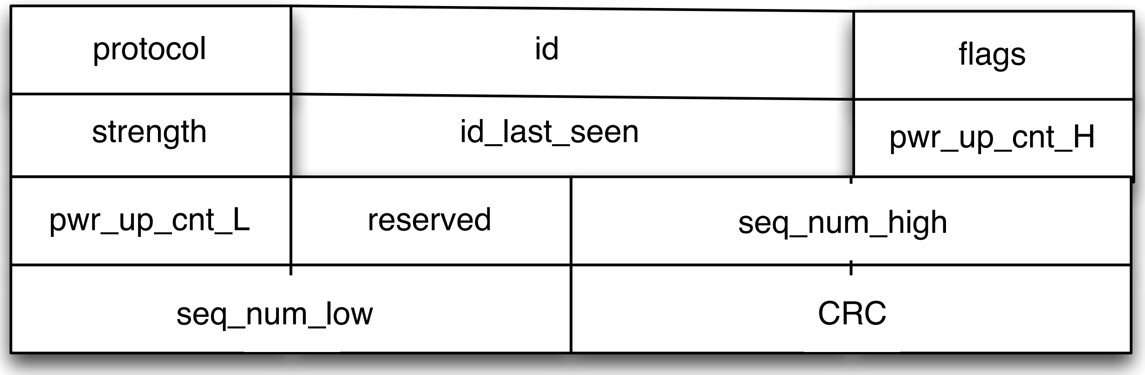
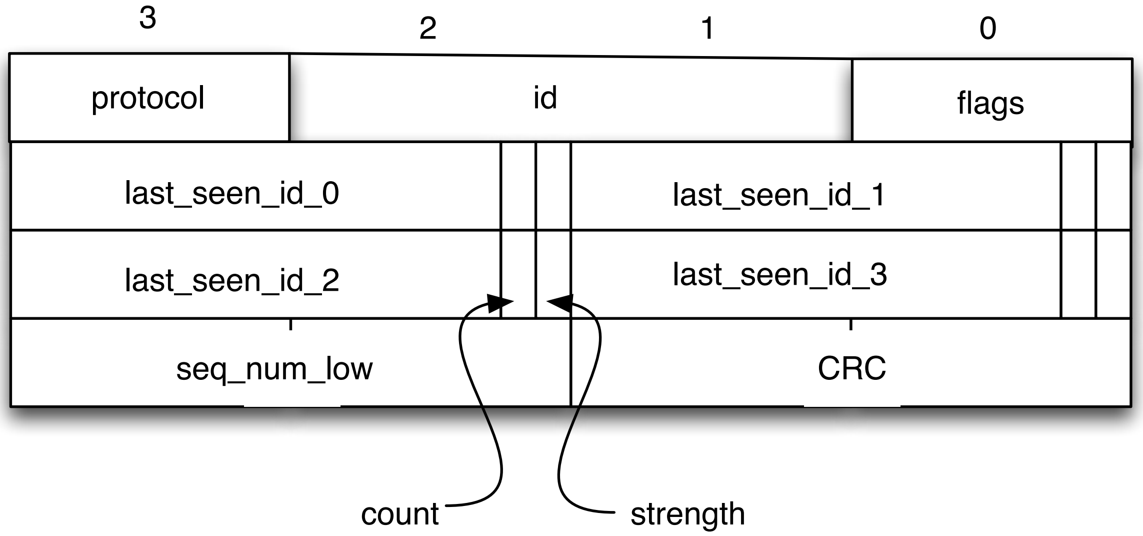
UQ Version of ‘openbeacon-tracker’

Packet Formats: Tag-to-Reader formats

Protocol BEACONTRACKER (0x18) Packet Format



Protocol PROXREPORT_EXT (0x46) Packet Format

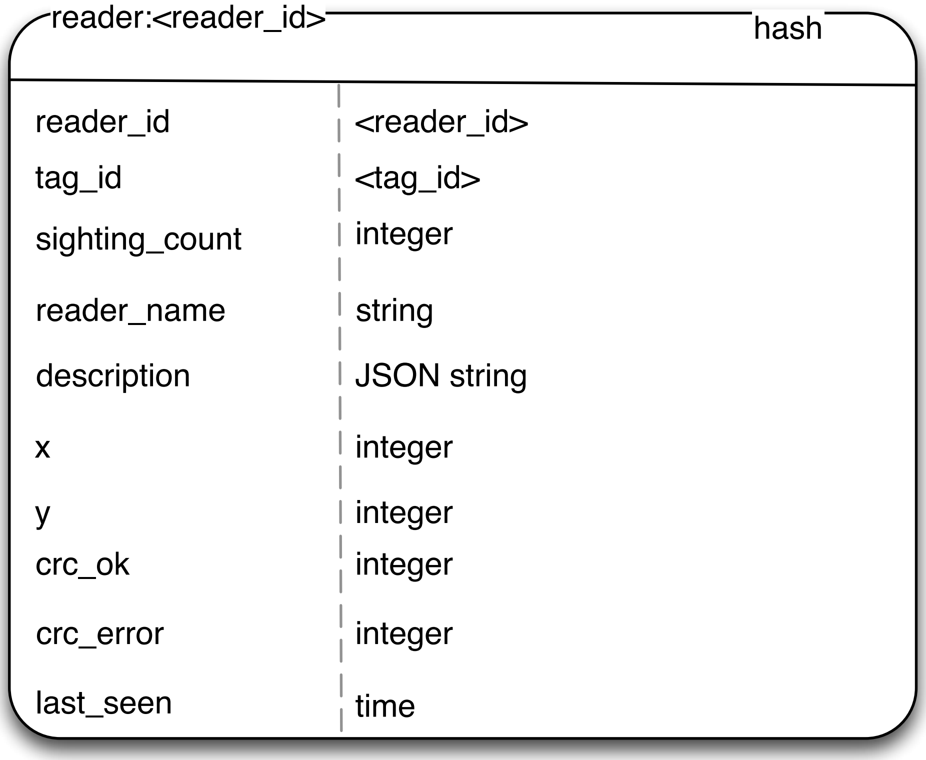


This remainder of this document details the data structures used in the Redis database.

Reader Statistics

KEY: reader:<reader_id>
EXAMPLE: reader:130.102.86.87

This structure is used to record data about each reader in the Openbeacon network. At this stage, it is just statistical information. The static information about each reader is held separately (for now).



reader:<reader_id> hash

reader_id	<reader_id>
tag_id	<tag_id>
sighting_count	integer
reader_name	string
description	JSON string
x	integer
y	integer
crc_ok	integer
crc_error	integer
last_seen	time

reader_id — The id of the reader for this sighting. Uses the IP address, e.g., 130.102.86.87

tag_id — The id of the tag sighted by the reader, unique to each tag, e.g., 1122.

sighting_count — This is a counter, used to generate a unique key when we enter the signal strength the tag transmitted with. Remember, nRF24L01+ chips do not supply RSSI. The solution is to have the tag transmit four separate messages each at a different power level. We record the level of the received message in an ordered set, where the score is the timeout on the life of the reading. The id of the entry is the sighting_count, as we may have more than one sighting in the same time period (not observed in practice though).

reader_name — This is a printable name of the reader, rather than its IP address, e.g., 'openbeacon1' or 'marksOffice'.

description This is a string contain JSON data, whatever you want here. The end application can make use of the data, but not this particular application.

x X-coordinate of the reader, based on some external mapping.

y Y-coordinate of the reader, based on some external mapping.

crc_ok Count of the number of valid packets received by this reader.

crc_error Count of the number of invalid packets (including CRC errors) sighted by this reader.

last_seen Last time this reader sighted anything. This was used in the original system in conjunction with the timeout READER_TIMEOUT_SECONDS (default value of 60*15 seconds) to remove the reader the ‘reader_list, and so suppress any reporting.

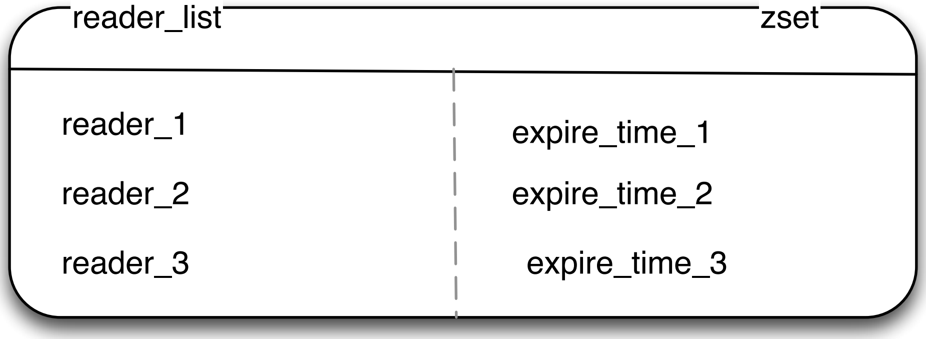
The ‘crc_ok’ and ‘crc_error’ allow us to calculate packet statistics, of a sort. ‘crc_error’ records both the packet which cannot be decoded and the decoded packet which has a checksum error. The XXTEA algorithm does not give an error while decoding unless there are invalid characters in the string to be decoded - these are what we count here and call ‘CRC errors’. These sort of errors were not observed with anywhere near the frequency as the CRC errors.

Typical calculation:

Percent CRC errors = crc_error / (crc_ok + crc_error) * 100

KEY: reader_list

EXAMPLE: [reader:130.102.86.87 1373240802, reader:130.102.86.141 1373240822]
EXPIRES: READER_TIMEOUT_SECONDS via helper process

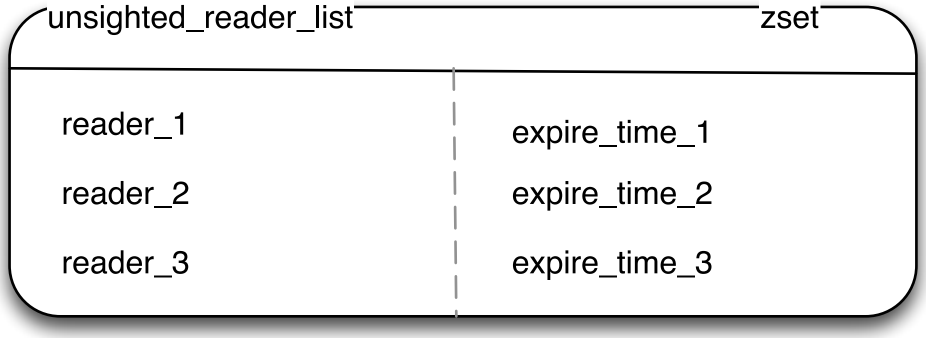


reader_*i* This is a set of all the readers defined as in use by the system, regardless of how long since a reader was last sighted, i.e., it is a static set. reader_ids take the form of IP addresses, e.g., 130.102.86.87

expire_time_*i* Time at which reader_*i* is to move from this list to the ‘unsighted_reader’ list.

KEY: unsighted_reader_list

EXAMPLE: [reader:130.102.86.87 1373240902, reader:130.102.86.141 1373240922]
EXPIRES: 10 * READER_TIMEOUT_SECONDS via helper process



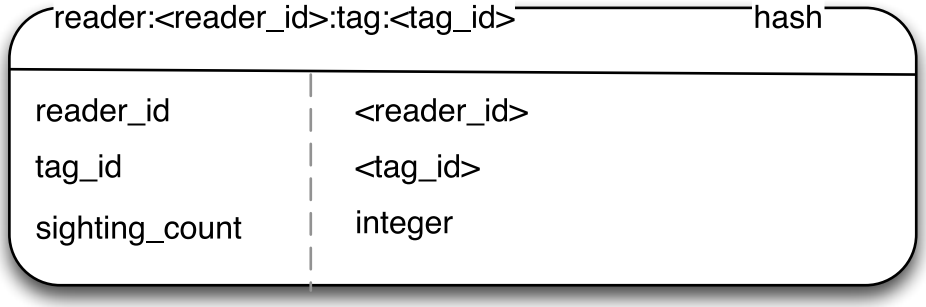
reader_*i* This is the reader_id of an unsighted tag,, e.g., 130.102.86.87

expire_time_*i* Time at which reader_*i* is to move from ‘unsighted_reader’ list permanently. NOTE: For now, let’s never do this!!!

Reader-Tag Relationships

KEY: reader:<reader_id>:tag:<tag_id>

EXAMPLE: reader:130.102.86.87:tag:1122
EXPIRES: READER_TIMEOUT_SECONDS (default: 60*15), software determined



reader:<reader_id>:tag:<tag_id> hash

reader_id	<reader_id>
tag_id	<tag_id>
sighting_count	integer

reader_id The id of the reader for this sighting. Uses the IP address, e.g., 130.102.86.87

tag_id The id of the tag sighted by the reader, unique to each tag, e.g., 1122.

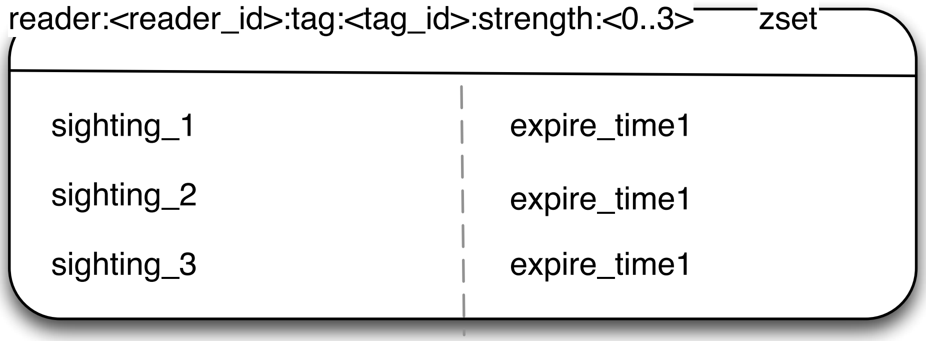
sighting_count This is a counter, used to generate a unique key when we enter the signal strength the tag transmitted with. Remember, nRF24L01+ chips do not supply RSSI. The solution is to have the tag transmit four separate messages each at a different power level. We record the level of the received message in an ordered set, where the score is the timeout on the life of the reading. The id of the entry is the sighting_count, as we may have more than one sighting in the same time period (not observed in practice though).

KEY: reader:<reader_id>:tag:<tag_id>:strength:<0..3>

EXAMPLE: reader:130.102.86.141:tag:975:strength:3
ENTRY EXAMPLE: [10023 1373240802, 10026 1373240842]
EXPIRES: MAX_AGGREGATION_SECONDS via helper process.

There are four of these strings (ending in 0 to 3) for each reader-tag pair. This structure records the number of sightings that have been made in the current time window. After MAX_AGGREGATION_SECONDS, each sighting is deleted.

The number of sightings in the current time window is given by the cardinality of each set (ZCARD) for each transmit power (‘strength’).



reader:<reader_id>:tag:<tag_id>:strength:<0..3> zset

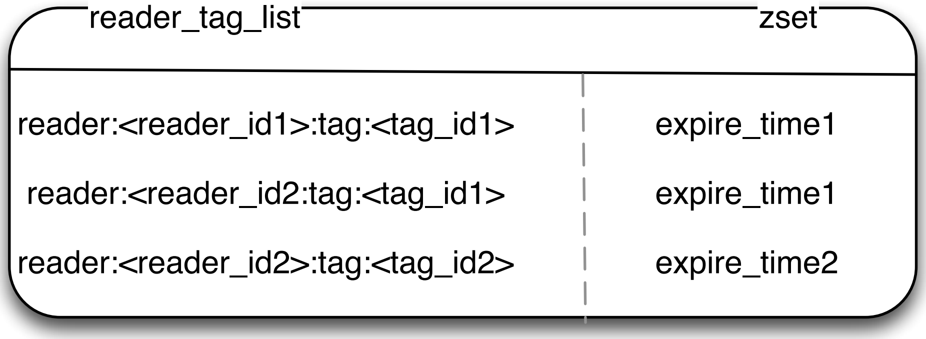
sighting_1	expire_time1
sighting_2	expire_time1
sighting_3	expire_time1

sighting_1 This is a unique identifier, generated from the corresponding ‘sighting_count’ held in the reader-tag pair record. Each ‘sighting_n’ is unique, as we may have more than one sighting from the same tag expire at the same time (due to the granularity of the expire_time value).

expire_time_1 The time at which the sighting expires and is removed from the ordered set, e.g., 1373240802.

KEY: reader_tag_list

This is an ordered set of all the reader-tag pairs that are currently active; reader-tag pairs that have not been active for MAX_AGGREGATION_SECONDS are removed from this ordered set.

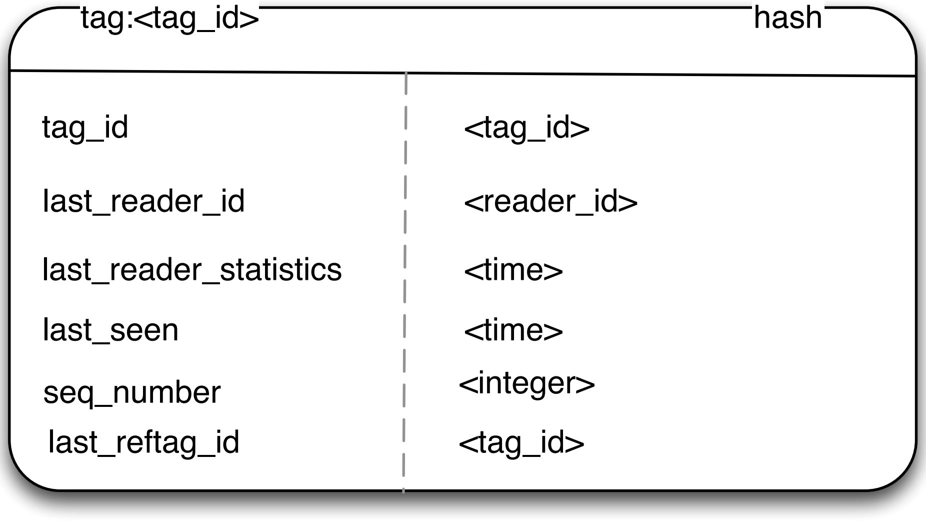


reader_tag_list	zset
reader:<reader_id1>:tag:<tag_id1>	expire_time1
reader:<reader_id2:tag:<tag_id1>	expire_time1
reader:<reader_id2>:tag:<tag_id2>	expire_time2

KEY: tag_id:<tag_id>

EXAMPLE: tag:1112

This records all the information about a tag that is in current use. Tags that have not been sighted for RESET_TAG_POSITIONS_SECONDS are removed.



tag:<tag_id> hash

tag_id	<tag_id>
last_reader_id	<reader_id>
last_reader_statistics	<time>
last_seen	<time>
seq_number	<integer>
last_reftag_id	<tag_id>

tag_id The id of the tag (duplicate from the key for ease of use), e.g., 1112

last_reader_id last reader to sight this tag, e.g., 130.102.86.87

last_reader_statistics The last time reader (and reftag) statistics were calculated. Used in the display software.

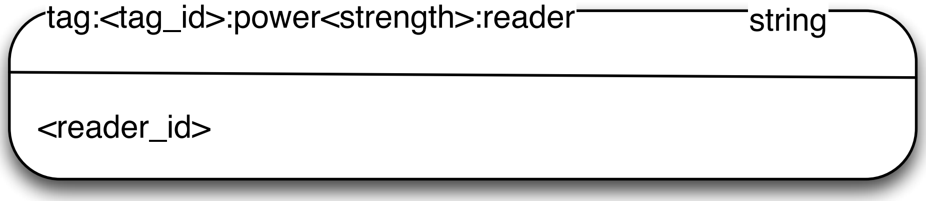
last_seen This is the time this tag was last seen by a reader or a reference tag. Somewhat obsolete now that we age out the records with a helper process, which searches the tag_list ordered set every 200 ms for entries to remove (well, to move to the 'unsighted_tags_list').

description This is a string contain JSON data, whatever you want here. The end application can make use of the data, but not this particular application.

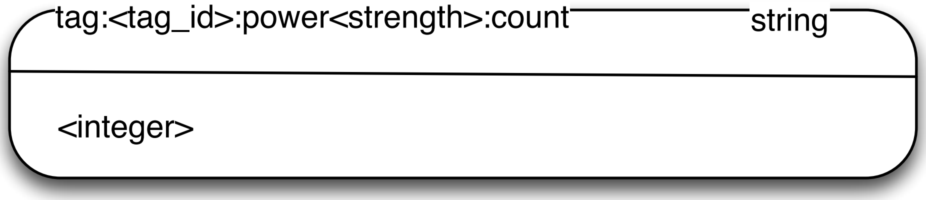
seq_number Records the sequence number of the meesage reporting the sighting. Not used in any UQ project at the moment.

last_reftag_id The ID of the last reference tag to sight this tag.

KEY: tag:<tag_id>:power:<strength>:reader
EXAMPLE: tag:1112:power:1:reader



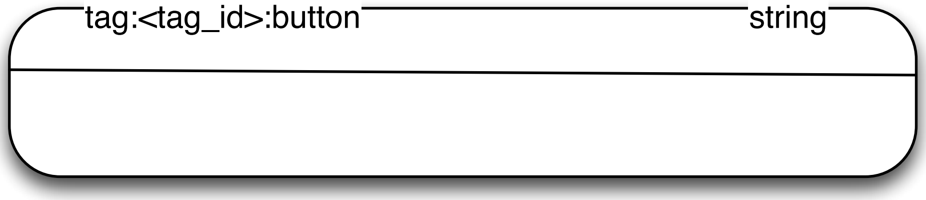
KEY: tag:<tag_id>:power:<strength>:count
EXAMPLE: tag:1112:power:1:count



These two keys are used to record the information about the 'nearest' reader at each transmission level; the nearest reader should be able to hear the signal at the lowest level, so the count should be greatest for that reader.

KEY: tag:<tag_id>:button
KEY:tag:1122:button
EXPIRES: TAGSIGHTING_BUTTON_TIME_SECONDS by Redis key xpire

This key is set when the button on a badge is pressed. It remains present for TAGSIGHTING_BUTTON_TIME_SECONDS and is then expired.



KEY: tag_list
EXAMPLE: [1112 1373240902, 1122 1373240922]
EXPIRE: MAX_AGGREGATION_SECONDS via helper process

tag_list	zset
tag_id_1	expire_time_1
tag_id_2	expire_time_2
tag_id_3	expire_time_3

tag_id_*i* This is the tag_id of a tag,, e.g., 1112

expire_time_*i* Time at which tag_id_*i* is to move to the ‘unsighted_tag_list’.

KEY: unsighted_tag_list
EXAMPLE: [1112 1373240902, 1122 1373240922]
EXPIRE: `REMOVE_UNSIGHTED_TAGS_SECONDS` via helper process

Tags are permanently removed from any list by this time. Is a tag is re-sighted before expiration, then the tag_id needs to be removed from this list and a new entry made in the ‘tag_list’.

tag_list	zset
tag_id_1	expire_time_1
tag_id_2	expire_time_2
tag_id_3	expire_time_3

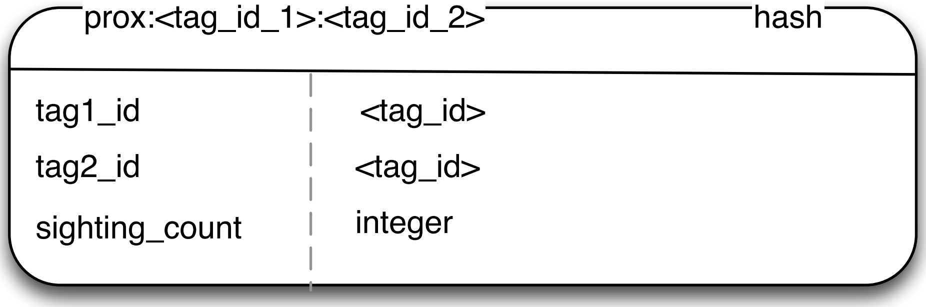
tag_id_*i* This is the tag_id of a tag,, e.g., 1112

expire_time_*i* Time at which tag_id_*i* is to move to the ‘unsighted_tag_list’.

PROXIMITY INFORMATION

On a side channel, the tags broadcast their identity and listen for other badges nearby. The tags have a much smaller and weaker antenna, so can only do this over a short range, as opposed to the lang range detection that readers can perform.

KEY: prox:tag_id_1>:<tag_id_2>
EXAMPLE: prox:1112:1122



prox:<tag_id_1>:<tag_id_2> hash

tag1_id	<tag_id>
tag2_id	<tag_id>
sighting_count	integer

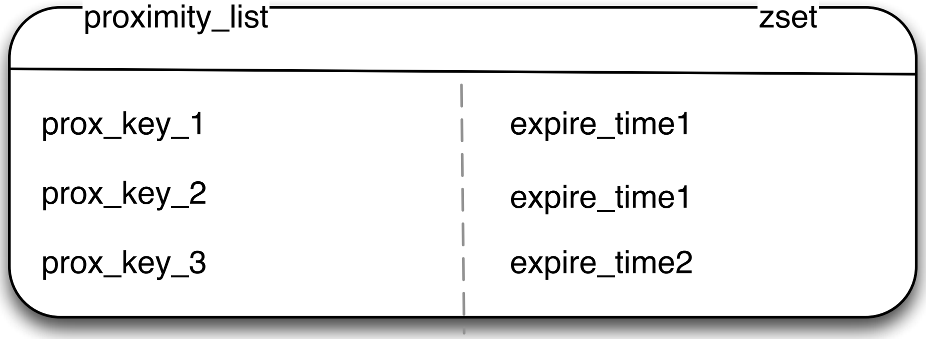
tag1_id The id of the tag sighted by this tag. The tags IDs are ordered in the key, with the lowest ID first, e.g., 1112.

tag2_id The id of the tag sighted by the first tag, with the larger ID second, e.g., 1122.

sighting_count This is a counter, used to generate a unique key when we enter the signal strength reported between the tags.

KEY: proximity_list
EXAMPLE: [prox:975:1127, prox:1112:1122]

This ordered set contains the set of all currently active pairs of tags reporting proximity relationships. Proximity keys are valid only for PROXAGGREGATION_TIME, after which they are removed from the set. The expiry time is used as the score used to order the items.



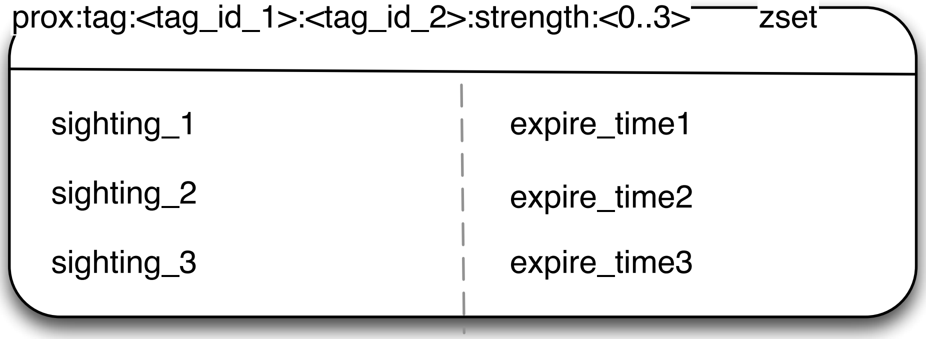
proximity_list	zset
prox_key_1	expire_time1
prox_key_2	expire_time1
prox_key_3	expire_time2

prox_key_*i* This is the key identify the two tags which reported a proximity sighting.

expire_time*i* This is the time at which the sighting will expire, unless a new sighting occurs which restarts the expiry time.

KEY: prox:<tag_id_1>:<tag_id_2>:strength:<0..3>
EXAMPLE: prox:975:1127:strength:1

This ordered set records the most recent sighting strengths. Sightings older than PROXAGGREGATION_TIME are removed from the set. The number of sightings at a particular strength is given by the cardinality of the ordered set (ZCARD)



prox:tag:<tag_id_1>:<tag_id_2>:strength:<0..3> zset

sighting_1	expire_time1
sighting_2	expire_time2
sighting_3	expire_time3

sighting_1 This is a unique identifier, generated from the corresponding ‘sighting_count’ held in the proximity-tag pair record. Each ‘sighting_n’ is unique, as we may have more than one sighting from the same tag expire at the same time (due to the granularity of the expire_time value).

expire_time_1 The time at which the sighting expires and is removed from the ordered set, e.g., 1373240802.

These entries take the form of KEY:<sighting><expire_time_score>, e.g., [10023 1373240902, 10024 1373240922]

REFERENCE TAGS

The UQ version of the Openbeacon system introduces the concept of Reference Tags. Reference tags are normal tags placed around a space, but the tags do not move. These tags report any tags that are then ‘proximate’. A typical use would be to put tags on paintings in an art gallery and then record the time spent at each painting (the dwell time) and the number of distinct visitors to each painting; this information can then be used to rank the popularity of each painting, which may then suggest an alternative layout of paintings in the gallery.

KEY: reftag:<tag_id>

reftag:<reftag_id>	hash
reader_id	<reader_id>
tag_id	<tag_id>
sighting_count	integer
reader_name	string
description	JSON string
x	integer
y	integer
crc_ok	integer
crc_error	integer
last_seen	time

reader_id — The id of the reference tag for this sighting. Uses the ID of the reference tag, e.g., 954

tag_id — The id of the tag sighted by the reference tag, unique to each tag, e.g., 1122.

sighting_count — This is a counter, used to generate a unique key when we enter the signal strength the tag transmitted with. Remember, nRF24L01+ chips do not supply RSSI. The solution is to have the tag transmit four separate messages each at a different power level. We record the level of the received message in an ordered set, where the score is the timeout on the life of the reading. The id of the entry is the sighting_count, as we may have more than one sighting in the same time period (not observed in practice though).

reader_name

This is a printable name of the reference tag, rather than its IID, e.g., ‘frontdoorsouth’ or ‘marksOffice’.

description

This is a string contain JSON data, whatever you want here. The end application can make use of the data, but not this particular application.

x

X-coordinate of the reference tag, based on some external mapping.

y

Y-coordinate of the reference tag, based on some external mapping.

crc_ok

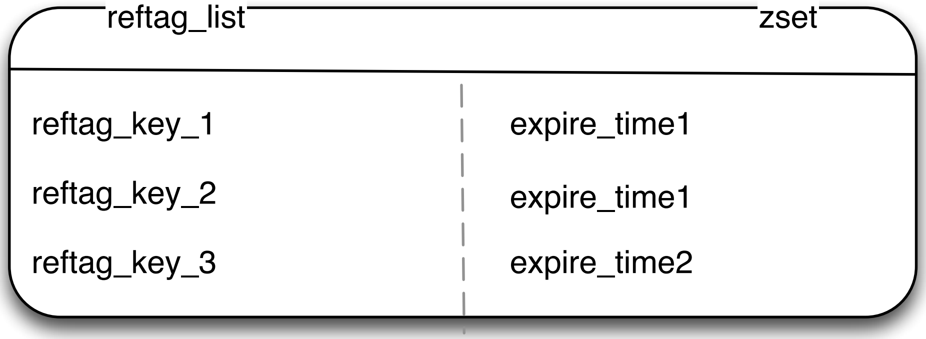
Count of the number of valid packets received by this reference tag. NOTE: we can’t measure ‘crc_error’, because this is only performed by readers, and not in proximity sightings.

last_seen

Last time this reference tag was sighted anything. In terms of readers, this was used in the original system in conjunction with the timeout READER_TIMEOUT_SECONDS (default value of 60*15 seconds) to remove the reader the ‘reader_list, and so suppress any reporting.

KEY: reftag_list
EXAMPLE: [reftag:975 1373240902, reftag:1112, 1373240922]
EXPIRED: MAX_AGGREGATION_SECONDS by a helper process

The reftag_list is an ordered set containing all the currently active reftags. Reftags which have not been sighted for more than MAX_AGGREGATION_SECONDS are removed from this list and placed in the 'unsighted_reftag_list'.



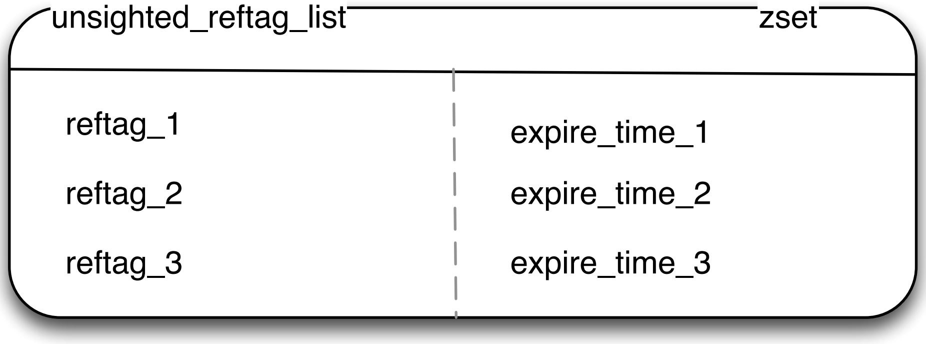
reftag_list	zset
reftag_key_1	expire_time1
reftag_key_2	expire_time1
reftag_key_3	expire_time2

reftag_key_*i* This is a set of all the reference tags defined as in use by the system, e.g., 954.

expire_time_*i* Time at which reftag_*i* is to move from this list to the 'unsighted_reader' list.

KEY: unsighted_reftag_list
EXAMPLE: [reftag:975 1373240902, reftag:1112, 1373240922]
EXPIRED: READER_TIMEOUT_SECONDS by a helper process

The reftag_list is an ordered set containing all the currently active reftags. Reftags which have not been sighted for more than MAX_AGGREGATION_SECONDS are removed from this list and placed in the ‘unsighted_reftag_list’.

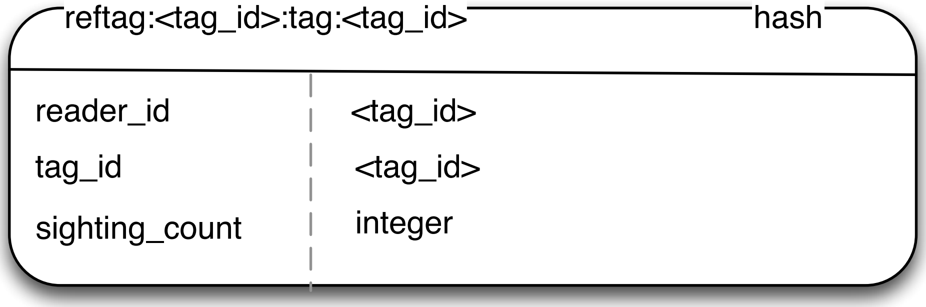


unsighted_reftag_list	zset
reftag_1	expire_time_1
reftag_2	expire_time_2
reftag_3	expire_time_3

reftag_key_*i* This is a set of all the reference tags defined as in use by the system, e.g., 954.

expire_time_*i* Time at which reftag_*i* is to move from this list to the ‘unsighted_reader’ list.

KEY: reftag:<tag_id>:tag:<tag_id>
EXAMPLE: reftag:1131:tag:1127
EXPIRES: MAX_AGGREGATION_SECONDS via a helper process



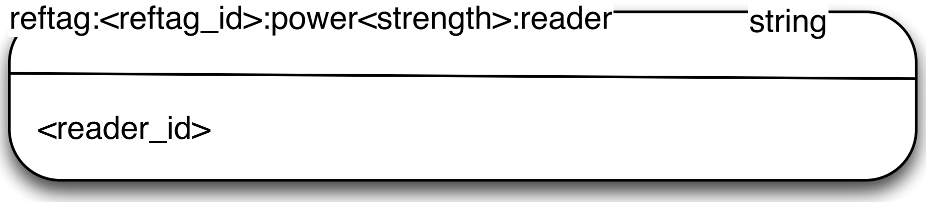
reftag:<tag_id>:tag:<tag_id>	hash
reader_id	<tag_id>
tag_id	<tag_id>
sighting_count	integer

reader_id — The id of the reference tag for this sighting. Uses the tag ID, e.g., 1131

tag_id — The id of the tag sighted by the reader, unique to each tag, e.g., 1127.

sighting_count — This is a counter, used to generate a unique key when we enter the signal strength that the tags are sighted with.

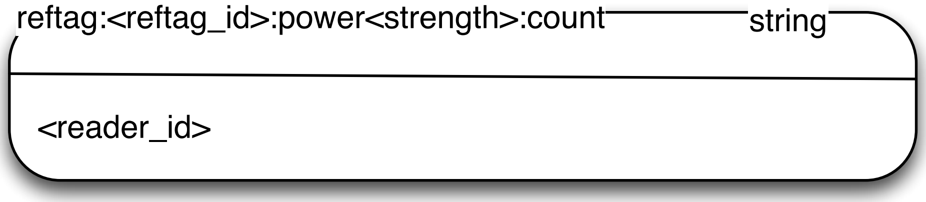
KEY: reftag:<reftag_id>:power:<strength>:reader
EXAMPLE: reftag:1112:power:1:reader
EXPIRES: MAX_AGGREGATION_SECONDS via a helper process



reftag:<reftag_id>:power<strength>:reader string

<reader_id>

KEY: reftag:<tag_id>:power:<strength>:count
EXAMPLE: reftag:1112:power:1:count
EXPIRES: MAX_AGGREGATION_SECONDS via a helper process



reftag:<reftag_id>:power<strength>:count string

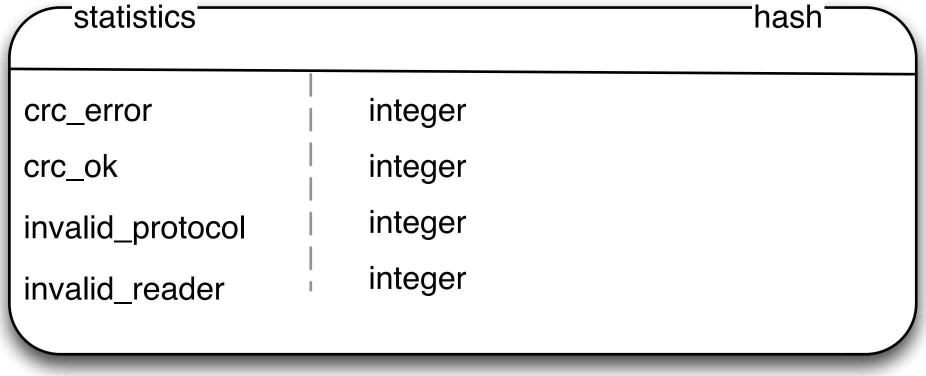
<reader_id>

These two keys are used to record the information about the ‘nearest’ reference tag at each transmission level; the nearest reference tag should be able to hear the signal at the lowest level, so the count should be greatest for that reference tag.

STATISTICS

We record some statistics of the overall system performance. These are held under the following keys in Redis.

KEY: statistics



statistics hash

crc_error	integer
crc_ok	integer
invalid_protocol	integer
invalid_reader	integer