



SmartSpace<sup>®</sup>

## Multi-tag Configuration

From version 3.8

Copyright © 2023, Ubisense Limited 2014 - 2023. 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.com>

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.

UBISENSE®, the Ubisense motif, SmartSpace® and AngleID® are registered trademarks of Ubisense Ltd. DIMENSION4™ and UB-Tag™ are trademarks of Ubisense Ltd.

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

---

- Multi-tag Configuration** ..... **2**
  - Audience ..... 2
  - Requirements ..... 2
- Configuring Multi-tag** ..... **4**
  - Using Raw Location Events versus Computed Tag Positions with D4 Tags ..... 7
  - What Happens With Tag Association and Tag Disassociation ..... 7
  - Parameters for Multi-tag ..... 8
  - Assertions for Multi-tag ..... 9
  - Tuning Filter Parameters in a Multi-tag Installation ..... 9
- Example Multi-tag in an HMI** ..... **10**





# Multi-tag Configuration

---

In SmartSpace releases earlier than SmartSpace 3.7, only a single tag could be associated with an object at any given time regardless of the number of tag positions defined. Associating a second tag with an object would replace the first tag. With the introduction of the Multi-tag feature, you can associate two or more tags with an object located at given offsets from the object's origin. This enables SmartSpace to more reliably track an object's location and determine its orientation.

Two operational modes are available:

- For DIMENSION4 tags, *raw sensor events* can be used to boost the accuracy of the observations
  - For non-DIMENSION4 tags, the position measurements (x, y, z) of individual tags are used
- x, y, z measurements can also be used with DIMENSION4 tags. However, see [Using Raw Location Events versus Computed Tag Positions with D4 Tags](#) for further discussion of the implications of choosing x, y, z measurements over raw sensor events.

The movement of a multi-tagged object is controlled using one of several motion models (described in the [parameters section](#)). These models should be selected with care to provide a realistic representation of the object's movement in the physical world. To allow tracking of objects with articulation, one or more types can be configured to share the same origin and these can be "joined" to one another by use of the ['Object' is articulation of 'Object'](#) assertion. This can, for example, be used to track the vertical position of a forklift's forks.

An API allows the development of HMIs for tag association, and tags can also be manually associated with objects using the Tag association workspace in SmartSpace Config.

## Audience

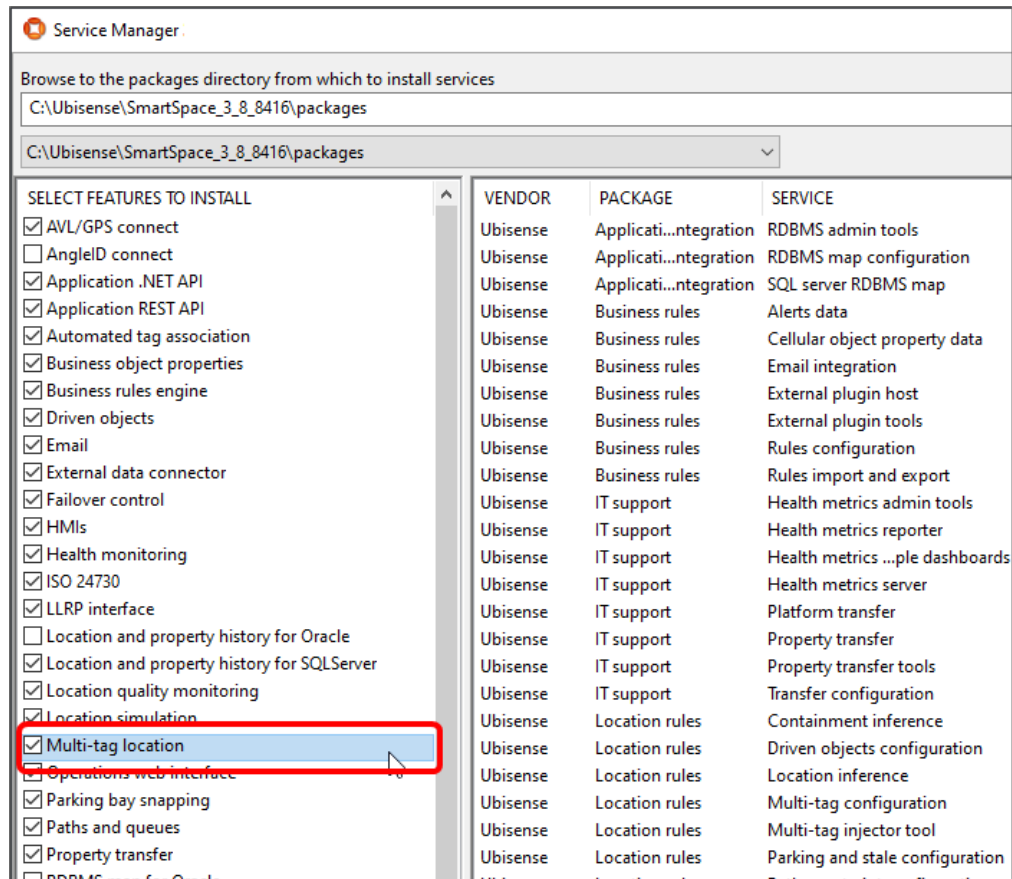
This guide provides guidance for anyone setting up a SmartSpace system to use the Multi-tag feature of Location rules, or those who are maintaining a system that uses this feature.

## Requirements

### SmartSpace

You will need SmartSpace 3.7 SP1 with a license for Location rules.

Click **Install services...** in Service Manager to install service packages. If you have a license, Multi-tag location will appear in the list of features available for you to install:



Configuration of Multi-tag is carried out using the SmartSpace Config application. If you do not have that installed, you can download it using the Ubisense Application Manager after the services have been installed. (See SmartSpace Installation for more information.)

If you want to develop and use HMIs with Multi-tag, you must also have a license for Visibility. See Introduction to HMIs for details on the exact requirements and for information on installing HMIs. An example HMI using Multi-tag is described in [Example Multi-tag in an HMI](#).

If you want to use Health monitoring to monitor multi-tag, you must also have a license for Advanced IT. See Overview of Health monitoring for information on configuring this feature.

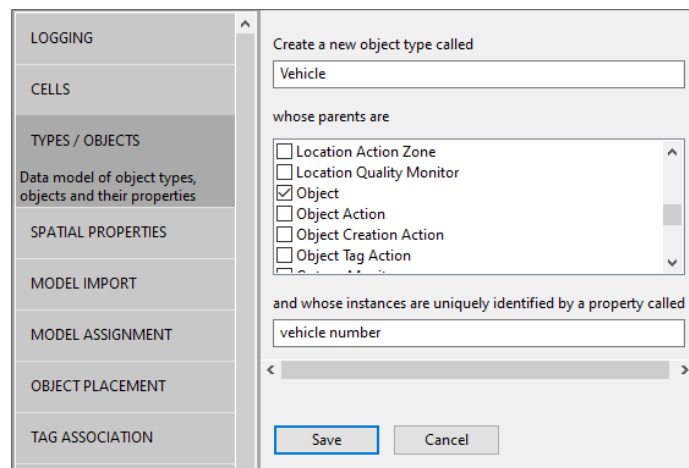
#### DIMENSION4

If you are using DIMENSION4 raw events, you will also need DIMENSION4 version 3.7 or later and you must also set the "Forward to Multitag Service" flag to "True" in relevant tag filters. See the description of the ["Forward to Multitag Service" flag on the DIMENSION4 website](#).

## Configuring Multi-tag

In this example, we will work through creating a vehicle with two tag locations.

1. Create the type we will use to represent the vehicle. In SmartSpace Config, select the TYPES / OBJECTS task, and double-click **<Create new type>**. In the dialog enter the type name "Vehicle" and tick "Object" from the list of parents. Enter "vehicle number" as the unique identification property. Click **Save**.



2. Import and assign representation models for the type. See [Model import](#) and [Model assignment](#) for how to import a model and set a representation on a type in SmartSpace Config. Ensure you scale the model correctly, and set the origin to a suitable position for example a point at the center of the vehicle.
3. Define tag positions. Tag positions are defined as an offset from the origin of an object. Accurate offsets are important for the successful functioning of Multi-tag. We will add two tag positions to the Vehicle type, one near the front and the other near the rear of the vehicle. In SmartSpace Config, select the TAG ASSOCIATION task, and double-click **<Create new tag position>**. In the dialog, enter "Rear" as the name of the position, select the Vehicle type, and enter the x, y and z coordinates for the tag position relative to the type's origin. Here, we have located the rear tag position 1.8 m behind the origin. Click **Save**.



Create a new tag position called

For type

X

Y

Z

4. Repeat to create a "Front" tag position 1.9 m in front of the origin.
5. Now we will set up the parameters for the vehicle. Go to the SERVICE PARAMETERS task, choose Multi-tag, and then drag the Vehicle type into the object browser. In the dialog that displays, double-click '**Vehicle**' objects to set the parameters for all vehicles. Click **Edit** in the parameters window.

We are going to use the steering motion model, which is the most relevant model for a twin-axle vehicle, and will also need to set the wheelbase, maximum velocity, acceleration time and orientation rate.

The "use computed tag positions" parameter is set to false (the default). This means that if we are using DIMENSION4 as a source for our tag location data, we will be expecting to receive raw location data. To make this happen, we must also have the correct tag filter parameter configuration in DIMENSION4: the "Forward to Multitag Service" flag must be set to "True". See the description of the ["Forward to Multitag Service" flag on the DIMENSION4 website](#).

Enter the parameters as shown below and Click **Save**.

Dialog box titled "Edit Multi-tag : 'Vehicle' objects" with the following configuration:

Applies to	'Vehicle' objects
acceleration time (s)	4
maximum measurement innovation	2
maximum uncertainty (m)	0.5
maximum velocity (m)	10
motion model	steering
observation noise multiplier	1
orientation rate (deg/s)	180
use computed tag positions	false
wheelbase (m)	2

Buttons: Save, Cancel

6. Create an instance of Vehicle. In TYPES / OBJECTS, drag the Vehicle type into the object browser, then double-click <Create new object> and enter Car 1 as the vehicle number. Click **Save**.
7. Open the TAG ASSOCIATION task and double-click <Associate tag with object>. Enter a tag number, choose Car 1 as the owner and the Rear tag position. Click **Save**.
8. Repeat for the Front tag position using a different tag number.

Tags associated with objects									
Q <Show only matching items>									
TAG ID.	OWNER	POSITION NAME	X	Y	Z	ACTIVITY	BATTERY	TAG TYPE	
<Associate tag with object>									
00:11:CE:00:00:67:D2	Car 1		0	1.9	0	Inactive	Unknown	Industrial tag (C cell)	
00:11:CE:00:00:67:EE	Car 1		0	-1.5	0	Inactive	Unknown	Industrial tag (C cell)	

Tag Ranges			Tag Ranges allowed for Types			Tag Positions for Types			
TAG RANGE NAME	FROM TAG ID.	TO TAG ID.	TYPE	ALLOWED TA...	TYPE	POSITION NAME	X	Y	
<Create new tag range>			<Assign tag range to type>			<Create new tag position>			
					Vehicle	Front	1.9	0	
					Vehicle	Rear	-1.8	0	

## Using Raw Location Events versus Computed Tag Positions with D4 Tags

By default multi-tagged objects with DIMENSION4 tags, use raw sensor events to determine their locations. When configuring a multi-tagged object, you can choose to force the use of x, y, z tag positions by setting the "use computed tag positions" parameter to "true". If you choose this configuration, then in Location System Config standard tag filters are used to compute the x, y, z location for each tag independently. This carries the risk that tags attached to the same object can, for example, appear to far apart from one another. Using the default raw location events with a suitable tag filter will provide more accurate tracking: for example an incorrect angle from a sighting could be discarded, whilst a valid tdoa could still be used.

## What Happens With Tag Association and Tag Disassociation

With Multi-tag, a single object can have multiple tag associations at the same time. Associating additional tags does not remove current associations (as happened in releases prior to 3.7).

Without Multi-tag, you can assign multiple tags to an object, but they will drive the object independently, and the orientation will not be estimated based on their relative positions on the object. For this reason, it is not recommended to associate multiple tags to an object without installing the Multi-tag feature.

On a SmartSpace installation with a Series 7000 deployment, deleting tag ownership from LEC for a single object with multiple tag associations will result in all ownerships/associations for this object being removed from LEC/SmartSpace Config. Similarly, adding a tag ownership from LEC, for an object that already has one or more tag associations, will replace all existing associations. This is the behavior regardless of whether Multi-tag is installed.

## Parameters for Multi-tag

The following parameters are defined for objects using the multi-tag functionality.

### **acceleration time**

The time in seconds it takes to achieve the velocity given at maximum velocity.

### **maximum measurement innovation**

Number of standard deviations away from the expected sighting before an outlier is discarded. By default, this is 2 and should not need to be changed.

### **maximum uncertainty**

If the uncertainty in the object position exceeds this threshold, no object position update will be generated. This might happen if the observations are particularly infrequent or very noisy, so that they get discarded.

### **maximum velocity**

The maximum velocity at which the object is typically expected to move in m/s. This is a guideline rather than a hard limit.

### **motion model**

This can be:

- *steering*. This model is useful for vehicles such as cars or forklifts and assumes the object has two axles, one of which steers the object, with the movement constraints implied by such a configuration.
- *omnidirectional*. This model is useful for objects that can move in any direction regardless of their current orientation, such as a car in production that is on a rotatable carrier.
- *stationary*. This model assumes the object is mostly stationary with any movement of the tag occurring within a localized area.

### **observation noise multiplier**

Identifies how noisy the location system is where 1 is the noise in a well-calibrated DIMENSION4 system. A number higher than 1 is a noisier location system.

### **orientation rate**

Defines how fast the object can turn in degrees per second.

### **use computed tag positions**

By default, if there is a DIMENSION4 system installed and D4 tags are being used, raw DIMENSION4 events are used to locate the tags. If 'use computed tag positions' is set to true, it overrides the default and uses the x, y, z positions of the tags instead. The default is false, i.e. to use the raw data by preference. For further discussion of the 'use computed tags' parameter, see [Using Raw Location Events versus Computed Tag Positions with D4 Tags](#).

If the tags driving a multi-tagged object are *not* D4 tags, this parameter is ignored and computed tag positions are always used.

In DIMENSION4, by default, raw tag measurement data is *not* sent to SmartSpace. Setting the "Forward to Multitag Service" flag to "True" in a tag filter enables sending of raw tag data. See the description of the ["Forward to Multitag Service" flag on the DIMENSION4 website](#).

#### **wheelbase**

This is the distance in meters between the two axles on an object such as a forklift or other vehicle. Used by the steering motion model.

## Assertions for Multi-tag

### **'Object' is articulation of 'Object'**

Objects can share the same origin and move together along the x- and y-axes. Additionally, one or more of them can move along the z-axis (i.e. up and down). This is how you can, for example, achieve the articulation of a forklift's forks.

For a group of objects to be driven by multi-tag, each object in the group must have at least one tag location defined, and at least one tag must be attached to each object.

## Tuning Filter Parameters in a Multi-tag Installation

The **ubisense\_multitag\_injector** tool is provided to support the tuning of filter parameters. It can be used to replay captured **.ubievent** files through the multi-tag location inference method. To get the tool, run **Application Manager**, open the **DOWNLOADABLES** task, and expand **Location rules/ Multi-tag injector tool**. Select **ubisense\_multitag\_injector.exe** and click **Download selected items**. Optionally, change the download destination directory, and then click **Start download**.

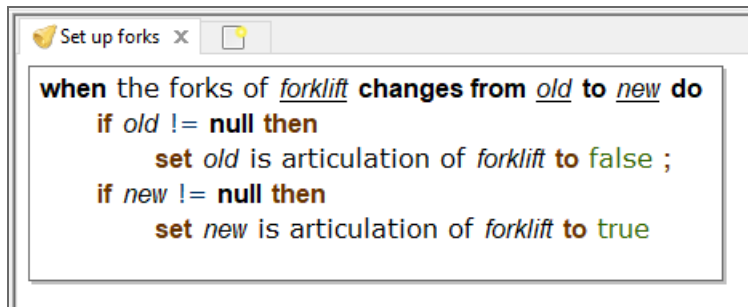
## Example Multi-tag in an HMI

---

We have created type "Forklift", with name property "name", and type "Forklift Forks", with name property "name".

We also created a property "forks<Forklift>: <Forklift Forks> "

We added a business rule so that adding a row to "forks<Forklift>" puts a row in <object> is articulation of <object>.



```
when the forks of forklift changes from old to new do
  if old != null then
    set old is articulation of forklift to false ;
  if new != null then
    set new is articulation of forklift to true
```

We created tag positions "Back" for type "Forklift" and "Forks" for the type "Forklift Forks".

Now the method "associate" shows an example of creating two objects for a forklift/forks combination, and associating them with their tags.

The method "checkTag" gets the status of a given tag ID into the data field "tagStatus".

The rest of the component logic is omitted.

```

var vm = new Vue({
  mixins: [ UbiHMI.mixin({ forklifts: "Forklifts" }) ],
  data: {
    objectName: null,
    tagidFront: null,
    tagidBack: null,
    tagStatus: {},
    statusMessage: "unknown"
  },
  methods: {
    associate: function () {
      // Should do some validation here, check that the forklift name is a valid
      // format, etc.
      if (!this.objectName) return;
      if (!this.tagidFront) return;
      if (!this.tagidBack) return;

      // Create the instances of forklift and forks.
      let forklift = this.createObject("UserDataModel::[Custom]Forklift");
      this.setProperty(forklift, "[Custom]name<[Custom]Forklift>",
this.objectName);
      let forks = this.createObject("UserDataModel::[Custom]Forklift_Forks");
      this.setProperty(forks, "[Custom]name<[Custom]Forklift_Forks>",
this.objectName + "F");

      // Associate the forks as an articulation of the forklift.
      this.setProperty(forklift, "[Custom]forks<[Custom]Forklift>", forks);

      // Associate the two tags using pre-defined tag positions.
      let me = this;
      let tagType = "Industrial tag (C cell)";
      this.tagManager().associate(
        this.tagidFront, forks, "Forks", tagType,
        function (res) { me.statusMessage = res; }
      );
      this.tagManager().associate(
        this.tagidBack, forklift, "Back", tagType,
        function (res) { me.statusMessage = res; me.associated = forklift; }
      );
      this.statusMessage = "sending";
    },

    checkTag: function (tagId) {
      // Get the status of a given tag id "v".
      let me = this;
      let v = tagId;
      this.tagStatus[v] = undefined;
      this.tagManager().getStatus(
        v,
        function (res) {
          me.tagStatus[v] = res;
          // tagStatus[tagId] will now have { Owner, Activity, Battery,

```

```

TagType } fields to be checked before association is allowed.
    },
    function (err) {
        me.tagStatus[v] = { Error: err };
        // tagStatus[tagId] will have { Error: message }.
    }
    );
}
},
watch: {
    'tagidFront': function (v) {
        this.checkTag(v)
    },
    'tagidBack': function (v) {
        this.checkTag(v)
    }
}
});

```

```

var vm = new Vue({
  mixins: [ UbiHMI.mixin({ forklifts: "Forklifts" }) ],
  data: {
    objectName: null,
    tagidFront: null,
    tagidBack: null,
    tagStatus: {},
    statusMessage: "unknown"
  },
  methods: {
    associate: function () {
      // Should do some validation here, check that the forklift name is a valid
      // format, etc.
      if (!this.objectName) return;
      if (!this.tagidFront) return;
      if (!this.tagidBack) return;

      // Create the instances of forklift and forks.
      let forklift = this.createObject("UserDataModel::[Custom]Forklift");
      this.setProperty(forklift, "[Custom]name<
[Custom]Forklift>", this.objectName);
      let forks = this.createObject("UserDataModel::[Custom]Forklift_Forks");
      this.setProperty(forks, "[Custom]name<[Custom]Forklift_
Forks>", this.objectName + "F");

      // Associate the forks as an articulation of the forklift.
      this.setProperty(forklift, "[Custom]forks<[Custom]Forklift>", forks);

      // Associate the two tags using pre-defined tag positions.
    }
  }
});

```



```

    let me = this;
    let tagType = "Industrial tag (C cell)";
    this.tagManager().associate(
        this.tagidFront, forks, "Forks", tagType,
        function (res) { me.statusMessage = res; }
    );
    this.tagManager().associate(
        this.tagidBack, forklift, "Back", tagType,
        function (res) { me.statusMessage = res; me.associated = forklift; }
    );
    this.statusMessage = "sending";
},

checkTag: function (tagId) {
    // Get the status of a given tag id "v".
    let me = this;
    let v = tagId;
    this.tagStatus[v] = undefined;
    this.tagManager().getStatus(
        v,
        function (res) {
            me.tagStatus[v] = res;
            // tagStatus[tagId] will now have { Owner, Activity, Battery,
TagType } fields to be checked before association is allowed.
        },
        function (err) {
            me.tagStatus[v] = { Error: err };
            // tagStatus[tagId] will have { Error: message }.
        }
    );
}
},
watch: {
    'tagidFront': function (v) {
        this.checkTag(v)
    },
    'tagidBack': function (v) {
        this.checkTag(v)
    }
}
});

```

For a description of the tagManager() method, see Tag Management in SmartSpace HMIs.