-09-11T13:06:01.000",
 "Spot":"Seen","InvCnt":25,"PC":":3592:0025",
 "AFI":":01","UII-PROPRIETARY":":0123:4567:89AB:CDEF:89AB:CDEF",
 "MB":[{"ID":3,"Start":0,"Data":":2323:2323:2323:2323"},{…}]}
```

Example: A **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","Scheme":"SGTIN","EPC":":0123:4567:89AB:CDEF:89AB:CDEF",
 "MB":[{"ID":2,"Start":0,"Data":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 feature specification which includes the information fields and their values, and a list of all the commands, fields, and field values a reader supports.

Annex A Abbreviations

Addr	Address
AirProt	Air Protocol
Ant(s)	Antenna(s)
BAP	Battery assisted passive
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
RFU	Reserved for Future Use
RSSI	Read Signal Strength Indicator
RZ	ReadZone

SN	Serial number
Temp	Temperature
TBD	To Be Determined
Time	See Date.
TO	Timeout
Val	Value
XPC	Extended PC

Annex B Error numbers and descriptions

Number (ErrID)	**Description (ErrDesc)**	**Optional information (ERRInfo)**	**Notes**
0	No error(s)	-	No error on the command when in response to a command. 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 runs 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.
9	Message length error	Number of bytes missing.	"Too many bytes" is indicated with a negative value.
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	**SpotProfiles** full	-	-
31	**SpotProfile** error	Array with: JSON number: **SpotProfile** number, JSON string: <more info>	A **SpotProfile** resulted in an air protocol configuration error.

Number (ErrID)	**Description (ErrDesc)**	**Optional information (ERRInfo)**	**Notes**
32	Illegal **SpotProfile**	Number array listing illegal **SpotProfile**s	-
33	**ThisTag** timeout	String stating one of the following: "No tags inventoried." "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.
35	Kill error	String describing the error	The result is that the tag killed status is not known.
36	Access error	String describing the error	-
37	Write check error	String describing the error	-
38	**ThisTag** active	"Command not executed; wait for a **ThisTag** from another connection to complete."	Only one **ThisTag** may be active at a given time. A **ThisTag** shall not be interfered with besides stopping it.
39	**ThisTag** ignored	**ThisTag** is not allowed on this connection.	-
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	-
50	**GPIO** toggle value the same	Array of numbers identifying the **GPIO** IDs with the problem	A toggle could not be performed.
51	**GPIO** not settable	Array of numbers identifying the **GPIO** IDs with the problem	The **GPIO** is not an output, D2A or register.
52	Trigger not an input switch	Array of number identifying the offending **GPIO**	-
60	Connection access error	TBD	TBD
≥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 Air-protocol summary

Tags are energised by the reader. Once energised a tag listens 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 three 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).

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. Tags may control access to their stored data using passwords and on-chip crypto. A crypto tag informs the reader that it has encrypted information and which crypto suite is supported. The application must provide the keys and use the indicated crypto method to gain access to the protected data, and/or verify the tag and/or the data.

C.2 TargetTags explained

The RCI makes use of a "schema" to inform the reader of its expected behaviour. This is to assist the reader vendor to build and configure readers optimised for specific read scenarios. A key principle is that the system designer actually knows the types of tags to be read within a specific reader's **ReadZone**. Tags have different features. The combination of these features may complicate the reader firmware substantially and reduce the reader performance. For example, most GS1 EPC tags are only inventoried, i.e. only the UII/EPC is read with no other tag access.

TargetTags is used to tell the reader the type of tags it will access and report upon. RCI specifies the following tag target types:

SIMPLE: Access to these tags is limited to inventory where only the PC/XPC bits and the UII/EPC are read.

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 power to store the data permanently. It takes longer to complete a tag write interrogation. Write actions often reduce the read range of the tag.

BAP: Battery assisted passive tags are defined in ISO/IEC 18000-63. The battery of a BAP tag provides power for the tag intelligence (and sensors), ensuring optimal read range for all types of interrogations. BAP tags may support additional dedicated commands and backscatter modulation schemes to optimise the use of such tags.

ALARMSENSOR: Alarm sensor tags are the most basic implementation of all sensor tags. Alarm sensors are vendor defined but are typically tripwire sensors. See Annex H for additional information on sensors.

SNAPSHOTSENSOR: Snapshot sensor tags are the most basic implementation of all sensor tags using defined sensors. See Annex H for additional information on sensors.

SIMPLESENSOR: Simple sensor tags are the most basic implementation of self-monitoring sensor tags using defined sensors. See Annex H for additional information on sensors.

FULLSENSOR: Full-function sensor tags are the most advanced implementation of self-monitoring sensor tags using defined sensors. See Annex H for additional information on sensors.

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 require more power and time to complete.

Example: When a reader is expected to only read UII/EPC, then **SIMPLE** can be used to say so, and then the reader may ignore all other parts of a **SpotProfile**. The reader may also ignore all tag indicated access beyond the UII/EPC inventory. A reader vendor should state in the RCI feature statement an RCI reader capability using the **TargetTags** values.

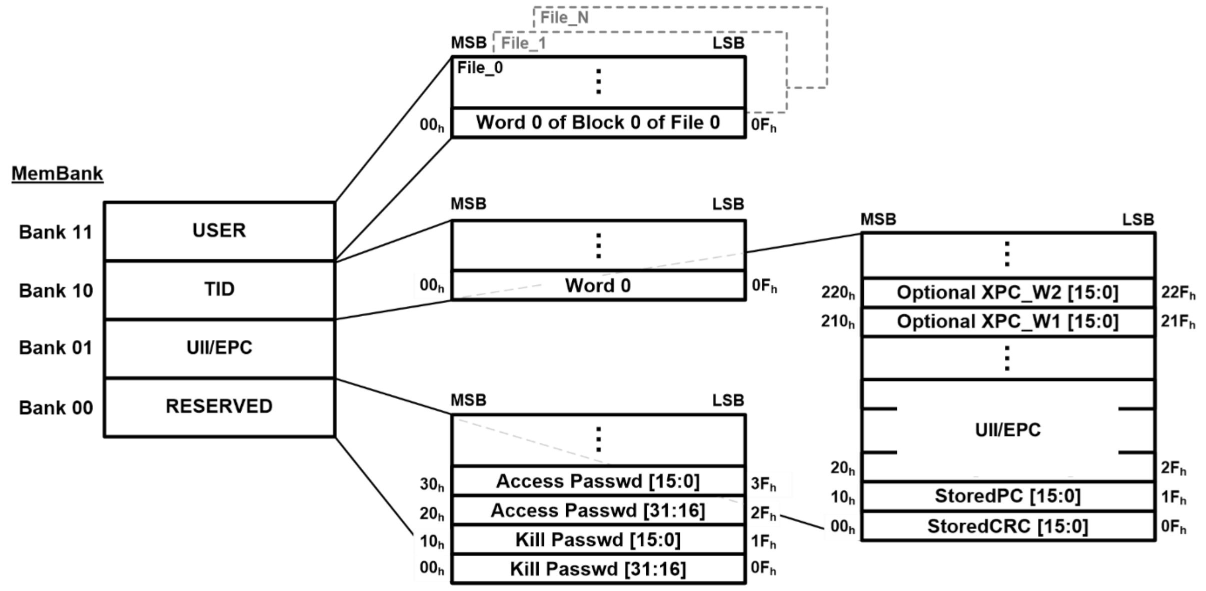
C.3 Tag memory organisation

This is a brief explanation of the RAIN tag memory map to a level of detail required by the RCI. The reader deals with more complex data elements, like CRC, as required by the air protocol and access to the tag.

A RAIN tag contains a number (UII/EPC), a chip identifier (TID), optional user data and optional sensor data. Access to the data on a tag can be controlled using a password and/or crypto suites as specified by ISO/IEC 29167.

RAIN tags can be killed by using the kill password.

RAIN tags have the following memory map:



Note: MB and MemBank are abbreviations of "memory bank".

Key tag data elements described are:

The Protocol Control (PC) word and the Extended Protocol Control words (XPC).

The PC word contains the Toggle (Standard) bit which indicates which standard group's numbering system is contained in the tag; ISO or GS1.

The PC and the tag optional XPC words may contain data about the tag use, access methods, and sensor data.

If the tag is ISO, then the PC bit also contains an 8-bit Application Family Identifier (AFI). In this way several ISO numbering systems are specified by ISO/IEC 15961. This number is called a Unique Item Identifier (UII). Each of these AFI numbering systems are specified by an application specification, e.g. ISO/IEC 20248 and IATA.

If the tag is GS1, then the number of the tag is an Electronic Product Code (EPC) as specified and issued by GS1. The EPC header, the first 8 bits of the EPC, is similar to the AFI in ISO. It specifies the EPC number schema, e.g. SGTIN.

The Data Storage Format Identifier (DSFID) is 8 bits used by ISO and GS1 to specify the data storage format. It may be present in an UII and shall be the first 8 bits of the User Memory (MB11) unless the AFI or GS1 Schema specifies it otherwise, e.g. ISO/IEC 20248.

Each RAIN chip also contains a unique identifier called the TID which is in MB10. The chip manufacturer programs the TID.

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 file type and file number 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 TDS or to ISO/IEC

15961/15962, ISO/IEC 1736x and ISO/IEC 20248. A specific AFI may also specify the structure of User memory.

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

MB01 (UII/EPC) has the following data elements which is reported by the reader:

- Protocol bits as contained in the PC word and XPC (extended) words. Sensor data may be contained in the XPC words.
- The UII/EPC number as specified by ISO and GS1. The Toggle bit in the PC word indicates whether the number is ISO or GS1; T=1 ➔ ISO, T=0 ➔ GS1.

Simple ISO tag with a 128-bit 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

It is important to note the UII type is specified by ISO/IEC 15961 and then by application standards like ISO/IEC 20248 and the IATA specifications.

Simple GS1 tag with a 96-bit 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

It is important to note that the EPC number is specified by GS1 TDS. It contains several data elements. The first 8 bits of the EPC number, the header, is important for the RCI since it specifies the schema of the EPC number, e.g. SGTIN. It is typically followed by a filter value, partition, GS1 Company Prefix, Item Reference and Serial Number.

Adhering to these numbering systems is critical to ensure that an application's tags does not look like another's tags and thus interfere with the operations of the application. This is called "acid RAIN".

Following some tag data examples:

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

MB01 PC Word					MB01 UII	MB11 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	8 bits	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

MB01 PC Word					MB01 UII	MB11 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	8 bits	96 bits	8 bits	≥ 0 bits

ISO tag with a simple sensor

MB01 PC Word					MB01 UII	MB01 XPC Word 1	MB01 Simple Sensor Data
UII len	UserMem	XI	Standard	AFI	UII as specified by the AFI	Simple sensor bit set	As specified by ISO
01001	0	1	1 (ISO)	8 bits	128 bits	16 bits	n*16 bits

C.4 Proper handling of the PC and XPC words

It is important to note that the XPC words (XPC_W1 and XPC_W2) are stored in a different sequence than how the tag will deliver it during inventory. The use of XPC words by the tag has an impact on the maximum length of the UII/EPC.

The RAIN air protocol specifications (ISO/IEC 18000-63 and GS1 EPC Gen2) specify the following:

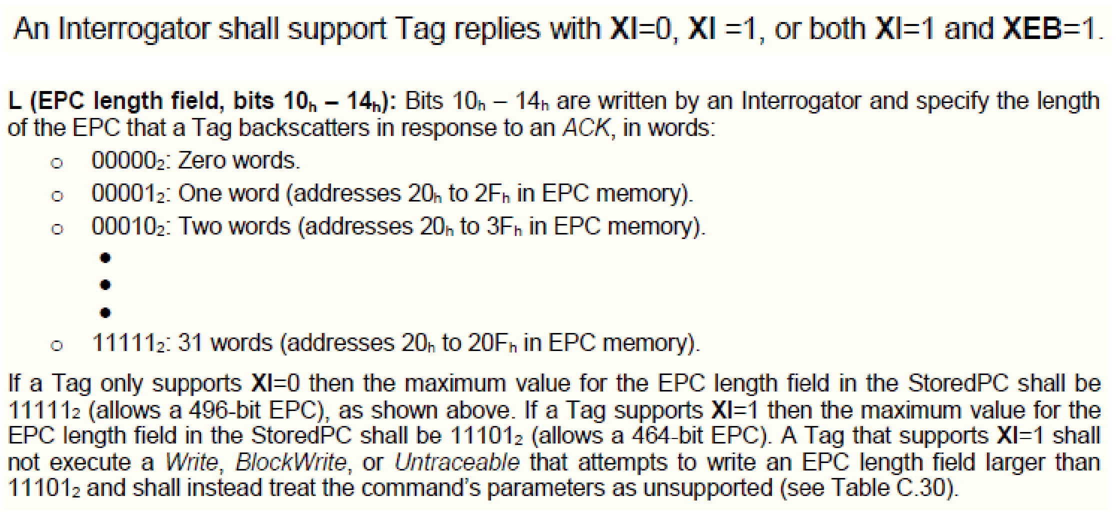
1. The XPC words are optional for tags. The optional XPC words are stored in MB01 address from bit 210_h.

2. During inventory, the tag shall backscatter/send the PC word, then the optional XPC words and then the UII/EPC.

   a. XPC_W1 shall be sent following the PC word when the PC word flag XI=1.

   b. XPC_W2 shall be sent following XPC_W1 when the XPC_W1 flag XEB=1.

3. The reader shall be able to handle XPC words; see the following **6.3.2.1.2.2 Protocol-control** extracts from ISO/IEC 18000-63 (which is the same in GS1 EPC Gen2).

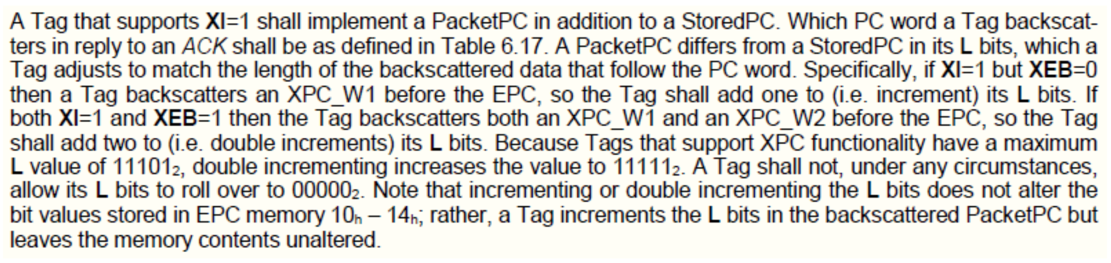


An Interrogator shall support Tag replies with **XI**=0, **XI** =1, or both **XI**=1 and **XEB**=1.

L (EPC length field, bits 10_h – 14_h): Bits 10_h – 14_h are written by an Interrogator and specify the length of the EPC that a Tag backscatters in response to an *ACK*, in words:

- 00000_2: Zero words.
- 00001_2: One word (addresses 20_h to $2F_h$ in EPC memory).
- 00010_2: Two words (addresses 20_h to $3F_h$ in EPC memory).
  - •
  - •
  - •
- 11111_2: 31 words (addresses 20_h to $20F_h$ in EPC memory).

If a Tag only supports **XI**=0 then the maximum value for the EPC length field in the StoredPC shall be 11111_2 (allows a 496-bit EPC), as shown above. If a Tag supports **XI**=1 then the maximum value for the EPC length field in the StoredPC shall be 11101_2 (allows a 464-bit EPC). A Tag that supports **XI**=1 shall not execute a *Write*, *BlockWrite*, or *Untraceable* that attempts to write an EPC length field larger than 11101_2 and shall instead treat the command's parameters as unsupported (see Table C.30).



A Tag that supports **XI**=1 shall implement a PacketPC in addition to a StoredPC. Which PC word a Tag backscatters in reply to an *ACK* shall be as defined in Table 6.17. A PacketPC differs from a StoredPC in its **L** bits, which a Tag adjusts to match the length of the backscattered data that follow the PC word. Specifically, if **XI**=1 but **XEB**=0 then a Tag backscatters an XPC_W1 before the EPC, so the Tag shall add one to (i.e. increment) its **L** bits. If both **XI**=1 and **XEB**=1 then the Tag backscatters both an XPC_W1 and an XPC_W2 before the EPC, so the Tag shall add two to (i.e. double increments) its **L** bits. Because Tags that support XPC functionality have a maximum **L** value of 11101_2, double incrementing increases the value to 11111_2. A Tag shall not, under any circumstances, allow its **L** bits to roll over to 00000_2. Note that incrementing or double incrementing the **L** bits does not alter the bit values stored in EPC memory 10_h – 14_h; rather, a Tag increments the **L** bits in the backscattered PacketPC but leaves the memory contents unaltered.

Note 1: Readers which do not comply with the RAIN air protocol specifications report the XPC word(s) prepended to the UII/EPC, causing application confusion.

Note 2: The XPC word(s) shall be reported in its stored location when accessed with a **Read**.

Examples: Let us consider the following three GS1 tags:

1. A standard "EPC tag". The PC word flag XI=0. The tag memory is as follows:

MB01 PC Word					MB01 UII/EPC
EPC len	UserMem	XI	Standard	RFU	EPC
00110	0	0	0 (GS1)	0x00	0x30123456790123456789012

RCI Spot report:

```
{"Report":"TagEvent","PC":":3000","Scheme":"SGTIN",
                          "EPC":":3012:3456:7890:1234:5678:9012"}
```

A non-compliant reader reports the tag inventory correctly, but often omits the PC word resulting in the inability to detect if this is a genuine GS1 EPC tag (note, the AFI = 0x00 and as such may be an GS1 tag or a not-configured ISO tag):

PC = 0x3000 and EPC = 0x30123456790123456789012

2. With the PC word flag XI=1; during inventory the tag transmits the PC word, XPC_W1 and then the EPC (note, the transmitted EPC len is incremented by 1; transmitted EPC len=00111). The tag memory is as follows (note, the stored EPC len is not changed):

MB01 PC Word					MB01 UII/EPC	MB01 XPC Word 1
EPC len	UserMem	XI	Standard	RFU	EPC	XPC_W1
00110	0	1	0 (GS1)	0x00	0x30123456790123456789012	0x0800

RCI Spot report:

```
{"Report":"TagEvent","PC":":3A00:0800","Scheme":"SGTIN",
                          "EPC":":3012:3456:7890:1234:5678:9012"}
```

A non-compliant reader typically reports the tag inventory incorrectly with the XPC prepended to the EPC as it received by the reader:

PC = 0x**3A00** and EPC = 0x**0800**30123456790123456789012

The XPC_W1 is prepended to the EPC which results in the wrong EPC.

This problem is worsened when the PC word is not reported. With the PC word reported, the application can detect the error.

The correct report is:

PC = 0x**3A00**, XPC_W1 = 0x**0800** and EPC = 0x30123456790123456789012, or

PC = 0x**3A000800** and EPC = 0x30123456790123456789012.

3. With the PC word flag XI=1 and the XPC word flag XEB=1; during inventory the tag transmits the PC word, XPC_W1, XPC_W2 and then the EPC (note, the transmitted EPC len is incremented by 2; transmitted EPC len=01000). The tag memory is as follows (note, the stored EPC len is not changed):

MB01 PC Word					MB01 UII/EPC	MB01 XPC Word 1	MB01 XPC Word 2
EPC len	UserMem	XI	Standard	RFU	EPC	XPC_W1	XPC_W2
00110	0	1	0 (GS1)	0x00	0x30123456790123456789012	0x8100	0x2222

RCI Spot report:

```
{"Report":"TagEvent","PC":":4200:8100:2222","Scheme":"SGTIN",
                      "EPC":":3012:3456:7890:1234:5678:9012"}
```

A non-compliant reader will report the tag inventory incorrectly with the XPC prepended to the EPC as it received by the reader:

PC = 0x**4200** and EPC = 0x**81002222**30123456790123456789012

The XPC_W1 and XPC_W2 are prepended to the EPC which results in the wrong EPC.

This problem is worsened when the PC word is not reported. With the PC word reported, the application can detect the error.

The correct report is:

PC = 0x**3A00**, XPC_W1 = 0x**8100**, XPC_W1 = 0x**2222** and EPC = 0x3012…012, or

PC = 0x**3A0081002222** and EPC = 0x30123456790123456789012.

C.5 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 when there are too many tags for the reader to cope with. It is also very useful to control battery assisted passive tag responses, since they may have substantially longer read ranges.

C.6 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 a tag has very little power available (since it harvested electric power from the reader radiation) and a small chip. It 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 numbers of tags while lower values are good for small populations 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 is useful 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 specify 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.

Annex D Tag spot example implementation

D.1 Detection workflow

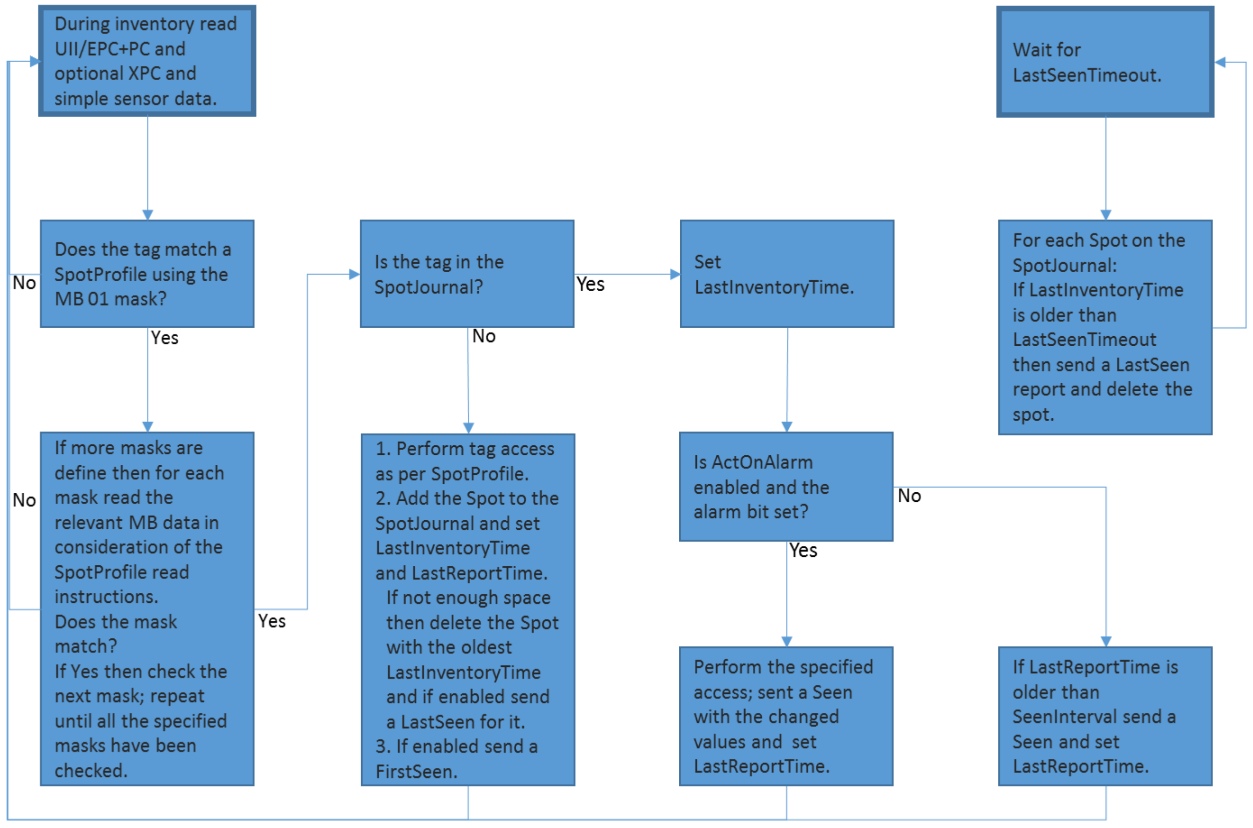
This example implementation makes use of the following data structures:

1. The **SpotProfileList** is ordered in decreasing priority. This can easily be achieved using linked lists.

2. A **SpotJournal** 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 in their own time in separate processing streams.



During inventory read UII/EPC+PC and optional XPC and simple sensor data.
Wait for LastSeenTimeout.
Does the tag match a SpotProfile using the MB 01 mask?
No
Yes
Is the tag in the SpotJournal?
Yes
No
Set LastInventoryTime.
For each Spot on the SpotJournal: If LastInventoryTime is older than LastSeenTimeout then send a LastSeen report and delete the spot.
If more masks are define then for each mask read the relevant MB data in consideration of the SpotProfile read instructions. Does the mask match? If Yes then check the next mask; repeat until all the specified masks have been checked.
No
Yes
1. Perform tag access as per SpotProfile. 2. Add the Spot to the SpotJournal and set LastInventoryTime and LastReportTime. If not enough space then delete the Spot with the oldest LastInventoryTime and if enabled send a LastSeen for it. 3. If enabled send a FirstSeen.
Is ActOnAlarm enabled and the alarm bit set?
No
Yes
Perform the specified access; sent a Seen with the changed values and set LastReportTime.
If LastReportTime is older than SeenInterval send a Seen and set LastReportTime.

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:

2. Drive the air protocol.
3. Perform the configuration and connectivity functions.
4. Perform the **LastSeen** pruning and data decoding functions.
5. Perform the crypto functions.
6. The **SpotProfileList**, **SpotJournal** and communications buffer are dual access memory locations controlled with semaphores.
7. 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). All the features, except for basic reading and **FirstSeen**, are optional. Defaults have been chosen to support lean implementations.

D.2 Dealing with SAMEs

It is possible and often feasible for RAIN tags to have the UII/EPC and User Memory (MB11) to be encoded with the same values. This may be by design, to facilitate untraceability, by mistake or even maliciously.

It is therefore possible that tags with the same UII/EPC, and/or same UII/EPC and User Memory data, appear in the read zone. This may confuse applications.

TID is currently unique. Chip manufacturers ensure this uniqueness and lock the TID during the chip manufacturing process. As such dealing with SAMEs can safely be ignored when the SpotProfile instructs the reading of the TID.

Note, a reader can typically only detect SAMEs when the SAME-tags are at the same time in the read zone and respond to the reader in the same inventory round, though smart algorithms in the reader firmware may extend the SAMEs detection beyond a single inventory round. Other mechanisms need to be used to detect copied tags in general. See GS1 LLRP (2020 version) for a detailed description and method to detect SAMEs.

RCI tag spot reporting is continues using **FirstSeen**, **Seen** and **LastSeen Spot**s as the tag-in-the-beam boundaries. The reporting of SAMEs should have the following behaviour:

Consider the following tags, with UII/EPC X and Y and User Memory data A and B, to be simultaneously in the read zone:

T1[UII/EPC=X, TID=1, UM=A]
T2[UII/EPC=X, TID=2, UM=A]
T3[UII/EPC=X, TID=3, UM=B]
T4[UII/EPC=X, TID=4, UM=B]
T5[UII/EPC=Y, TID=5, UM=B]

A **SpotProfile** reading only UII/EPC the following **Spot**s should be reported:

{X,SAMEs=false}, {X,SAMEs=true}, {Y,SAMEs=false}

A **SpotProfile** reading UII/EPC and User Memory the following **Spot**s should be reported:

{X,A,SAMEs=false}, {X,A,SAMEs=true}, {X,B,SAMEs=false}, {Y,B,SAMEs=false}

A **SpotProfile** also reading TID should report all tags as SAMEs=false, since all TIDs are unique for each chip.

Annex E Protocol Control bits interpretation

E.1 General

The Protocol Control (PC and XPC) bits indicate by default the length of numbering system standard. It may additionally contain tag use and sensor information.

E.2 Tag use flags

The air protocol control bits (PC) assist the reader to determine the type of tag and type of data stored in the tag. The extended air protocol control bits (XPC) contain indicator flags of tag use which may be important to the application, see ISO/IEC 18000-63 or GS1 EPC Gen2 Specification.

When **TAGUSE** is set, then these indicators shall be included in the **Spot** report with the following field:

```
    "TagIndicator":<array of strings of tag indicators set in the tag XPC>
```

XPC_W1 flag	**RCI TagIndicator value**
SA (sensor alarm)	"ALARMSENSOR"
SS (simple sensor tag)	"SIMPLESENSOR"
FS (full-function sensor tag)	"FULLSENSOR"
SN (snapshot sensor tag)	"SNAPSHOTSENSOR"
B (battery assisted passive tag)	"BAP"
TN (tag notification)	"TAGNOTE"
U (untraceable)	"UNTRACEABLE"
K (killable)	"KILLABLE"
NR (nonremoveable)	"NONREMOVE"
H (Hazmat)	"HAZMAT"

Note: The XEB (XPC_W2 indicator), C (ResponseBuffer State), and SLI (SELECT Flag status) have no meaning within the RCI. They are used within the air protocol.

Annex F ISO/IEC 20248 data interpretation

The RCI 20248 data interpretation shall comply with ISO/IEC 20248. This annex assumes a knowledge of ISO/IEC 20248 data presentations.

ISO/IEC 20248 specifies a method and data description language (using JSON) to specify verifiable tag data. The ISO/IEC 20248 data structure schema is called a DigSig Data Description (DDD). The DDD is distributed within a X.509 Version 3 Digital Certificate which also contains a public key with which the **DDDdata** can be verified.

The 20248 data interpretation is enabled by including the field **20248** in the value array of the field **InterpretData** of a **SpotProfile**.

The 20248 data interpretation is configured using the following **20248** values (the default values are shown):

```
"20248":{"DDDdata":false,       # when true include DDDdata in the Spot
         "SigData":false,       # when true include SigData in the Spot
         "DDDdataTagged":true,  # when true include DDDdataTagged
                                                         in the Spot
         "DDDdataDisplay":false,# when true include DDDdataDisplay
                                                         in the Spot
         "Timezone":"+0000",    # time zone offset in minutes
         "Language":"en"}       # language code per ISO/IEC 20248
```

The ISO/IEC 20248 interpreted data shall be reported in a **Spot** report using the following JSON object:

```
"20248":{"ResponseCode":<DigSig response code>,
         "DDDdata":{<DDDdata fields and values>},
         "SigData":{<SigData fields and values>},
         "DDDdataTagged":{<DDDdataTagged fields and values>},
         "DDDdataDisplay":{<DDDdataDisplay fields and values>}}
```

The fields **DDDdata**, **SigData**, **DDDdataTagged** and **DDDdataDisplay** shall be omitted when the associated **20248** configuration field value is set to false.

The following represents a complete Spot report example:

```
{"Report":"TagEvent","ErrID":0,
  "DT":"2017-09-11T13:06:01.000",
  "TimeStamp":687652641.324,
  "PC":":8592","AFI":":92"
  "UII":"wJgLT3UAawN5RRiWnEEAA==",
  "MB":[{"ID":2,"Start":0,"Data":"4sBokiAACwAeOqun"},
        {"ID":3,"Start":0,
                "Data":"IAG1kIrW9xoL_oRifldd7xrq4w0A_JDdn_z7t50ktMU"}],
  "20248":{
    "ResponseCode":{"Code":0,
                    "Desc":"DigSig Verification accepted; No error"},
    "DDDdataTagged":{
      "cid":107,
      "daid":"QC FVXX",
      "dauri":"https://da.fleetvalid.info",
      "license_plate":"569QS",
      "plate_placing":"REAR",
      "signature":"IAG1kIrW9xoL_oRifldd7xrq4w0A_JDdn_z7t50ktMU",
      "specificationversion":"ISO/IEC 20248:2018",
      "tid":"4sBokiAACwAeOqun",
      "timestamp":4752000,
      "vehicle_colour":"BLACK",
      "vehicle_shape":"COMPACT"}}}
```

The **ResponseCode** shall be a tuple of two values as per ISO/IEC 20248 Annex F *DigSig error codes*:

```
    "ResponseCode":{"Code":<DigSig error code - Number>,
                    "Desc":<DigSig error description - String>}
```

ISO/IEC 20248 supports the interpretation of partial data. Missing field values shall take the value **null** with the **ResponseCode** set to the appropriate ISO/IEC 20248 error code (17, 18 or 19).

Annex G GS1 interpretation

The GS1 EPC Tag URI interpretation shall comply with the GS1 TDS specification.

RCI provides for EPC interpretation to the ***EPC Pure Identity URI*** as per GS1 EPC TDS.

Note, the ***EPC Pure Identity URI*** shall be the default RCI GS1 EPC Tag URI interpretation. Future RCI versions may provide for configuration to achieve other GS1 EPC Tag URI interpretation.

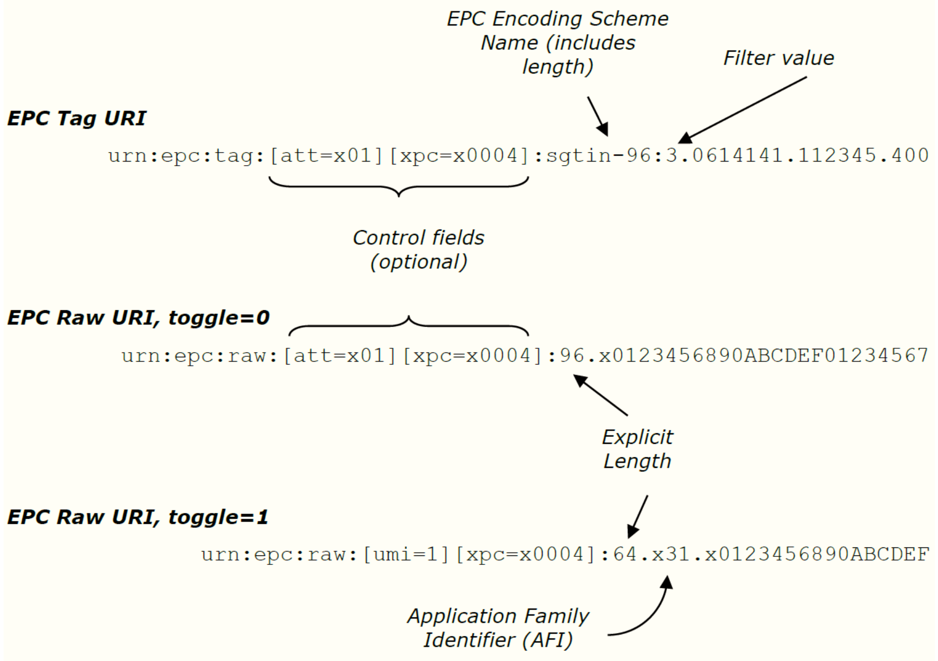
Example, an SGTIN is interpreted to the following ***EPC Pure Identity URI***:

```
urn:epc:id:sgtin:<Company Prefix>.<Indicator><Item Reference>.<Serial Number>
```

Example: `urn:epc:id:sgtin:0614141.112345.400`

Note: The ***EPC TAG URI*** contains length information as well as optional attribute and XPC values, see GS1 EPC TDS clause 12 (see extract below); as such only the ***EPC Pure Identity URI*** is reported by RCI. RCI provides attribute and XPC values as independent fields.

Figure 12-1 Illustration of EPC Tag URI and EPC Raw URI



EPC Encoding Scheme Name (includes length)
Filter value
EPC Tag URI
urn:epc:tag:[att=x01][xpc=x0004]:sgtin-96:3.0614141.112345.400
Control fields (optional)
EPC Raw URI, toggle=0
urn:epc:raw:[att=x01][xpc=x0004]:96.x0123456890ABCDEF01234567
Explicit Length
EPC Raw URI, toggle=1
urn:epc:raw:[umi=1][xpc=x0004]:64.x31.x0123456890ABCDEF
Application Family Identifier (AFI)

RCI does not currently provide a method to write an ***EPC Pure Identity URI***.

GS1 EPC Tag URI interpretation is enabled by adding the string "EPC-URI" to the value array of the field **InterpretData** in the **SportProfile**, see 6.6.2.

Example: `"InterpretData":["EPC-URI"]`

In this RCI version GS1 EPC Tag URI interpretation has no configuration.

When GS1 EPC Tag URI is enabled, then the **Spot** shall contain the field **EPC-URI** with the following format:

```
"EPC-URI":{
  "ResponseCode":{"Code":<code>,"Desc":<description>},
  "URI":<"EPC Pure Identity URI">}
```

The following error codes shall be used:

Code	**Desc**
0	"OK"
1	"EPC code not recognised"
2	"Binary format error"
3	"Interpretation failed"

Annex H Sensors

This annex provides a summary of RAIN sensor tags that are supported by ISO and GS1 standards.

H.1 Types of sensor tags

Sensor support for RAIN tags is defined in ISO/IEC 18000-63 and supported by GS1 EPC Gen2. A RAIN tag declares itself to be a sensor tag via XPC_W1 flags which are reported as RCI **TagIndicator**s. There are four types of RAIN sensor tags:

1. Alarm Sensor (**TagIndicator** = **ALARMSENSOR**). Alarm sensor tags are the most basic implementation of all sensor tags. Alarm sensors are vendor defined but are typically tripwire sensors indicating a continuity loop is either open or closed (e.g. tamper detection for digital seals). The continuity loop might be implemented using a simple continuity loop, with heat sensitive fuse wire, with sensors having both high and low impedance states, or with external devices controlling an electronic switch such as a MOSFET to open/close the loop. Alarm sensor tags may be either passive tags or BAP tags. Alarm sensor data is provided during inventory using the Sensor Alarm (SA) in XPC_W1 or it may be read from the tags. Other types of sensors may also use SA in XPC_W1.

2. Snapshot Sensor (**TagIndicator** = **SNAPSHOTSENSOR**). Snapshot sensor tags are defined in ISO/IEC 18000-63 and are the most basic implementation of all sensor tags using defined sensors. Snapshot sensor tags may be either passive tags or BAP tags. Snapshot sensor data may be provided during inventory using XPC_W2 or may be read from the tags.

3. Simple Sensor (**TagIndicator** = **SIMPLESENSOR**). Simple sensor tags are defined in ISO/IEC 18000-63 and are the most basic implementation of self-monitoring sensor tags using defined sensors. Simple sensor tags are required to be BAP tags. Simple sensor tags may support additional dedicated commands to optimise the use of such tags. Simple sensor data may be appended to the UII/EPC during inventory as indicated by the protocol or may be read from the tags. Memory mapped simple sensor data is defined in ISO/IEC 18000-63 and ported simple sensor data is defined in ISO/IEC 24753.

   Note: The additional dedicated commands and handling of the complex data structures used by ported simple sensors are beyond the scope of the RCI.

4. Full-function Sensor (**TagIndicator** = **FULLSENSOR**). Full-function sensor tags are defined in ISO/IEC 18000-63 and are the most advanced implementation of self-monitoring sensor tags using defined sensors. Full-function sensor tags are required to be BAP tags. Full-function sensor data is defined in ISO/IEC 24753 and ISO/IEC/IEEE 21451-7 and may be read from the tags using additional dedicated commands.

   Note: **TagIndicator** = **FULLSENSOR** is provided for completeness in the identification of sensor tags but the additional dedicated commands and handling of the complex data structures used by full-function sensors are beyond the scope of the RCI.

H.2 Alarm sensor operation

Alarm sensors are typically implemented on passive tags. A sensor status check occurs when the tag becomes energized by the RF field and prior to processing of the first command from a reader. The sensor status may be kept in volatile memory, non-volatile memory, or both. Tags that keep the status in non-volatile memory usually do not permit clearing or resetting the alarm sensor. BAP tags may implement alarm sensors and their behaviour is slightly different since the BAP tag is always energized. A BAP tag performs a sensor status check when the tag detects the RF field and prior to processing of the first command from a reader. Additionally, a BAP tag might implement an asynchronous interrupt when a change in sensor status occurs regardless if an RF field is present or not.

The status of the alarm sensor is mapped to the Sensor Alarm (SA) in XPC_W1. This also corresponds to **TagIndicator** = **ALARMSENSOR**. A reader automatically obtains XPC_W1 during normal tag inventory operations. It may also be obtained using a *Read* command to directly read XPC_W1. A reader may be commanded to specifically look for alarm sensor tags using the **TargetTags** field in **SetCfg** or the **MBMask** field in **SpotProfile**.

The Sensor Alarm may also be used by snapshot sensors, simple sensors, and full-function sensors to indicate one or more alarms exist for those types of sensors. A tag having only an alarm sensor will not declare itself as having any other sensor types (i.e. SN, SS, and FS will not be set in XPC_W1).

H.3 Snapshot sensor operation

H.3.1 Air protocol

Snapshot sensors generate a sensor measurement essentially in real-time as a reader inventories or reads the tag. The sensor measurement may occur during power-up or on demand from a reader and the tag delivers the sensor measurement via XPC_W2. Consequently, a tag will set XEB=0 when snapshot sensor information is not available meaning that XPC_W2 = 0000_h, and a tag will set XEB=1 when snapshot sensor information is available meaning that XPC_W2 ≠ 0000_h. Available snapshot sensor information is included in the tag reply to an *ACK*. It may also be obtained using a *Read* command to directly read XPC_W2. If snapshot sensor information is not available, then a reader can demand the tag to make a sensor measurement by writing $FFFF_h$ to XPC_W2 immediately followed by reading XPC_W1.

Snapshot sensor information in XPC_W2 uses the following formats (see ISO/IEC 18000-63 as the master specification):

1. XPC_W2 = (sensor type || sensor data) as defined in the table below. This format is used for reporting a sensor measurement from only one snapshot sensor and the sensor type must be in the range 0000_2 to 1011_2. An example for this type of reporting is the following:

   - XPC_W1 = 8100_h indicating a snapshot sensor tag with sensor information available
   - XPC_W2 = 1190_h indicating a temperature sensor with a measurement of 25°C

2. XPC_W2 = (11_2 || Memory Bank || Word Address). This format is used for reporting sensor measurements from a multi-dimension sensor, or a multi-sensor tag, or when one or more sensors provides additional vendor defined supplemental data. A reader may obtain the snapshot sensor information using a *Read* command with <u>MemBank</u> = Memory Bank, <u>WordPtr</u> = Word Address converted to EBV format, and <u>WordCount</u> = 00_h. The snapshot sensor information consists of a sequence of words, each having the format (sensor type || sensor data) as defined in the table below. The sequence of words is terminated with a **null** value = 0000_h which does not correspond to any valid value for (sensor type || sensor data). Sensor type = 1111_2 is reserved for a tag to include optional vendor defined supplemental data for a snapshot sensor. The supplemental data consists of a sequence of words and the

number of data words is specified following the sensor type, i.e. (sensor type = 1111_2 || number of data words). An example for this type of reporting is the following:

- XPC_W1 = 8100_h indicating a snapshot sensor tag with sensor information available
- XPC_W2 = $E100_h$ indicating sensor data is in TID memory starting at word address 100_h
- TID word 100_h = 1190_h indicates a temperature sensor with measurement of 25°C
- TID word 101_h = $F002_h$ indicates 2 words follow of supplemental data for the temperature sensor
- TID word 102_h = 1234_h is supplemental data word #1 for the temperature sensor
- TID word 103_h = 5678_h is supplemental data word #2 for the temperature sensor
- TID word 104_h = 2500_h indicates a relative humidity sensor with measurement of 80%
- TID word 105_h = 0000_h is the **null** value terminator indicating the end of sensor data

Type	**Sensor**	**Units**	**Sensor Data**
0000_2	Vendor Defined (VenDef)	VenDef	2-bit data type || 10-bit data value as follows: 00_2 || 1111111111_2 (error) 01_2 || 10 discrete bits (e.g. switches) 10_2 || 10-bit unsigned integer (e.g. A2D converter) 11_2 || 10-bit signed integer (e.g. A2D converter)
0001_2	Temperature	°C	12-bit signed integer
0010_2	Relative Humidity	%	12-bit unsigned integer
0011_2	Barometric Pressure	hPa (mbar)	12-bit unsigned integer
0100_2	Light	lux	1-bit scale factor || 11-bit data value as follows: 0_2 || 11-bit unsigned integer (scale factor is 0.5) 1_2 || 11-bit unsigned integer (scale factor is 50)
0101_2	Voltage	V	2-bit scale factor || 10-bit data value as follows: 00_2 || 10-bit unsigned integer (scale factor is 0.001) 01_2 || 10-bit unsigned integer (scale factor is 0.01) 10_2 || 10-bit unsigned integer (scale factor is 0.1) 11_2 || 10-bit unsigned integer (scale factor is 1)
0110_2	Magnetic Field	mT (10G)	12-bit signed integer
0111_2	Angular Position	°	12-bit unsigned integer
1000_2	Rotational Speed	rpm	1-bit scale factor || 11-bit data value as follows: 0_2 || 11-bit signed integer (scale factor is 1) 1_2 || 11-bit signed integer (scale factor is 10)
1001_2	Weight	kg	2-bit scale factor || 10-bit data value as follows: 00_2 || 10-bit unsigned integer (scale factor is 0.1) 01_2 || 10-bit unsigned integer (scale factor is 1) 10_2 || 10-bit unsigned integer (scale factor is 10) 11_2 || 10-bit unsigned integer (scale factor is 100)
1010_2	Liquid Flow	ml/min	2-bit scale factor || 10-bit data value as follows: 00_2 || 10-bit signed integer (scale factor is 0.00002) 01_2 || 10-bit signed integer (scale factor is 0.002) 10_2 || 10-bit signed integer (scale factor is 0.2) 11_2 || 10-bit signed integer (scale factor is 20)

Type	Sensor	Units	Sensor Data
1011_2	Gas Flow	l_n/min	1-bit scale factor || 11-bit data value as follows: 0_2 || 11-bit unsigned integer (scale factor is 0.000625) 1_2 || 11-bit unsigned integer (scale factor is 0.0625)
1100_2	Accelerometer	G ($9.81m/s^2$)	2-bit axis || 10-bit data value as follows: 00_2 || 10-bit signed integer (X-axis measurement) 01_2 || 10-bit signed integer (Y-axis measurement) 10_2 || 10-bit signed integer (Z-axis measurement) 11_2 || 10-bit signed integer (not used)
1101_2	Gyroscope	°/s	2-bit axis || 10-bit data value as follows: 00_2 || 10-bit signed integer (X-axis measurement) 01_2 || 10-bit signed integer (Y-axis measurement) 10_2 || 10-bit signed integer (Z-axis measurement) 11_2 || 10-bit signed integer (not used)
1110_2	Magnetometer	μT	2-bit axis || 10-bit data value as follows: 00_2 || 10-bit signed integer (X-axis measurement) 01_2 || 10-bit signed integer (Y-axis measurement) 10_2 || 10-bit signed integer (Z-axis measurement) 11_2 || 10-bit signed integer (not used)
1111_2	VenDef Supplemental Data	n/a	12-bit unsigned integer for number of words that follow which is supplemental sensor information for the preceding sensor

H.3.2 RCI

Snapshot sensor data interpretation is enabled by adding the string "SNAPSHOTSENSOR" to the value array of the field **InterpretData** in the **SpotProfile**, see 6.6.2.

Example: `"InterpretData":["SNAPSHOTSENSOR"]`

Snapshot sensor data interpretation has no configuration.

When snapshot sensor data interpretation is enabled, then the **Spot** shall contain the field **SNAPSHOTSENSOR** with the following format:

```
"SNAPSHOTSENSOR":{"ResponseCode":{"Code":<code>,"Desc":<description>},
                  <sensor name>:<sensor value>,…}
```

Sensor	Sensor name	Units	Sensor value
Vendor Defined (VenDef)	VenDef	VenDef	For the 2-bit data type: 00_2: **null** (measurement error) 01_2: 10-bit **HexString** 10_2: JSON non-negative number 11_2: JSON number
Temperature	TempC	°C	JSON number

Sensor	Sensor name	Units	Sensor value
Relative Humidity	RelHumidity	%	JSON number
Barometric Pressure	BarPress	hPa (mbar)	JSON number
Light	Light	lux	JSON number
Voltage	Voltage	V	JSON number
Magnetic Field	MagField	mT (10G)	JSON number
Angular Position	AngularPos	°	JSON number
Rotational Speed	RotSpeed	rpm	JSON number
Weight	Weight	kg	JSON number
Liquid Flow	LiquidFlow	ml/min	JSON number
Gas Flow	GasFlow	l_n/min	JSON number
Accelerometer	Accelerometer	G (9.81m/s^2)	A JSON number tuple [X,Y,Z]
Gyroscope	Gyroscope	°/s	A JSON number tuple [X,Y,Z]
Magnetometer	Magnetometer	µT	A JSON number tuple [X,Y,Z]
VenDef Supplemental Data	VenDefData	n/a	**HexString**

The following error codes shall be used:

Code	Desc
0	"OK"
1	"Measurement error"
2	"Binary format error"

Example:

```
"SNAPSHOTSENSOR":{"ResponseCode":{"Code":0,"Desc":"OK"},
                   "Weight":2.3,"Accelerometer":[213,4,789],"VenDefData":":ABCD"}
```

H.4 Simple sensor operation

Simple sensors monitor the environmental characteristic for which they are designed. A simple sensor tag is self-monitoring and takes samples at defined intervals, computes pass/fail based on its characteristics, and report its status. Memory mapped simple sensors report their data using the Simple Sensor Data (SSD) block. The *Flex_Query* command is required to command tags to append the SSD block to the UII/EPC in reply to an *ACK* command. Alternatively, the SSD block can be obtained from the tag memory using a normal read operation.

H.4.1 Air protocol

The Simple Sensor Data (SSD) Block is specified as follows (see ISO/IEC 18000-63 as the master specification):

Most significant bit																															**Least significant bit**
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
$=0000_2$ temperature sensor with span of 14°C				measure span			accuracy		sampling regime				high in-range limit			low in-range limit			monitor delay			high out-of-range alarm delay			low out-of-range alarm delay			alarms			
$=0001_2$ temperature sensor with span of 28°C				measure span			accuracy		sampling regime				high in-range limit			low in-range limit			monitor delay			high out-of-range alarm delay			low out-of-range alarm delay			alarms			
$=0010_2$ relative humidity sensor				measure span			accuracy		sampling regime				high in-range limit			low in-range limit			monitor delay			high out-of-range alarm delay			low out-of-range alarm delay			alarms			
$=0011_2$ impact sensor				measure span			accuracy		0000_2				defined by measure span			000_2			000_2			high out-of-range alarm delay			000_2			alarms			
$=0100_2$ tilt sensor				measure span			accuracy		0000_2				defined by measure span			000_2			000_2			high out-of-range alarm delay			000_2			alarms			

The Sensor Alarm (SA) in XPC_W1 is the logical OR of the alarm bits which have the following meanings:

$1xxx_2$	Low battery
$x1xx_2$	Tamper
$xx1x_2$	Delayed high out-of-range
$xxx1_2$	Delayed low out-of-range (not used by impact or tilt)

H.4.2 RCI

SSD interpretation is enabled by adding the string "SIMPLESENSOR" to the value array of the field **InterpretData** in the **SpotProfile**, see 6.6.2.

Example: `"InterpretData":["SIMPLESENSOR"]`

Simple sensor data interpretation has no configuration.

RCI does not make provision for interpreted configuration of the Simple Sensors. Simple Sensor configuration is viewed as an expert operation to be performed using **Write{VAL}**, see 6.6.3.

When simple sensor data interpretation is enabled, then the **Spot** shall contain the field **SIMPLESENSOR** with the following format:

```
    "SIMPLESENSOR":{"ResponseCode":{"Code":<code>,"Desc":<description>},
                    <simple sensor>:[<list of alarms>],…}
```

RCI field name	Sensor
TempC14	0000_2 temperature sensor with span of 14°C
TempC28	0001_2 temperature sensor with span of 28°C
Humidity	0010_2 relative humidity sensor
Impact	0011_2 impact sensor
Tilt	0100_2 tilt sensor

All the detected sensors shall be listed with an empty list if no alarms are reported.

RCI sensor alarm value	Sensor alarm
"LowBat"	Low battery
"Tamper"	Tamper
"TooHigh"	Delayed high out-of-range
"TooLow"	Delayed low out-of-range (not used by impact or tilt)

Example:

```
    "SIMPLESENSOR":{"ResponseCode":{"Code":0,"Desc":"OK"},
      "SimpleSensor":{"TempC14":[],"Impact":["BatLow","TooHigh"]}
```

The following error codes shall be used:

Code	Desc
0	"OK"
1	"Binary format error"
2	"Interpretation failed"

Annex I   Crypto

A familiarity of ISO/IEC 29167 parts 10 and 13, the RAIN air protocol crypto commands and the basic RCI tag access methods is assumed.

RCI Version 4 specifies the tag crypto read methods for TagAuth (TAM1/TA.1) and private data (TAM2/TAR) for current products implementing ISO/IEC 29167 parts 10 and/or 13.

Note: TAM1 and TAM2 is specified by ISO/IEC 29167 part 10 and TA.1 and TAR by part 13.

It is important to note that RCI crypto methods can be achieved in more than one way which may be influenced by the set of crypto commands implemented by the various chip vendors. RCI provides a singular method to achieve the outcomes of tag crypto (by example see Annex I.4.1 for a discussion on methods to achieve TagAuth). Reader vendors are encouraged to implement methods aimed at their intended reader use-cases.

Note: The TID provides information about the RAIN chip and its capabilities.

The application informs a reader with **SetCfg TargetTags** (6.3.6) that crypto tags is part of the set of target tags allowing the reader to optimise its internal operations to deal with crypto tags in an efficient manner.

I.1 General

Encrypted tag data may be decrypted by the reader (directly) and/or by an application typically remote from the reader (by proxy).

The reader, in lieu of the application, shall generate the random challenge for the air protocol crypto suite functions.

Note 1: The random challenge prevents over-the-air replay attacks; the random challenge is encrypted with the encrypted tag response ensuring the tag response is random between tag access sessions.

Note 2: RCI assumes that the connectivity between the application and the reader is secured appropriately, as such the complexity of an application generating the challenge is avoided.

Note 3: A proxy, to perform the required description, requires both the challenge and the crypto suite used in generating the encrypted response from the tag.

I.2 SpotProfile configuration

Crypto access to a tag is tag specific. Crypto functions are therefore specified using a **SpotProfile**, see 6.6.2. The fields **TagAuth** and **PrivateData** are used to configure the crypto tag read access.

```
    "TagAuth":{"Crypto":<crypto suite>,
               "Key":<key or key pointer>,"KeyID":<key identifier>,
               "KeyDiv":{"Method":<diversification method>,
                         "Data":[<MB>,<start>,<words>],
               "Report":<report method>}
```

Fieldname	Value type	Default	Notes
Crypto	"AES-128-0", "AES-128-1" or "GRAIN-128A"	-	Defines cryptography method to use. This is a compulsory field for **TagAuth** and **PrivateData**.
Key[1,2]	Binary or URI	null	The **Key** value is a binary key or a key pointer. A key pointer is an URI which points to the location where the key can be obtained. The default value is **null**. When **Key** is set to **null** or omitted, then the reader does not decrypt and simply report the challenge and the cypher text. Note: The key pointer method is TBD. Proposals will be welcomed.
KeyID	Number	1	The **KeyID** value is a non-negative number.
KeyDiv	"NONE", "AES-128" or "HKDF"	"NONE"	See Annex I.4 below.
Report	"OPEN", "OPROXY" or "PROXY"	"OPEN"	See Annex I.3 below.

Note 1: A URI easily is distinguished from binary; a **HexString** starts with ":", a **Base64String** contains no ":" and a URI contains at least one ":" which is not the first character.

Note 2: The URI may also provide the method and access control to obtain the key.

PrivateData specifies read access to one or more tag data segments to be encrypted before transmission to the reader. The value of **PrivateData** is an array of objects, each specifying a data segment to be accessed. Each object uses the same fields than **TagAuth** and includes a memory profile for the data segment.

```
"PrivateData":[{"Crypto":<…>,"Key":<…>,"KeyID":<…>,"KeyDiv":{…},"Report":<…>,
              "MemProf":[<memory profile number>,<block start>]}…]
```

MemProf (crypto memory profile) specifies the predefined memory segment that can be either a memory bank (0: MB01 containing the UII/EPC | 1: MB10 - TID) | 2: MB11 - User Memory) or a vendor defined tag memory segment which is not part of any memory bank specified with values 3 and above. The optional block start value specifies a block size offset within the memory profile, with 0 the default.

Note 1: Memory specified using **MemProf** is not specified using **Read** in the **SpotProfile**.

Note 2: The **PrivateData** function only uses blocks of memory. The block size is dependent on the crypto suite, typically 16 or 64 bits.

I.3 Tag spot report

The result of a crypto tag access shall be reported with the fields **TagAuth** and **PrivateData** as part of a **Spot** report, see 7.4.

The reader shall never reveal a key in any manner. **Key** shall only be included in a report if it contains a key pointer.

The fields included in **TagAuth** and **PrivateData** are set by the value of **Report**. The default value of **Report** is **OPEN**. When **Key** is set to **null** or omitted, then **Report** shall be set and default to **PROXY**.

When **Key** has a non-**null** value, then the reader shall attempt to perform the decryption function. **Report** shall be set to **OK** on success, **FAIL** on failure and **?** if no decryption was done.

Note: It is recommended that the key pointer method provides support for key cashing inside the reader.

The **OPEN** reports have the following format:

```
"TagAuth":{"Result":<"OK" or "FAIL">}

"PrivateData":{"Result":<"OK", "FAIL">,
               "MemProf":[{"ID":<memory profile number>,"Start":<block start>,
                           "Data":<decrypted bitstring>}…]}
```

When **Result** have the value of **FAIL**, then the relevant **Data** shall be set to **null**.

The **PROXY** reports have the following format:

```
"TagAuth":{"Result":<"OK", "FAIL" or "?">,
           "Suite":<crypto suite>,
           "Key":<key-pointer>,
           "Challenge":<bitstring>,
           "Response":<bitstring"}

"PrivateData":{"Result":<"OK", "FAIL" or "?">,
               "MemProf":[{"ID":<memory profile number>,"Start":<block start>,
                           "Suite":<crypto suite>,
                           "Key":<key-pointer>,
                           "Challenge":<bitstring>,
                           "Response":<bitstring"}…]}
```

The **OPROXY** (**OPEN** and **PROXY**) is only valid for **PrivateData**. The report shall contain all the fields of **OPEN** and **PROXY**.

I.4 Key diversification

The practice of key diversification provides a method for symmetric cyphers whereby a set of tags is provided each their own key derived from a master key using some identifying data stored on the tag; for example, the UII/EPC or the TID. In this way only the master key need to be distributed to authorised readers and proxy applications. RCI makes provision for key diversification with the field **KeyDiv**.

```
"KeyDiv":{"Method":<diversification method>,"Data":[<MB>,<start>,<words>]}
```

Note: UII/EPC is obtained during inventory. It is specified as "Data":[1,2,<words>] with words being the number of 16-bit words of the UII/EPC, which is 6 for a 96-bit UII/EPC. The reader should not reread UII/EPC.

The following diversification methods are defined for **KeyDiv**:

Method	**Description**
NONE	No diversification is used. This is the default. **Data** is ignored.
AES-128-CMAC	See I.4.1.
HKDF	See I.4.2. This is the recommended method.

I.4.1 AES-128-CMAC

AES-128-CMAC key diversification is in common use with HF and UHF RFID systems. It is based on NIST Special Publication 800-38B (doi.org/10.6028/NIST.SP.800-38B) and IETF RFC4493 which provides excellent support for implementation.

The RCI AES-128-CMAC implementation uses simplified input values **Key** and **Data** shown below in context of **PrivateData**:

```
"PrivateData":[{"Crypto":<…>,"Key":"The master Key","KeyID":<…>,
                "KeyDiv":{"Method":"AES-128-CMAC",
                          "Data":[<MB>,<start>,<words>]},
                …}…]
```

RCI specifies AES-128-CMAC(key, data) with:

key the master key, is set to the 128-bit value of **Key**, and

data is set the value to the binary value (≤ 31 bytes or 248 bits) pointed to by **Data**

which results in a 128-bit derived key.

AES-128-CMAC(key, data) is calculated in the following steps with $\mathrm{CIPH}_K(x)$ the AES-128(K, x) encryption with K = key and x the input data:

1. First calculate subkeys K1 and K2, as specified by NIST 800-38B 6.1, from the master key, the value of **key**.
2. Create diversification data M by concatenating 0x01, data and padding to result in a 256-bit (32 byte) binary value. The padding shall start with 0x80 followed by zero.

   M := 01h || M || PADDING

3. If M is padded, then XOR the last 16 bytes of M with K1, else with K2.
4. Calculate MAC = $\mathrm{CIPH}_K(M)$ using AES-128 encryption in CBC mode and the initialisation vector set to zero.
5. The 128-bit diversified key is the last 16 bytes of the MAC.

Example:

Master key (K) = 0x00112233445566778899AABBCCDDEEFF

Date = 0x04782E21801D803042F54E585020416275

Results in the diversified key: 0xA8DD63A3B89D54B37CA802473FDA9175

I.4.2 HKDF

IETF RFC 5869 specifies the HMAC-based Extract-and-Expand Key Derivation Function (HKDF) method.

This is the recommended key diversification method for RCI. The input values **Key**, **Data** and the optional **Salt** are shown below in context of **PrivateData**:

```
"PrivateData":[{"Crypto":<…>,"Key":"The master Key","KeyID":<…>,
                 "KeyDiv":{"Method":"HKDF",
                           "Data":[<MB>,<start>,<words>],
                           "Salt":<bitstring>},
                 …}…]
```

The HKDF operation recommends the use of the optional binary parameter salt. Its purpose and use are well specified by IETF RFC 5869. The salt value is included in the **KeyDiv** object using the following field:

```
"Salt":<bitstring with a maximum length of 128 bits with default is all zeros>
```

IETF RFC 5869 specifies HKDF(salt, IKM, info, L) with:

salt is set to the value of **Salt** truncate or padded with zeros to 128 bits,

IKM (input keying material) is set to the 128-bit value of **Key**,

info is set to the binary value pointed to by **Data**, and

L is set to 16

which results in a 128-bit derived key.

IETF RFC 5869 allows various hash algorithms. For RCI SHA256 (IETF RFC 4634) shall be used as the Hash function. SHA256 padding shall be used.

I.5 TagAuth air protocol implementation examples

By mid-2020 two families of crypto chips where EPC Gen2v2 Certified. These chips crypto support is as follows:

Family 1:

- Implements ISO/IEC 29167-10 Rev 0 (AES-128) and supports Tag Authentication.
- Supports *Authenticate* and *ReadBuffer* commands.

Family 2:

- Implements ISO/IEC 29167-10 Rev 1 (AES-128) and supports Tag Authentication, Interrogator Authentication, and Mutual Authentication
- Implements ISO/IEC 29167-13 (GRAIN-128A) and supports Tag Authentication, Mutual Authentication, and Authenticated Communication
- Supports Challenge, Authenticate, ReadBuffer, and AuthComm commands

Following examples of Reader ⬄ Tag Crypto Dialogs for Tag Authentication:

1. The most efficient method to authenticate tags uses the following command sequence:

   a. Challenge (plaintext data for crypto challenge, instruct all crypto tags to append crypto response to ACK reply)
   b. Keep tags energized (CW ON) to perform crypto operation
   c. Select (select all crypto tags that have 'C-flag' asserted indicating crypto response is available)
   d. Query -- tag replies with RN16
   e. ACK -- tag replies with EPC || crypto ciphertext response
   f. QueryRep -- tag replies with RN16
   g. ACK -- tag replies with EPC || crypto ciphertext response
   h. QueQueryRep -- tag replies with RN16
   i. …

2. The next most efficient method to authenticate tags uses the following command sequence:

   a. *Challenge* (plaintext data for crypto challenge, instruct all crypto tags to store response in ResponseBuffer)
   b. Keep tags energized (CW ON) to perform crypto operation
   c. Select (select all crypto tags that have 'C-flag' asserted indicating crypto response is available)
   d. Query -- tag replies with RN16
   e. ACK -- tag replies with EPC
   f. REQ_RN -- tag replies with handle
   g. ReadBuffer -- tag replies with stored crypto ciphertext response in ResponseBuffer
   h. QueryRep -- tag replies with RN16
   i. …

3. The next to least efficient method to authenticate tags uses the following command sequence:

   a. *Select* (select all tags that have 'S' asserted in TID indicating a crypto tag)
   b. *Query* -- tag replies with RN16
   c. *ACK* -- tag replies with EPC
   d. *REQ_RN* -- tag replies with handle
   e. *Authenticate* (plaintext data for crypto challenge, instruct the crypto tag to send the crypto response now) -- tag replies with crypto ciphertext response
   f. *QueryRep* -- tag replies with RN16
   g. …

4. The least efficient method to authenticate tags uses the following command sequence:

   a. *Select* (select all tags that have 'S' asserted in TID indicating a crypto tag)

b. *Query* -- tag replies with RN16
c. *ACK* -- tag replies with EPC
d. *REQ_RN* -- tag replies with handle
e. *Authenticate* (plaintext data for crypto challenge, instruct the crypto tag to store response in ResponseBuffer)
f. *ReadBuffer* -- tag replies with stored crypto ciphertext response in ResponseBuffer
g. *QueryRep* -- tag replies with RN16
h. …

5. One possibility of working with a mixed population of vendor tags with different implementations:

   a. *Challenge* (plaintext data for crypto challenge, instruct all crypto tags to append crypto response to ACK reply)
   b. Keep tags energized (CW ON) to perform crypto operation
   c. *Select* (select all tags that have 'S' asserted in TID indicating a crypto tag)
   d. *Query* -- tag replies with RN16
   e. *ACK* -- tag replies with 'C' asserted and EPC || crypto ciphertext response OR tag replies with 'C' deasserted and EPC
   f. If 'C' was asserted, then crypto response was provided in ACK reply, so *QueryRep* to go to next slot
   g. If 'C' was deasserted, then crypto response was NOT provided in ACK reply, so continue with dialog
   h. *REQ_RN* -- tag replies with handle
   i. *Authenticate* (plaintext data for crypto challenge, instruct the crypto tag to send the crypto response now) -- tag replies with crypto ciphertext response
   j. *QueryRep* -- tag replies with RN16
   k. …

Annex J Multiple connections

Support for multiple connections to an RCI reader allows for:

- Connection roles whereby the data of each connection can be specified allowing for:
  - Separation of data and control.
  - Separation of different event reports.
  - Separation of tag event (Spots) reports.
- Connection authentication and access control

J.1 General

Multiple connections provide for the ability to have separate connections for different types of monitoring and configuration. Allowing for the client to have, perhaps, tag events on one connection, heartbeat events on another, etc.

Each connection has an associated connection object describing connection; its identification, login authentication level, as well as the "information" it will delivers. The connection object can be applied to TCP, as well as serial ports. It facilitates the wrapping of the RCI within WebSockets and serve as Topics in MQTT.

Note: In the case of serial ports, there is a limit of one connection per serial port. A reader may have multiple serial ports.

A connection is controlled with the **SetConnection** command. It has the following format:

```
{"Cmd":"SetConnection",<connection fields>}
```

The operating flow is:

1. Connect to the reader. The connection assumes the guest role.
2. Login as the required user.
3. Perform the required functions and listen to events. Typically, admin will configure all SpotProfiles and ops selects those it needs to listen to.

Role	**Description**	**Allowed functions**
admin	For configuration and reports. It is recommended to limit admin connections to one at a time.	GetInfo SaveFields, ReadFields, DefaultFields, Reboot, GetCfg, SetCfg GetRZ, SetRZ, AddRZ, DelRZ GetGPIOs, SetGPIOs GetProf, SetProf, AddProf, DelProf [1] StartRZ [2], GetActRZ, StopRZ [2] ThisTag [3], ThisTagStop [3] SetConnection, GetConnection Vendor selected commands

Role	Description	Allowed functions
guest	For the initial open access. It shall only provide the configured heartbeat message and the ability to perform a login.	GetInfo, GetCfg and SetConnection{Login} Vendor selected commands
installer	Optional. For installer specific function as specified and implemented by the vendor. Configuration of the regulatory settings fall in this class of access as well as low level vendor specific reader controls and settings. Recommend being limited to one connection.	All RCI and vendor selected commands
ops	For report selection and reports.	GetInfo, GetCfg, GetRZ, GetGPIOs GetProf, GetActRZ SetConnection, GetConnection Vendor selected commands
support	Optional. For vendor specific functions. Recommend being limited to one connection.	All RCI and vendor commands

Note 1: Care should be taken when deleting **SpotProfiles** from more than one connection. The behaviour may be confusing.

Note 2: Care should be taken when starting and stopping read zones from more than one connection. The behaviour may be confusing.

Note 3: Only one ThisTag may be active at any time.

J.2 Change notification

When a reader function is changed, then all other open connections shall receive the following unsolicited report:

```
{"Report":"ChangeEvent","Changed":<command name>}
```

Example:

```
{"Report":"ChangeEvent","Changed":"DelProf"}
```

The following commands is deemed to cause a functional change: **Reboot**, **SetCfg**, **SetRZ**, **AddRZ**, **DelRZ**, **SetGPIOs**, **SetProf**, **AddProf**, **DelProf**, **StartRZ**, **StopRZ**, **ThisTag** and **ThisTagStop**.

J.3 Access control

The **Connection** command shall be used to perform a login and logout:

```
{"Cmd":"SetConnection","Login":[<role>,<password>]}
{"Cmd":"SetConnection","Logout":<delay in seconds>}
```

An immediate logout shall be performed whenever a **Login** is requested on a logged-in connection or when the connection is terminated (detectable on TCP connections, but not on serial connections). The reader shall not terminate the TCP connection on a logout.

The reader shall respond with one of:

```
{"Report":"SetConnection","Login":"OK"}
{"Report":"SetConnection","Login":"Failed"}
{"Report":"SetConnection","Logout":"OK"}
{"Report":"SetConnection","Logout":"OK-delayed"}
```

The reader shall report the logout event when it occurs with:

```
{"Report":"SetConnection","Logout":"OK"}
```

The following **SetConnection** command change a password:

```
{"Report":"SetConnection",
 "Password":[<connected role password>,<new password>,<role>}
```

with the connected role the default for <role>

support and **installer** may change all passwords and **admin** may change the **password** of ops.

The reader shall respond with one of:

```
{"Report":"SetConnection","Password":"Changed"}
{"Report":"SetConnection","Password":"Failed"}
```

J.4 Connection report filtering

SpotProfile reports are filter using the **AllowTagEvents** field with the following format:

```
{"Cmd":"SetConnection","AllowTagEvents":<list of SpotProfile IDs>}
```

To report no tag events on this connection, set this value to an empty array:

```
{"Cmd":"SetConnection","AllowTagEvents":[]}
```

To report all tag events on this current connection, set this value to an array with a single 0 (meaning "ALL"):

```
{"Cmd":"SetConnection","AllowTagEvents":[0]}
```

To report tag events from specific **SpotProfile**s on this connects list their IDs in the array:

```
{"Cmd":"SetConnection","AllowTagEvents":[1, 10]}
```

The defaults values are set to **DefaultAllowTagEvents** see J.6.

The reader shall respond with:

```
{"Report":"SetConnection","AllowTagEvents":<list of selected SpotProfile IDs>}
```

Note 1: The list is expanded for [0].

Note 2: A SpotProfile selected which does not exist is not listed. Error 32 is returned.

Reports are also filtered with the following **Connection** command Boolean fields:

Fieldname	**Description**
AllowHBEvents	true: Report **HB** events on this connection: false: Do not report **HB** events on this connection: `{"Cmd":"SetConnection","AllowHBEvents":<true or false>}` Defaults to **DefaultAllowHBEvents** see J.6.
AllowRdrEvents	true: Report reader events on this connection: false: Do not report reader events on this connection: `{"Cmd":"SetConnection","AllowRdrEvents":<true or false>}` Defaults to **DefaultAllowRdrEvents** see J.6.
TagEventFirstSeen	true: Report **FirstSeen Spot**s on this connection: false: Do not report **FirstSeen Spot**s on this connection: `{"Cmd":"SetConnection","TagEventFirstSeen":<true or false>}` Note: The selected **SpotProfiles** must report **FirstSeen** events for any of these events to be reported on this connection. Defaults to **DefaultTagEventFirstSeen** see J.6.
TagEventLastSeen	true: Report **LastSeen Spot**s on this connection: false: Do not report **LastSeen Spot**s on this connection: `{"Cmd":"SetConnection","TagEventLastSeen":<true or false>}` Note: The selected **SpotProfiles** must report **LastSeen** events for any of these events to be reported on this connection. Defaults to **DefaultTagLastEventSeen** see J.6.
TagEventSeen	true: Report **Seen Spot**s on this connection: false: Do not report **Seen Spot**s on this connection: `{"Cmd":"SetConnection","TagEventSeen":<true or false>}` Note: The selected **SpotProfiles** must report **Seen** events for any of these events to be reported on this connection. Defaults to **DefaultTagEventSeen** see J.6.
TagEventThisTag	true: Handle **ThisTag** on this connection: false: Ignore **ThisTag** on this connection: `{"Cmd":"SetConnection","TagEventThisTag":<true or false>}` Defaults to **DefaultTagEventThisTag** see J.6.

The reader shall respond with:

```
{"Report":"SetConnection",<field and value as set>}
```

J.5 Connection field values

The field values of this connection (J.4) are obtained by using **GetConnection{Fields}** in the same way than the **GetInfo** command, see 6.1.

The additional field **Role** returns:

```
"Role":[<"guest", "admin", "ops", "installer" or "support">,<connection ID>]
```

Note: Each connection shall have a unique identifier.

Example:

```
{"Cmd":"GetConnection","Fields":["Role","TagEventFirstSeen","AllowTagEvents"]}
```

results in:

```
{"Report":"GetConnection","Role":["admin",3],"TagEventFirstSeen":true,
                          "AllowTagEvents":[3,6]}
```

J.6 Default settings for report filtering

The following fields provides the default settings for connection report filtering. The fields are set with **SetCfg** and read with **GetCfg**, see6.3.

Fieldname	**Field type**	**Default**	**Description**
DefaultAllowTagEvents	Array of integers	[0]	Default value for AllowTagEvents.
DefaultAllowHBEvents	Boolean	True	Default value for AllowHBEvents.
DefaultAllowRdrEvents	Boolean	True	Default value for AllowRdrEvents.
DefaultTagEventFirstSeen	Boolean	True	Default value for TagEventFirstSeen.
DefaultTagEventLastSeen	Boolean	True	Default value for TagEventLastSeen.
DefaultTagEventSeen	Boolean	True	Default value for TagEventSeen.
DefaultTagEventThisTag	Boolean	True	Default value for TagEventThisTag.

Annex K Contributors

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

Stefano Coluccini (Co-Chair up to 2018-04)	CAEN RFID srl
Thomas Frederick	Clairvoyant Technology LLC
Craig Alan Repec	GS1
Claude Tetelin	GS1
Bryan Berezdivin	Impinj, Inc.
Rob Collins	Impinj, Inc.
David Dzenitis	Impinj, Inc
Matt Robshaw	Impinj, Inc.
Srikanth Kodimela	JADAK a Novanta Company
Dan Ratner	JADAK a Novanta Company
Harry Tsai	JADAK a Novanta Company
Harinath Reddy	JADAK a Novanta Company
Georg Michel	Kathrein Solutions GmbH
Bertus Pretorius (Author, Original Proposal, and editor, Co-Chair from 2018-04)	LicenSys Pty. Ltd.
Lars Thuring (Chair)	Logopak Systeme GmbH & Co. KG
David Ciampi	Mimeme LLC
John T. Armstrong	MonsoonRF, Inc.
Danny Haak	Nedap N.V.
Nikias Klohr	RACE RESULT
James Goodland	NXP Semiconductors N.V.
Dene Taylor	SPF-Inc
Mattias Edlund	TagMaster AB
Benjamin Bekritsky	Zebra Technologies
Sajan Wilfred	Zebra Technologies

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.