



SmartSpace

Grenade Tag Support

For version 3.3.6828 and above

Copyright © 2019, Ubisense Limited 2014 - 2019. All Rights Reserved. You may not reproduce this document in whole or in part without permission in writing from Ubisense at the following address:

Ubisense Limited
St Andrew's House
St Andrew's Road
Cambridge CB4 1DL
United Kingdom

Tel: +44 (0)1223 535170

WWW: <https://www.ubisense.net>

All contents of this document are subject to change without notice and do not represent a commitment on the part of Ubisense. Reasonable effort is made to ensure the accuracy of the information contained in the document. However, due to on-going product improvements and revisions, Ubisense and its subsidiaries do not warrant the accuracy of this information and cannot accept responsibility for errors or omissions that may be contained in this document.

Information in this document is provided in connection with Ubisense products. No license, express or implied to any intellectual property rights is granted by this document.

Ubisense encourages all users of its products to procure all necessary intellectual property licenses required to implement any concepts or applications and does not condone or encourage any intellectual property infringement and disclaims any responsibility related thereto. These intellectual property licenses may differ from country to country and it is the responsibility of those who develop the concepts or applications to be aware of and comply with different national license requirements.

Windows® is a registered trademark of Microsoft Corporation in the United States and/or other countries. The other names of actual companies and products mentioned herein are the trademarks of their respective owners.

Contents

- Overview of Grenade tag support** 1
- Application-level overview of tag behavior** 2
- How to configure the sensors** 4
 - Configuring Countdown Ack Threshold 4
 - Configuring Mode and Value 0 5
- Example output using the tracing options** 7
 - Tracing options 7
 - Simple test setup 7
 - Sensor (mis)configuration 8
 - Test traces 8
 - Test 1: Multiple failed attempts to ack a tag (due to misconfiguration) 8
 - Test 2: Successful ack of the tag 10
 - Test 3: Successful ack of two tags simultaneously 10

Overview of Grenade tag support

This guide describes the behavior of a tag attached to a grenade: how it is controlled by using tag status message acknowledgment via 2.4 GHz wireless, and how its status is pushed to the server-side schemas to make it available to application-level code. An example configuration and test results are provided.

Application-level overview of tag behavior

The following describes the behavior of a tag attached to a grenade:

1. *Initial state.* Initially the tag is not armed and does not transmit anything.
2. *When the grenade is armed.* At some point the grenade's pin is pulled so that it is armed; the tag wakes up and transmits at rate 2 Hz, sending the data message 0x0660 (with data payload 0x0000). This status will be pushed to the TagStatus schema in an expedited way (i.e. we do not use the normal TagStatus batching for these messages). This allows client application code to be notified and take some action when a grenade is armed.
3. *When the grenade is thrown.* When the grenade spoon is released (i.e. when the grenade is thrown) the tag transmits at 10 Hz, sending the status message 0x0666, for a period of 5 seconds before it 'explodes' and ceases transmissions. The data payload of the status message is:
 - **MSB:** 'ack count', the number of 2.4 GHz ack messages received (i.e. initially zero)
 - **LSB:** 'message count', the number of transmissions since the spoon was released
4. *Tracking the grenade and updating its status during countdown.* If the tag transmissions are picked up by a sensor group, they will be forwarded (in the usual way) to a master sensor for the tag in that group. The master sensor will update the TagStatus schema every second (i.e. after 10, 20, 30, 40, 50 messages) if it has managed to generate a location for the tag in the previous second or if the count of received ack messages is greater than zero, effectively giving a 4, 3, 2, 1, 0 countdown for tags which are receiving or have received good sightings. There are no special mechanisms to deal with 'dropped' tag transmissions because they are highly unlikely in the context (the tag has to get located by the master sensor so each transmission will be received by a few sensors). This notification mechanism allows client application code to be notified and take action in response to the tag countdown.
5. *Acknowledging tag transmissions.* The sensor software will send ack messages to the tag if it has a good position for the tag sufficiently close to the time when the grenade explodes. The purpose of the ack messages is solely to disable the third-party hardware mechanism for weapons effects. The timing of the ack is determined by the **Countdown Ack Threshold** parameter which should be set to some value T , where $0 \leq T < 50$, such that an ack will be sent for the tag if:

1. The tag has just been located in this measurement, and
2. The message count in this measurement is greater than T, and
3. The ack count in this measurement is zero

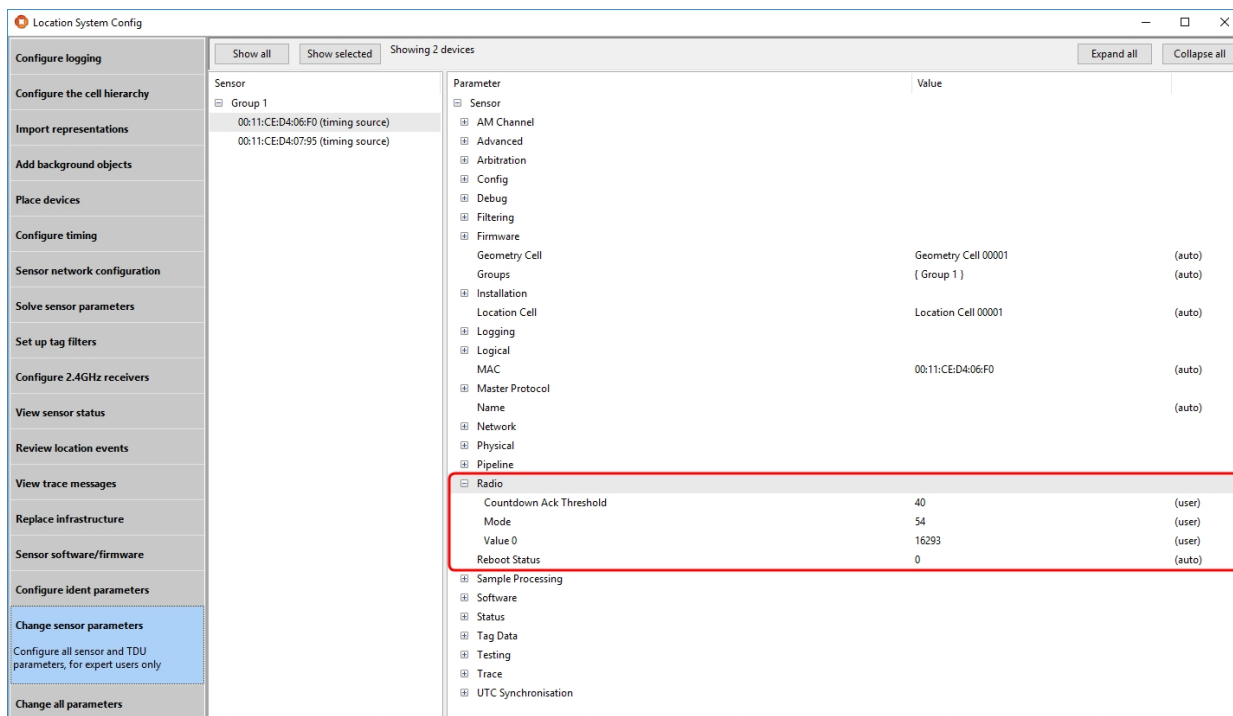
Thus the **Countdown Ack Threshold** setting allows the user to control how long to wait before the tag is acknowledged. The default value of **Countdown Ack Threshold** is 0, meaning that the tag will be acked as soon as it is thrown. But in practice it may be preferable to set the threshold higher, because effective calculation of area weapons effects requires that we know the position of the grenade at the time that it detonated. If the grenade is thrown into an area where it cannot be tracked then we are better off using the third-party area weapons mechanism, so we should not ack the tag. However it is not possible to leave it too long before acking the tag because we would not have time to get the ack message to the tag before the third-party mechanism was triggered. The example traces shown later in the document show the behavior when **Countdown Ack Threshold** is set to 40.

6. *Grenade explosion.* The grenade will explode 5 seconds (i.e. fifty transmissions) after the spoon was released. When the grenade explodes the tag ceases transmitting and communicates its ack status to the third-party hardware. If no acks have been received then the third-party hardware kicks off some appropriate area weapons effect evaluation. When the tag status has message count of 50, the grenade has detonated: this allows application-level client code to be notified of the grenade detonation.
7. *Evaluating area weapons effects.* The area weapons effects software should operate whenever any tag (for a grenade) has status **0x0666** with data **0xXX32**, i.e. where XX is the ack count and **0x32** (decimal 50) is the message count.
 - If $XX == 0$ then the third-party hardware mechanism will be operating, so the area weapons software must make some arbitration decision (e.g. leave the weapons effects to the third-party hardware, override the hardware in some way).
 - If $XX > 0$ then the third-party hardware mechanism will have been disabled, in which case there will also be a valid location available for the tag with the location at which it was seen shortly before the explosion. By using the tag location, together with the locations of other players at the same time, the area weapons effects of the grenade detonation may be calculated.

How to configure the sensors

The sensors each require three parameters to be set in order to control the acknowledgement functions.

The parameters are configured in Location System Config in the **Radio** section of the **Change sensor parameters** configuration screen.



The screenshot shows the 'Location System Config' window with the 'Change sensor parameters' configuration screen. The 'Radio' section is highlighted with a red box, showing the following parameters:

Parameter	Value	Setting
Countdown Ack Threshold	40	(user)
Mode	54	(user)
Value 0	16293	(user)
Reboot Status	0	(auto)

Their values must be configured according to the following guidelines:

Configuring Countdown Ack Threshold

As previously discussed, **Countdown Ack Threshold** should be set fairly close to the value of the tag message count when the grenade detonates but low enough such that the sensors have time to acknowledge the tag. A good compromise is to set the value to the value of the message count one second before detonation, so if the tag's location is calculated one second before detonation an ack will be sent. In the current implementation this corresponds to a value of **40**.

Configuring Mode and Value 0

The radio **Mode** parameter must be configured to the value **54**. This sets up the radio in the sensor and starts it running. If the radio mode is not configured to the value **54** then a warning message will be generated by the sensor if it receives a request to send an ack to the tag. By default this value is set to zero, which ensures that the sensor will not transmit on the 2.4 GHz channel (this is required in some installations).

The radio **Value 0** parameter must be configured in order to set a suitable transmit power and transmission 'on-time' for each radio transmission. When a sensor transmits an ack to a tag it actually transmits a repeated sequence of messages over a period of 40 ms, where each message is about 1 ms duration, and where the sensor can choose randomly whether or not to transmit each message in the sequence. The **Value 0** parameter combines two values:

- The 'on time', which represents the probability of transmitting each message, and
- The 'power setting' which controls the transmit power of the radio.

The parameters are combined into the single **Value 0** parameter using this table:

Approximate Range:		100%	75%	50%	25%	10%
Power setting:		255	210	165	138	129
Power Byte:		0xFF	0xD2	0xA5	0x8A	0x81
On-Time	On-Time Byte	Decimal "Value 0"				
0% *	0x0000	255	210	165	138	129
5%	0x0C00	3327	3282	3237	3210	3201
10%	0x1900	6655	6610	6565	6538	6529
15%	0x2600	9983	9938	9893	9866	9857
20%	0x3300	13311	13266	13221	13194	13185
25%	0x3F00	16383	16338	16293	16266	16257
30%	0x4C00	19711	19666	19621	19594	19585
35%	0x5900	23039	22994	22949	22922	22913

40%	0x6600	26367	26322	26277	26250	26241
45%	0x7200	29439	29394	29349	29322	29313
50%	0x7F00	32767	32722	32677	32650	32641
55%	0x8C00	36095	36050	36005	35978	35969
60%	0x9900	39423	39378	39333	39306	39297
65%	0xA500	42495	42450	42405	42378	42369
70%	0xB200	45823	45778	45733	45706	45697
75%	0xBF00	49151	49106	49061	49034	49025
80%	0xCC00	52479	52434	52389	52362	52353
85%	0xD800	55551	55506	55461	55434	55425
90%	0xE500	58879	58834	58789	58762	58753
95%	0xF200	62207	62162	62117	62090	62081
100%	0xFF00	65535	65490	65445	65418	65409

* Note: The 0% setting actually corresponds to 1/256, or 0.4%.

A suitable recommended value for the **Value 0** parameter is **16293**, corresponding to a 25% duty cycle and 50% power setting. This should permit the acknowledgement of large numbers of grenade tags simultaneously thrown into the same building. This is the value used in the [example configurations](#).

Example output using the tracing options

Tracing options

In order to simplify user setup, a tracing option is provided. In normal use this option would be disabled, but it provides a 'running commentary' on the sensors' response to tag events which can be helpful if the user is trying to understand the operation of their particular configuration.

To enable the tracing option, set the **platform_monitor** global configuration parameter to contain the string **tag_countdown**, before rebooting the sensors.

For example, execute the command:

```
ubisense_configuration_client set platform_monitor tag_countdown
```

and then reboot the sensors.

Simple test setup

The traces below show some sample output for a simple test setup using a two sensor system; both sensors can see UWB from the tag(s). Up to four tags are used, all with the tag filter initially set to **Always succeed** to ensure that we get a location event for each tag.

Sensor (mis)configuration

Tests 1 and 2 purposefully contain a misconfiguration: the radio has only been configured on one of the two sensors. This is shown on line 2 of the table below. Line 3 shows the correct configuration as used in test 3.

Sensor	Countdown Ack Threshold	Radio Mode	Radio Value 0	Test
00:11:CE:D4:06:F0	40	54	16293	all tests
00:11:CE:D4:07:95	40	0	0	tests 1 and 2
00:11:CE:D4:07:95	40	54	16293	test 3

Test traces

Test 1: Multiple failed attempts to ack a tag (due to misconfiguration)

This test shows the tag being processed by 07:95. Both sensors 'see' the tag UWB and there is a random choice between them to determine the ack-ing sensor. This comes out for 07:95 three times. Each of these ack attempts (msg count = 40, 43, 46) causes a warning message to be generated because the 07:95 radio has not been configured (and the attempt fails). The fourth time the other sensor 06:F0 is selected and this sends an ack to the tag at msg count 49, but this doesn't get transmitted until too late and the tag has not received the acknowledgement at msg count = 50 and therefore the status 50 gets sent to the tag status service.

Tag 1 powered up, button pressed, tracked, acknowledged, but ack not received

```
[15/06/2018 16:19:29] tag_countdown: 00:11:CE:D4:06:F0: 00:11:CE:D4:06:F0 countdown ack threshold = 40
[15/06/2018 16:19:40] tag_countdown: 00:11:CE:D4:07:95: 00:11:CE:D4:07:95 countdown ack threshold = 40
[15/06/2018 16:21:05] tag_countdown: 00:11:CE:D4:07:95: 00:11:CE:D4:07:95 initializes new tag countdown state for
00:11:CE:00:00:00:00:01
[15/06/2018 16:21:05] tag_countdown: tag status server asserts 00:11:CE:00:00:00:00:01 countdown_armed_flag
[15/06/2018 16:21:11] tag_countdown: tag status server asserts 00:11:CE:00:00:00:00:01 countdown_value = 10
[15/06/2018 16:21:12] tag_countdown: tag status server asserts 00:11:CE:00:00:00:00:01 countdown_value = 20
[15/06/2018 16:21:13] tag_countdown: tag status server asserts 00:11:CE:00:00:00:00:01 countdown_value = 30
[15/06/2018 16:21:14] tag_countdown: tag status server asserts 00:11:CE:00:00:00:00:01 countdown_value = 40
[15/06/2018 16:21:14] tag_countdown: 00:11:CE:D4:07:95: 00:11:CE:D4:07:95 sends countdown ack to 00:11:CE:00:00:00:00:01
[15/06/2018 16:21:14] sensor_warning: 00:11:CE:D4:07:95: attempt to send radio ack message with power/duty cycle unset
[15/06/2018 16:21:14] tag_countdown: 00:11:CE:D4:07:95: 00:11:CE:D4:07:95 sends countdown ack to 00:11:CE:00:00:00:00:01
[15/06/2018 16:21:14] sensor_warning: 00:11:CE:D4:07:95: attempt to send radio ack message with power/duty cycle unset
[15/06/2018 16:21:15] tag_countdown: 00:11:CE:D4:07:95: 00:11:CE:D4:07:95 sends countdown ack to 00:11:CE:00:00:00:00:01
[15/06/2018 16:21:15] sensor_warning: 00:11:CE:D4:07:95: attempt to send radio ack message with power/duty cycle unset
[15/06/2018 16:21:15] tag_countdown: 00:11:CE:D4:07:95: 00:11:CE:D4:07:95 forwards acknowledgement request for
00:11:CE:00:00:00:00:00:01 to 00:11:CE:D4:06:F0 at 10.42.5.112:53536
[15/06/2018 16:21:15] tag_countdown: 00:11:CE:D4:06:F0: 00:11:CE:D4:06:F0 sends countdown ack to 00:11:CE:00:00:00:00:01
[15/06/2018 16:21:15] tag_countdown: tag status server asserts 00:11:CE:00:00:00:00:01 countdown_value = 50
[15/06/2018 16:21:26] tag_countdown: 00:11:CE:D4:07:95: 00:11:CE:D4:07:95 ages out old tag countdown state for
00:11:CE:00:00:00:00:00:01
[15/06/2018 16:27:36] tag_countdown: tag status server ages out old entry for 00:11:CE:00:00:00:00:01 countdown_armed_flag
[15/06/2018 16:27:36] tag_countdown: tag status server ages out old entry for 00:11:CE:00:00:00:00:01 countdown_value
```

Test 2: Successful ack of the tag

This is the same configuration as (1) but in this case the sensor 06:F0 is selected for the first ack attempt and the tag is successfully ack-ed.

Tag 1 powered up, button pressed, tracked, acknowledged, ack received

```
[15/06/2018 16:44:24] tag_countdown: 00:11:CE:D4:07:95: 00:11:CE:D4:07:95 initializes new tag countdown state for
00:11:CE:00:00:00:00:01
[15/06/2018 16:44:24] tag_countdown: tag status server asserts 00:11:CE:00:00:00:00:01 countdown_armed_flag
[15/06/2018 16:44:29] tag_countdown: tag status server asserts 00:11:CE:00:00:00:00:01 countdown_value = 10
[15/06/2018 16:44:30] tag_countdown: tag status server asserts 00:11:CE:00:00:00:00:01 countdown_value = 20
[15/06/2018 16:44:31] tag_countdown: tag status server asserts 00:11:CE:00:00:00:00:01 countdown_value = 30
[15/06/2018 16:44:32] tag_countdown: tag status server asserts 00:11:CE:00:00:00:00:01 countdown_value = 40
[15/06/2018 16:44:32] tag_countdown: 00:11:CE:D4:07:95: 00:11:CE:D4:07:95 forwards acknowledgement request for
00:11:CE:00:00:00:00:01 to 00:11:CE:D4:06:F0 at 10.42.5.112:53536
[15/06/2018 16:44:32] tag_countdown: 00:11:CE:D4:06:F0: 00:11:CE:D4:06:F0 sends countdown ack to 00:11:CE:00:00:00:00:01
[15/06/2018 16:44:32] tag_countdown: 00:11:CE:D4:07:95: 00:11:CE:D4:07:95 receives 00:11:CE:00:00:00:00:01 confirmation
that it has received an ack message
[15/06/2018 16:44:33] tag_countdown: tag status server asserts 00:11:CE:00:00:00:00:01 countdown_value = 306
[15/06/2018 16:44:48] tag_countdown: 00:11:CE:D4:07:95: 00:11:CE:D4:07:95 ages out old tag countdown state for
00:11:CE:00:00:00:00:01
```

Test 3: Successful ack of two tags simultaneously

Using the correct radio configuration for both sensors (to improve radio reliability) we show two tags being ack-ed simultaneously. Note that both final values are 306 which is 1 ack + 50 messages, demonstrating that each tag only heard one ack message (the one destined for it).

Tag 1 and 2 powered up, button pressed, tracked, acknowledged, but ack not received

```
[15/06/2018 18:04:46] tag_countdown: 00:11:CE:D4:07:95: 00:11:CE:D4:07:95 initializes new tag countdown state for
00:11:CE:00:00:00:00:01
[15/06/2018 18:04:46] tag_countdown: 00:11:CE:D4:06:F0: 00:11:CE:D4:06:F0 initializes new tag countdown state for
00:11:CE:00:00:00:00:02
[15/06/2018 18:04:46] tag_countdown: tag status server asserts 00:11:CE:00:00:00:00:02 countdown_armed_flag
[15/06/2018 18:04:46] tag_countdown: tag status server asserts 00:11:CE:00:00:00:00:01 countdown_armed_flag
[15/06/2018 18:04:54] tag_countdown: tag status server asserts 00:11:CE:00:00:00:00:02 countdown_armed_flag
[15/06/2018 18:04:55] tag_countdown: tag status server asserts 00:11:CE:00:00:00:00:01 countdown_armed_flag
[15/06/2018 18:04:56] tag_countdown: tag status server asserts 00:11:CE:00:00:00:00:01 countdown_value = 10
[15/06/2018 18:04:57] tag_countdown: tag status server asserts 00:11:CE:00:00:00:00:02 countdown_value = 10
[15/06/2018 18:04:57] tag_countdown: tag status server asserts 00:11:CE:00:00:00:00:01 countdown_value = 20
[15/06/2018 18:04:57] tag_countdown: tag status server asserts 00:11:CE:00:00:00:00:02 countdown_value = 20
[15/06/2018 18:04:58] tag_countdown: tag status server asserts 00:11:CE:00:00:00:00:01 countdown_value = 30
[15/06/2018 18:04:58] tag_countdown: tag status server asserts 00:11:CE:00:00:00:00:02 countdown_value = 30
[15/06/2018 18:04:59] tag_countdown: tag status server asserts 00:11:CE:00:00:00:00:01 countdown_value = 40
[15/06/2018 18:04:59] tag_countdown: 00:11:CE:D4:07:95: 00:11:CE:D4:07:95 forwards acknowledgement request for
00:11:CE:00:00:00:00:01 to 00:11:CE:D4:06:F0 at 10.42.5.112:53536
[15/06/2018 18:04:59] tag_countdown: 00:11:CE:D4:06:F0: 00:11:CE:D4:06:F0 sends countdown ack to 00:11:CE:00:00:00:00:01
[15/06/2018 18:04:59] tag_countdown: tag status server asserts 00:11:CE:00:00:00:00:02 countdown_value = 40
[15/06/2018 18:04:59] tag_countdown: 00:11:CE:D4:06:F0: 00:11:CE:D4:06:F0 sends countdown ack to 00:11:CE:00:00:00:00:02
[15/06/2018 18:05:00] tag_countdown: 00:11:CE:D4:06:F0: 00:11:CE:D4:06:F0 receives 00:11:CE:00:00:00:00:02 confirmation that
it has received an ack message
[15/06/2018 18:05:00] tag_countdown: 00:11:CE:D4:07:95: 00:11:CE:D4:07:95 sends countdown ack to 00:11:CE:00:00:00:00:01
[15/06/2018 18:05:00] tag_countdown: 00:11:CE:D4:07:95: 00:11:CE:D4:07:95 receives 00:11:CE:00:00:00:00:01 confirmation that
it has received an ack message
[15/06/2018 18:05:00] tag_countdown: tag status server asserts 00:11:CE:00:00:00:00:01 countdown_value = 306
[15/06/2018 18:05:00] tag_countdown: tag status server asserts 00:11:CE:00:00:00:00:02 countdown_value = 306
[15/06/2018 18:05:18] tag_countdown: 00:11:CE:D4:06:F0: 00:11:CE:D4:06:F0 ages out old tag countdown state for
00:11:CE:00:00:00:00:02
[15/06/2018 18:05:19] tag_countdown: 00:11:CE:D4:07:95: 00:11:CE:D4:07:95 ages out old tag countdown state for
00:11:CE:00:00:00:00:01
[15/06/2018 18:08:22] tag_countdown: tag status server ages out old entry for 00:11:CE:00:00:00:00:01 countdown_armed_flag
[15/06/2018 18:08:22] tag_countdown: tag status server ages out old entry for 00:11:CE:00:00:00:00:01 countdown_value
[15/06/2018 18:08:22] tag_countdown: tag status server ages out old entry for 00:11:CE:00:00:00:00:02 countdown_armed_flag
[15/06/2018 18:08:22] tag_countdown: tag status server ages out old entry for 00:11:CE:00:00:00:00:02 countdown_value
```