Application Programming Interface

Active Wave Inc.

Version: 40
2 September 2013

1. Introduction

The API functions may execute in a synchronous or asynchronous manner. Most of API functions will execute asynchronously. The following descriptions and diagram illustrate several possible scenarios that may result from an API function call.

1. The function returns RF_S_DONE indicating that a valid response was sent either by API or the device and this is the final return value for the function call sent to API.

2. The function returns RF_E_XXX indicating an error condition. The specific return value identifies the cause of the error. Any output arguments are not guaranteed to be valid.

3. The function returns RF_S_PEND indicating that it will be executed asynchronously. The application should expect one or more callbacks corresponding to this function. A callback with RF_S_OK_PEND status indicates the function sent to API was a broadcast command and this return value is a device response to this command. There might be more responses from other devices. The last callback will have RF_S_DONE status indicating that it is the last callback for the function.

4. The function returns RF_S_PEND indicating that it will be executed asynchronously. A callback with RF_S_DONE status indicates the function that was sent to API was executed successfully and this is the last response for this function from API.

5. The function returns RF_S_PEND indicating that it will be executed asynchronously. A callback with RF_E_XXX status indicates the function that there was an error executing this command.

All functions will have a unique packet ID. The value of this argument will be passed to the callback function. The application can use this value to match the callback to originating function call.

2. Commands

2.1 rfOpen

The **rfOpen** function opens a communications channel for communicating with readers. All readers connected to the channel must support the communication parameters specified in this function. This function executes synchronously.

```
long rfOpen(
  UInt32 baudRate,    // baud rate
  UInt32 compport,    // comm. Port number
  HANDLE* hConn       // connection handle
);
```

Parameters

baudRate

[in] Baud rate in bits per second.

comPort

[in] Communication port number.

hConn

[out] Connection handle. Used as input parameter to other functions. This handle is valid only if the return value indicates success.

Return Values

RF_S_DONE if successful, error code otherwise.

Table Index

2.2 rfClose

The **rfClose** function closes a communications channel that was previously opened with the **rfOpen** function. This function executes synchronously.

```
long rfClose(
  HANDLE hConn                  // connection handle
);
```

Parameters

hConn

[in] Connection handle returned by open function. The handle value is no longer valid and must not be used if this function returns a success value.

Return Values

RF_S_DONE if successful, error code otherwise.

Table Index

2.3 rfScanNetwork

The **rfScanNetwork** searches the network for any reader with active network connection.

```
long rfSacnNetwork (UInt16 pktID);
```

Parameters

pktID

[in] Packet identifier used to match callbacks to a specific function call.

Return Values

RF_S_PEND if function acknowledged by API;
RF_S_OK_PEND if function acknowledged and responded successfully by each reader or tag when broadcasting, it should be followed by RF_S_DONE when function times out and there are no more responses from the reader, repeater, or tag;
RF_S_DONE if function successful and acknowledged by reader, repeater, or API;
Error code otherwise.

rfReaderEvent_t structure in reader call back function contains information about ip address (xxx.xxx.xxx.xxx) of the reader. The table below describes which items in the rfReaderEvevt_t structure are valid for this function.

Items	
	Valid
host	Yes
reader	No
repeater	No
Ip	Yes
port	Yes
relay	No
fGenerator	No
eventType	Yes
cmdType	Yes
eventStatus	Yes
errorStatus	Yes
pktID	Yes
cmdRef	No
data	No
versionInfo	No

Table Index

2.4 rfScanIP

The **rfScanIP** searches the network for any reader with specific IP and OEM address.

```
long rfSacnIP (
    Byte ip[ ],
    UInt16 pktID);
```

Parameters

ip

[in] Reader internet address in form of xxx.xxx.xxx.xxx, or a valid host name.

pktID

[in] Packet identifier used to match callbacks to a specific function call. Range from 1 to 223. Each consecutive API function **Must** have different packet ID than previous one.

Return Values

RF_S_DONE if function successful and acknowledged by reader and API;
Error code otherwise.

rfReaderEvent_t structure in reader call back function contains information about ip address (xxx.xxx.xxx.xxx) of the reader. The table below describes which items in the rfReaderEvevt_t structure are valid for this function.

Items	Valid
host	Yes
reader	No
repeater	No
Ip	Yes
port	Yes
relay	No
fGenerator	No
eventType	Yes
cmdType	Yes
eventStatus	Yes
errorStatus	Yes
pktID	Yes
cmdRef	No
data	No
versionInfo	No

Table Index

2.5 rfOpenSocket

The **rfOpenSocket** function opens a communications channel over network for communicating with reader. This function executes synchronously.

```
long rfOpenSocket (
    Byte ip[ ],              // IP Address
    UInt16 host,             // function type
    Boolean  encrypt,        //encryption
    UInt16 cmdType,          // function type
    UInt16 pktID             // packet id used in callback
) ;
```

Parameters

ip

[in] Reader internet address in form of xxx.xxx.xxx.xxx, or a valid host name.

host

[in] Address of the host.

Encrypt

[in] Encrypt the transmit and receive data between the host and the reader over network. If true encrypt it, false otherwise.

cmdType

[in] Send the command to specific enabled socket or broadcast it to all enabled sockets.

Value	Meaning
ALL_IPS	All enabled socket
SPECIFIC_IP	Specific enabled socket

pktID

[in] Packet identifier used to match callbacks to a specific function call. Range from 1 to 223. Each consecutive API function **Must** have different packet ID than previous one.

Return Values

RF_S_DONE if function successful and acknowledged by reader, repeater, or API; Error code otherwise.

Table Index

2.6 rfCloseSocket

The **rfCloseSocket** function closes the communications channel already opened with rfOpenSocket(). This function executes synchronously.

```
Long  rfCloseSocket (
    Byte  ip[ ],
    UInt16  cmdType
);
```

Parameters

ip

[in] Reader internet address in form of xxx.xxx.xxx.xxx, if cmdType is SPECIFIC_IP, otherwise NULL.

cmdType

[in] Send the command to specific enabled socket or broadcast it to all enabled sockets.

Value	Meaning
ALL_IPS	All enabled socket
SPECIFIC_IP	Specific enabled socket

Return Value

RF_S_DONE if function successful and acknowledged by reader, repeater, or API;
Error code otherwise.

Table Index

2.7 rfChangeIPAddress

The **rfChangeIPAddress** function changes network IP address of a reader to a new IP address. This function executes synchronously.

```
long rfChangeIPAddress (
  Byte oldIP[],           // Old IP Address
  Byte newIP[]            // New IP Address
);
```

Parameters

oldIP

[in] Reader network address in form of xxx.xxx.xxx.xxx

NewIP

[in] Reader new network address in form of xxx.xxx.xxx.xxx

Return Values

RF_S_DONE if function successful and acknowledged by reader, repeater, or API; Error

Table Index

2.8 rfResetReader

The **rfResetReader** function requests a reader to perform a reset sequence. The API will call the registered **rfReaderEvent** callback function when the reader is back on line.

```
long rfResetReader(
  UInt16  host       // address of reader
  UInt16  reader     // address of host
  UInt16  repeater   // address of repeater
  UInt16  cmdType    // command type
  UInt16  pktID      // packet ID
);
```

Parameters

host
[in] Address of the host.

reader
[in] Address of the reader.

repeater
[in] Address of the repeater.

cmdType
[in] Send the command to specific reader or broadcast it to all readers.

Value	Meaning
ALL_READERS	All readers in the system
ALL_REPEATERS	All repeaters in the system
SPECIFIC_READER	Specified reader address
SPECIFIC_REPEATER	Specified repeater address

pktID
[in] Packet identifier used to match callbacks to a specific function call. Range from 1 to 223. Each consecutive API function **Must** have different packet ID than previous one.

Return Values

RF_S_PEND if function acknowledged by API;
RF_S_OK_PEND if function acknowledged and responded successfully by each reader or tag when broadcasting, it should be followed by RF_S_DONE when function times out and there are no more responses from the reader, repeater, or tag;
RF_S_DONE if function successful and acknowledged by reader, repeater, or API;
Error code otherwise.

rfReaderEvent_t structure in reader call back function contains information about reader being reset. The table below describes which items in the rfReaderEvevt_t structure are valid for this function.

CmdType : ALL_READERS ALL_REPEATERS				CmdType: SPECIFIC_READER SPECIFIC_REPEATER	
Items	Valid			Valid	
	RF_S_OK_PEND	RF_S_DONE	RF_E_XXX	RF_S_DONE	RF_E_XXX
host	Yes	NO	Yes	Yes	Yes
reader	Yes	No	Yes	Yes	Yes
repeater	Yes	No	Yes	Yes	Yes
relay	No	No	No	No	No
fGenerator	No	No	No	No	No
eventType	Yes	Yes	Yes	Yes	Yes
cmdType	Yes	Yes	Yes	Yes	Yes
eventStatus	Yes	Yes	Yes	Yes	Yes
errorStatus	No	No	Yes	No	Yes
pktID	Yes	Yes	Yes	Yes	Yes
cmdRef	No	No	No	No	No
data	No	No	No	No	No
versionInfo	No	No	No	No	No

Table Index

2.9 rfResetReaderSocket

The **rfResetReaderSocket** function requests a reader with **specific IP address** to perform a reset sequence. The API will call the registered **rfReaderEvent** callback function when the reader is back on line.

```
long rfResetReaderSocket (
  UInt16 host,
  Byte ip[],
  UInt16 pktID);
```

Parameters

host

[in] Address of the host.

ip

[in] Reader internet address in form of xxx.xxx.xxx.xxx, or a valid host name.

pktID

[in] Packet identifier used to match callbacks to a specific function call. Range from 1 to 223. Each consecutive API function **Must** have different packet ID than previous one.

Return Values

RF_S_PEND if function acknowledged by API;
RF_S_OK_PEND if function acknowledged and responded successfully by each reader or tag when broadcasting, it should be followed by RF_S_DONE when function times out and there are no more responses from the reader, repeater, or tag;
RF_S_DONE if function successful and acknowledged by reader, repeater, or API;
Error code otherwise.

rfReaderEvent_t structure in reader call back function contains information about reader being reset. The table below describes which items in the **rfReaderEvevt_t** structure are valid for this function.

CmdType : ALL_READERS ALL_REPEATERS				CmdType: SPECIFIC_READER SPECIFIC_REPEATER	
Items	Valid			Valid	
	RF_S_OK_PEND	RF_S_DONE	RF_E_XXX	RF_S_DONE	RF_E_XXX
host	Yes	NO	Yes	Yes	Yes
reader	Yes	No	Yes	Yes	Yes
repeater	Yes	No	Yes	Yes	Yes
relay	No	No	No	No	No
fGenerator	No	No	No	No	No
eventType	Yes	Yes	Yes	Yes	Yes
cmdType	Yes	Yes	Yes	Yes	Yes
eventStatus	Yes	Yes	Yes	Yes	Yes
errorStatus	No	No	Yes	No	Yes

pktID	Yes	Yes	Yes	Yes	Yes
cmdRef	No	No	No	No	No
data	No	No	No	No	No
versionInfo	No	No	No	No	No

Table Index

2.10 rfQueryReader

The **rfQueryReader** function acknowledges that a reader is on-line and able to process commands. The function can be targeted to a specific reader or broadcast to all readers. The broadcast mode allows the application to find out the address of all readers that are on-line. The API will call the registered **rfReaderEvent** callback function for each reader response.

```
long rfQueryReader(
  UInt16 host,        // address of host
  UInt16 reader,      // address of reader
  UInt16 repeater,    // address of repeater
  UInt16 cmdType,     // command type
  UInt16 pktID        // packet id used in callback
);
```

Parameters

host
[in] Address of the host.

reader
[in] Address of the reader.

repeater
[in] Address of the repeater.

cmdType
[in] Send the command to specific reader or broadcast it to all readers.

Value	Meaning
ALL_READERS	All readers in the system
ALL_REPEATERS	All repeaters in the system
SPECIFIC_READER	Specified reader address
SPECIFIC_REPEATER	Specified repeater address

pktID
[in] Packet identifier used to match callbacks to a specific function call. Range from 1 to 223. Each consecutive API function **Must** have different packet ID than previous one.

Return Values

RF_S_PEND if function acknowledged by API;
RF_S_OK_PEND if function acknowledged and responded successfully by each reader or tag when broadcasting, it should be followed by RF_S_DONE when function times out and there are no more responses from the reader, repeater, or tag;
RF_S_DONE if function successful and acknowledged by reader, repeater, or API;
Error code otherwise.

rfReaderEvent_t structure in reader call back function contains information about reader being queried. **Data** item in the rfReaderEvent_t structure contains information for **reader configuration** and **reader type**. The table below describes which items in the rfReaderEvevt_t structure are valid for this function.

CmdType : ALL_READERS ALL_REPEATERS				**CmdType: SPECIFIC_READER SPECIFIC_REPEATER**	
Items	Valid			Valid	
	RF_S_OK_PEND	RF_S_DONE	RF_E_XXX	RF_S_DONE	RF_E_XXX
host	Yes	NO	Yes	Yes	Yes
reader	Yes	No	Yes	Yes	Yes
repeater	Yes	No	Yes	Yes	Yes
relay	No	No	No	No	No
fGenerator	No	No	No	No	No
eventType	Yes	Yes	Yes	Yes	Yes
cmdType	Yes	Yes	Yes	Yes	Yes
eventStatus	Yes	Yes	Yes	Yes	Yes
errorStatus	No	No	Yes	No	Yes
pktID	Yes	Yes	Yes	Yes	Yes
cmdRef	No	No	No	No	No
data	Yes	No	No	Yes	No
versionInfo	No	No	No	No	No

data (1 byte) bit assignment for response to rfQueryReader.

Reader Type:

3	2	1	0	Reader Type

Reader Type = 0 No change
Reader Type = 1 Programming Station
Reader Type = 2 Standard Reader
Reader Type = 3 Access Control Reader
Reader Type = 4 Small RF Reader
Reader Type = 5 PDA Reader

Reader Configuration:

7	6	5	4	Reader Configure Status

Bit 4: 0 = Respond to broadcast commands.
1 = Do not respond

.

Bit 5: 0 = Enable at power up.
1 = Disable at power up.

Bit 6: 0 = Do not send RSSI byte.
1 = Send RSSI byte.

Bit 7: 0 = Do not change configuration.
1 = Change configuration.

2.11 rfPowerupReader

This function should **only** be called in response to reader's callback function eventType RF_READER_POWERUP. The reader sends this command when it first powers up. rfReaderEvent_t structure in reader call back function contains information about reader being powered up. The reader sets host and reader address to zero if it has not been initialized before. The cmdRef should not be altered, it is used by API internally. This function executes synchronously by the API and there is no callback function for this command.

```
long rfPowerupReader(
  UInt16 host,          // address of the current host
  UInt16 reader,        // address of the current reader
  UInt16 repeater,      // address of the current repeater
  UInt16 newHost,       // address of the new host
  UInt16 newReader,     // address of the new reader
  UInt16 newRepeater,   // address of the new repeater
  UInt16 config,        // configuration of the reader
  UInt16 type,          // type of reader
  UInt16 dynamicRdr,    // address of the dynamic reader or repeater
  UInt16 cmdRef         // API internal use
  UInt16 pktID,         // packet id used in callback
);
```

Parameters

host

[in] Address of the current host.

reader

[in] Address of the current reader.

repeater

[in] Address of the current repeater.

newHost

[in] Address of the new host.

newReader

[in] Address of the new reader.

newRepeater

[in] Address of the new repeater.

config

[in] Identifies which reader parameter to configure. These parameters can be combined (Ored) together.

Configuration
RF_NO_BROADCAST_RESPONSE
RF_POWERUP_DISABLE

RF_SEND_RSSI_BYTE
RF_NO_CHANGE

type

[in] Identifies which type of readers to configure. Must be one of the values in the table below.

Reader Type
RF_RDR_PROG_STATION
RF_RDR_STANDARD
RF_RDR_ACCESS_CTRL
RF_RDR_SMALL_RF
RF_RDR_PDA
RF_RDR_NO_CHANGE

dynamicReader

[in] Dynamic reader address.

cmdRef

[in] Reserved for internal use by the API. The application must not change its value.

pktID

[in] Packet identifier used to match callbacks to a specific function call. Range from 1 to 223. Each consecutive API function **Must** have different packet ID than previous one.

Return Values

RF_S_PEND if function acknowledged by API;
RF_S_OK_PEND if function acknowledged and responded successfully by each reader or tag when broadcasting, it should be followed by RF_S_DONE when function times out and there are no more responses from the reader, repeater, or tag;
RF_S_DONE if function successful and acknowledged by reader, repeater, or API;
Error code otherwise.

Table Index

2.12 rfConfigureReader

The **rfConfigureReader** function changes reader or repeater address and configures its's type and mode of operation.

```
long rfConfigureReader(
  UInt16 host,          // address of the host
  UInt16 reader,        // address of the reader
  UInt16 repeater,      // address of the repeater
  UInt16 cmdType,       // command type
  UInt32 configType,    // Config type
  UInt16 param1,        // parameter 1
  UInt16 param2,        // parameter 2
  UInt16 param3,        // parameter 3
  UInt16 param4,        // parameter 4
  UInt16 pktID          // packet ID
);
```

Parameters

host

[in] Address of the host.

reader

[in] Address of the reader.

repeater

[in] Address of the repeater.

cmdType

[in] Send the command to specific reader or broadcast it to all readers.

Value	Meaning
ALL_READERS	All readers in the system
ALL_REPEATERS	All repeaters in the system
SPECIFIC_READER	Specified reader address
SPECIFIC_REPEATER	Specified repeater address

configType

[in] Identifies which reader parameter to configure. Must be values **from ONE of the groups in following tables** below.

Group1

The following 4 values from **group 1** can be (Ored) together. Exception: CFG_READER_ID and CFG_REPEATER_ID can not be ored together.

Value	Meaning
CFG_HOST_ID	Configure Reader with new Host address. (**param1**)
CFG_READER_ID	Configure Reader with new address. (**param2**)
CFG_REPEATER_ID	Configure Repeater with new address (**param2**)
CFG_READER_TYPE	Configure Reader Type (**param3**)
CFG_READER_OPTIONS	Configure Reader Options (**param4**)

Group2

Value	Meaning
CFG_READER_TX_FGEN	Configure reader Transmit time (**param1)** , field gen.Wait time (**param2),** time unit for wait time (**param3)** and motion detector configuration (**param4**).

Param1

[in] If *ConfigType* = CFG_HOST_ID then Address of the new host **.**
If *ConfigType* = CFG_TRANSMITTER then Reader Transmit Time value.

Param2

[in] If *ConfigType* = CFG_READER_ID then address of the new reader**.**
If *ConfigType* = CFG_REPEATER_ID then address of the new repeater**.**
If *ConfigType* = CFG_TX_FGEN then Wait Time value. **Not applicable to RF_RDR_STANDARD type reader.**

Param3

[in] If *ConfigType* = CFG_READER_TYPE then one of the values from **group 1**.
If *ConfigType* = CFG_TX_FGEN then motion detector configuration value. **Not applicable to RF_RDR_STANDARD type reader.** Motion Detector. Configuration can be values from the **group2** and **group3**.Group2 and group3 can be combined together (Ored).

Group 1 - CFG_READER_TYPE

Value	Meaning
RF_RDR_PROG_STATION	Not available.
RF_RDR_STANDARD	Default setting for all readers at the factory.
RF_RDR_ACCESS_CTRL	Not available.
RF_RDR_SMALL_RF	Not available.
RF_RDR_PDA	Not available.
RF_FGEN_READER	Reader acting as field generator.

Group 2 - CFG_TX_FGEN

Value	Meaning
RF_MD_ACTIVE_HIGH	Set the Motion detector setting to Activate High
RF_MD_ACTIVE_LOW	Set the Motion detector setting to Activate Low

Group 3 - CFG_TX_FGEN

Value	Meaning
RF_MD_ENABLE	Enable Motion Detector.
RF_MD_DISABLE	Disable Motion Detector.

Param4

[in] If *ConfigType* = CFG_TX_FGEN then one of the value from the **group1** table should be used for **wait time**. **Not applicable to RF_RDR_STANDARD type reader.**
If *ConfigType* = CFG_READER_OPTIONS then one or more values from the **group2** table should be used. These values can be combined(ORED) together.

Group 1 - CFG_TX_FGEN

Value	Meaning
RF_TIME_HOUR	Unit of time in Hour
RF_TIME_MINUTE	Unit of time in Minute
RF_TIME_SECOND	Unit of time in second

Group 2 - CFG_READER_OPTIONS

Value	Meaning
RF_NO_CHANGE	Do not change any reader option setting.
RF_RESPOND_TO_BROADCAST	Respond to all broadcast reader commands.
RF_ENABLE_AT_POWERUP	Enable reader when it first powers up.
RF_SEND_RSSI	Send field strength value with call tag command or when tag send its tag id unsolicited.

pktID

[in] Packet identifier used to match callbacks to a specific function call. Range from 1 to 223. Each consecutive API function **Must** have different packet ID than previous one.

Return Values

RF_S_DONE if function successful and acknowledged by reader, repeater, or API;
Error code otherwise.

Table Index

2.13 rfGetReaderConfig

The **rfGetReaderConfig** function requests reader transmit time, reader field generator wait time, wait time type (sec, min, hour) and reader field strength from the reader. The API will call the registered **rfReaderEvent** callback function for each reader response.

```
long rfEnableReader(
  UInt16 host,        // address of host
  UInt16 reader,      // address of reader
  UInt16 repeater,    // address of repeater
  UInt16 cmdType,     // command type
  UInt16 pktID        // packet id used in callback
);
```

Parameters

host

[in] Address of the host.

reader

[in] Address of the reader.

repeater

[in] Address of the repeater.

cmdType

[in] Send the command to specific reader or broadcast it to all readers.

Value	Meaning
ALL_READERS	All readers in the system
ALL_REPEATERS	All repeaters in the system
SPECIFIC_READER	Specified reader address
SPECIFIC_REPEATER	Specified repeater address

pktID

[in] Packet identifier used to match callbacks to a specific function call. Range from 1 to 223. Each consecutive API function **Must** have different packet ID than previous one.

Return Values

RF_S_PEND if function acknowledged by API;
RF_S_DONE if function successful and acknowledged by reader, repeater, or API;
Error code otherwise.

rfReaderEvent_t structure in reader call back function contains information about reader configuration. This information will be available in **smartFgen** in **rfReaderEvent_t** structure. ***txTime***, ***waitTime***, **wTimeType**, and ***fsValue*** parameters in ***rfSmartFGen_t*** structure.

2.14 rfEnableReader

The **rfEnableReader** function enables or disables the RF functions of the reader. When disabled, the reader will continue to process all commands from the host, but any commands requiring RF communications will return an error status. The API will call the registered **rfReaderEvent** callback function for each reader response.

```
long rfEnableReader(
  UInt16 host,        // address of host
  UInt16 reader,      // address of reader
  UInt16 repeater,    // address of repeater
  BooLean enable,     // enable/disable
  UInt16 cmdType,     // command type
  UInt16 pktID        // packet id used in callback
);
```

Parameters

host

[in] Address of the host.

reader

[in] Address of the reader.

repeater

[in] Address of the repeater.

enable

[in] If true, the reader is enabled. If false, the reader is disabled.

cmdType

[in] Send the command to specific reader or broadcast it to all readers.

Value	Meaning
ALL_READERS	All readers in the system
ALL_REPEATERS	All repeaters in the system
SPECIFIC_READER	Specified reader address
SPECIFIC_REPEATER	Specified repeater address

pktID

[in] Packet identifier used to match callbacks to a specific function call. Range from 1 to 223. Each consecutive API function **Must** have different packet ID than previous one.

Return Values

RF_S_PEND if function acknowledged by API;

RF_S_DONE if function successful and acknowledged by reader, repeater, or API; Error code otherwise.

rfReaderEvent_t structure in reader call back function contains information about reader being enabled or disabled. The table below describes which items in the **rfReaderEvent_t** structure are valid for this function.

CmdType : ALL_READERS ALL_REPEATERS				**CmdType: SPECIFIC_READER SPECIFIC_REPEATER**	
Items	Valid			Valid	
	RF_S_OK_PEND	RF_S_DONE	RF_E_XXX	RF_S_DONE	RF_E_XXX
host	Yes	NO	Yes	Yes	Yes
reader	Yes	No	Yes	Yes	Yes
repeater	Yes	No	Yes	Yes	Yes
relay	No	No	No	No	No
fGenerator	No	No	No	No	No
eventType	Yes	Yes	Yes	Yes	Yes
cmdType	Yes	Yes	Yes	Yes	Yes
eventStatus	Yes	Yes	Yes	Yes	Yes
errorStatus	No	No	Yes	No	Yes
pktID	Yes	Yes	Yes	Yes	Yes
cmdRef	No	No	No	No	No
data	No	No	No	No	No
versionInfo	No	No	No	No	No

Table Index

2.15 rfEnableRelay

The **rfEnableRelay** function controls the status of the output relay on the reader. The function can be targeted to a specific reader or broadcast to all readers. The API will call the registered **rfReaderEvent** callback function for each reader response.

```
long rfEnableRelay(
  UInt16 host,        // address of the host
  UInt16 reader,      // address of the reader
  UInt16 repeater,    // address of the repeater
  UInt16 relay,       // address of the output relay
  Boolean enable,     // enable/disable flag
  UInt16 cmdType,     // command type
  UInt16 pktID        // packet id used in callback
);
```

Parameters

host

[in] Address of the host.

reader

[in] Address of the reader.

repeater

[in] Address of the repeater.

relay

[in] Address of the relay. Bit 0 to bit 7 represents 8 relay address (1 – 8) .

enable

[in] If true, relay output is enabled. If false, relay output is disabled.

cmdType

[in] Send the command to specific reader or broadcast it to all readers.

Value	Meaning
ALL_READERS	All readers in the system
SPECIFIC_READER	Specified reader address

pktID

[in] Packet identifier used to match callbacks to a specific function call. Range from 1 to 223. Each consecutive API function **Must** have different packet ID than previous one.

Return Values

RF_S_DONE if function successful and acknowledged by reader, repeater, or API;
Error code (**RF_E_XXX**) otherwise.

Return Value	Value	Meaning
RF_S_DONE	0	All readers in the system
RF_E_ARGUMENT	-104	There is an error with one or more parameters.
RF_E_TX	-116	Error in transmitting the packet to the reader.
RF_E_NOT_SUPPORTED	-102	Function is not supported.
RF_E_QUE_FULL	-118	Queue is full for calling callback function to application.

rfReaderEvent_t structure in reader call back function contains information about relay being enabled or disabled. The table below describes which items in the **rfReaderEvent_t** structure are valid for this function.

CmdType : ALL_READERS ALL_REPEATERS				CmdType: SPECIFIC_READER SPECIFIC_REPEATER	
Items	Valid			Valid	
	RF_S_OK_PEND	RF_S_DONE	RF_E_XXX	RF_S_DONE	RF_E_XXX
host	Yes	NO	Yes	Yes	Yes
reader	Yes	No	Yes	Yes	Yes
repeater	Yes	No	Yes	Yes	Yes
relay	No	No	No	No	No
fGenerator	No	No	No	No	No
eventType	Yes	Yes	Yes	Yes	Yes
cmdType	Yes	Yes	Yes	Yes	Yes
eventStatus	Yes	Yes	Yes	Yes	Yes
errorStatus	No	No	Yes	No	Yes
pktID	Yes	Yes	Yes	Yes	Yes
cmdRef	No	No	No	No	No
data	No	No	No	No	No
versionInfo	No	No	No	No	No

Table Index

2.16 rfGetInputPortStatus

The **rfGetInputPortStatus** function requests the status of input port from the reader. The function can be targeted to a specific reader or broadcast to all readers. The API will call the registered **rfReaderEvent** callback function for each reader response.

```
long rfGetInputPortStatus(
  UInt16 host,        // address of the host
  UInt16 reader,      // address of the reader
  UInt16 repeater,    // address of the repeater
  UInt16 cmdType,     // command type
  UInt16 pktID        // packet id used in callback
);
```

Parameters

host

[in] Address of the host.

reader

[in] Address of the reader.

repeater

[in] Address of the repeater.

cmdType

[in] Send the command to specific reader or broadcast it to all readers.

Value	Meaning
ALL_READERS	All readers in the system
SPECIFIC_READER	Specified reader address

pktID

[in] Packet identifier used to match callbacks to a specific function call. Range from 1 to 223. Each consecutive API function **Must** have different packet ID than previous one.

Return Values

RF_S_PEND if function acknowledged by API;
RF_S_OK_PEND if function acknowledged and responded successfully by each reader or tag when broadcasting, it should be followed by RF_S_DONE when function times out and there are no more responses from the reader, repeater, or tag;
RF_S_DONE if function successful and acknowledged by reader, repeater, or API;
Error code otherwise.

rfReaderEvent_t structure in reader callback function contains information about status of input port of the reader. The **data**[0] and **data**[1] items in rfReaderEvent_t structure contain data for input status for port 1 and port 2 of the reader, respectively. The size of each input status field is one byte. The table below describes the possible values for status of reader input:

State	Value	Meaning
NORMAL_CLOSED	0	
NORMAL_OPEN	1	
FAULTY_CLOSED	2	
FAULTY_OPEN	3	

Table Index

2.17 rfConfigInputPort

The **rfConfigInputPort** function configures the reader input port. The function can be targeted to a specific reader or broadcast to all readers. The API will call the registered **rfReaderEvent** callback function for each reader response. The status change generates **RF_GET_INPUT_STATUS** or **RF_GET_INPUT_STATUS_ALL** events for the active inputs.

```
long rfConfigInputPort(
  UInt16 host,           // address of the host
  UInt16 reader,         // address of the reader
  UInt16 repeater,       // address of the repeater
  UInt16 input1Config,   // configuration for input1
  UInt16 input2Config,   // configuration for input2
  Boolean supervised,    // inputs are supervised
  UInt16 cmdType,        // command type
  UInt16 pktID           // packet id used in callback
);
```

Parameters

host

[in] Address of the host.

reader

[in] Address of the reader.

repeater

[in] Address of the repeater.

input1Config

[in] Input1 can be configured as one of the following parameters.

Value	Meaning
IGNOR_INPUT_CHANGE	Do not report when input status changes.
REPORT_INPUT_CHANGE	Report when input status changes.
NO_CHANGE_INPUT	Do not set the input port with new configuration.

input2Config

[in] Input2 can be configured as one of the following parameters.

Value	Meaning
IGNOR_INPUT_CHANGE	Do not report when input status changes.
REPORT_INPUT_CHANGE	Report when input status changes.
NO_CHANGE_INPUT	Do not set the input port with new configuration.

supervised

[in] Inputs are supervised by the calling application.

cmdType

[in] Send the command to specific reader or broadcast it to all readers.

Value	Meaning
ALL_READERS	All readers in the system
SPECIFIC_READER	Specified reader address

pktID

[in] Packet identifier used to match callbacks to a specific function call. Range from 1 to 223. Each consecutive API function **Must** have different packet ID than previous one.

Return Values

RF_S_PEND if function acknowledged by API;
RF_S_OK_PEND if function acknowledged and responded successfully by each reader or tag when broadcasting, it should be followed by RF_S_DONE when function times out and there are no more responses from the reader, repeater, or tag;
RF_S_DONE if function successful and acknowledged by reader, repeater, or API;
Error code otherwise.

Table Index

2.18 rfSetReaderFS

The **rfSetReaderFS** function requests the reader to set its Field Strength to a specific value from 0 to 20 or to increment or decrement the Field Strength value and/or to change the reader 433MHz from low to high level. The reader will ack this command by sending the current field strength back to calling application. The API will call the registered **rfReaderEvent** callback function for each reader response.

```
long rfSetReaderFS(
  UInt16 host,         // address of the host
  UInt16 reader,       // address of the reader
  UInt16 repeater,     // address of the repeater
  UInt16 actionType,   // type of setting
  Byte   fsValue,      // field strength value(0-20)
  Boolean range,       // short or long range
  UInt16 cmdType,      // command type
  UInt16 pktID         // packet id used in callback
);
```

Parameters

host

[in] Address of the host.

reader

[in] Address of the reader.

repeater

[in] Address of the repeater.

actionType

[in] Send the command to reader to set field strength or increment or decrement.

Value	Meaning
RF_ABSOLUTE	Field strength absolute value. Uses *fsValue and range*
RF_INCREMENT	Increment current field strength by one. Ignores *fsValue and range*
RF_DECREMENT	Decrement current field strength by one. Ignore *fsValue and range*

fsValue

[in] Field strength value to be set in the reader (0-20). If actionType is RF_ABSOLUTE.

range

[in] Field strength level. If true long range otherwise short range. If actionType is RF_ABSOLUTE.

cmdType

[in] Send the command to specific reader or broadcast it to all readers.

pktID

[in] Packet identifier used to match callbacks to a specific function call. Range from 1 to 223. Each consecutive API function **Must** have different packet ID than previous one.

Return Values :

RF_S_PEND if function acknowledged by API;
RF_S_OK_PEND if function acknowledged and responded successfully by each reader or tag when broadcasting, it should be followed by RF_S_DONE when function times out and there are no more responses from the reader, repeater, or tag;
RF_S_DONE if function successful and acknowledged by reader, repeater, or API; Error code (**RF_E_XXX**) otherwise.

The following table represents some of the error codes that reader would send to the calling application in case of an error:

Return Value	Value	Meaning
RF_E_ARGUMENT	-104	There is an error with one or more parameters.
RF_E_TX	-116	Error in transmitting the packet to the reader.
RF_E_NOT_SUPPORTED	-102	Function is not supported.
RF_E_QUE_FULL	-118	Queue is full for calling callback function to application.
RF_E_READER_BUSY	-131	Reader is still processing last received data or command.

Table Index

2.19 rfGetReaderFS

The **rfGetReaderFS** function requests the reader to send its Field Strength value. The API will call the registered **rfReaderEvent** callback function for each reader response.

```
long rfGetReaderFS(
  UInt16 host,         // address of the host
  UInt16 reader,       // address of the reader
  UInt16 repeater,     // address of the repeater
  UInt16 cmdType,      // command type
  UInt16 pktID         // packet id used in callback
);
```

Parameters

host

[in] Address of the host.

reader

[in] Address of the reader.

repeater

[in] Address of the repeater.

cmdType

[in] Send the command to specific reader or broadcast it to all readers.

Value	Meaning
ALL_READERS	All readers in the system
SPECIFIC_READER	Specified reader address

pktID

[in] Packet identifier used to match callbacks to a specific function call. Range from 1 to 223. Each consecutive API function **Must** have different packet ID than previous one.

Return Values

RF_S_PEND if function acknowledged by API;
RF_S_OK_PEND if function acknowledged and responded successfully by each reader or tag when broadcasting, it should be followed by RF_S_DONE when function times out and there are no more responses from the reader, repeater, or tag;
RF_S_DONE if function successful and acknowledged by reader, repeater, or API; Error code otherwise.

Table Index

2.20 rfRegisterReaderEvent

The **rfRegisterReaderEvent** function is used to register an application implemented callback function that will be called by the API when a reader event occurs. This function executes synchronously.

```
void rfRegisterReaderEvent(rfReaderEvent fnCallback,     // callback function
                           void* userArg                 // user-defined value
);
```

Parameters

fnCallback

[in] Callback function to be called whenever a reader comes on-line. This value can be NULL to disable callbacks. The callback function prototype is:

```
typedef long (*rfReaderEvent)(Int32 status,
                              HANDLE funcId,
                              rfReaderEvent_t* readerEvent,
                              void* userArg);
```

Note that the **readerEvent** argument is only valid within the scope of the function.

userArg

[in] User defined value passed to callback function.

Table Index

2.21 rfQuerySmartFGen

The **rfQuerySmartFGen** function reads configuration data from one or more Smart Field Generator. This function executes asynchronously. The API will call the registered **rfReaderEvent** callback function for each Smart FGen response.

```
Int32 rfQuerySmartFGen(
  UInt16  host,         // address of reader
  UInt16  reader,       // address of host
  UInt16  repeater,     // address of repeater
  UInt16  sFGen,        // address of smart field generator
  UInt16  cmdType,      // command type
  Boolean broadcast,    // broadcast to all smart field generator
  UInt16  broadcastType,   // broadcast type
  UInt16  pktID         // packet ID
);
```

Parameters

host
[in] Address of the host.

reader
[in] Address of the reader.

repeater
[in] Address of the repeater.

sFGen
[in] Address of the smart field generator.

broadcast
[in] Broadcast to all smart field generator when true, otherwise to a specific smart field generator.

broadcastType
[in] If broadcast param is set to **true** API will check the *broadcastType* param otherwise API will ignore it. For RESPOND_SPEC_RDR param Smart fgen will check the incoming reader address with its assigned reader address, if it matches then it will respond to that reader. For RESPOND_SPEC_RDR param smart fgen do not check the reader address and it will respond to any reader address.

Value	Meaning
RESPOND_SPEC_RDR	Smart fgen respond to specific reader address
RESPOND_ANY_RDR	Smart fgen respond to any reader address

cmdType

[in] Send the command to specific reader or broadcast it to all readers.

Value	Meaning
ALL_READERS	All readers in the system
SPECIFIC_READER	Specified reader address

pktID

[in] Packet identifier used to match callbacks to a specific function call. Range from 1 to 223. Each consecutive API function **Must** have different packet ID than previous one.

Return Values

RF_S_PEND if function acknowledged by API;

RF_S_OK_PEND if function acknowledged and responded successfully by each reader, smart FGen tag when broadcasting, it should be followed by RF_S_DONE when function times out and there are no more responses from the reader, repeater, smart FGen, or tag;

RF_S_DONE if function successful and acknowledged by reader, repeater, smart FGen, or API; Error code otherwise.

Table Index

2.22 rfCallTagSmartFGen

The **rfCallTagSmartFGen** function requests one or more tags to transmit their tag id and status. The reader will send the request to the Smart Field Generator and also acknowledge this command without getting response from the tag. The tag will send its tag id and status asynchronously. The API will call the registered **rfTagEvent** callback function when a tag is detected. The purpose of this command is to wakeup all or particular tag(s) in the field using Smart Field Generator. The application calling this function is responsible for tracking the response form the tag in the **rfTagEvent** callback function.

```
long rfCallTagSmartFGen(
  UInt16  host,         // address of the host
  UInt16  reader,       // address of the reader
  UInt16  repeater,     // address of the repeater
  UInt16  sFieldGen,    // address of the smart field Generator
  UInt16  cmdType,      // command type
  Boolean broadcast,    // broadcast to all smart field generator
  rfTagSelect_t* tagSelect, // tag list
  Boolean setTxTime,        // set time interval between tag transmit time
  Boolean longInterval,     // time interval between each tag transmission
  UInt16 pktID          // packet id used in callback
);
```

Parameters

host

[in] Address of the host.

reader

[in] Address of the reader.

repeater

[in] Address of the repeater.

sFieldGen

[in] Address of the smart field generator.

cmdType

[in] Send the command to specific reader or broadcast it to all readers.

Value	Meaning
ALL_READERS	All readers in the system
SPECIFIC_READER	Specified reader address

broadcast

[in] Broadcast to all smart field generator when true, otherwise to a specific smart field generator.

tagSelect

[in] Pointer to a tag select structure that identifies which tags should be selected. Refer to the definition of the **`rfTagSelect_t`** structure for more information.

setTxTime

[in] If true, the tag will use longInterval parameter value to set its transmit time interval

longInterval

[in] time interval between each tag transmission.

tagSelect

[in] Pointer to a tag select structure that identifies which tags should be selected. Refer to the definition of the **`rfTagSelect_t`** structure for more information.

pktID

[in] Packet identifier used to match callbacks to a specific function call. Range from 1 to 223. Each consecutive API function **Must** have different packet ID than previous one.

Return Values

RF_S_PEND if function acknowledged by API;
RF_S_OK_PEND if function acknowledged and responded successfully by each reader, smart FGen tag when broadcasting, it should be followed by RF_S_DONE when function times out and there are no more responses from the reader, repeater, smart FGen, or tag;
RF_S_DONE if function successful and acknowledged by reader, repeater, smart FGen, or API; Error code otherwise.

Table Index

2.23 rfSetConfigSmartFGen

The **rfSetConfigSmartFGen** function reads configuration data from one or more Smart Field Generator. This function executes asynchronously. The API will call the registered **rfReaderEvent** callback function for each Smart FGen response.

```
Int32 rfSetConfigSmartFGen(
  UInt16   host,          // address of reader
  UInt16   reader,        // address of host
  UInt16   repeater,      // address of repeater
  UInt16   sFGen,         // address of smart field generator
  UInt16   cmdType,       // command type
  rfSmartFGen_t* smartFGenData,   // smart field generator data structure
  UInt16   pktID,         // packet ID
);
```

Parameters

host
[in] Address of the host.

reader
[in] Address of the reader.

repeater
[in] Address of the repeater.

sFGen
[in] Address of the smart field generator.

cmdType
[in] Send the command to specific reader or broadcast it to all readers.

Value	Meaning
ALL_READERS	All readers in the system
SPECIFIC_READER	Specified reader address

smartFGenData
[in] Pointer to a smart field generator structure that holds information for the smart fgen. See the definition of the **rfSmartFGen_t** structure for more information.

pktID
[in] Packet identifier used to match callbacks to a specific function call. Range from 1 to 223. Each consecutive API function **Must** have different packet ID than previous one.

Return Values

RF_S_PEND if function acknowledged by API;
RF_S_OK_PEND if function acknowledged and responded successfully by each reader, smart FGen tag when broadcasting, it should be followed by RF_S_DONE when function times out and there are no more responses from the reader, repeater, smart FGen, or tag;
RF_S_DONE if function successful and acknowledged by reader, repeater, smart FGen, or API; Error code otherwise.

Table Index

2.24 rfResetSmartFGen

The **rfResetSmartFGen** function requests a smart Field Generator to perform a soft reset . The API will call the registered **rfReaderEvent** callback function when API receives the response from the smart field generator.

```
Int32 rfResetSmartFGen(
  UInt16   host,        // address of reader
  UInt16   reader,      // address of host
  UInt16   repeater,    // address of repeater
  UInt16   sFGen,       // address of smart field generator
  UInt16   cmdType,     // command type
  Boolean  broadcast,   // broadcast to all smart field generator
  UInt16   pktID        // packet ID
);
```

Parameters

host

[in] Address of the host.

reader

[in] Address of the reader.

repeater

[in] Address of the repeater.

sFGen

[in] Address of the smart field generator.

broadcast

[in] Broadcast to all smart field generator when true, otherwise to a specific smart field generator.

cmdType

[in] Send the command to specific reader or broadcast it to all readers.

Value	Meaning
ALL_READERS	All readers in the system
SPECIFIC_READER	Specified reader address

pktID

[in] Packet identifier used to match callbacks to a specific function call. Range from 1 to 223. Each consecutive API function **Must** have different packet ID than previous one.

Return Values

RF_S_PEND if function acknowledged by API;
RF_S_OK_PEND if function acknowledged and responded successfully by each reader, Smart FGen tag when broadcasting, it should be followed by RF_S_DONE when function times out and there are no more responses from the reader, repeater, smart FGen, or tag;
RF_S_DONE if function successful and acknowledged by reader, repeater, smart FGen, or API; Error code otherwise.

Table Index

2.25 rfSetSmartFGenFS

The **rfSetSmartFGenFS** function sets a smart Field Generator **Field Strength**. The Field Strength can be incremented by one or decremented by one or set to absolute value. The Field Strength range is from 0 to 20. The API will call the registered **rfReaderEvent** callback function when API receives the response from the smart field generator.

```
Int32 rfSetSmartFGenFS (
  UInt16   host,         // address of reader
  UInt16   reader,       // address of host
  UInt16   repeater,     // address of repeater
  UInt16   sFGen,        // address of smart field generator
  UInt16   cmdType,      // command type
  Boolean  broadcast,    // broadcast to all smart field generator
  UInt16   actionType,   // action type
  UInt16   absValue,     // field strength absolute value
  UInt16   pktID,        // packet ID
);
```

Parameters

host
[in] Address of the host.

reader
[in] Address of the reader.

repeater
[in] Address of the repeater.

sFGen
[in] Address of the smart field generator.

broadcast
[in] Broadcast to all smart field generator when true, otherwise to a specific smart field generator.

cmdType
[in] Send the command to specific reader or broadcast it to all readers.

Value	Meaning
ALL_READERS	All readers in the system
SPECIFIC_READER	Specified reader address

actionType

[in] Type of action to be taken to set Smart FGen Field Strength.

Value	Meaning
RF_INC_FS	Increment Field Strength by one
RF_DEC_FS	Decrement Field Strength by one
RF_ABS_FS	Set Field Strength to an absolute value

absValue

[in] Field Strength absolute value when *actionType* is ABS_FS; otherwise set to zero. This parameter will be ignored by API if actionType is not ABS_FS.

pktID

[in] Packet identifier used to match callbacks to a specific function call. Range from 1 to 223. Each consecutive API function **Must** have different packet ID than previous one.

Return Values

RF_S_PEND if function acknowledged by API;
RF_S_OK_PEND if function acknowledged and responded successfully by each reader, Smart FGen tag when broadcasting, it should be followed by RF_S_DONE when function times out and there are no more responses from the reader, repeater, smart FGen, or tag;
RF_S_DONE if function successful and acknowledged by reader, repeater, smart FGen, or API; Error code otherwise.

Table Index

2.26 rfGetSmartFGenFS

The **rfGetSmartFGenFS** function gets a smart Field Generator **Field Strength**. The Field Strength range is from 0 to 20. The API will call the registered **rfReaderEvent** callback function when API receives the response from the smart field generator.

```
Int32 rfGetSmartFGenFS (
  UInt16   host,         // address of reader
  UInt16   reader,       // address of host
  UInt16   repeater,     // address of repeater
  UInt16   sFGen,        // address of smart field generator
  UInt16   cmdType,      // command type
  Boolean  broadcast,    // broadcast to all smart field generator
  UInt16   pktID         // packet ID
);
```

Parameters

host

[in] Address of the host.

reader

[in] Address of the reader.

repeater

[in] Address of the repeater.

sFGen

[in] Address of the smart field generator.

broadcast

[in] Broadcast to all smart field generator when true, otherwise to a specific smart field generator.

cmdType

[in] Send the command to specific reader or broadcast it to all readers.

Value	Meaning
ALL_READERS	All readers in the system
SPECIFIC_READER	Specified reader address

pktID

[in] Packet identifier used to match callbacks to a specific function call. Range from 1 to 223. Each consecutive API function **Must** have different packet ID than previous one.

Return Values

RF_S_PEND if function acknowledged by API;
RF_S_OK_PEND if function acknowledged and responded successfully by each reader, Smart FGen tag when broadcasting, it should be followed by RF_S_DONE when function times out and there are no more responses from the reader, repeater, smart FGen, or tag;
RF_S_DONE if function successful and acknowledged by reader, repeater, smart FGen, or API; Error code otherwise.

Table Index

2.27 rfQuerySTDFGen

The **rfQuerySTDFGen** function reads configuration data from Standard Field Generator. This function executes synchronously. The API will call the registered **rfReaderEvent** callback function for Standard FGen response. The communication baud rate should be set to 9600 when calling this function.

```
Int32 rfQuerySTDFGen(
  UInt16  host,         // address of reader
  UInt16  fGen,        // address of standard field generator
  UInt16  pktID         // packet ID
);
```

Parameters

host
[in] Address of the host.

fGen
[in] Address of the standard field generator.

pktID
[in] Packet identifier used to match callbacks to a specific function call. Range from 1 to 223. Each consecutive API function **Must** have different packet ID than previous one.

Return Values

RF_S_DONE if function successful and acknowledged by standard field generator; Error code otherwise.

Table Index

2.28 rfConfigSTDFGen

The **rfConfigSTDFGen** function reads configuration data from Standard Field Generator. This function executes synchronously. The API will call the registered **rfReaderEvent** callback function for Standard FGen response.

```
Int32 rfConfigSTDFGen(
  UInt16   host,              // address of reader
  UInt16   fGen,              // address of standard field generator
  UInt16   reserved,          // reserved for future use
  rfSmartFGen_t* fGenData,    // standard field generator data structure
  UInt16   pktID              // packet ID
);
```

Parameters

host
[in] Address of the host.

fGen
[in] Address of the standard field generator.

reserved
[in] Reserved for future use. Set to zero.

fGenData
[in] Pointer to a smart field generator structure that holds information for the std fgen. See the definition of the **`rfSmartFGen_t`** structure for more information.

pktID
[in] Packet identifier used to match callbacks to a specific function call. Range from 1 to 223. Each consecutive API function **Must** have different packet ID than previous one.

Return Values

RF_S_DONE if function successful and acknowledged by standard field generator; Error code otherwise.

Table Index

2.29 rfEnableTags

The **rfEnableTags** function enables or disables the tags ability to initiate commands to the reader. Depending on the tag configuration, an enabled tag may transmit information periodically or when awoken by an RF field. A disabled tag will never transmit information except in response to a reader command. The tags to be enabled or disabled can be selected based on tag id or tag type. The API will call the registered **rfTagEvent** callback function for each tag response.

```
long rfEnableTags(
  UInt16 host,            // address of the reader
  UInt16 reader,          // address of the reader
  UInt16 repeater,        // address of the reader
  rfTagSelect_t* tagSelect,    // tag list
  Boolean enable,              // enable/disable flag
  Boolean setTxTimeInterval,  // set time interval between tag transmit time
  Boolean timeInterval,  // time interval between each tag transmission
  UInt16 cmdType,         // command type
  UInt16 pktID            // packet id used in callback
);
```

Parameters

host

[in] Address of the host.

reader

[in] Address of the reader.

repeater

[in] Address of the repeater.

tagSelect

[in] Pointer to a tag select structure that identifies which tags should be selected. Refer to the definition of the **rfTagSelect_t** structure for more information. If NULL, all tags in the field will be selected.

enable

[in] If true, the tag is enabled. If false, the tag is disabled. When the tag is disabled, it will not initiate any commands to the reader, however it will still responds to all reader commands.

setTxTimeInterval

[in] If true, the tag will use *timeInterval* parameter value to set its transmit time interval between each packet that it transmits.

timeInterval

[in] if true **long** time interval between each tag transmission otherwise **short** time interval.

cmdType

[in] Send the command to specific reader or broadcast it to all readers.

Value	Meaning
ALL_READERS	All readers in the system
SPECIFIC_READER	Specified reader address

pktID

[in] Packet identifier used to match callbacks to a specific function call. Range from 1 to 223. Each consecutive API function **Must** have different packet ID than previous one.

Return Values

RF_S_PEND if function acknowledged by API;
RF_S_OK_PEND if function acknowledged and responded successfully by each reader or tag when broadcasting, it should be followed by RF_S_DONE when function times out and there are no more responses from the reader, repeater, or tag;
RF_S_DONE if function successful and acknowledged by reader, repeater, or API;
Error code otherwise.

rfTagEvent_t structure in tag callback function contains information about the tag being enabled or disabled. The **tag** structure in rfTagEvent_t structure holds information about tag ID and tag type. The table below describes which items in the rfTagEvent_t structure are valid for this function.

Select Type : RF_SELECT_FIELD RF_SELECT_TAGLIST RF_SELECT_FIELDFACTORY				**SelectType: RF_SELECT_TAGID RF_SELECT_TAGFACTORY**	
Items	Valid			Valid	
	RF_S_OK_PEND	RF_S_DONE	RF_E_XXX	RF_S_DONE	RF_E_XXX
host	Yes	NO	Yes	Yes	Yes
reader	Yes	No	Yes	Yes	Yes
repeater	Yes	No	Yes	Yes	Yes
eventType	Yes	Yes	Yes	Yes	Yes
cmdType	Yes	Yes	Yes	Yes	Yes
eventStatus	Yes	Yes	Yes	Yes	Yes
errorStatus	No	No	Yes	No	Yes
pktID	Yes	Yes	Yes	Yes	Yes
RSSI	No	No	No	No	No
tag	Yes	No	Yes	Yes	Yes
status	No	No	No	No	No

Table Index

2.30 rfQueryTags

The **rfQueryTags** function reads configuration data from one or more tags. The tags to be read can be selected based on tag id or tag type. This function executes asynchronously. The API will call the registered **rfTagEvent** callback function for each tag response.

```
long rfQueryTags(
  UInt16 host,          // connection handle
  UInt16 reader,        // address of the reader
  UInt16 repeater,      // address of the repeater
  rfTagSelect_t* tagSelect,    // tag list
  Boolean setTxTimeInterval,  // set time interval between tag transmit time
  Boolean timeInterval,       // time interval between each tag transmission
  UInt16 cmdType,       // command type
  UInt16 pktID          // packet id used in callback
);
```

Parameters

host

[in] Address of the host.

reader

[in] Address of the reader.

repeater

[in] Address of the repeater.

tagSelect

[in] Pointer to a tag select structure that identifies which tags should be selected. Refer to the definition of the **`rfTagSelect_t`** structure for more information. If NULL, all tags in the field will be selected.

setTxTimeInterval

[in] If true, the tag will use *timeInterval* parameter value to set its transmit time interval between each packet that it transmits.

timeInterval

[in] if true long time interval between each tag transmission otherwise short time interval.

cmdType

[in] Send the command to specific reader or broadcast it to all readers.

Value	Meaning
ALL_READERS	All readers in the system
SPECIFIC_READER	Specified reader address

pktID

[in] Packet identifier used to match callbacks to a specific function call. Range from 1 to 223. Each API function **Must** have different packet ID than previous one.

Return Values

RF_S_PEND if function acknowledged by API;
RF_S_OK_PEND if function acknowledged and responded successfully by each reader or tag when broadcasting, it should be followed by RF_S_DONE when function times out and there are no more responses from the reader, repeater, or tag;
RF_S_DONE if function successful and acknowledged by reader, repeater, or API;
Error code otherwise.

rfTagEvent_t structure in tag callback function contains information about the tag being queried. The **tag** structure in rfTagEvent_t structure holds information about tag ID, tag type, tag version, TIF/GC and resend time and **status** structure in **tag** structure holds information about status of the tag. The table below describes which items in the rfTagEvent_t structure are valid for this function.

Select Type : RF_SELECT_FIELD RF_SELECT_TAGLIST RF_SELECT_FIELDFACTORY				**SelectType: RF_SELECT_TAGID RF_SELECT_TAGFACTORY**	
Items	Valid			Valid	
	RF_S_OK_PEND	RF_S_DONE	RF_E_XXX	RF_S_DONE	RF_E_XXX
host	Yes	NO	Yes	Yes	Yes
reader	Yes	No	Yes	Yes	Yes
repeater	Yes	No	Yes	Yes	Yes
eventType	Yes	Yes	Yes	Yes	Yes
cmdType	Yes	Yes	Yes	Yes	Yes
eventStatus	Yes	Yes	Yes	Yes	Yes
errorStatus	No	No	Yes	No	Yes
pktID	Yes	Yes	Yes	Yes	Yes
RSSI	No	No	No	No	No
tag	Yes	No	Yes	Yes	Yes
status	Yes	No	No	Yes	No

Table Index

2.31 rfReadTags

The **rfReadTags** function reads data from one or more tags. The tags to be read can be selected based on tag id or tag type. The API will call the registered **rfTagEvent** callback function for each tag response.

```
long rfReadTags(
  UInt16 host,                  // address of the host
  UInt16 reader,                // address of the reader
  UInt16 repeater,              // address of the repeater
  rfTagSelect_t* tagSelect,     // tag list
  UInt16 address,               // starting tag memory address
  UInt16 length,                // num bytes to read
  Boolean setTxTimeInterval,    // set time interval between tag transmit time
  Boolean timeInterval,         // time interval between each tag transmission
  UInt16 cmdType,               // reader command type
  UInt16 pktID                  // packet id used in callback
);
```

Parameters

host

[in] Address of the reader.

reader

[in] Address of the reader if *cmdType* = SPECIFIC_READER otherwise zero.

repeater

[in] Address of the repeater = 0.

tagSelect

[in] Pointer to a tag select structure that identifies which tags should be selected. Refer to the definition of the **`rfTagSelect_t`** structure for more information. If NULL, all tags in the field will be selected.

address

[in] Starting memory address to read from the tag. The starting address should be within 0x00E0 and 0x3F00 range. The starting **address** + **length** should not exceed 0x3F00. The starting **address** + **length** should be within 16-byte memory boundary.

Example: 0x00100 + 16 -> within boundary.
0x00109 + 5 -> within boundary.
0x00100 + 18 -> outside boundary
0x0010A + 7 -> outside boundary

length

[in] Number of bytes that can read from the tag at a time (**Max 12 bytes**).

setTxTimeInterval

[in] If true, the tag will use *timeInterval* parameter value to set its transmit time interval between each packet that it transmits.

timeInterval

[in] If true **long** time interval between each tag transmission otherwise **short** time interval.

cmdType

[in] Reader Command Type, ALL_READERS or SPECIFIC_READER.

pktID

[in] Packet identifier used to match callbacks to a specific function call. Range from 1 to 223. Each consecutive API function **Must** have different packet ID than previous one.

Return Values:

Return Value	Value	Meaning
RF_S_DONE	0	All readers in the system
RF_E_ARGUMENT	-104	There is an error with one or more parameters.
RF_E_IP_NOTFOUND	-160	***IP*** address does not exist in API active IP list.
RF_E_MAX_READ_MEM_LEN	-127	Parameter ***length*** exceeds max length.
RF_E_USER_RAM_MEM_ADDR	-129	Parameter ***address*** is not within user ram address range.
RF_E_USER_MEM_BOUNDRY	-130	***Address*** + ***length*** exceeds user ram address range.
RF_E_QUE_FULL	-118	Queue is full for calling callback function to application.

rfTagEvent_t structure in tag callback function contains information about the tag being asked to send data from its memory. The **tag** structure in **rfTagEvent_t** structure holds information about tag ID, tag type and required tag data. The table below describes which items in the **rfTagEvent_t** structure are valid for this function.

Select Type : RF_SELECT_FIELD RF_SELECT_TAGLIST RF_SELECT_FIELDFACTORY				SelectType: RF_SELECT_TAGID RF_SELECT_TAGFACTORY	
Items	Valid			Valid	
	RF_S_OK_PEND	RF_S_DONE	RF_E_XXX	RF_S_DONE	RF_E_XXX
host	Yes	NO	Yes	Yes	Yes
reader	Yes	No	Yes	Yes	Yes
repeater	Yes	No	Yes	Yes	Yes
eventType	Yes	Yes	Yes	Yes	Yes
cmdType	Yes	Yes	Yes	Yes	Yes
eventStatus	Yes	Yes	Yes	Yes	Yes
errorStatus	No	No	Yes	No	Yes
pktID	Yes	Yes	Yes	Yes	Yes
RSSI	No	No	No	No	No
tag	Yes	No	Yes	Yes	Yes
status	Yes	No	No	Yes	No

Table Index

2.32 rfWriteTags

The **rfWriteTags** function writes data to one or more tags. The tags to be written can be selected based on tag id or tag type. The API will call the registered **rfTagEvent** callback function for each tag response.

```
long rfWriteTags(
  UInt16 host,                  // address of the host
  UInt16 reader,                // address of the reader
  UInt16 repeater,              // address of the repeater
  rfTagSelect_t* tagSelect,     // tag list
  UInt16 address,               // starting tag memory address
  UInt16 length,                // num bytes to write
  Byte data[],                  // bytes of data to write
  Boolean setTxTimeInterval,    // set time interval between tag transmit time
  Boolean timeInterval,         // time interval between each tag transmission
  UInt16 cmdType,               // reader command type
  UInt16 pktID                  // packet id used in callback
);
```

Parameters

host

[in] Address of the host.

reader

[in] Address of the reader if *cmdType* = SPECIFIC_READER otherwise zero.

repeater

[in] Address of the repeater = 0.

tagSelect

[in] Pointer to a tag select structure that identifies which tags should be selected. Refer to the definition of the **`rfTagSelect_t`** structure for more information. If NULL, all tags in the field will be selected.

address

[in] Starting memory address to write to the tag. The starting address should be within 0x00E0 and 0x3F00 range. The starting **address** + **length** should not exceed 0x3F00. The starting **address** + **length** should be within 8-byte memory boundary.
Example: 0x00100 + 8 -> within boundary.
0x00103 + 3 -> within boundary.
0x00100 + 9 -> outside boundary
0x00107 + 6 -> outside boundary

length

[in] Number of bytes that can be written to the tag at a time (**Max 12 bytes**).

data

[in] Data buffer that hold data to be written to the tag, must be at least *length* bytes long.

setTxTimeInterval

[in] If true, the tag will use *timeInterval* parameter value to set its transmit time interval between each packet that it transmits.

timeInterval

[in] If true **long** time interval between each tag transmission otherwise **short** time interval.

cmdType

[in] Reader Command Type, ALL_READERS or SPECIFIC_READER.

pktID

[in] Packet identifier used to match callbacks to a specific function call. Range from 1 to 223. Each consecutive API function **Must** have different packet ID than previous one.

Return Values:

Return Value	Meaning
RF_S_DONE	All readers in the system
RF_E_ARGUMENT	There is an error with one or more parameters.
RF_E_IP_NOTFOUND	***IP*** address does not exist in API active IP list.
RF_E_MAX_READ_MEM_LEN	Parameter ***length*** exceeds max length.
RF_E_USER_RAM_MEM_ADDR	Parameter ***address*** is not within user ram address range.
RF_E_USER_MEM_BOUNDRY	***Address*** + ***length*** exceeds user ram address range.
RF_E_QUE_FULL	Queue is full for calling callback function to application.

rfTagEvent_t structure in tag callback function contains information about the tag being asked to write data to its memory. The **tag** structure in **rfTagEvent**_t structure holds information about tag ID and tag type. The table below describes which items in the **rfTagEvent_t** structure are valid for this function.

Select Type : RF_SELECT_FIELD RF_SELECT_TAGLIST RF_SELECT_FIELDFACTORY				SelectType: RF_SELECT_TAGID RF_SELECT_TAGFACTORY	
Items	Valid			Valid	
	RF_S_OK_PEND	RF_S_DONE	RF_E_XXX	RF_S_DONE	RF_E_XXX
Host	Yes	NO	Yes	Yes	Yes
Reader	Yes	No	Yes	Yes	Yes
Repeater	Yes	No	Yes	Yes	Yes
eventType	Yes	Yes	Yes	Yes	Yes
cmdType	Yes	Yes	Yes	Yes	Yes
eventStatus	Yes	Yes	Yes	Yes	Yes
errorStatus	No	No	Yes	No	Yes
pktID	Yes	Yes	Yes	Yes	Yes

RSSI	No	No	No	No	No
Tag	Yes	No	Yes	Yes	Yes
Status	No	No	No	No	No

Table Index

2.33 rfCallTags

The **rfCallTags** function requests one or more tags to transmit their tag id and status. The reader will acknowledge this command without getting response from the tag. The tag will send its tag id and status asynchronously. The API will call the registered **rfTagEvent** callback function when a tag is detected. The purpose of this command is to wakeup all or particular tag(s) in the field. The application calling this function is responsible for tracking the response form the tag in the **rfTagEvent** callback function.

```
long rfCallTags(
  UInt16 host,                  // address of the host
  UInt16 reader,                // address of the reader
  UInt16 repeater,              // address of the repeater
  UInt16 fieldGen,              // address of the field Generator
  rfTagSelect_t* tagSelect,     // list of tags to be called
  Boolean setTxTimeInterval,    // set time interval between tag transmit time
  Boolean timeInterval,         // time interval between each tag transmission
  UInt16 cmdType,               // reader command type
  UInt16 pktID                  // packet id used in callback
);
```

Parameters

host

[in] Address of the host.

reader

[in] Address of the reader.

repeater

[in] Address of the repeater.

fieldGen

[in] Address of the field generator.

setTxTimeInterval

[in] If true, the tag will use *timeInterval* parameter value to set its transmit time interval between each packet that it transmits.

timeInterval

[in] if true **long** time interval between each tag transmission otherwise **short** time interval.

tagSelect

[in] Pointer to a tag select structure that identifies which tags should be selected. Refer to the definition of the **rfTagSelect_t** structure for more information.

cmdType

[in] Reader Command Type, ALL_READERS or SPECIFIC_READER.

pktID

[in] Packet identifier used to match callbacks to a specific function call. Range from 1 to 223. Each consecutive API function **Must** have different packet ID than previous one.

Return Values

RF_S_PEND if function acknowledged by API;
RF_S_OK_PEND if function acknowledged and responded successfully by each reader or tag when broadcasting, it should be followed by RF_S_DONE when function times out and there are no more responses from the reader, repeater, or tag;
RF_S_DONE if function successful and acknowledged by reader, repeater, or API;
Error code otherwise.

rfTagEvent_t structure in tag callback function contains information about the tag being asked to wakeup and send its ID address and status. The **tag** structure in rfTagEvent_t structure holds information about tag ID and tag type. The table below describes which items in the rfTagEvent_t structure are valid for this function.

Select Type : RF_SELECT_FIELD RF_SELECT_TAGLIST RF_SELECT_FIELDFACTORY				**SelectType: RF_SELECT_TAGID RF_SELECT_TAGFACTORY**	
Items	Valid			Valid	
	RF_S_OK_PEND	RF_S_DONE	RF_E_XXX	RF_S_DONE	RF_E_XXX
Host	Yes	NO	Yes	Yes	Yes
Reader	Yes	No	Yes	Yes	Yes
Repeater	Yes/No	No	Yes/No	Yes/No	Yes/No
eventType	Yes	Yes	Yes	Yes	Yes
cmdType	Yes	Yes	Yes	Yes	Yes
eventStatus	Yes	Yes	Yes	Yes	Yes
errorStatus	No	No	Yes	No	Yes
pktID	Yes	Yes	Yes	Yes	Yes
RSSI	No	No	No	No	No
Tag	Yes	No	Yes	Yes	Yes
Status	No	No	No	No	No

Table Index

2.34 rfConfigureTags

The **rfConfigureTags** function sets one or more tag configuration parameters. The reader will acknowledge this command without getting response from the tag. The API will call the registered **rfTagEvent** callback function when a tag sends its acknowledgement for this command. The purpose of this command is to configure all or particular tag parameter(s) in the field. The application calling this function is responsible for tracking the response form the tag in the **rfTagEvent** callback function.

```
long rfConfigureTags(
  UInt16 host,                  // address of the host
  UInt16 reader,                // address of the reader
  UInt16 repeater,              // address of the repeater
  rfTagSelect_t* tagSelect,     // list of tags
  rfNewTagConfig_t newTagCfg,   // new configuration for tag
  UInt16 configType,            // configuration type
  Boolean setTxTimeInterval,    // set time interval between tag transmit time
  Boolean longInterval,         // time interval between each tag transmission
  UInt16 cmdType,               // reader command type
  UInt16  pktID                 // packet id used in callback
);
```

Parameters

host

[in] Address of the host.

reader

[in] Address of the reader.

repeater

[in] Address of the repeater.

tagSelect

[in] Pointer to a tag select structure that identifies which tags should be selected. Refer to the definition of the **rfTagSelect_t** structure for more information.

newTagCfg

[in] Pointer to a tag new tag structure that contains tag configuration parameters for the tag selected. Refer to the definition of the **rfNewTagConfig_t** structure for more information.

configType

[in] Identifies which tag parameter to configure. All Config values can be combined (ored) together except RF_CFG_TAG_FACTORY_SETTING.

Config Value	Meaning
RF_CFG_TAG_ID	change tag ID.
RF_CFG_TAG_TYPE	change tag type.
RF_CFG_TAG_TIF_GC	change tag time in field and group count.
RF_CFG_TAG_RESEND_TIME	change tag resend time.
RF_CFG_TAG_REPORT_TAMPER	set tag to report tamper.

RF_CFG_TAG_REPORT_TAMPER_HISTORY	set tag to report tamper history.
RF_CFG_TAG_REPORT_NO_TAMPER	set tag to not to report tamper.
RF_CFG_TAG_FACTORY_SETTING	set tag configuration to factory setting.

setTxTimeInterval

[in] If true, the tag will use long interval parameter value to set its transmit time interval between each packet that it transmits.

longInterval

[in] If true **long** time interval between each transmission otherwise **short** time interval.

cmdType

[in] Reader Command Type, ALL_READERS or SPECIFIC_READER.

pktID

[in] Packet identifier used to match callbacks to a specific function call. Range from 1 to 223. Each consecutive API function **Must** have different packet ID than previous one.

Return Values

RF_S_PEND if function acknowledged by API;
RF_S_OK_PEND if function acknowledged and responded successfully by each reader or tag when broadcasting, it should be followed by RF_S_DONE when function times out and there are no more responses from the reader, repeater, or tag;
RF_S_DONE if function successful and acknowledged by reader, repeater, or API;
Error code otherwise.

rfTagEvent_t structure in tag callback function contains information about the tag being asked to send information based on the command. The table below describes which items in the **rfTagEvent_t** structure are valid for this function.

Valid Items in **rfTagEvent_t** for this function	
Host	Tag.type
Reader	
eventType	
cmdType	
eventStatus	
errorStatus	
pktID	
Status	
tag.id	

Table Index

2.35 rfGetTagTempConfig

The **rfGetTagTempConfig** function requests one or more tags to transmit their tag temperature configuration. The reader will acknowledge this command without getting response from the tag. The tag will send its tag temperature configuration asynchronously. The API will call the registered **rfTagEvent** callback function when a tag is detected. The purpose of this command is to obtain configuration all or particular tag(s) in the field. The application calling this function is responsible for tracking the response form the tag in the **rfTagEvent** callback function.

```
long rfGetTagTempConfig(
  UInt16 host,                  // address of the host
  UInt16 reader,                // address of the reader
  UInt16 repeater,              // address of the repeater
  rfTagSelect_t* tagSelect,     // tag list
  Boolean setTxTimeInterval,    // set time interval between tag transmit time
  Boolean timeInterval,         // time interval between each tag transmission
  UInt16 cmdType,               // reader command type
  UInt16 pktID                  // packet id used in callback
);
```

Parameters

host

[in] Address of the host.

reader

[in] Address of the reader.

repeater

[in] Address of the repeater.

tagSelect

[in] Pointer to a tag select structure that identifies which tags should be selected. Refer to the definition of the **`rfTagSelect_t`** structure for more information.

setTxTimeInterval

[in] If true, the tag will use longInterval parameter value to set its transmit time interval between each packet that it transmits.

timeInterval

[in] If true **long** Time interval between each tag transmission otherwise **short** time interval.

cmdType

[in] Reader Command Type, ALL_READERS or SPECIFIC_READER.

pktID

[in] Packet identifier used to match callbacks to a specific function call. Range from 1 to 223. Each consecutive API function **Must** have different packet ID than previous one.

Return Values

RF_S_PEND if function acknowledged by API;
RF_S_OK_PEND if function acknowledged and responded successfully by each reader or tag when broadcasting, it should be followed by RF_S_DONE when function times out and there are no more responses from the reader, repeater, or tag;
RF_S_DONE if function successful and acknowledged by reader, repeater, or API;
Error code otherwise.

rfTagEvent_t structure in tag callback function contains information about the tag being asked to send information based on the command. The table below describes which items in the **rfTagEvent_t** structure are valid for this function.

Valid Items in **rfTagEvent_t** for this function	
Host	tag.type
Reader	tag.temp.rptUnderLowerLimit
eventType	tag.temp.rptOverUpperLimit
cmdType	tag.temp.rptperiodicRead
eventStatus	tag.temp.numReadAve
errorStatus	tag.temp.periodicRptTime
pktID	
Status	
tag.id	

Table Index

2.36 rfSetTagTempConfig

The **rfSetTagTempConfig** function sets one or more tags temperature configuration. The reader will acknowledge this command without getting response from the tag. The API will call the registered **rfTagEvent** callback function when a tag sends its acknowledgement for this command. The purpose of this command is to configure all or particular tag(s) temperature in the field. The application calling this function is responsible for tracking the response form the tag in the **rfTagEvent** callback function.

```
long rfSetTagTempConfig(
  UInt16 host,                  // address of the host
  UInt16 reader,                // address of the reader
  UInt16 repeater,              // address of the repeater
  rfTagSelect_t* tagSelect,     // list of tags
  rfTagTemp_t tagTemp,          // tag temperature structure
  Boolean setTxTimeInterval,    // set time interval between tag transmit time
  Boolean longInterval,         // time interval between each tag transmission
  UInt16 cmdType,               // reader command type
  UInt16  pktID                 // packet id used in callback
);
```

Parameters

host

[in] Address of the host.

reader

[in] Address of the reader.

repeater

[in] Address of the repeater.

tagSelect

[in] [in] Pointer to a tag select structure that identifies which tags should be selected. Refer to the definition of the **rfTagSelect_t** structure for more information. Tag structure contains tag temperature configuration in the rfTagTemp_t structure.

tagTemp

[in] Pointer to a tag temperature structure that contains tag temperature configuration for the tag selected. Refer to the definition of the **rfTagTemp_t** structure for more information.

setTxTimeInterval

[in] If true, the tag will use long interval parameter value to set its transmit time interval between each packet that it transmits.

longInterval

[in] If true **long** time interval between each transmission otherwise **short** time interval.

cmdType

[in] Reader Command Type, ALL_READERS or SPECIFIC_READER.

pktID

[in] Packet identifier used to match callbacks to a specific function call. Range from 1 to 223. Each consecutive API function **Must** have different packet ID than previous one.

Return Values

RF_S_PEND if function acknowledged by API;
RF_S_OK_PEND if function acknowledged and responded successfully by each reader or tag when broadcasting, it should be followed by RF_S_DONE when function times out and there are no more responses from the reader, repeater, or tag;
RF_S_DONE if function successful and acknowledged by reader, repeater, or API;
Error code otherwise.

rfTagEvent_t structure in tag callback function contains information about the tag being asked to send information based on the command. The table below describes which items in the **rfTagEvent_t** structure are valid for this function.

Valid Items in **rfTagEvent_t** for this function	
Host	Tag.type
Reader	
eventType	
cmdType	
eventStatus	
errorStatus	
pktID	
Status	
tag.id	

Table Index

2.37 rfGetTagTemp

The **rfGetTagTemp** function requests one or more tags to transmit their temperature reading. The reader will acknowledge this command without getting response from the tag. The tag will send its tag id, status and temperature asynchronously. The API will call the registered **rfTagEvent** callback function when the tag information is received. The application calling this function is responsible for tracking the response form the tag in the **rfTagEvent** callback function.

```
long rfGetTagTemp(
  UInt16 host,               // address of the host
  UInt16 reader,             // address of the reader
  UInt16 repeater,           // address of the repeater
  rfTagSelect_t* tagSelect,  // tag structure
  Boolean longInterval,      // time interval between each tag transmission
  UInt16 cmdType,            // reader command type
  UInt16 pktID               // packet id used in callback
);
```

Parameters

host
[in] Address of the host.

reader
[in] Address of the reader.

repeater
[in] Address of the repeater.

tagSelect
[in] Tag structure containing information about tag.

setTxTimeInterval
[in] If true, the tag will use long interval parameter value to set its transmit time interval between each packet that it transmits.

longInterval
[in] Time interval between each tag transmission. True for long interval and false for short interval.

cmdType
[in] Reader Command Type, ALL_READERS or SPECIFIC_READER.

pktID
[in] Packet identifier used to match callbacks to a specific function call. Range from 1 to 223. Each consecutive API function **Must** have different packet ID than previous one.

Return Values

RF_S_PEND if function acknowledged by API;
RF_S_OK_PEND if function acknowledged and responded successfully by each reader or

tag when broadcasting, it should be followed by RF_S_DONE when function times out and there are no more responses from the reader, repeater, or tag;
RF_S_DONE if function successful and acknowledged by reader, repeater, or API;
Error code otherwise.

rfTagEvent_t structure in tag callback function contains information about the tag being asked to send information based on the command. The table below describes which items in the **rfTagEvent_t** structure are valid for this function.

rfTagEvent_t structure in tag callback function contains information about the tag being asked to send information based on the command. The table below describes which items in the **rfTagEvent_t** structure are valid for this function.

Valid Items in **rfTagEvent_t** for this function	
Host	tag.type
Reader	tag.temp.temperature
eventType	Tag.temp.status
cmdType	
eventStatus	
errorStatus	
pktID	
Status	
tag.id	

Table Index

2.38 rfGetTagTempCalib

The **rfGetTagTempCalib** function requests API to send tag temp calibration value. The API will send the tag temperature calibration value synchronously. The purpose of this command is to obtain tag calibration value that is used to adjust tag temperature and temperature limits for low and high.

Single rfGetTagTempCalib( );

Return Values

Tag temperature calibration value.

Table Index

2.39 rfSetTagTempCalib

The **rfSetTagTempCalib** function requests API to set tag temp calibration value. The API will set the tag temperature calibration value synchronously. The purpose of this command is to set tag calibration value that is used to adjust tag temperature and temperature limits for low and high.

```
int rfSetTagTempCalib(Single calib);
```

Parameters

calib

[in] Tag temperature calibration value .

.

Return Values

RF_S_DONE if function successful, Error code otherwise.

.

Table Index

2.40 rfGetTagLEDConfig

The **rfGetTagLEDConfig** function requests one or more tags to send their LED setting. The setting represent the number of times the LED will flash on the tag. The API will call the registered **rfTagEvent** callback function when the information is received. These information will be in **rfTag_t** structure in **rfTagEvent_t** structure.

```
long rfGetTagLEDConfig(
  UInt16 host,                // address of the host
  UInt16 reader,              // address of the reader
  UInt16 repeater,            // address of the repeater
  rfTagSelect_t* tagSelect,   // tag structure
  Boolean timeInterval,       // time interval between each tag transmission
  UInt16 cmdType,             // reader command type
  UInt16 pktID                // packet id used in callback
);
```

Parameters

host

[in] Address of the host.

reader

[in] Address of the reader.

repeater

[in] Address of the repeater.

tagSelect

[in] Tag structure containing information about tag.

cmdType

[in] Reader Command Type, ALL_READERS or SPECIFIC_READER.

timeInterval

[in] Time interval between each tag transmission. True for long interval and false for short interval.

pktID

[in] Packet identifier used to match callbacks to a specific function call. Range from 1 to 223. Each consecutive API function **Must** have different packet ID than previous one.

Return Values

RF_S_PEND if function acknowledged by API;
RF_S_OK_PEND if function acknowledged and responded successfully by each reader or tag when broadcasting, it should be followed by RF_S_DONE when function times out and there are no more responses from the reader, repeater, or tag;
RF_S_DONE if function successful and acknowledged by reader, repeater, or API;

Error code otherwise.

rfTagEvent_t structure in tag callback function contains information about the tag being asked to send information based on the command. **rfTag_t** structure in rfTagEvent_t holds the setting for LED, data parameter in rfTag_t with index 0 has the value. data[0] = LED setting.

Table Index

2.41 rfGetTagSpeakerConfig

The **rfGetTagSpeakerConfig** function requests one or more tags to send their speaker setting. The setting represent the number of times the speaker will beep on the tag. The API will call the registered **rfTagEvent** callback function when the information is received. These information will be in **rfTag_t** structure in **rfTagEvent_t** structure.

```
long rfGetTagSpeakerConfig(
  UInt16 host,                // address of the host
  UInt16 reader,              // address of the reader
  UInt16 repeater,            // address of the repeater
  rfTagSelect_t* tagSelect,   // tag structure
  Boolean timeInterval,       // time interval between each tag transmission
  UInt16 cmdType,             // reader command type
  UInt16 pktID                // packet id used in callback
);
```

Parameters

host

[in] Address of the host.

reader

[in] Address of the reader.

repeater

[in] Address of the repeater.

tagSelect

[in] Tag structure containing information about tag.

cmdType

[in] Reader Command Type, ALL_READERS or SPECIFIC_READER.

timeInterval

[in] Time interval between each tag transmission. True for long interval and false for short interval.

pktID

[in] Packet identifier used to match callbacks to a specific function call. Range from 1 to 223. Each API function **Must** have different packet ID than previous one.

Return Values

RF_S_PEND if function acknowledged by API;
RF_S_OK_PEND if function acknowledged and responded successfully by each reader or tag when broadcasting, it should be followed by RF_S_DONE when function times out and there are no more responses from the reader, repeater, or tag;
RF_S_DONE if function successful and acknowledged by reader, repeater, or API;

Error code otherwise.

rfTagEvent_t structure in tag callback function contains information about the tag being asked to send information based on the command. **rfTag_t** structure in rfTagEvent_t holds the setting for speaker, data parameter in rfTag_t with index 0 has the value. data[0] = speaker setting.

Table Index

2.42 rfSetTagLEDConfig

The **rfSetTagLEDConfig** function configures tag LED settings. The setting represent the number of times the LED will flash on the tag upon request from the application. The calling application is responsible for tracking the response form the tag in the **rfTagEvent** callback function.

```
long rfSetTagLEDConfig(
  UInt16 host,                  // address of the host
  UInt16 reader,                // address of the reader
  UInt16 repeater,              // address of the repeater
  rfTagSelect_t* tagSelect,     // tag structure
  UInt16 led,                   // number of times tag LED to flash
  Boolean timeInterval,         // time interval between each tag transmission
  UInt16 cmdType,               // reader command type
  UInt16 pktID                  // packet id used in callback
);
```

Parameters

host
[in] Address of the host.

reader
[in] Address of the reader.

repeater
[in] Address of the repeater.

tagSelect
[in] Tag structure containing information about tag.

led
[in] Number of times tag LED to flash. (0 - 63)

cmdType
[in] Reader Command Type, ALL_READERS or SPECIFIC_READER.

timeInterval
[in] Time interval between each tag transmission. True for long interval and false for short interval.

pktID
[in] Packet identifier used to match callbacks to a specific function call. Range from 1 to 223. Each consecutive API function **Must** have different packet ID than previous one.

Return Values

RF_S_PEND if function acknowledged by API;
RF_S_OK_PEND if function acknowledged and responded successfully by each reader or tag when broadcasting, it should be followed by RF_S_DONE when function times out and there are no more responses from the reader, repeater, or tag;

RF_S_DONE if function successful and acknowledged by reader, repeater, or API;
Error code otherwise.

Table Index

2.43 rfSetTagSpeakerConfig

The **rfSetTagSpeakerConfig** function configures tag speaker settings. The setting represent the number of times the speaker will beep on the tag upon request from the application. The calling application is responsible for tracking the response form the tag in the **rfTagEvent** callback function.

```
long rfSetTagLEDConfig(
  UInt16 host,                // address of the host
  UInt16 reader,              // address of the reader
  UInt16 repeater,            // address of the repeater
  rfTagSelect_t* tagSelect,   // tag structure
  UInt16 speaker,             // number of times tag speaker to beep
  Boolean timeInterval,       // time interval between each tag transmission
  UInt16 cmdType,             // reader command type
  UInt16 pktID                // packet id used in callback
);
```

Parameters

host
[in] Address of the host.

reader
[in] Address of the reader.

repeater
[in] Address of the repeater.

tagSelect
[in] Tag structure containing information about tag.

speaker
[in] Number of times tag speaker to beep. (0 - 63)

cmdType
[in] Reader Command Type, ALL_READERS or SPECIFIC_READER.

timeInterval
[in] Time interval between each tag transmission. True for long interval and false for short interval.

pktID
[in] Packet identifier used to match callbacks to a specific function call. Range from 1 to 223. Each consecutive API function **Must** have different packet ID than previous one.

Return Values

RF_S_PEND if function acknowledged by API;
RF_S_OK_PEND if function acknowledged and responded successfully by each reader or tag when broadcasting, it should be followed by RF_S_DONE when function times out and there are no more responses from the reader, repeater, or tag;

RF_S_DONE if function successful and acknowledged by reader, repeater, or API;
Error code otherwise.

Table Index

2.44 rfRegisterTagEvent

The **rfRegisterTagEvent** function is used to register an application implemented callback function that will be called by the API when a tag event occurs. This function executes synchronously.

```
void rfRegisterTagEvent( rfTagEvent fnCallback,     // callback function
                         void* userArg              // user-defined value
);
```

Parameters

fnCallback

[in] Callback function to be called for any tag event. This value can be NULL to disable callbacks. The callback function prototype is:

```
typedef long (*rfTagEvent)(Int32 status,
                           HANDLE funcId,
                           rfTagEvent_t tagEvent,
                           void* userArg);
```

Note that the **tagEvent** argument is only valid within the scope of the function.

userArg

[in] User defined value passed to callback function.

Table Index

3. Unsolicited Event Messages

3.1 RF_TAG_DETECTED

The **RF_TAG_DETECTED** event is sent from reader to host, indicating that a tag has been detected in the field. API will call application tag callback function to provide the information about the tag.

rfTagEvent_t structure in tag callback function contains information about the tag being detected. The **tag** structure in rfTagEvent_t structure holds information about tag ID, and field generator address. **Status** structure in the **tag** structure holds information about status of the tag. The table below describes which items in the rfTagEvent_t structure are valid for this function.

Select Type : RF_SELECT_FIELD RF_SELECT_TAGLIST RF_SELECT_FIELDFACTORY				**SelectType: RF_SELECT_TAGID RF_SELECT_TAGFACTORY**	
Items	Valid			Valid	
	RF_S_OK_PEND	RF_S_DONE	RF_E_XXX	RF_S_DONE	RF_E_XXX
host	Yes	NO	Yes	Yes	Yes
reader	Yes	No	Yes	Yes	Yes
repeater	Yes/No	No	Yes/No	Yes/No	Yes/No
fGenerator	Yes/No	No	No	Yes/No	No
eventType	Yes	Yes	Yes	Yes	Yes
cmdType	Yes	Yes	Yes	Yes	Yes
eventStatus	Yes	Yes	Yes	Yes	Yes
errorStatus	No	No	Yes	No	Yes
pktID	Yes	Yes	Yes	Yes	Yes
RSSI	No	No	No	No	No
tag	Yes	No	Yes	Yes	Yes
status	No	No	No	No	No

Table Index

Command Table

3.2 RF_TAG_DETECTED_RSSI

The **RF_TAG_DETECTED_RSSI** event is sent from reader to host, indicating that a tag has been detected in the field. API will call application tag callback function to provide the information about the tag.

rfTagEvent_t structure in tag callback function contains information about the tag being detected. Field strength(RSSI) is stored in the rfTagEvent_t. The **tag** structure in rfTagEvent_t structure holds information about tag ID, and field generator address. **Status** structure in the **tag** structure holds information about status of the tag. The table below describes which items in the rfTagEvent_t structure are valid for this function.

Select Type : RF_SELECT_FIELD RF_SELECT_TAGLIST RF_SELECT_FIELDFACTORY				**SelectType: RF_SELECT_TAGID RF_SELECT_TAGFACTORY**	
Items	Valid			Valid	
	RF_S_OK_PEND	RF_S_DONE	RF_E_XXX	RF_S_DONE	RF_E_XXX
host	Yes	NO	Yes	Yes	Yes
reader	Yes	No	Yes	Yes	Yes
repeater	Yes/No	No	Yes/No	Yes/No	Yes/No
fGenerator	Yes/No	No	No	Yes/No	No
eventType	Yes	Yes	Yes	Yes	Yes
cmdType	Yes	Yes	Yes	Yes	Yes
eventStatus	Yes	Yes	Yes	Yes	Yes
errorStatus	No	No	Yes	No	Yes
pktID	Yes	Yes	Yes	Yes	Yes
RSSI	Yes	No	No	Yes	No
tag	Yes	No	Yes	Yes	Yes
status	No	No	No	No	No

Table Index

Command Table

3.3 RF_TAG_DETECTED_SANI

The **RF_TAG_DETECTED_SANI** event is sent from reader to host, indicating that a SaniFaucet tag has been detected in the field. API will call application tag callback function to provide the information about the tag.

rfTagEvent_t structure in tag callback function contains information about the tag being detected. Field strength(RSSI) is stored in the rfTagEvent_t.

The **tag** structure in rfTagEvent_t structure holds information about tag ID, and field generator address.

Status structure in the **tag** structure holds information about status of the tag. The table below describes which items in the rfTagEvent_t structure are valid for this function.

	SelectType: RF_SELECT_TAGID RF_SELECT_TAGFACTORY
Items	Valid
	RF_S_DONE
host	Yes
reader	Yes
repeater	Yes/No
fGenerator	Yes/No
eventType	Yes
cmdType	Yes
eventStatus	Yes
errorStatus	No
pktID	Yes
RSSI	Yes
tag	Yes
status	No

Table Index

Command Table

3.4 RF_INVALID_PACKET

The **RF_INVALID_PACKET** event is sent from reader to host, indicating that the packet received from the host is invalid. API will call application reader or tag callback function to provide the information about the error.

rfReaderEvent_t structure in reader callback function contains information about the error condition if error was as result of reader function call or rfTagEvent_t structure if error condition was caused by a tag function. In both structure **errorStatus** has information about the type of error that has occurred.

Select Type : RF_SELECT_FIELD RF_SELECT_TAGLIST RF_SELECT_FIELDFACTORY				**SelectType: RF_SELECT_TAGID RF_SELECT_TAGFACTORY**	
Items	Valid			Valid	
	RF_S_OK_PEND	RF_S_DONE	RF_E_XXX	RF_S_DONE	RF_E_XXX
host	Yes	NO	Yes	Yes	Yes
reader	Yes	No	Yes	Yes	Yes
repeater	Yes	No	Yes	Yes	Yes
fGenerator	Yes	No	No	Yes	No
eventType	Yes	Yes	Yes	Yes	Yes
cmdType	Yes	Yes	Yes	Yes	Yes
eventStatus	Yes	Yes	Yes	Yes	Yes
errorStatus	Yes	No	No	Yes	No
pktID	Yes	Yes	Yes	Yes	Yes
RSSI	No	No	No	No	No
tag	Yes	No	Yes	Yes	Yes
status	No	No	No	No	No

CmdType : ALL_READERS ALL_REPEATERS				**CmdType: SPECIFIC_READER SPECIFIC_REPEATER**	
Items	Valid			Valid	
	RF_S_OK_PEND	RF_S_DONE	RF_E_XXX	RF_S_DONE	RF_E_XXX
host	Yes	NO	Yes	Yes	Yes
reader	Yes	No	Yes	Yes	Yes
repeater	Yes	No	Yes	Yes	Yes
relay	No	No	No	No	No
fGenerator	No	No	No	No	No
eventType	Yes	Yes	Yes	Yes	Yes
cmdType	Yes	Yes	Yes	Yes	Yes
eventStatus	Yes	Yes	Yes	Yes	Yes
errorStatus	Yes	No	No	Yes	No
pktID	Yes	Yes	Yes	Yes	Yes
cmdRef	No	No	No	No	No
data	No	No	No	No	No
versionInfo	No	No	No	No	No

4. Structures

Structure	Description
rfVersionInfo_t	Contains info about reader's data, prog and host code version
rfReaderEvent_t	Callback function parameter for reader, Contains info about reader, smart fgen, std fgen and status of the reader, This structure gets populated by the API based on the command from the host.
rfTagEvent_t	Callback function parameter for tag. Contains info about tag This structure gets populated by the API based on the command from the host.
rfTagSelect_t	This structure is used with tag commands. It holds information about the tag that is been asked to perform an operation.
rfTag_t	This structure is used to return information about the tag for most of tag commands. It gets populated in the API.
rfNewTagConfig_t	This structure is used by application to configure a new tag,
rfTagStatus_t	This structure is used by API to send the status of a tag to the application.
rfTagTemp_t	This structure is used to configure tag temperature settings and also is used by API to send information about tag temperature configuration and values to the application.
rfSmartFGen_t	This structure is used to configure field generator and also return information about smart and STD field generators. It is used with field generator commands.

Table Index

4.1 rfVersionInfo_t

```
typedef struct {
  Byte dataCodeVer;
  Byte ProgCodeVer;
  Byte hostCodeVer;
} rfVersionInfo_t;
```

Members

dataCodeVer
ProgCodeVer
hostCodeVer

Table Index

4.2 rfReaderEvent_t

```
typedef struct {
  UInt16 host;                      // address of host
  UInt16 reader;                    // address of reader
  UInt16 repeater;                  // address of repeater
  UInt16 relay;                     // address of relay
  UInt16 fGenerator;                // address of field generator
  UInt16 eventType;                 // type of event
  UInt16 cmdType;                   // reader command type
  UInt16 eventStatus;               // status of event
  UInt16 errorStatus;               // error status
  UInt16 pktID;                     // packet id used in callback
  Byte   data[MAX_DATA]       // Extra space for use by user & API   MAX_DATA=256
  Byte   ip[IP_MAX_DATA];     // address of IP(Internet Protocol) IP_MAX_DATA=20
  UInt32 port;                      // network port number of reader
  UInt16 cmdRef;                    // API internal use
  rfVersionInfo_t versionInfo;      // Reader version info structure
  rfSmartFGen_t smartFgen;          // smart and std field generator structure
} rfReaderEvent_t;
```

Members

host
Address of the host.

reader
Address of the reader.

repeater
Address of the repeater.

relay
Address of the relay 1 or 2.

fGenerator
Address of the standard field generator.

eventType
Indicates the event type as defined in the table below.

Value	Meaning
RF_READER_POWERUP	Reader completed power-up sequence. The application can change reader address by changing the value of **reader** before returning from the callback. If the value of **reader** is zero, the application must change it to a unique non-zero value before any other functions can be called for the reader.
RF_READER_QUERY RF_READER_QUERY_ALL	Response to rfQueryReader function. CmdType = SPECIFIC_READER “ CmdType = ALL_READER
RF_READER_ENABLE RF_READER_ENABLE_ALL	Response to rfEnableReader function. CmdType = SPECIFIC_READER “ CmdType = ALL_READER
RF_READER_DISABLE RF_READER_DISABLE_ALL	Response to rfEnableReader function. CmdType = SPECIFIC_READER “ CmdType = ALL_READER

RF_READER_RESET RF_READER_RESET_ALL	Response to rfResetReader function. CmdType = SPECIFIC_READER “ CmdType = ALL_READER
RF_READER_GET_VERSION RF_READER_GET_VERSION_ALL	Response to rfGetReaderVersion function. CmdType = SPECIFIC_READER “ CmdType = ALL_READER
RF_RELAY_ENABLE RF_RELAY_ENABLE_ALL	Response to rfEnableRelay function. CmdType = SPECIFIC_READER “ CmdType = ALL_READER
RF_RELAY_DISABLE RF_RELAY_DISABLE_ALL	Response to rfEnableRelay function. CmdType = SPECIFIC_READER “ CmdType = ALL_READER
RF_CONFIG_INPUT_PORT RF_CONFIG_INPUT_PORT_ALL	Response to rfConfigInputPort function. CmdType = SPECIFIC_READER “ CmdType = ALL_READER
RF_GET_INPUT_STATUS RF_GET_INPUT_STATUS_ALL	Response to rfGetInputPortStatus function. CmdType = SPECIFIC_READER “ CmdType = ALL_READER
RF_GET_READER_VERSION RF_GET_READER_VERSION_ALL	Response to rfGetReaderVersion function. CmdType = SPECIFIC_READER “ CmdType = ALL_READER
RF_SET_RDR_FS RF_SET_RDR_FS_ALL	Response to rfSetReaderFS function. CmdType = SPECIFIC_READER “ CmdType = ALL_READER
RF_GET_RDR_FS RF_GET_RDR_FS_ALL	Response to rfGetReaderFS function. CmdType = SPECIFIC_READER “ CmdType = ALL_READER
RF_RESET_SMART_FGEN	Response to rfResetSmartFGen function.
RF_QUERY_SMART_FGEN	Response to rfQuerySmartFGen function.
RF_SET_CONFIG_SMART_FGEN	Response to rfSetConfigSmartFGen function.
RF_SET_FS_SMART_FGEN	Response to rfSetSmartFGenFS function.
RF_GET_FS_SMART_FGEN	Response to rfGetSmartFGenFS function.
RF_READER_CONFIG	Response to rfConfigureReader function.
RF_SCAN_NETWORK	Response to rfScanNetwork function.
RF_SCAN_IP	Response to rfScanIP function.
RF_OPEN_SOCKET	Response to rfOpenSocket function.
RF_CLOSE_SOCKET	Response to rfCloseSocket function.

cmdType

Reader command type: ALL_READERS(broadcasting to all readers) or SPECIFIC_READER.

eventStatus

The status of the command that was sent to the API.

ID	Value	Meaning
RF_S_DONE	0	The command is completed.
RF_S_PEND	2	The command is pending upon reader or tag for response.
RF_S_OK_PEND	3	The command is a broadcast type and waiting for all readers to respond.
RF_S_ERROR	4	There was an error when the command executed.
RF_S_TIMEOUT	5	There was no response to this command from the reader within the specific time period.

errorStatus

The error reporting back to the calling application when the command executed. If there is no error RF_E_NO_ERROR will be returned.

ID	Value	Description
RF_E_NO_ERROR	0	
RF_E_UNSPECIFIED	-1	
RF_E_ERROR	-2	
RF_E_PACKET	-100	

RF_E_NOT_IMPLEMENTED	-101	
RF_E_NOT_SUPPORTED	-102	
RF_E_COMMAND	-103	
RF_E_ARGUMENT	-104	
RF_E_CRC	-105	
RF_E_TIMEOUT	-106	
RF_E_CREATE_FILE	-107	
RF_E_CREATE_EVENT	-108	
RF_E_SET_UP_COMM	-109	
RF_E_SET_COM_MASK	-110	
RF_E_CREATE_THREAD	-111	
RF_E_TERMINATE_THREAD	-112	
RF_E_SET_PIORITY	-113	
RF_E_SET_COM_STATE	-114	
RF_E_SET_COM_TIMEOUT	-115	
RF_E_TX	-116	
RF_E_ALLOCATE_MEM	-117	
RF_E_QUE_FULL	-118	
RF_E_RX_QUE_FULL	-119	
RF_E_PENDING_PKTID	-120	
RF_E_NO_RESPONSE	-121	
RF_E_CHECKSUM	-122	
RF_E_READ_MEMORY	-123	
RF_E_WRITE_MEMORY	-124	
RF_E_RESPOND_ERROR_STATUS	-125	
RF_E_BUILD_PACKET	-126	
RF_E_MAX_READ_MEM_LEN	-127	
RF_E_MAX_WRITE_MEM_LEN	-128	
RF_E_USER_RAM_MEM_ADDR	-129	
RF_E_USER_MEM_BOUNDRY	-130	
RF_E_READER_BUSY	-131	
RF_E_PARAM_OUT_OF_RANGE	-132	
RF_E_PKT_ID	-133	
RF_E_CLOSE_PORT	-135	
RF_E_GET_COM_STATE	-139	
RF_E_PORT_ALREADY_OPEN	-140	
RF_E_GET_COM_TIMEOUT	-149	
RF_E_WINSOCKET_VER_NOT_SUPPORTED	-150	
RF_E_CREATE_SOCKET	-151	
RF_E_INVALID_IP_ADDRESS	-152	
RF_E_INVALID_HOST_NAME	-153	
RF_E_BINDING_SOCKET	-154	
RF_E_CLOSING_SOCKET	-155	
RF_E_CONNECT	-156	
RF_E_GET_INTERFACE_LIST	-157	
RF_E_WSASTARTUP	-158	
RF_E_OPEN_SOCKET	-159	
RF_E_IP_NOT_FOUND	-160	
RF_E_LISTEN	-161	
RF_E_LOADING_LIB	-162	
RF_E_WRITE_IV	-163	
RF_E_SOCKET_EXCEPTION	-164	
RF_E_READING_SOCKET	-165	
RF_E_BLOCKENCRYPT_NOT_FOUND	-166	
RF_E_SOCKET_NOT_ACTIVE	-167	

RF_E_READER_NOT_POWERED_UP	-168	
RF_E_SOCKET_WRITE	-169	
RF_E_SOCKET_READ	-170	
RF_E_CHANGE_IP	-171	
RF_E_INVALID_OEMID	-172	
RF_E_IP_NOT_SCANNED	-173	
RF_E_TEMP_LIMIT	-174	
RF_E_SOCKET_ALREADY_CONNECTED	-175	
RF_E_NO_OPEN_SOCKET	-176	
RF_E_SOCKET_CONNECTION_PENDING	-177	
RF_E_MAX_SOCKET_OPEN_SLOT	-178	
RF_E_MAX_FS	-182	

pktID

Packet identifier to match the function pktID with callbacks pktID. Range from 1 to 223

data

The data which is passed to calling function. For example with rfReadTags() the data which is read from the tag will be stored in this parameter and pass to the application..

ip

The ip address of the reader on network.

port

The port number for the reader on the network.

cmdRef

Reserved for internal use by the API. The application must not change its value.

versionInfo

The structure that holds information about reader version. This structure will be populated when rfGetReaderVersion() is called.

smartFgen

The structure that holds information about standard field generator and smart field generator. This structure will be populated when smart Fgen and Std Fgen API functions are called.

Table Index

4.3 rfTagEvent_t

```
typedef struct {
  UInt16 host;          // address of host
  UInt16 reader;        // address of reader
  UInt16 repeater;      // address of repeater
  UInt16 fGenerator;    // address of field generator
  UInt16 eventType;     // type of event
  UInt16 eventStatus;   // status of event
  UInt16 errorStatus;   // status of error
  UInt16 RSSI;          // field strength
  UInt16 pktID;         // id of packet sent by application
  UInt16 cmdType;       // type of the command requested by application
  rfTag_t tag;          // tag that caused event
} rfTagEvent_t;
```

Members

host
Address of the host.

reader
Address of the reader.

repeater
Address of the repeater.

fGenerator
Address of the standard field generator.

eventType
Indicates the event type as defined in the table below.

Value	Meaning
RF_TAG_DETECTED	The reader detected an unsolicited tag id.
RF_TAG_CONFIGURE	Response to rfConfigureTags function.
RF_TAG_ENABLE	Response to rfEnableTags function.
RF_TAG_QUERY	Response to rfQueryTags function.
RF_TAG_READ	Response to rfReadTags function.
RF_TAG_WRITE	Response to rfWriteTags function.
RF_TAG_GET_TEMP_CONFIG	Response to rfGetTagTempConfig function.
RF_TAG_SET_TEMP_CONFIG	Response to rfSetTagTempConfig function.
RF_TAG_GET_TEMP	Response to rfGetTagTemp function.
RF_GET_TAG_LED_CONFIG	Response to the rfGetTagLEDConfig function.
RF_SET_TAG_LED_CONFIG	Response to the rfSetTagLEDConfig function.
RF_GET_TAG_SPEAKER_CONFIG	Response to the rfGetTagSpeakerConfig function.
RF_SET_TAG_SPEAKER_CONFIG	Response to the rfSetTagSpeakerConfig function.

eventStatus

The status of the command that was sent to the API.

ID	Value	Meaning
RF_S_DONE	0	The command is completed.
RF_S_PEND	2	The command is pending upon reader or tag for response.
RF_S_OK_PEND	3	The command is a broadcast type and waiting for all readers to respond.
RF_S_ERROR	4	There was an error when the command executed.
RF_S_TIMEOUT	5	There was no response to this command from the reader within the specific time period.

errorStatus

The error reporting back to the calling application when the command executed. If there is no error RF_E_NO_ERROR will be returned.

ID	Value	Description
RF_E_NO_ERROR	0	
RF_E_UNSPECIFIED	-1	
RF_E_ERROR	-2	
RF_E_PACKET	-100	
RF_E_NOT_IMPLEMENTED	-101	
RF_E_NOT_SUPPORTED	-102	
RF_E_COMMAND	-103	
RF_E_ARGUMENT	-104	
RF_E_CRC	-105	
RF_E_TIMEOUT	-106	
RF_E_CREATE_FILE	-107	
RF_E_CREATE_EVENT	-108	
RF_E_SET_UP_COMM	-109	
RF_E_SET_COM_MASK	-110	
RF_E_CREATE_THREAD	-111	
RF_E_TERMINATE_THREAD	-112	
RF_E_SET_PIORITY	-113	
RF_E_SET_COM_STATE	-114	
RF_E_SET_COM_TIMEOUT	-115	
RF_E_TX	-116	
RF_E_ALLOCATE_MEM	-117	
RF_E_QUE_FULL	-118	
RF_E_RX_QUE_FULL	-119	
RF_E_PENDING_PKTID	-120	
RF_E_NO_RESPONSE	-121	
RF_E_CHECKSUM	-122	
RF_E_READ_MEMORY	-123	
RF_E_WRITE_MEMORY	-124	
RF_E_RESPOND_ERROR_STATUS	-125	
RF_E_BUILD_PACKET	-126	
RF_E_MAX_READ_MEM_LEN	-127	
RF_E_MAX_WRITE_MEM_LEN	-128	
RF_E_USER_RAM_MEM_ADDR	-129	
RF_E_USER_MEM_BOUNDRY	-130	
RF_E_READER_BUSY	-131	
RF_E_PARAM_OUT_OF_RANGE	-132	
RF_E_PKT_ID	-133	

RF_E_CLOSE_PORT	-135	
RF_E_GET_COM_STATE	-139	
RF_E_PORT_ALREADY_OPEN	-140	
RF_E_GET_COM_TIMEOUT	-149	
RF_E_WINSOCKET_VER_NOT_SUPPORTED	-150	
RF_E_CREATE_SOCKET	-151	
RF_E_INVALID_IP_ADDRESS	-152	
RF_E_INVALID_HOST_NAME	-153	
RF_E_BINDING_SOCKET	-154	
RF_E_CLOSING_SOCKET	-155	
RF_E_CONNECT	-156	
RF_E_GET_INTERFACE_LIST	-157	
RF_E_WSASTARTUP	-158	
RF_E_OPEN_SOCKET	-159	
RF_E_IP_NOT_FOUND	-160	
RF_E_LISTEN	-161	
RF_E_LOADING_LIB	-162	
RF_E_WRITE_IV	-163	
RF_E_SOCKET_EXCEPTION	-164	
RF_E_READING_SOCKET	-165	
RF_E_BLOCKENCRYPT_NOT_FOUND	-166	
RF_E_SOCKET_NOT_ACTIVE	-167	
RF_E_READER_NOT_POWERED_UP	-168	
RF_E_SOCKET_WRITE	-169	
RF_E_SOCKET_READ	-170	
RF_E_CHANGE_IP	-171	
RF_E_INVALID_OEMID	-172	
RF_E_IP_NOT_SCANNED	-173	
RF_E_TEMP_LIMIT	-174	
RF_E_SOCKET_ALREADY_CONNECTED	-175	
RF_E_NO_OPEN_SOCKET	-176	
RF_E_SOCKET_CONNECTION_PENDING	-177	
RF_E_MAX_SOCKET_OPEN_SLOT	-178	
RF_E_MAX_FS	-182	

host
Address of the host.

pktID
Packet identifier to match the function pktID with callbacks pktID. Range from 1 to 223

cmdType
Reader command type: ALL_READERS(broadcasting to all readers) or SPECIFIC_READER

tag
The structure that holds information about reader version. This structure will be populated when rfGetReaderVersion() is called.

Table Index

4.4 rfTagSelect_t

```
typedef struct {
  UInt32 selectType;
  Byte tagType;
  UInt32 tagList[MAX_TAG_SELECT];    //MAX_TAG_SELECT = 50
  UInt32 numTags;
  Boolean ledOn;
  Boolean speakerOn;
  Byte rangeIndex;
} rfTagSelect_t;
```

Members

selectType

Specifies the tag selection criteria to apply. Must be one of the following:

ID	Meaning
RF_SELECT_FIELD	Selects all tags in the field.
RF_SELECT_TAG_TYPE	Select only tags in the field whose tag type matches the value of tagType.
RF_SELECT_TAG_ID	Select numTags tags from the tagList staring with index zero whose tags id match the value of the tagList.
RF_SELECT_TAG_RANGE	Select tags that their ID's are within the range.

tagType

Specifies which tag types are to be selected.

ID	Meaning
ALL_TAGS	Selects all tag types in the field.
ACCESS_TAG	
INVENTORY_TAG	
ASSET_TAG	
FACTORY_TAG	

TagList

An array of tag id's to be selected (max size 50). Used in conjunction with **numTags** to select the number of tags. Used when **selectType** is RF_SELECT_TAGID.

NumTags

Specifies number of tags to be selected from the **tagList** starting from index zero.

ledON

If true it will turn on the LED on the tag.

speakerON

If true it will turn on the speaker on the tag.

rangeIndex

The range of tag ids to be selected, starting from the first tag ID in the TagList to the first tag ID + rangeIndex. The valid rangeIndex are 2, 3, 4, 5, 6, 7, 8, 10, 15, 20, 40, 60, 80, 100, 150 And 200.

Table Index

4.5 rfTagStatus_t

```
typedef struct {

  Boolean batteryLow;
  Boolean tamperSwitch;
  Boolean continuousField;
  Boolean bi-directional;
  Boolean batteryRechargeable;
  Boolean enabled;
  Byte type;
} rfTagStatus_t;
```

Members

batteryLow
Indicates the tag battery is low if value is true.

tamperSwitch
Indicates the tag's tamper switch status. A true value means the tamper switch has been activated.

continuousField
Indicates the tag's RF field detection status. True value means the tag is detecting an RF field.

bi-directional
Indicates the tag's capability for bi-directional communications with the reader. A true value means the tag supports bi-directional communications.

Enabled
Indicates the tag's enable status. A true value means the tag is enabled. An enabled tag will periodically transmit its tag id and status without needing an explicit command from the reader. A disabled tag, will only responds to reader initiated commands.

Type
Indicates the tag type as defined in the table below.

ID	Value	Meaning
ALL_TAGS	0x00	All tags
ACCESS_TAG	0x01	Access tag
INVENTORY_TAG	0x02	Inventory tag
ASSET_TAG	0x03	Asset tag

Table Index

4.6 rfTagTemp_t

```
typedef struct {
  Boolean rptUnderLowerLimit,     // report under lower limit
  Boolean rptOverUpperLimit,      // report over upper limit
  Boolean rptPeriodicRead,        // report with each periodic read
  Boolean enableTempLogging;      // enable temperature logging
  Boolean logging;                // reports if tag is logging temperature
  UInt16  numReadAve,             // number of reads per average
  UInt16  wrapAround,             // temp logging will warp-around if memory full
  UInt16  periodicRptTime,        // periodic report time
  UInt16  periodicTimeType,       // periodic report time type
  Byte    status,                 // normal, low or high
  Single  lowerLimitTemp,         // tag temperature lower limit
  Single  upperLimitTemp,         // tag temperature upper limit
  Single  temperature             // tag temperature value
} rfTagTemp_t;
```

Members

rptUnderLowerLimit
Report tag temperature when its temperature falls below the Lower Limit. True report false otherwise.

rptOverUpperLimit
Report tag temperature when its temperature goes over the Upper Limit. True report false otherwise.

rptPeriodicRead
Report tag temperature when **periodicRptTime** expires. True report false otherwise.

enableTempLogging
Enable logging temperature into tag memory if True false otherwise.

logging
Report tag logging status. True if logging false otherwise.

numReadAve
Number of temperature reads tag performs before it takes average of them.

wrapAround
Wraps around if tag memory is full if set to True otherwise false.

periodicRptTime
Tag temperature periodic read. Wait this unit of time before attempting to report the tag temperature. If **rptPeriodicRead** is false set this field to zero.

periodicTimeType
Type of time for **periodicRptTime** parameter**.**

Value	Meaning

RF_TIME_HOUR	Unit of time for **periodicTimeType** parameter (Hour)
RF_TIME_MINUTE	Unit of time for **periodicTimeType** parameter (Minute)

.

tempStatus

Status of temperature being reported. It can be one of the following:

Value	Meaning
RF_TAG_TEMP_NORM	Normal temperature based on **upperLimitTemp** and **lowerLimitTemp** value.
RF_TAG_TEMP_HIGH	High temperature based on **upperLimitTemp** value.
RF_TAG_TEMP_LOW	Low temperature based on **lowerLimitTemp** value.

lowerLimitTemp

Tag lower limit temperature. Tag will report if its temperature falls below *`lowerLimitTemp and rptUnderLowerLimit`* is true.

upperLimitTemp

Tag upper limit temperature. Tag will report if its temperature goes over *`upperLimitTemp and rptOverUpperLimit`* is true.

temperature

Tag current temperature.

Table Index

4.7 rfTag_t

```
typedef struct {
  UInt32  id;
  Byte  tagType;
  Byte  version;
  rfTagStatus_t  status;
  rfTagTemp_t  temp;
  Byte  timeInField;
  Byte  groupCount;
  Byte  resendTime;
  UInt16  resendTimeType;
  Byte data[MAX_DATA];    //MAX_DATA = 255
  UInt16 dataLen;
  UInt16 assignedReader;
  UInt16 selectType;
  rfTagSani_t sani;
} rfTag_t;
```

Members

id

Unique tag identifier.

tagType

Indicates the tag type as defined in the table below.

ID	Meaning
ALL_TAGS	All tags
ACCESS_TAG	Access tag
INVENTORY_TAG	Inventory tag
ASSET_TAG	Asset tag

Version

Identifies the version of the tag.

Status

Contains various tag status indicators. Refer to the **rfTagStatus_t** structure for details.

temp

Contains tag temperature configuration. Refer to the **rfTagTemp_t** structure for details.

TimeInField

The time period that the tag must be in the field before sending its date.

groupCount

The number of times that the tag sends its data for each command or when it is detected.

resendTime

The time period between each packet that is sent by the tag.

resendTimeType
The type of the resendTime: seconds, minutes or hours..

tamperSwitch
Set Tamper Switch when true, clear Tamper Switch when false.

data
Array of 255 bytes – used for user data

dataLen
Length of the data..

assignedReader
Reader ID that has been assigned to the tag.

selectType
Specifies the tag selection criteria to apply. Must be one of the following:

ID	Meaning
RF_SELECT_FIELD	Selects all tags in the field.
RF_SELECT_TAG_TYPE	Select only tags in the field whose tag type matches the value of tagType.
RF_SELECT_TAG_ID	Select numTags tags from the tagList staring with index zero whose tags id match the value of the tagList.
RF_SELECT_TAG_RANGE	Select tags that their ID's are within the range.

sani
Sani Faucet tag information associated with tag.

Table Index

4.8 rfNewTagConfig_t

```
typedef struct {
  UInt32  newTagID;
  Byte  newTagType;
  Byte  configByte;
  Byte  timeInField;
  Byte  groupCount;
  Byte  resendTime;
  UInt16  resendTimeType;
  Boolean reportTamper;
  Boolean reportTamperHistory;
  Boolean noTamperReport;
  Boolean noChangeTamper;
  Boolean factorySetting;
  UInt16 assignedReader;
} rfNewTagConfig_t;
```

Members

id

Unique tag identifier.

newTagType

Indicates the tag type as defined in the table below.

ID	Meaning
ALL_TAGS	All tags
ACCESS_TAG	Access tag
INVENTORY_TAG	Inventory tag
ASSET_TAG	Asset tag

configByte

Identifies which tag parameter to configure. All Config values can be combined (ored) together except RF_CFG_TAG_FACTORY_SETTING.

Config Value	Meaning
RF_CFG_TAG_ID	change tag ID.
RF_CFG_TAG_TYPE	change tag type.
RF_CFG_TAG_TIF_GC	change tag time in field and group count.
RF_CFG_TAG_RESEND_TIME	change tag resend time.
RF_CFG_TAG_REPORT_TAMPER	set tag to report tamper.
RF_CFG_TAG_REPORT_TAMPER_HISTORY	set tag to report tamper history.
RF_CFG_TAG_REPORT_NO_TAMPER	set tag to not to report tamper.
RF_CFG_TAG_FACTORY_SETTING	set tag configuration to factory setting.

TimeInField

The time period that the tag must be in the field before sending its date.

groupCount
The number of times that the tag sends its data for each command or when it is detected.

resendTime
The time period between each packet that is sent by the tag.

resendTimeType
The type of the resendTime: seconds, minutes or hours..

reportTamper
Report real time when the tag is tampered.

reportTamperHistory
Report the tamper history for the tag

noTamperReport
Do not report when the tag is tampered.

noChangeTamper
Do not change the setting for the tag.

factorySetting
Change the tag setting to the factory setting.

assignedReader
The tag assigned Reader. If zero, tag will respond to any reader ID otherwise the tag will respond to the reader that its ID matches assignedReader.

Table Index

.

4.9 rfSmartFGen_t

```
typedef struct {
  Uint16  ID,             // smart Fgen ID
  Uint16  readerID,       // assigned reader ID
  Byte    txTime,         // transmit time in seconds
  Byte    waitTime,       // wait time
  Byte    wTimeType,      // wait time type sec, min or hour
  Byte    tagType,        // tag type
  UInt32  tagID,          // tag ID
  Byte    fsValue,        // field strength
  Boolean longDistance,   // short or long range
  Boolean mDetectStatus,  // motion detector status enable or disable
  Boolean mDetectActive,  // motion detector active high or low
  Boolean longInterval,   // long interval between each TX packet
  Boolean assignRdr       // Assign reader ID to the tag
} rfSmartFGen_t;
```

Members

ID

Smart Field Generator Field ID.

readerID

Smart field generator assigned reader ID. This reader ID will be used to communicate with the reader.

txTime

Transmit time in seconds.

waitTime

Smart Field Generator wait time.

wTimeType

Wait time unit of time:

Value	Meaning
RF_TIME_HOUR	Unit of time in Hour
RF_TIME_MINUTE	Unit of time in Minute
RF_TIME_SECOND	Unit of time in second

tagType

Smart Field Generator tag type:

Value	Meaning
ACCESS_TAG	Access tag
ASSET_TAG	Asset tag
INVENTORY_TAG	Inventory tag
ALL_TAGS	Any tag type

tagID
Smart Field Generator tag ID.

fsValue
Smart Field Generator field strength value.

mDetectStatus
Motion Detector status. True for Enable motion detector and false for disable.

mDetectActive
Motion Detector active high when true otherwise active low.

longInterval
Long interval between each packet transmission. True for longInterval and false for Short interval.

assignedRdr
Field Generator assigned reader ID.

Table Index

4.9 rfTagSani_t

```
typedef struct {
   rfSaniUnitType_e UnitType;
   rfSaniStatus_e Status;
   Byte EventCnt;
} rfTagSani_t;
```

Members

UnitType

Indicate the SaniFaucet unit type that engaged with the tag:

rfSaniUnitType_e	Meaning
0, NoUnit	Not a SaniFaucet unit
7, Door	SaniFaucet Door unit
8, Faucet	SaniFaucet Faucet unit
9, Sanitization	SaniFaucet Sanitization unit
10, Contamination	SaniFaucet Contamination unit
11, Bed	SaniFaucet Bed unit

Status

Indicate the SaniFaucet status of the tag:

rfSaniStatus_e	Meaning
0, Violation	Tag is in violation state; when contaminated for too long.
1, ContaminatedOther	Contaminated by other means (contamination unit).
2, ContaminatedPatient	Contaminated after engaging patient.
3, EngagingPatient	Engaging patient.
4, ContaminatedBathroom	Contaminated by bathroom
5, Clean	Clean.
6, AlcoholClean	It was just cleaned by alcohol.
7, SoapClean	It was just cleaned by soap.

EventCnt

Counter associated with event. It is used to discriminate redundant status broadcasts.

5 - Type Definition

```
UInt32     4 bytes - 32 bits unsigned integer
UInt16     2 bytes - 16 bits unsigned integer
Int16      2 bytes - 16 bits signed integer
```

Int32 4 bytes - 32 bits signed integer
Byte 1 byte - 8 bits unsigned integer
SByte 1 byte - 8 bits signed integer
Single; 4 bytes - 32 bits Single point number (precision 7 bits)
Boolean true or false

Table Index