SmartSpace®

Components and Features

From version 3.8

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

Telephone: +44 (0)1223 535170. Website: https://www.ubisense.com

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

Ubisense SmartSpace® ***1***

SmartSpace Core ***4***

Summary *4*

Type and object definition ***4***

Summary *4*

Site visualization ***5***

Summary *5*

Features *5*

Spatial relationship definition ***6***

Summary *6*

Spatial Property Editor *7*

Spatial Relationship Editor *7*

Spatial Monitoring at Runtime *7*

Service security ***8***

Summary *8*

Features *8*

Site connector ***9***

Summary *9*

Object to tag assignment ***9***

Summary *9*

.NET API ***10***

Summary *10*

C++ API ***10***

Summary *10*

Features *10*

Location store ***11***

Summary *11*

Service management ***11***

Summary ... 11
Logging ... **12**
Summary ... 12
Smart Workers ... **14**
Business rules ... **14**
Business object properties ... **14**
Summary ... 14
Simple Properties ... 14
Complex Properties ... 15
User Interfaces ... 16
Business rules engine ... **17**
Summary ... 17
Users and roles ... **17**
Summary ... 18
Email ... **19**
Summary ... 19
Shifts ... **19**
Summary ... 19
Visibility ... **21**
Business object properties ... **21**
Summary ... 21
Simple Properties ... 21
Complex Properties ... 22
User Interfaces ... 22
Web maps ... **23**
Summary ... 23
Web forms ... **24**
Summary ... 24
Features ... 24
HMIs ... **26**

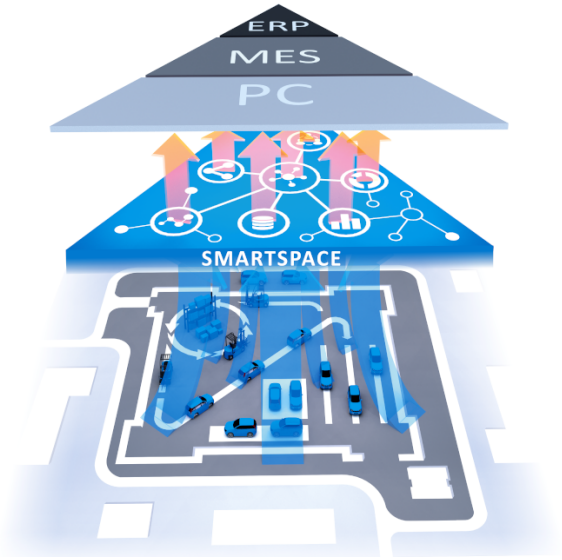
Summary ... 26
ObjectView API ... ***27***
Summary ... 27
Operations web interface ... ***28***
Summary ... 28
Shift Configuration UI ... 28
Manual Tag Association and Disassociation UI ... 28
Automatic Tag Association UI ... 28
Tag Status UI ... 28
Users and Roles UI ... 29
Users and roles ... ***29***
Summary ... 29
Location rules ... ***30***
Driven objects ... ***30***
Summary ... 30
Stale location detection ... ***30***
Summary ... 30
Robust location ... ***31***
Summary ... 31
Parking bay snapping ... ***31***
Summary ... 31
Automated tag association ... ***31***
Summary ... 32
Location removal ... ***32***
Summary ... 32
Paths and queues ... ***32***
Summary ... 32
Multi-tag ... ***34***
Summary ... 34
Benefits ... 34

Room snapping ***34***
Summary *34*
Location quality monitoring ***35***
Summary *35*
Benefits *35*
Reporting ***37***
Business object properties ***37***
Summary *37*
Simple Properties *37*
Complex Properties *38*
User Interfaces *38*
Property history ***39***
Summary *39*
Web reports ***40***
Summary *40*
Users and roles ***41***
Summary *41*
Applications integration ***42***
Business object properties ***42***
Summary *42*
Simple Properties *42*
Complex Properties *43*
User Interfaces *43*
RDBMS map ***44***
Summary *44*
Application .NET API ***45***
Summary *45*
Application REST API ***45***
Summary *45*
RFID integration ***47***

AngleID connect ***47***
Summary *47*
LLRP interface ***48***
Summary *48*
RTLS integration ***49***
External data connector ***49***
Summary *49*
ISO 24730 ***49***
Summary *49*
Advanced IT ***50***
Failover ***50***
Summary *50*
Benefits *50*
Replication ***51***
Summary *51*
Property transfer ***52***
Summary *52*
Health monitoring ***52***
Summary *52*
Benefits *53*
SmartSpace Developer ***55***
Rules engine developer ***55***
Summary *55*
Features *55*
Real-time rules engine ***56***
Summary *56*
Location simulation ***56***
Summary *56*
Features *57*
Benefits *57*

Reports engine developer ***58***
Summary *58*

Ubisense SmartSpace®



Ubisense SmartSpace® is a modular software platform that manages real-time location and identification data from multiple sources to support industrial-scale mission-critical visibility and control.

SmartSpace makes it easy to build, deploy and manage software applications that can detect and respond to real-world physical interactions between people, things, and the environment. The modularity of SmartSpace means that users can license only the functionality they will need for their specific location-aware applications.

To facilitate building sensor-rich, industrial-scale implementations, SmartSpace is fully sensor-independent, meaning it can take in and handle location and identification events from almost any data source, including third-party sensors, third-party tag readers, barcode or other software systems (such as RDBMS, ERP or MES).

Scalability and real-time performance lie at the core of the SmartSpace architecture, allowing users to build applications that scale from micro-installations (small control systems running on a single machine) through to large sites with several servers and thousands of sensors, devices, people, etc., all with predictable performance.

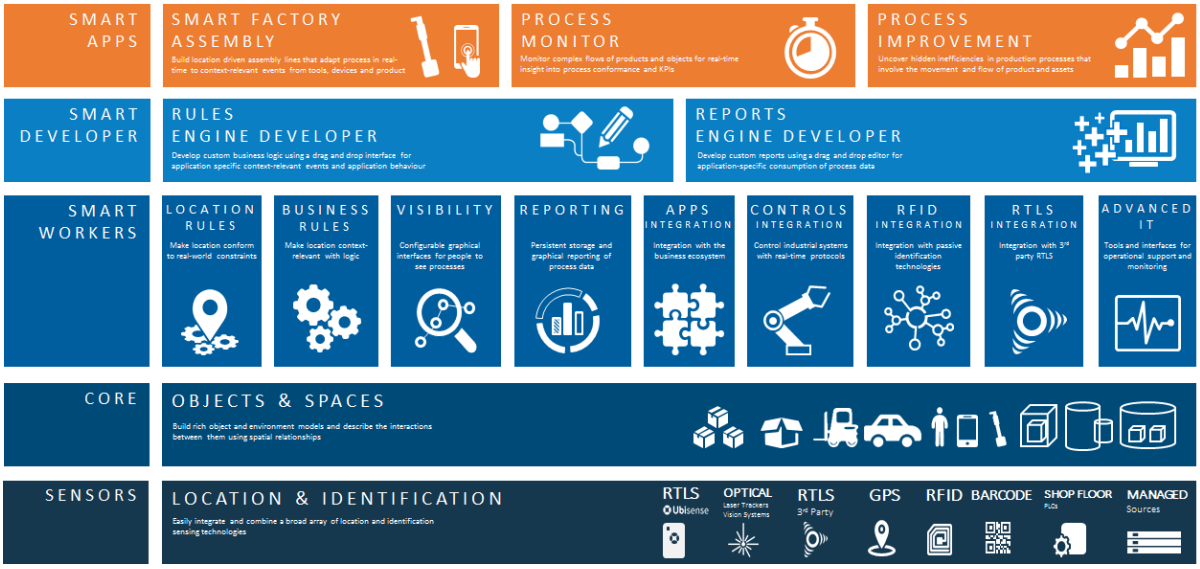
Modular Platform

SmartSpace is divided into a set of functional blocks that can be plugged together in different ways to create new types of location-aware business applications. Each component supports a number of different features that are common to the theme of that component. For example, the Location rules component provides application-level control over object location, giving you options to constrain object position via models of the environment, driving object location via the location of other objects, or detecting important location state for objects that can drive business process.

This document introduces the SmartSpace components and provides detailed descriptions of the available features.

Components are divided into three major layers:

- Core
- Smart Workers
- Smart Developer



- SmartSpace Core is a fundamental component for all SmartSpace applications, enabling the building of rich object and environment models and the description of the interactions between them.
- Smart Workers consume the object data provided by the Core to generate business activity events and deliver information to business systems and users through integration paths and graphical interfaces. Smart Workers are also used to integrate third-party sensor systems such as barcode, RFID and RTLS.
- Building new business logic or historical reports is enabled through the SmartSpace Developer, allowing SmartSpace customers to fully customize or extend defined functionality or create new application behavior and reports.
- Smart Apps are preconfigured implementations of SmartSpace for specific industries and applications. Smart Apps are not described further in this document.

SmartSpace Core

The fundamental services needed for storing, managing and making sense of location data on a large scale in real time

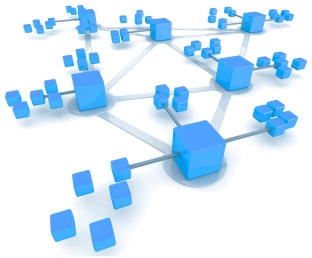
Summary

Successful development of scalable real-time location-aware applications is made possible using SmartSpace Core's:

- Object modeling
- Scalable location store and event management
- Spatial relationship detection
- APIs

Type and object definition

Tools and a framework to build and manage application-specific data models including types, objects, names, and spatial properties



Summary

In SmartSpace, *object types* are used to model real-world entities. An object type has *properties* (sometimes also called *attributes*), which reflect the entity's structure. Properties are defined using *built-in types* or other object types. Creating a collection of types and assigning them properties that are relevant to an application is called *data modeling*.

The first level of data modeling in SmartSpace is type and object definition. This feature allows application developers to define new types (such as product, process, trolley, forklift) and give them two fundamental properties: a *name* and optional spatial properties (or *spaces*).

When building a new application, the name property is used to give every object instance of a type a unique identifier, while spaces are used in spatial relations between object types (see *Spatial relationship definition*). Where an object type is involved in more than one spatial relationship, it is possible to give the type multiple different spaces to match each desired spatial relationship.

With the licensing of *Business object properties*, user-defined types can be created with additional properties. These can be any of the basic numeric or string types as well as user-defined types already created within the data model. Additionally more complex relationships between properties can be defined.

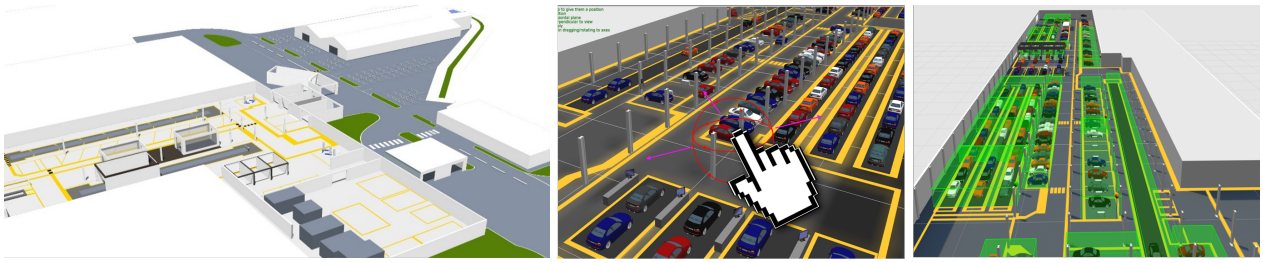
The SmartSpace configuration client includes a types and objects creation GUI.

Site visualization

Import 2D and 3D graphics into an easy-to-use configuration tool to visualize models of objects and buildings in a virtual representation of the real world

Summary

Building location-aware applications with SmartSpace is all about modeling the behavior and interaction of real-world objects in a real-world environment. The task of building and testing such applications is greatly simplified when the types and objects involved can be visualized and manipulated in a virtual representation of the real world.



To this end, SmartSpace supports a site visualization feature that renders a graphical model of the environment and the application-specific objects within it. This is enabled by importing simple graphical elements (both images and 3D models) that can be assigned to object types or used as a backdrop for the site. These image primitives are called *representations* (or reps) in SmartSpace and can be arbitrarily attached to objects or used as layers to construct a "scene" representative of the real-world environment.

Features

Site visualization includes the following capabilities:

- Importing standard graphics formats, including vector (SVG), raster (JPG, BMP, PNG), and 3D models (DAE) into a SmartSpace reps library

- Setting the scale and offset of all reps to match the scale and coordinate system of the application environment
- A scene visualization GUI which renders the environment scene and all located objects which have a representation assigned
- The ability to drag, drop, position and lock "scene reps" in the site visualization
- The ability to assign reps to object types so they appear in the site visualization
- The ability to move objects "virtually" by using the mouse
- The ability to visualize spatial relationships and containment events

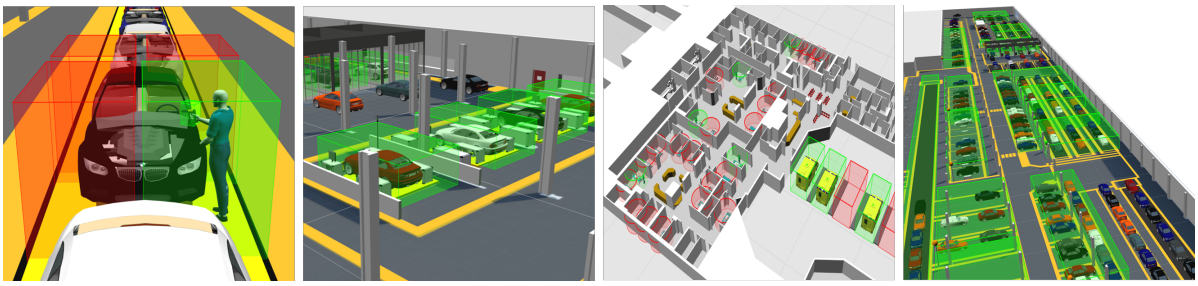
Spatial relationship definition

Define and monitor physical relationships between real-world objects via the interaction of their spatial properties

Summary

Object types can be assigned one or many spatial properties (3D spaces) in order to represent important geometric characteristics of the object. These could be fixed spaces around a set of workstations or 3D zones which move with the object wherever it goes, i.e. when the location of the object changes the location of the space changes with it.

Spatial relationships indicate the containment relationship between pairs of spaces. When the spaces of two different objects come together so that the 3D extent of one space completely encloses the 3D extent of the other, this fully-enclosed state between the two objects causes their containment relationship to become "true". When the objects move such that the 3D extents of the spaces no longer intersect at all, the containment relationship becomes "false". This "positive-containment, negative-overlap" behavior allows containment relationships to be stable when objects move about right on the edge of containment, avoiding multiple triggering of behavior.



Where an application has chosen to subscribe to events from a particular spatial relation, containment events provide the application with a very reliable, real-time handshake between two objects which can be used to drive connected systems and processes. Spatial relations turn locations (most of which are not interesting to business systems) into software-defined identifications, triggered by the spatial interactions between objects. These significant changes in state will only be triggered by a small fraction of sensed object locations.

Spatial Property Editor

SmartSpace provides an easy-to-use configuration client for defining the 3D geometries of the spaces owned by types and objects. This is a drag-and-drop interface which allows the user to define simple polygons around objects using mouse clicks.

Spatial Relationship Editor

The SmartSpace configuration client exposes an interface for defining which containment relations are actually of interest for the application. There may well be many possible containment permutations that are not actually of interest the application being configured, so it is necessary to explicitly request monitoring of the relationship between two spatial properties.

Spatial Monitoring at Runtime

At runtime the spatial relations that have been defined and the interaction events that are detected are available as real-time events for consumption by higher-level components (such as *Business rules*) or to be forwarded to external systems through one of the SmartSpace integration APIs or interfaces.

The spatial monitoring feature of SmartSpace is important as it acts as a filter between the noisy, very high bandwidth, low latency location data being generated by underlying sensor systems and the slower response, lower bandwidth information systems SmartSpace connects to. By turning high volumes of real-time location events into a few reliable and actionable business- and context-specific events, spatial monitoring converts sensed location data into a form that business systems can use.

Service security

Multi-level role-based access control for the basic Ubisense services

Summary

SmartSpace implements a service-oriented architecture where each service is a unit of functionality that works with other services to deliver some larger capability. Service security allows each service to be protected so only authorized users and clients can access the data or execute operations to modify it.

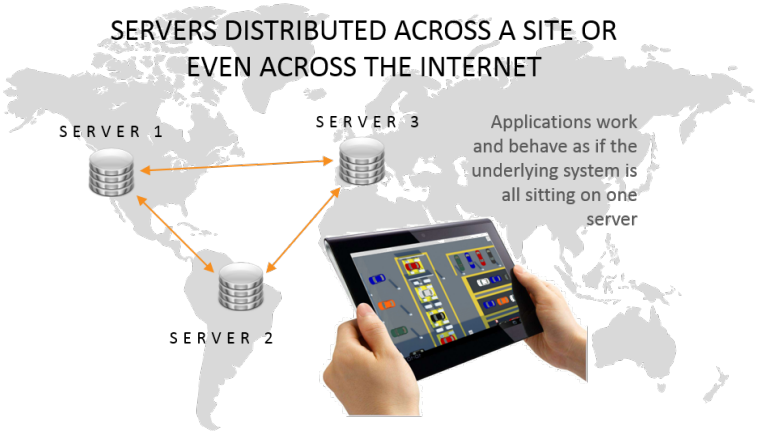
Features

Key features include:

- Protection of system state at the service level for security and data integrity
- Three service access levels: *Open*, *Read-only*, *Closed*
- Configurable role-based multi-level security model
    - Each (Role, Service, Cell) combination maps to a service access level
    - Uses AES-128 (Advanced Encryption Standard)
- Client credential entry integrated with Windows authentication

Site connector

Connect Ubisense platforms on separate networks using TCP/IP, either for multi-site operation, or for support purposes



Summary

Site connector has two related uses.

It can connect together distributed computers on disjoint networks, via a TCP/IP connection, so they appear to be connected to the same Ubisense platform. In this role, Site connector has been used to connect production sites around Europe to a single central platform.

Site connector can also be used to connect client computers to a platform when the network configuration prevents direct connection, such as for remote access support via a VPN.

Object to tag assignment

Manually assign tags to objects using the SmartSpace configuration client

Summary

Tags can be assigned to objects using the SmartSpace configuration client. The offset of the tag from the origin of the object can be defined per type. For example, if the tag is attached to a car's windscreen the offset from the origin can be set for cars.

Tags can be assigned to objects manually, one object at a time. The tag identifier must be typed in by hand and the object and tag position selected from drop-down lists.

Valid ranges of tags can be specified and associated with object types. An attempt to assign a tag which is outside the valid range for the object's type will result in an error.

.NET API

Exchange configuration and real-time data with third-party systems

Summary

The .NET API is a client-side library that uses the proprietary Ubisense protocols to communicate directly with the SmartSpace Core services. It supports the development of programs to:

- Change or retrieve SmartSpace configuration information, such as names, spatial properties, cell hierarchy, service management
- Exchange real-time data with SmartSpace, for example receiving or injecting real-time-location events or spatial relationships

C++ API

Exchange configuration and real-time data with third-party systems

Summary

The C++ API is a client-side library that uses the proprietary Ubisense protocols to communicate directly with the SmartSpace Core services.

Features

The C++ API supports the development of programs to:

- Change or retrieve SmartSpace configuration information such as names, spatial properties, cell hierarchy, service management
- Exchange real-time data with SmartSpace, for example receiving or injecting real-time-location events or spatial relationships

The C++ API is available for Windows (Win32) and Linux applications.

The C++ API uses the same code that has been proven in large-scale mission-critical real-time control systems for over a decade. It is only recommended for users familiar with C++ programming because it requires an understanding of important C++ principles such as RAII.

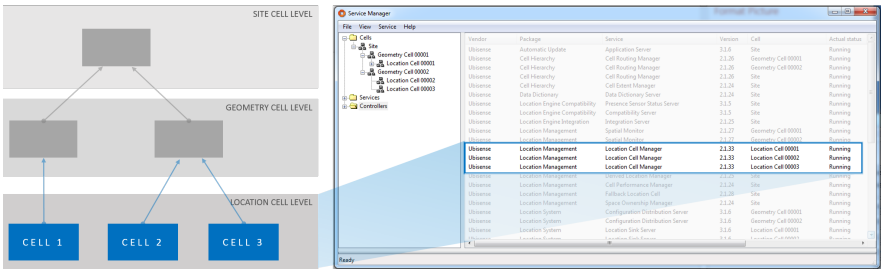
Location store

Persistent storage and real-time distribution of location data

Summary

Location store is a distributed, persistent store for location and orientation of objects: it is designed to support high event rates (for example, for fast-moving objects) and reliable storage of location data (for example, for objects whose position is only ever asserted once).
Location store has been proven in mission-critical industrial applications, handling several hundred million individual location events per day and providing reliable persistent storage.

Location store can scale to cover an arbitrarily-large area by splitting it up into multiple cells, each of which is managed by a separate *location cell manager* service.



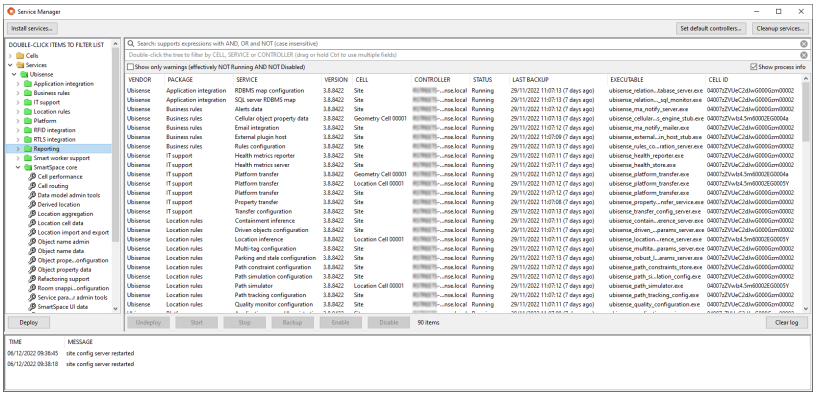
Service management

Centralized control of the Ubisense Service-Oriented Architecture

Summary

SmartSpace uses a Service-Oriented Architecture. Each SmartSpace installation comprises a few dozen separate services that map onto executable programs on a single machine or a set of communicating machines. This provides scalable real-time performance. Service management provides centralized control for the set of services in a system. It supports service installation and upgrade, assignment of services to machines, and service backup.

Service management provides a status screen running on the Windows operating system showing real-time status of all services along with support for installing, upgrading and removing services.



Logging

Cellular service-based logging of trace messages and DIMENSION4 location events

Summary

The logging services are designed for always-on and scalable logging of the SmartSpace system. They can be configured to run on a dedicated logging server, or be split between multiple logging servers for maximum scalability. The services ensure that only a configured maximum disk space on each logging server is used. Trace messages are logged from all Ubisense programs. Location messages are logged from DIMENSION4 sensors. The logging services support listening in real time and the retrieval of historical messages. There are GUI components supporting these features for both trace messages and location events and a command-line utility for retrieving trace messages for use with standard command-line text manipulation tools.

Smart Workers

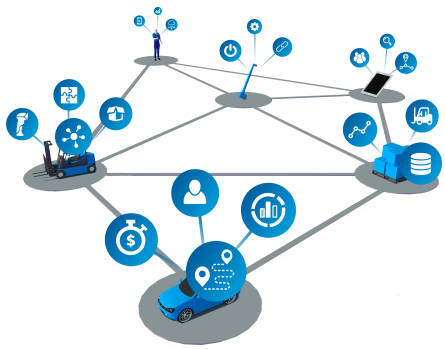
Optional extensions that are used to enrich the SmartSpace model of the environment

Business rules

Transform object location data into business intelligence, by integrating spatial facts about objects with their business properties, detecting important events and alerting the right people

Business object properties

Define data models to support context-aware business applications, tightly integrated with location data



Summary

Business object properties provides the following capabilities:

- User-extensible definition of functional and relational properties of types
- Creation and deletion of properties
- Query, creation, update and deletion of property assertions
- Query, creation, editing and deletion of 3D spatial properties of objects and spatial relationships between objects

SmartSpace allows the definition of relationships between objects by using either *Simple Properties* or *Complex Properties*.

Simple Properties

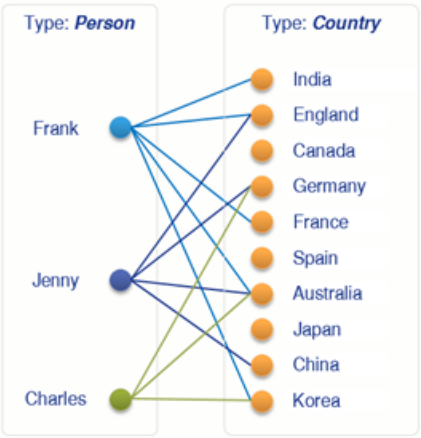
SmartSpace enables developers to flexibly create the types necessary to define a new application data model. User-defined types are created with appropriate properties, which can be any of the basic numeric or string types as well as user-defined types already defined in the data model. In SmartSpace, these are called *Simple Properties*.

Types which represent real-world objects can include one or more *Space Properties*, which are the geometric extents (zones) defined around the object that move and interact with the spaces of other object types in the data model when objects are given a location.

Additionally, multiple inheritance is fully supported, so that new types can be created from a combination of parent types which own the properties required.

Complex Properties

By using Simple Properties, one-to-one and many-to-one relations between objects can be specified. *Complex Properties* allow the inclusion of joins between types so that *one-to-many* or even *many-to-many* relations can be defined. This is really valuable when it becomes necessary to store or operate on sets of data, or some conditional fact must be asserted based on the state of multiple objects in the data model.



Where a relation between types is a fact, whether it is either true or false, the relation must be qualified by some underlying logic. This relation must be asserted either by an integrated third-party business system or through the SmartSpace Business rules engine (whose logic is implemented using the Rules engine developer).

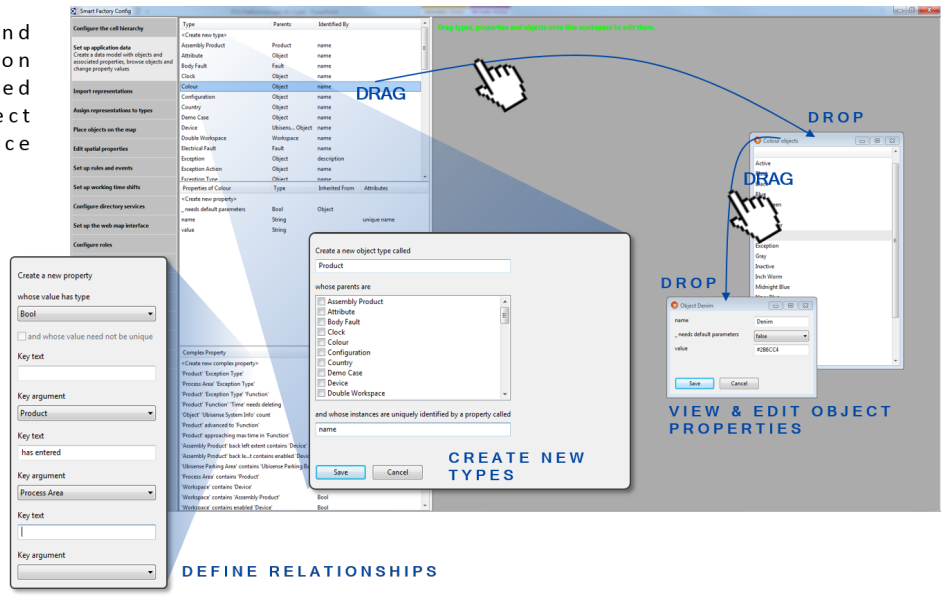
User Interfaces

The *Type and object definition* feature includes a full development and object explorer thick-client GUI, which provides a drag-and-drop interface for creating new types, instantiating new objects, and interrogating and editing object properties and states.

Data modelling and relationship definition through a detailed property and object browser interface

View, modify, delete any object, property or relationship

See changes to the underlying object model happen in real-time.



Business rules engine

Execution of

configura ble rules and event handlers to implement business logic

Summary

The Business rules engine includes the features necessary to encode a business process. It includes support for calculating derived properties of objects, enforcing constraints on data, setting timeouts, taking actions when timeouts expire, detecting error cases and raising alerts to users. Alerts can be routed to appropriate users using the inbuilt multi-level role definition, and also displayed graphically to a logged-in user if the Visibility component is licensed.

Users and roles

Manage user authorizations and integrate with external user and role definitions in LDAP

Summary

Users and their *roles* can be defined within SmartSpace or imported from external systems via LDAP.

SmartSpace roles control which parts of the web interface can be accessed by a user, and the maps, reports and other screens available to them. It also determines which users should receive notifications, emails and alerts generated through the Business rules engine.

This allows the application experience to be tailored to meet the needs of specific classes of end user.

The SmartSpace configuration client includes a workspace for defining roles and assigning users to them. In addition SmartSpace provides a browser-based view of the underlying users and roles model, giving system administrators simple access for editing or extending the underlying user and group allocations.

Email

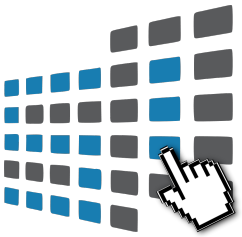
Send real-time notifications of platform alerts to specific end users

Summary

Email integration is configured to connect to common email systems and can be used to deliver notifications, such as business rules alerts, as they are raised by SmartSpace.

Shifts

Support the concept of working and non-working time within the business rules



Summary

Within the Business rules it is useful to differentiate between working and non-working time. For example, when measuring the time a process takes, only working time should be measured; non-working time, such as evenings and weekends, should not be included in the process time.

The working time can be defined in the SmartSpace configuration client. It is built up from the working hours in a single 24-hour period which can be assigned to days of the week in a named shift pattern.

The working time is then used by the Business rules engine to calculate the differences between two times in working hours, for example how many hours are worked between 16:30 Monday and 10:00 Tuesday, or to calculate times in the future in working hours, such as what time is four working hours after 17:00 Friday.

If the Visibility component is licensed, the currently configured working- and non-working time used by the Business rules engine can be viewed in a browser-based interface.

Some users can also be given access to change the working time through the web-site according to their SmartSpace roles. For example all workers may be able to view the working time, but only team leaders are allowed to change the working times.

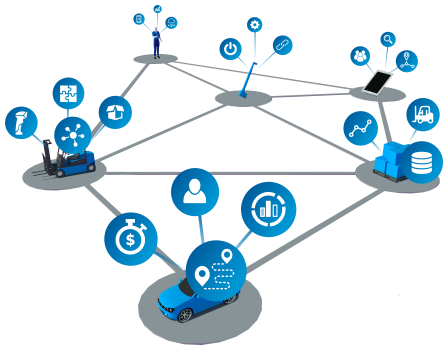
It will vary from place to place who, if anyone, is allowed to change working times through the website. For example, in some factories team leaders can extend their shifts and use overtime.

Visibility

Make location and business data visible via web browsers, using configurable personalized maps and forms to provide users with relevant graphical data

Business object properties

Define data models to support context-aware business applications, tightly integrated with location data



Summary

Business object properties provides the following capabilities:

- User-extensible definition of functional and relational properties of types
- Creation and deletion of properties
- Query, creation, update and deletion of property assertions
- Query, creation, editing and deletion of 3D spatial properties of objects and spatial relationships between objects

SmartSpace allows the definition of relationships between objects by using either *Simple Properties* or *Complex Properties*.

Simple Properties

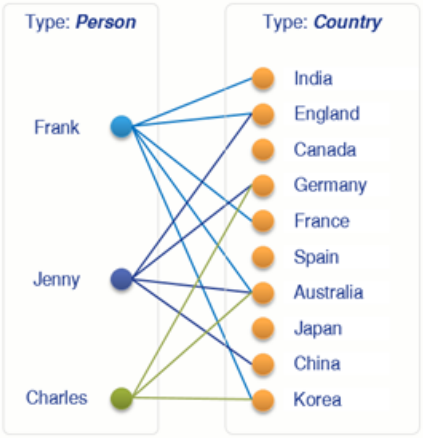
SmartSpace enables developers to flexibly create the types necessary to define a new application data model. User-defined types are created with appropriate properties, which can be any of the basic numeric or string types as well as user-defined types already defined in the data model. In SmartSpace, these are called *Simple Properties*.

Types which represent real-world objects can include one or more *Space Properties*, which are the geometric extents (zones) defined around the object that move and interact with the spaces of other object types in the data model when objects are given a location.

Additionally, multiple inheritance is fully supported, so that new types can be created from a combination of parent types which own the properties required.

Complex Properties

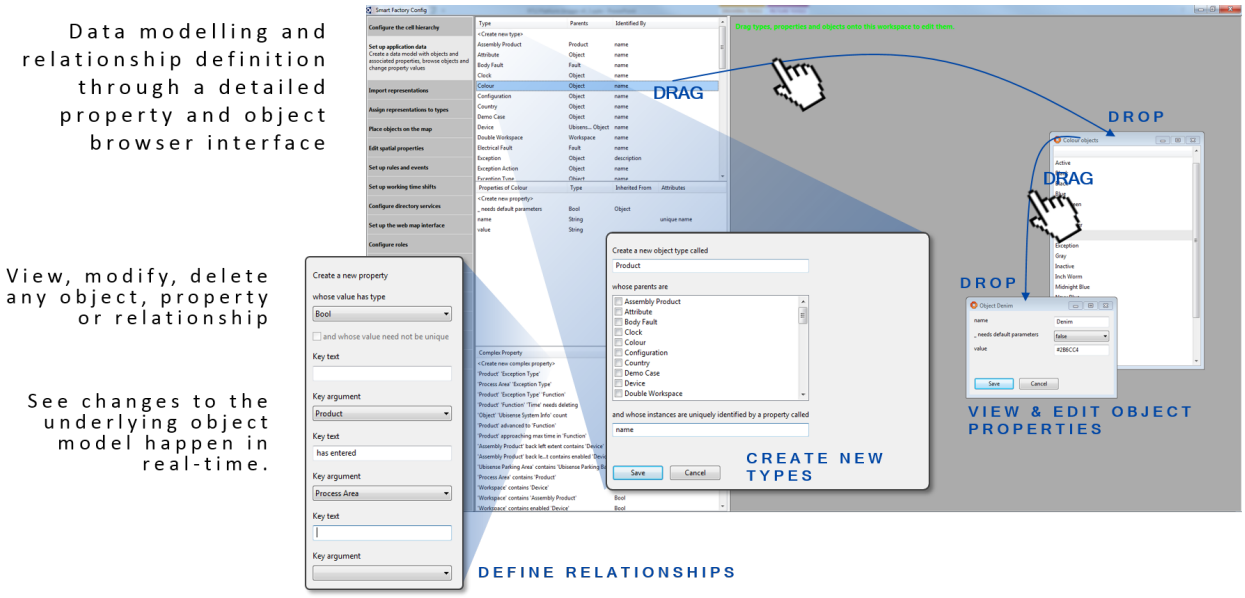
By using Simple Properties, one-to-one and many-to-one relations between objects can be specified. *Complex Properties* allow the inclusion of joins between types so that *one-to-many* or even *many-to-many* relations can be defined. This is really valuable when it becomes necessary to store or operate on sets of data, or some conditional fact must be asserted based on the state of multiple objects in the data model.



Where a relation between types is a fact, whether it is either true or false, the relation must be qualified by some underlying logic. This relation must be asserted either by an integrated third-party business system or through the SmartSpace Business rules engine (whose logic is implemented using the Rules engine developer).

User Interfaces

The *Type and object definition* feature includes a full development and object explorer thick-client GUI, which provides a drag-and-drop interface for creating new types, instantiating new objects, and interrogating and editing object properties and states.



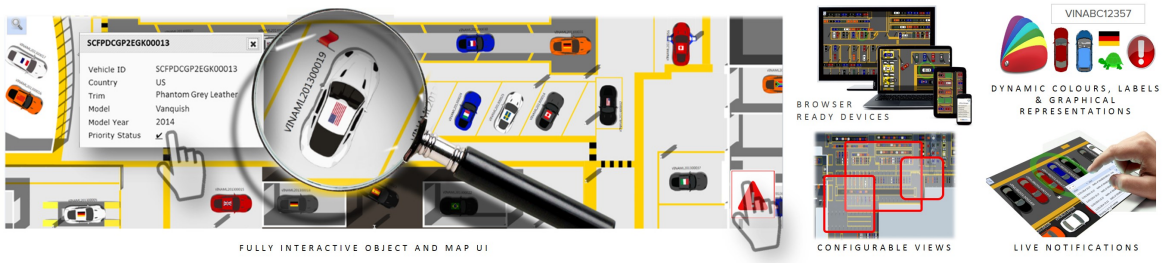
Web maps

Real-time views of the locations of tracked assets, with role-specific searches, representations, details and alerts

Summary

The web map is a browser-based interface that displays the current locations of tracked assets. It supports role-specific searches that can be configured to the needs of specific types of end user. Each search can be set up with appropriate representations of assets, including labels and annotations based on *Business object properties*. Searches also return textual data about assets, both in summary and detailed form. The map can also display current raised alerts relevant to the user.

Web maps can be accessed from a wide variety of client devices, ranging from desktop computers to tablets and phones, and do not require installation.



Web forms

Customizable screens showing role-specific lists and forms based on current data objects, types and business object properties

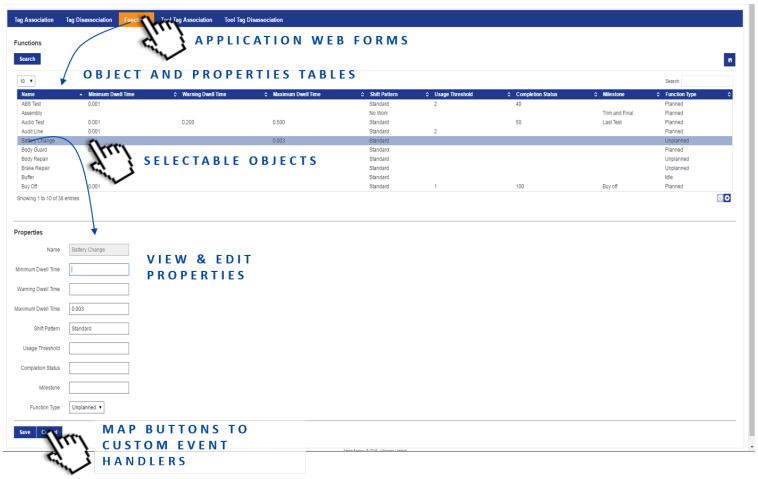
Summary

Web forms are a browser-based interface that display textual data from the Ubisense data model to end users, and optionally allow the entry of data.

Features

The set of interface screens shown to a user is configurable based on their role memberships. Similarly, users can be allowed to enter data into these screens using configurable forms, and this data is written into the data model. The screens update dynamically, and columns and rows can be also be colored based on data.

Web forms can be accessed from a wide variety of client devices, ranging from desktop computers to tablets and phones, and do not require installation.


APPLICATION WEB FORMS
OBJECT AND PROPERTIES TABLES
SELECTABLE OBJECTS
VIEW & EDIT PROPERTIES
MAP BUTTONS TO CUSTOM EVENT HANDLERS

HMIs

Custom interfaces for displaying current business objects and properties in the SmartSpace website

Summary

HMIs (Human Machine Interfaces) allow custom web interfaces to be developed and deployed within the Ubisense SmartSpace web site. They can display content based on web searches and reporting queries defined within SmartSpace in a variety of formats including:

- visual components such as progress bars or gauges
- embedded versions of web maps or web reports (if licensed)

Administrators can use a web-based editor to develop and test interfaces, before publishing the finished versions to designated SmartSpace roles.

A simple declarative binding API is provided to allow HMI content to be generated using only attributed HTML; whilst, for more advanced uses, CSS and JavaScript can be added to the interfaces and external assets, such as images and script libraries, can be hosted in the SmartSpace web site for use by the feature.

ObjectView API

Provides access to SmartSpace data using custom website code

Summary

The ObjectView API provides convenient access to the SmartSpace user data model from custom website code. It allows *views* to be configured based on specific object types defined within SmartSpace, containing the values of simple and complex properties of those objects. Clients can access these views, either by subscribing to them and receiving updates pushed from the server, or by retrieving the view contents for a specific object. Access to views is controlled by assigning them to roles within SmartSpace.

The ObjectView API uses SignalR to push view updates to clients, using WebSockets, if available, or falling back to long polling. This means that client applications can receive low-latency notification of data model changes, and react to present them quickly to users.

The ObjectView API also allows special properties to be configured for a view, including:

- associated tags
- configured representations
- external object names
- spatial extents

A special view can also be configured containing the current location of an object, with specialized features designed to allow this to scale well with high update rates.

Operations web interface

Web user-interfaces for the operation of applications and maintenance of tags in the Ubisense location system

Summary

Operations and maintenance users can use browser-based interfaces to perform or monitor associations, configure application users and working time, and perform periodic tag maintenance such as battery changes.

Operations web interface provides a browser-based interface to the following areas of operations:

Shift Configuration UI

Users can update and modify working time schedules through a browser-based calendar and shift pattern editor. Shifts and working time are immediately updated in the underlying data model.

Manual Tag Association and Disassociation UI

Manual tag association requires the operator to enter a tag identifier and the name of an object. This can be done by typing them in manually or using a barcode scanner. The interface can also be configured to create new objects at tag association time.

Automatic Tag Association UI

The user can see the status of automatic tag association. This is useful if an operator is attaching a tag to an object, say a car, and wishes to know whether automatic association has happened correctly.

Tag Status UI

The user can see the status of all tags, including whether they are owned, i.e. assigned to, an object, whether they are currently active, and the status of the battery (whether it is OK or

running low). The user can also reset the tag battery status after the battery has been replaced by an operator.

Users and Roles UI

A browser-based view of the underlying users and roles model, giving system administrators simple access for editing or extending the underlying user and group allocations.

Users and roles

Manage user authorizations and integrate with external user and role definitions in LDAP

Summary

Users and their *roles* can be defined within SmartSpace or imported from external systems via LDAP.

SmartSpace roles control which parts of the web interface can be accessed by a user, and the maps, reports and other screens available to them. It also determines which users should receive notifications, emails and alerts generated through the Business rules engine.

This allows the application experience to be tailored to meet the needs of specific classes of end user.

The SmartSpace configuration client includes a workspace for defining roles and assigning users to them. In addition SmartSpace provides a browser-based view of the underlying users and roles model, giving system administrators simple access for editing or extending the underlying user and group allocations.

Location rules

Use real-world information to process raw location information to improve its quality and to detect relevant business information

Driven objects

Track and locate objects without RTLS tags in real time

Summary

Driven objects allows objects which do not have an RTLS tag to be given locations by associating them with an object which does have a location.

There are a number of use cases.

1. Crates can be associated with a trolley or forklift by using RFID or a barcode scanner. The crates are then given a position by the trolley or forklift and move with it.
2. Crates can be associated with a storage area by using RFID or a barcode scanner. The crate is then given a location within the storage area. It will not be the exact location of the crate but it will show that the crate is in the area and will allow an operator to locate it.
3. Objects which do have RTLS tags can be located in areas with no RTLS sensor coverage. For example cars can be parked in a warehouse where there is no coverage. The car will be given a position, which is not exact, but which allows the operator to know the car is located somewhere in the warehouse.

Stale location detection

Monitor streams of location events for tagged objects and detect out-of-date location data

Summary

Stale location detection can be configured, for any individual object, with a maximum delay between expected location events for that object; it monitors the location event stream for configured objects, detects when objects are no longer receiving up-to-date location information, and asserts that they are *stale*. These stale assertions can then be used by third-party

systems, via the *.NET API* or the *C++ API*, or by logic configured using the *Business rules* component (if the Business rules component is also licensed).

Robust location

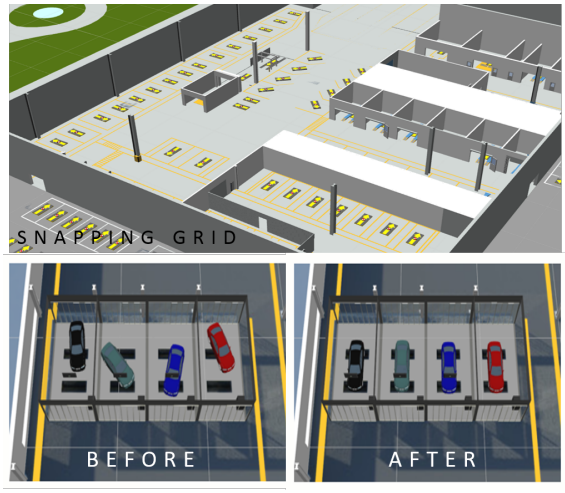
Detection of an object's location at a given place using strong evidence, including distance and speed, to be robust to transient process errors

Summary

An assertion point detects when an object is located at a given place using strong evidence, including distance and speed. This allows the point to be robust to transient process errors in production, such as carrying assets close to a gate location.

Parking bay snapping

Definition of positions in which objects are parked, real-time detection of parking, and snapping and orientation of objects into parking spaces



Summary

Parking bay snapping controls the position and orientation of objects when they stop at user-configured parking bays. It is configured by placing parking bays, grouping them into *parking areas*, and setting parameters on the parking areas to control parking detection. When Parking bay snapping detects that an object is parked in a bay, it snaps it to the bay, orients it to line up to the bay and asserts that it is parking in the bay. These *parking assertions* can then be used by third-party systems, via the *.NET API* or the *C++ API*, or by logic configured using the *Business rules* component (if the Business rules component is also licensed).

Automated tag association

Associate tags to products using location data from tags

Summary

In production environments, if a product is to be tagged its tag is normally attached early in the product's lifecycle and subsequently removed when the product is completed. Automated tag association allows the user to place and configure *association objects* and *disassociation objects*. When a product and a tag are both positioned at an association object, the tag is associated to the product; similarly when the tagged product is positioned at a disassociation object, its tag is removed. This provides a way of managing tag-to-product association that is much less error-prone and more efficient than a manual configuration process. Automated tag association has been used for many years in large-scale production environments to associate and remove hundreds of tags every day.

If the *Visibility* component is also licensed, SmartSpace provides a browser-based interface to display automated association status data.

Location removal

Provides a mechanism by which the current location of an object is removed

Summary

The location removal feature is similar to tag removal and object deletion supported via assertions in SmartSpace core. Location removal allows an assertion to be made that causes the current location of the object to be removed, and the assertion will be reset once this has been done.

Paths and queues

Definition of fixed paths along which objects travel, snapping of located objects to paths, and the application of constraints to form queues

Summary

The purpose of Paths and queues is to allow you to introduce prior knowledge of object locations into SmartSpace, in particular when objects travel along fixed paths.

With Paths and queues in control, objects will be snapped to nearby paths as tags move around. Configuration intended to match the real-world, physical, immutable constraints of the objects being modeled, such as speed and separation, ensures that objects form queues along the paths.

The object sequence information in your application can be used to report things like “number of vehicles in front”.

Multi-tag

Tracking and location with multiple tags per object

Summary

The Multi-tag feature supports the tracking of objects to which two or more tags have been attached using pre-defined motion models in SmartSpace or DIMENSION4 installations. The measurement source can be either the position measurements of individual tags, or in DIMENSION4 systems, the raw sensor events can be used. Where available, measurement sources can be combined, with some tags using raw events and some using positions.

Benefits

Multi-tag tracking can be useful in a number of situations:

- Forklift tracking. The tracking environment for forklifts may be a canyon-like warehouse. Orientation, and possibly height, is used to record the location of products. Multiple tags can be attached to a forklift, even to the forks, to accurately track location and alignment. Additionally, multi-tag tracking on forklifts can aid in safety and the avoidance of collisions, for example by providing forklift locations to other vehicles.
- Object tracking where orientation is not always constrained by location, and the application requires correct orientation tracking in this unconstrained case.
- AGV (autonomous guided vehicle) tracking, where the tracked object may be able to rotate freely, or may have a single steering axis like a car or forklift.
- Robust tracking through diversity of tag mounting points, especially in an environment where there may be obstruction of tags in some positions and orientations.

Room snapping

Consistently snap object locations within rooms

Summary

Room snapping consistently locates objects within defined areas called *rooms* by snapping them a defined distance inside the room's boundary until the object has completely left the room. It allows a clean exit through *doorways*—areas at the perimeter of rooms where snapping is

disabled. Rooms and doorways exist within buildings which can be defined to reflect the physical layout of a site.

Room snapping eliminates through wall error where tagged objects close to the edge of a defined area are incorrectly located.

Location quality monitoring

Analyze the accuracy and robustness of location systems

Summary

A key requirement for SmartSpace is to be able to analyze the accuracy and robustness of the location systems that feed location data into the system. With Location quality monitoring various location quality monitors can be set up across a deployment, and each can generate health metrics.

Monitors can also raise sensor errors when the number of detected issues within some time interval exceeds a threshold.

Benefits

Location quality monitoring is useful in several contexts:

- in qualifying the suitability of a given location system to be used for a planned application
- in deployment and commissioning to ensure that the planned location system is functioning as expected

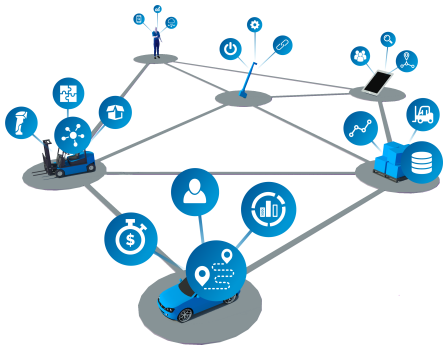
- in production to monitor for location system issues and failures due to changing environment and infrastructure

Reporting

Store historical data in a relational database and generate web-enabled graphical reports or integrate with third-party report builders

Business object properties

Define data models to support context-aware business applications, tightly integrated with location data



Summary

Business object properties provides the following capabilities:

- User-extensible definition of functional and relational properties of types
- Creation and deletion of properties
- Query, creation, update and deletion of property assertions
- Query, creation, editing and deletion of 3D spatial properties of objects and spatial relationships between objects

SmartSpace allows the definition of relationships between objects by using either *Simple Properties* or *Complex Properties*.

Simple Properties

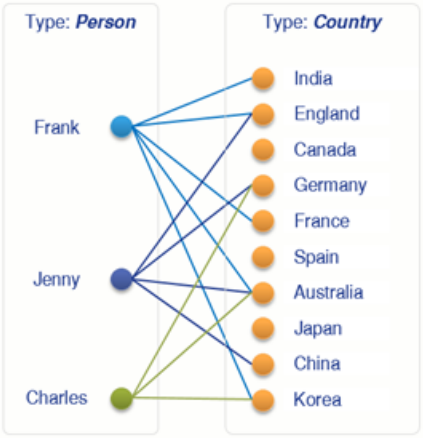
SmartSpace enables developers to flexibly create the types necessary to define a new application data model. User-defined types are created with appropriate properties, which can be any of the basic numeric or string types as well as user-defined types already defined in the data model. In SmartSpace, these are called *Simple Properties*.

Types which represent real-world objects can include one or more *Space Properties*, which are the geometric extents (zones) defined around the object that move and interact with the spaces of other object types in the data model when objects are given a location.

Additionally, multiple inheritance is fully supported, so that new types can be created from a combination of parent types which own the properties required.

Complex Properties

By using Simple Properties, one-to-one and many-to-one relations between objects can be specified. *Complex Properties* allow the inclusion of joins between types so that *one-to-many* or even *many-to-many* relations can be defined. This is really valuable when it becomes necessary to store or operate on sets of data, or some conditional fact must be asserted based on the state of multiple objects in the data model.



Where a relation between types is a fact, whether it is either true or false, the relation must be qualified by some underlying logic. This relation must be asserted either by an integrated third-party business system or through the SmartSpace Business rules engine (whose logic is implemented using the Rules engine developer).

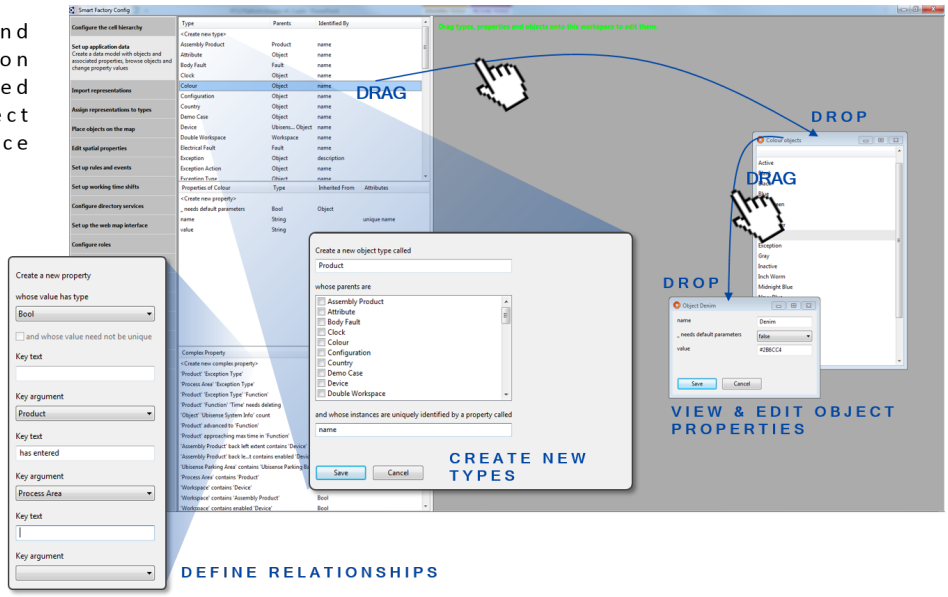
User Interfaces

The *Type and object definition* feature includes a full development and object explorer thick-client GUI, which provides a drag-and-drop interface for creating new types, instantiating new objects, and interrogating and editing object properties and states.

Data modelling and relationship definition through a detailed property and object browser interface

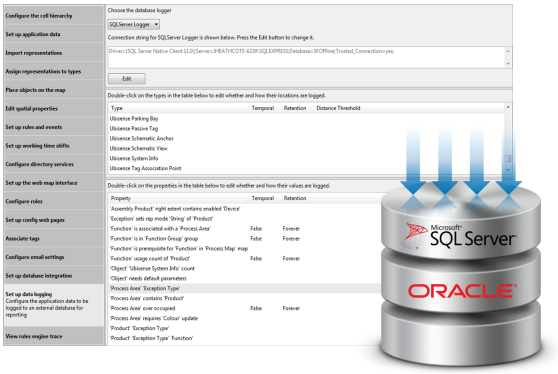
View, modify, delete any object, property or relationship

See changes to the underlying object model happen in real-time.



Property history

Record the locations of tracked assets, and their other properties, over time



Summary

The locations of configurable asset types can be recorded into a relational database and retained for a given time period. Similarly the values of asset properties and other business properties can be stored in the database.

Customers can license SmartSpace Reporting or run their own reporting engine over the recorded data.

Web reports

Deliver reports based on location and property history

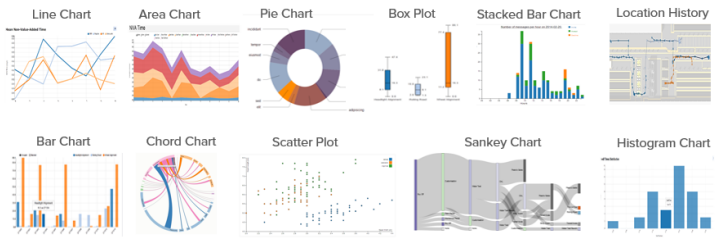
BROWSER BASED CHARTS AND TABLES ON ANY WEB-ENABLED DEVICE

Summary

The Web reports feature runs queries over the recorded location and property history to generate sophisticated tabular and chart-based reports displayed in the browser. On its own, this feature can deliver pre-built reports, such as those available as part of a vertical application. With the *Reports engine developer* feature, reports can be edited on line.

Web reports can be accessed from a wide variety of devices, ranging from desktop computers to tablets and phones, and do not require installation.

Supported charts include line, discrete bar, multi-bar, horizontal bar, stacked area, scatter, cumulative line, line and focus, pie, histogram, box plot, Sankey diagram, chord diagram and location history.



Users and roles

use
r authorizations and integrate with external user and role definitions in LDAP

Summary

Users and their *roles* can be defined within SmartSpace or imported from external systems via LDAP.

SmartSpace roles control which parts of the web interface can be accessed by a user, and the maps, reports and other screens available to them. It also determines which users should receive notifications, emails and alerts generated through the Business rules engine.

This allows the application experience to be tailored to meet the needs of specific classes of end user.

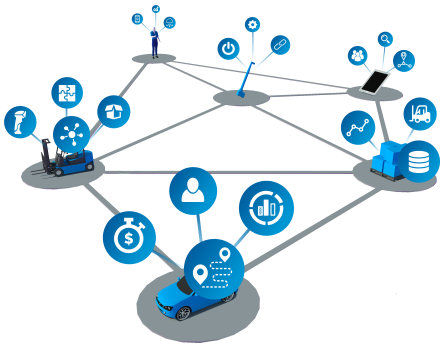
The SmartSpace configuration client includes a workspace for defining roles and assigning users to them. In addition SmartSpace provides a browser-based view of the underlying users and roles model, giving system administrators simple access for editing or extending the underlying user and group allocations.

Applications integration

Exchange business data with other systems using extensible APIs that support integration via various interfacing technologies including REST services, .NET, and direct RDBMS integration

Business object properties

Define data models to support context-aware business applications, tightly integrated with location data



Summary

Business object properties provides the following capabilities:

- User-extensible definition of functional and relational properties of types
- Creation and deletion of properties
- Query, creation, update and deletion of property assertions
- Query, creation, editing and deletion of 3D spatial properties of objects and spatial relationships between objects

SmartSpace allows the definition of relationships between objects by using either *Simple Properties* or *Complex Properties*.

Simple Properties

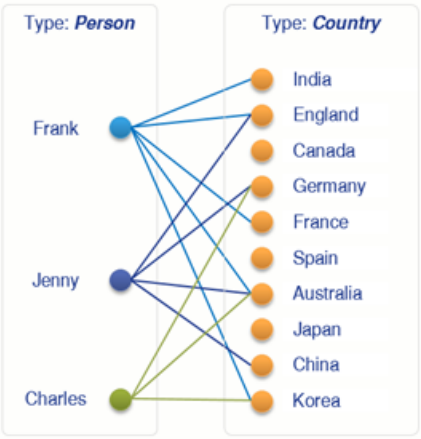
SmartSpace enables developers to flexibly create the types necessary to define a new application data model. User-defined types are created with appropriate properties, which can be any of the basic numeric or string types as well as user-defined types already defined in the data model. In SmartSpace, these are called *Simple Properties*.

Types which represent real-world objects can include one or more *Space Properties*, which are the geometric extents (zones) defined around the object that move and interact with the spaces of other object types in the data model when objects are given a location.

Additionally, multiple inheritance is fully supported, so that new types can be created from a combination of parent types which own the properties required.

Complex Properties

By using Simple Properties, one-to-one and many-to-one relations between objects can be specified. *Complex Properties* allow the inclusion of joins between types so that *one-to-many* or even *many-to-many* relations can be defined. This is really valuable when it becomes necessary to store or operate on sets of data, or some conditional fact must be asserted based on the state of multiple objects in the data model.



Where a relation between types is a fact, whether it is either true or false, the relation must be qualified by some underlying logic. This relation must be asserted either by an integrated third-party business system or through the SmartSpace Business rules engine (whose logic is implemented using the Rules engine developer).

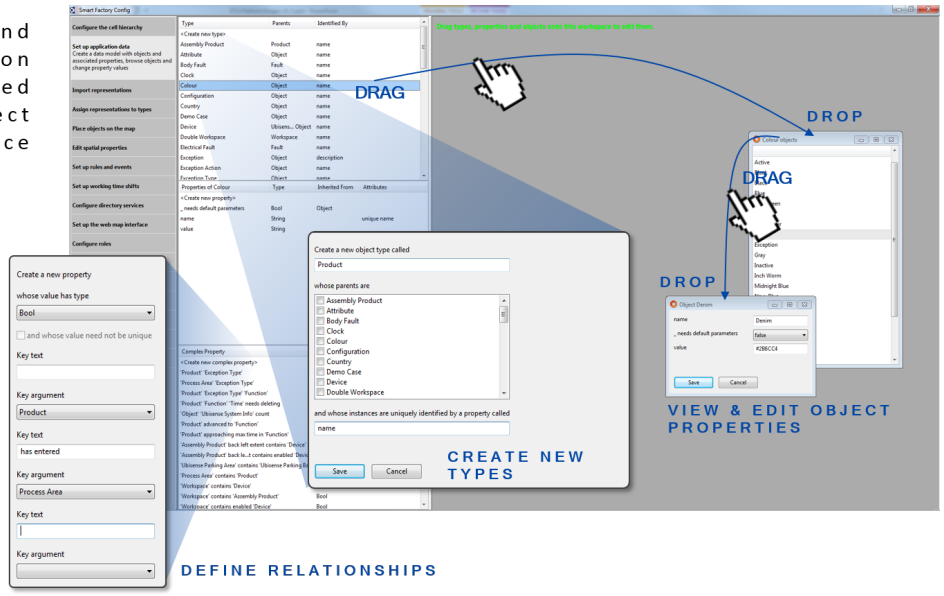
User Interfaces

The *Type and object definition* feature includes a full development and object explorer thick-client GUI, which provides a drag-and-drop interface for creating new types, instantiating new objects, and interrogating and editing object properties and states.

Data modelling and relationship definition through a detailed property and object browser interface

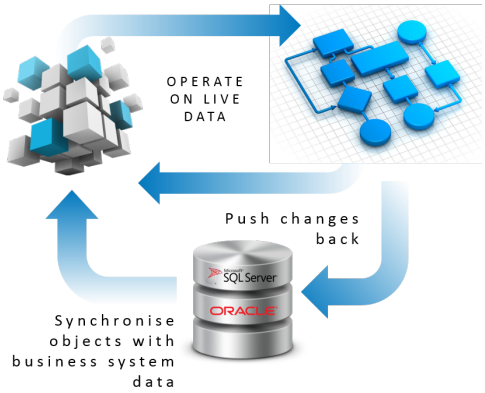
View, modify, delete any object, property or relationship

See changes to the underlying object model happen in real-time.



RDBMS map

Integrate with external relational database management systems to read or write application data



Summary

The mapping allows the configuration of the import and export of object properties between the Ubisense platform and external database schemas. SQL Server and Oracle are supported directly.

RDBMS map allows operations to be scheduled, or to be triggered based on changes to object properties.

Application .NET API

Enable .NET programs to read and write the defined SmartSpace object properties

Summary

Application .NET API provides a set of related .NET assemblies which can be referenced in a .NET project to access the Ubisense data model. The API supports creating and removing object instances, setting and reading object properties, including custom business object properties. It also supports callbacks when object properties are inserted, updated or removed.

Application REST API

A web API serving UDM objects/properties

Summary

Application REST API is an http base API allowing the viewing or creation of objects and properties from the UDM.

Application REST API:

- Provides an interface to the Ubisense platform without requiring knowledge of or access to Ubisense protocols
- Allows for easy access to the platform information without needing to write software. This is useful for viewing information in a one time/non-automated fashion or when writing lightweight web scripts

Ubisense Web API Help Page

TypeHierarchy

Operations for exploring type structure.

API	Description
GET api/TypeHierarchy/{type}	Get a list of ancestors and descendants for a given type.
GET api/TypeHierarchy	Get all types with their ancestors and descendants.

ObjectProperties

Operations for gettings and setting objects' properties.

API	Description
GET api/ObjectProperties/{type}/{name}/{property}	Get a property's value for the object of given name and type. The property must be defined at this level i.e. {type} defines the property or inherits from a type that does.
GET api/ObjectProperties/{type}/{property}	Get a property's value for all objects of a given type. The property must be defined at this level i.e. {type} defines the property or inherits from a type that does.
GET api/ObjectProperties/{type}/{name}/{property}/{key}	Get all complex property values involving this object for a given property signature. Signatures can be found using the TypeProperties operations.
PUT api/ObjectProperties/{type}/{name}/{property}	Set a property's value for the object of a given name and type.

TypeObjects

Operations for the creation/browsing of objects based on type and inherited types.

API	Description
GET api/TypeObjects/{type}	Get all objects of given type.
GET api/TypeObjects/{type}/{name}	Get the named object with given type.
PUT api/TypeObjects/{type}/{name}	Create an object of given name and type.

TypeProperties

Operations for exploring properties of types.

API	Description

RFID integration

Receive identification data from a wide range of suppliers using industry-standard protocols

AngleID connect

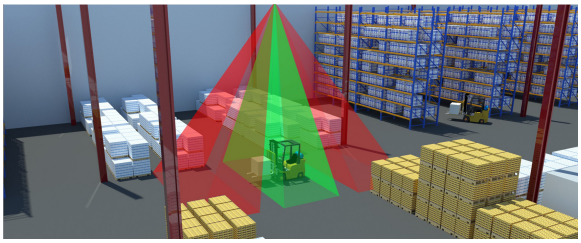
Collect data from an AngleID system and add it to the Ubisense platform

Summary

The entry_exit_connector service collects data from an (entry/exit) AngleID system and adds it to the platform in the form of UDM properties in real time. The following UDM elements are maintained by the service:

- **Ubisense AngleID Recipe** objects
- Object satisfies **Ubisense AngleID Recipe** property which contains entries for object inside a recipe detection zone

This allows a user to program logic based on this property using the *Business rules engine*, for example to place an object at location “x” when it satisfies recipe “y”.



LLRP interface

Collect data from LLRP readers and add it to the Ubisense platform

Summary

The LLRP interface enables configurable integration and mapping of identification events from 3rd-party RFID readers that support LLRP to objects and events.

The LLRP interface collects tag reports from readers and stores the data (what tags have been seen by what readers) in the assertion store/UDM, allowing a user to write logic based on this data using the *Business rules engine*.

RTLS integration

Receive real-time location data from a wide range of suppliers using industry-standard protocols

External data connector

Import data from external location systems in real time

Summary

The External data connector provides out-of-the-box configurable integration for any real-time location system.

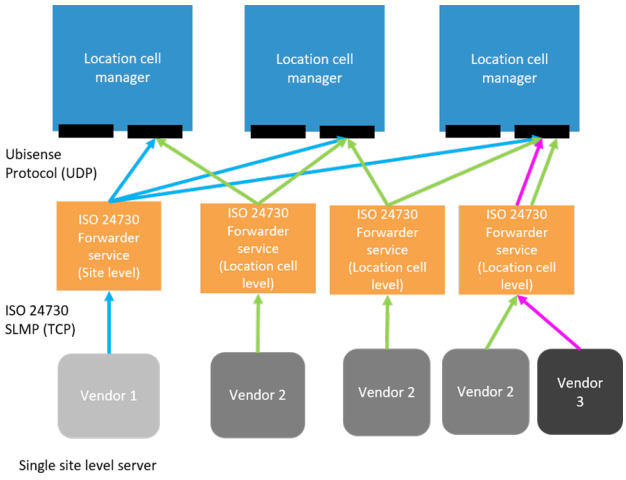
ISO 24730

Scalable standards-based real-time data import from a wide range of RTLS systems

Summary

The ISO 24730 feature provides a configurable cellular ISO 24730 client for the integration of multiple third-party location sources.

The ISO/IEC 24730 standard defines protocols for transferring information between different Real-time Location Systems.



The Ubisense ISO 24730 feature consists of Ubisense services and tools that allow a Ubisense RTLS to receive real-time locations using a *Text over Socket* connection over TCP/IP.

The Ubisense services act as a client connecting to a server within a third-party RTLS system. The services are added to a Ubisense dataset which contains the Ubisense platform, and optionally ISO 24730 applications.

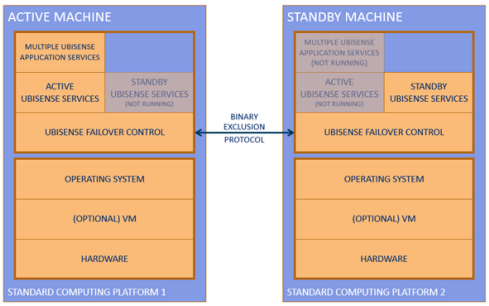
ISO 24730 is supported on Microsoft® Windows® and Linux operating systems.

Advanced IT

Manage a large-scale mission-critical production system using SmartSpace's tools

Failover

Automated high-availability for Ubisense systems, using a simple protocol running over a standard network to co-ordinate between N active machines and N standby machines



Summary

SmartSpace Failover automates switching from active to standby machines in case of machine failure, or to enable maintenance or backups, in a manner that protects data integrity by ensuring that active and standby machines never run at the same time.

Failover is based around the 'two machine' setup in which pairs of machines are configured so that at any one time only one machine is active whilst the other is on standby. If the active machine stops or fails, the standby machine takes over.

SmartSpace Failover ensures platform availability by automating switchover to an alternative host. It assumes an up-to-date copy of production data is made available by some method, for example by using SmartSpace Replication.

Ubisense Failover runs on Linux and Windows platforms.

Benefits

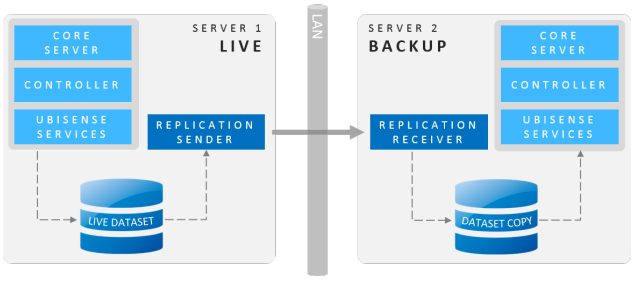
The benefits of SmartSpace Failover are:

- Automated
  Failover is automated so that the standby machine will take over from the active machine if it detects that the active machine has timed out. This ensures continuity without manual intervention.

- Maintenance and backups
  Maintenance that requires a machine to be taken out of service can be performed while the system is live so long as only one machine in a pair is halted at a time with Failover ensuring continuity of service on the other machine.
  Failover also allows scheduled dataset backups to be taken while the system is live.
- Data integrity
  Failover implements a protocol between two machines to make them work in a two machine setup, controlling Ubisense services on both machines, and minimizing the likelihood of both machines simultaneously becoming active.
- Scalable
  Failover can be implemented with a single pair of machines running all necessary Ubisense services or it can be scaled to encompass many pairs of machines, including pairs running different operating systems. For example the real-time control elements could run on Linux machines whilst the web visibility features run on Windows machines.
  Failover uses standard hardware and software and can be conveniently configured across whatever virtual machine and hardware configuration is required.

Replication

Maintain asynchronous hot copies of datasets to support hot backup and fail over



Summary

Replication provides software support for the configuration of a continuously-updated backup of the live dataset. This is stored on a separate (backup) machine, which also has the Ubisense platform installed, but inactive. In case the production machine fails, the Ubisense platform on the backup machine can be started and take over as the production platform server.

The backup machine can also share the same data as the live machine, for example by using a Storage Area Network (SAN) or Network-Attached Storage (NAS).

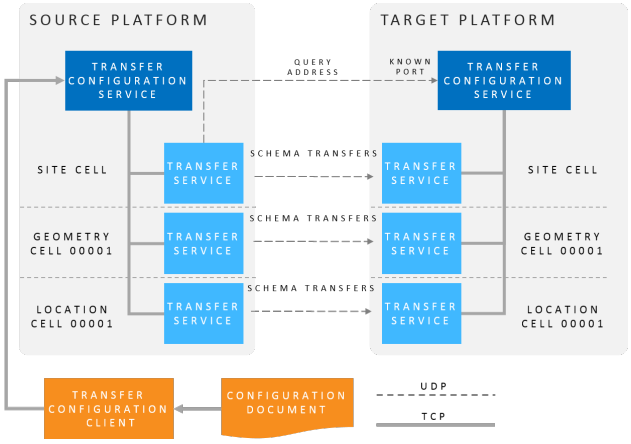
The benefits of using Replication are:

- Replication is performed in software, meaning there is no need for specialized high availability storage hardware
- The copy of data on the backup machine can be used as a *hot backup*
- Live machine is immune from latency problems commonly caused by synchronous replication
- Good for long-distance off-site replication where latency would be a serious issue

Property transfer

Maintain side-by-side test systems using production data

Summary



Property transfer provides software support for copying live data such as object locations, names, battery status, spatial ownerships, data messages, and sensor status, from one instance of the Ubisense platform to another.

The transfer operates at a low level in the platform architecture, and can be configured to copy data in a *best-effort* or *reliable* mode.

Individual transfers can be stopped and started using a configuration tool, and the current status and statistics of each transfer are recorded to a central schema.

From version 3.7, an additional tool enables the real-time transfer of user data model (UDM) properties.

Health monitoring

Monitoring of the status and performance of a SmartSpace system over time

Summary

Recording and analyzing system health is often a requirement for enterprise location system deployments. Ubisense Health monitoring works by periodically delivering metrics to a central

health monitoring service. Separate services expose these measurements to an external metrics database system such as Prometheus. Metrics can then be viewed on configurable dashboards in a web browser. Isolation of monitoring from the operation of the SmartSpace platform ensures a minimal impact on system performance.

Benefits

Benefits of Health monitoring are:

- Proactive support
  Threshold conditions can be set to trigger support before the system fails
- Swifter fault diagnosis
  The availability of recorded health data can speed up the diagnosis and correction of issues
- Modularity and scalability
  The modular health monitoring architecture composes with the particular SmartSpace features you license and can scale to large volumes of recorded data
- Visibility
  Use of web-based configurable dashboards brings metrics to users in an easily-accessible format

SmartSpace Developer

Development tools which are used to implement and deliver new functionality across the enterprise

Rules engine developer

User tools for programming the Business rules engine

Summary

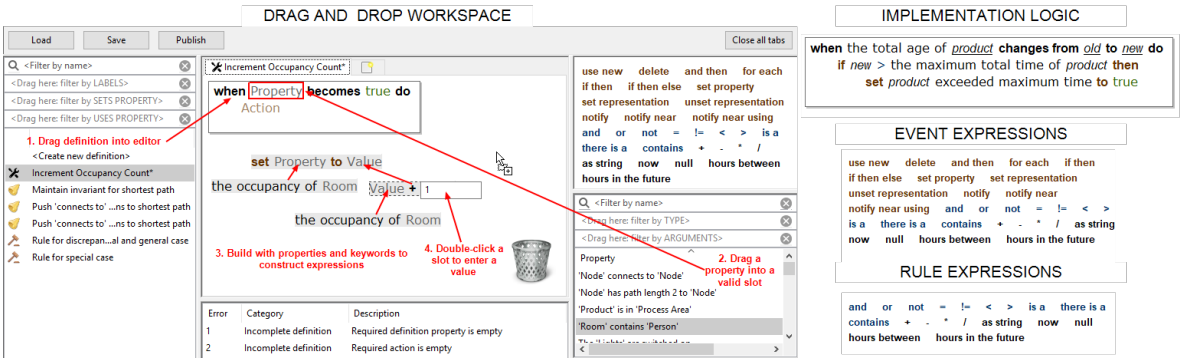
The Business rules engine allows complex logic and relationships between objects to be defined. The user specifies these rules by using graphical programming techniques.

Features

Defining important events or actions within an application requires a way to describe:

- The conditions under which an event has become true
- The actions that must occur when the prerequisite conditions have been satisfied

The easiest way to describe such constructs is through language. SmartSpace offers a rule and event definition interface that allows the user to drag and drop objects, properties and relations and join them together to create language-like structures.



The Rules engine developer also includes tools for the import and export of rules to allow developers to ship reusable packages of rules to other sites or customers and to manage development projects using standard version control systems.

Real-time rules engine

Use rules and event handlers to create scalable real-time control applications

Summary

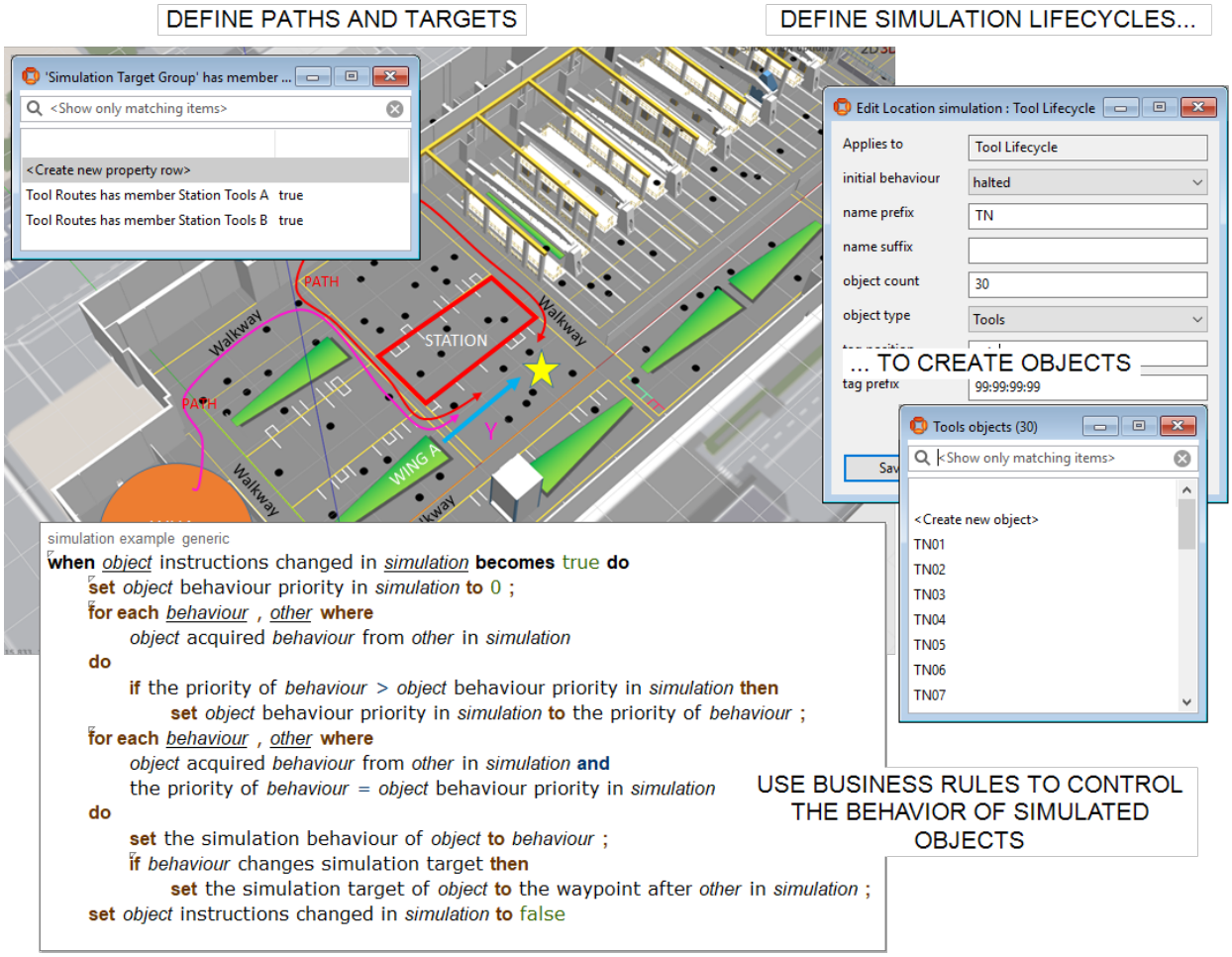
The standard rules engine uses a centralized, persistent data store. The Real-time rules engine extends the standard rules engine with multiple parallel data stores (one for each geometry cell), where each store manages non-persistent data. This makes it possible to integrate user data seamlessly into low-latency real-time control applications.

Location simulation

Model the movement of simulated objects within the SmartSpace digital environment

Summary

Before you begin to track physical objects in your workplace, or make changes to an existing real-time location system, you need to understand the how objects move through space and interact with their environment. The core Site visualization feature allows you to create a 3D representation of your site and populate it with objects that mirror your real-world site. With the Location simulation feature, used in conjunction with the Rules engine developer, you can create simulated objects and the rules that define how they move through your site—and then run and re-run these simulations. For example, you might want to model how AGVs move through a manufacturing plant, stopping at workstations for the correct length of time, and avoiding collisions when routes merge or cross.



Features

Location simulation includes the following capabilities:

- Creation of simulations to encapsulate real-world scenarios
- Automated creation of simulated objects
- Definition of the behavior of objects in simulations using business rules

Benefits

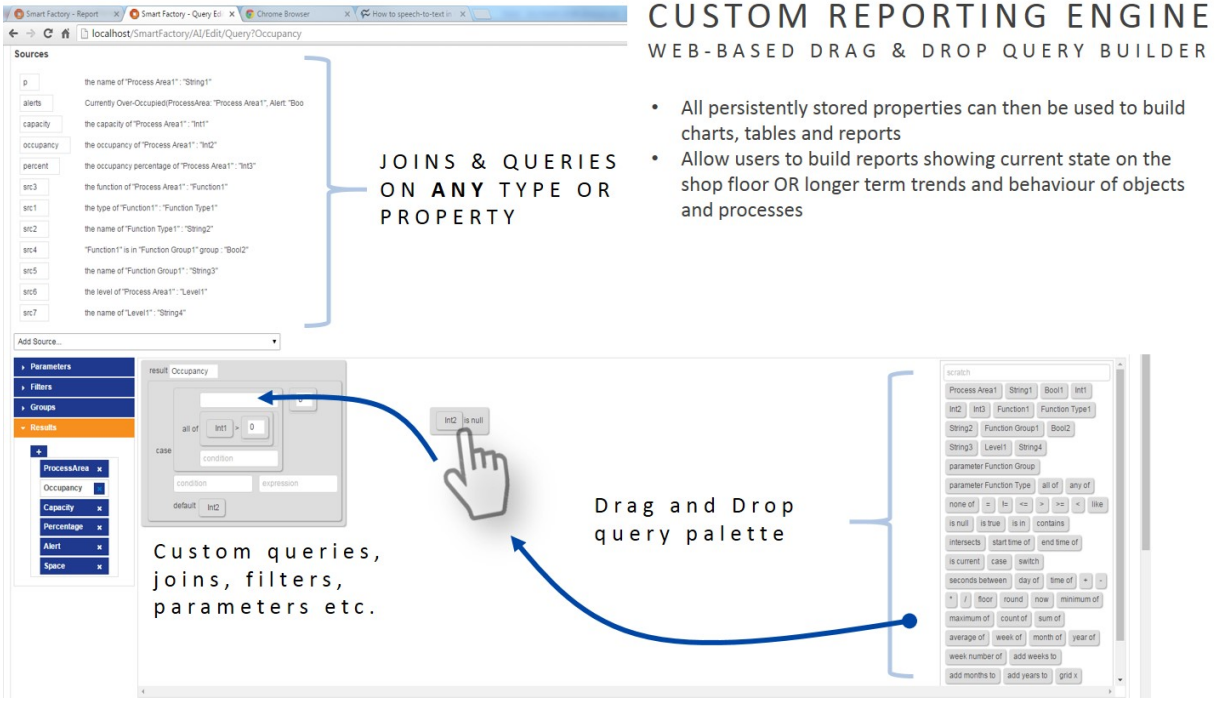
- Better understanding of the mechanics of existing real-world sites
- Improved efficiency in routing objects through a site
- Planning new sites made easier and less prone to error

Reports engine developer

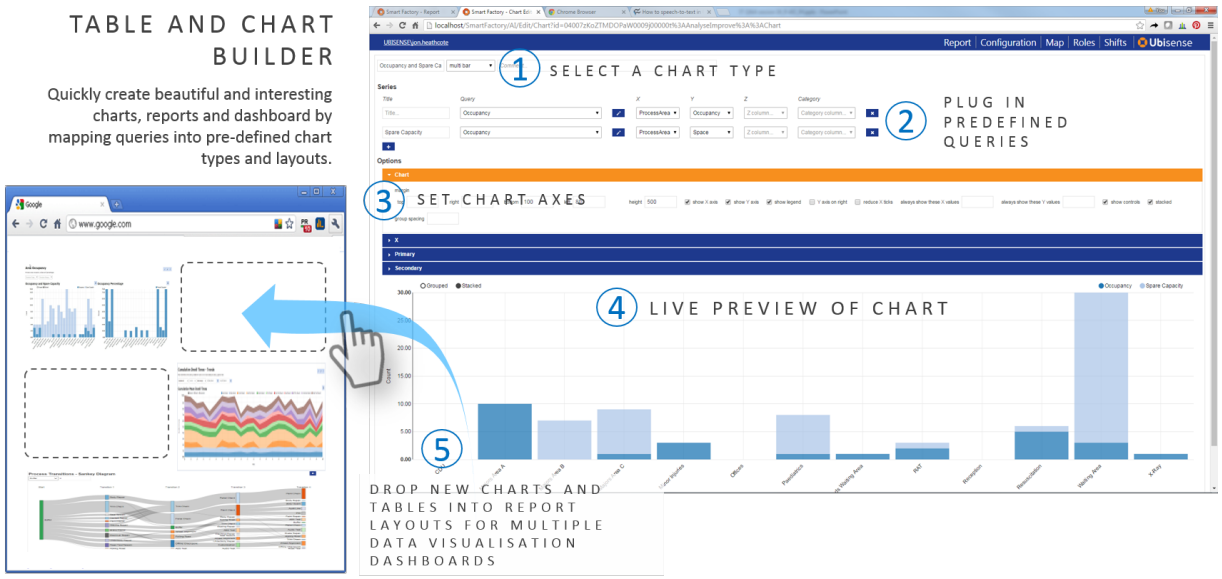
Build custom web reports within the browser

Summary

The Reports engine developer allows authorized users to create and modify reports based on recorded location and property history. The feature provides editor web pages that allow drag-and-drop construction of queries over the recorded data.



These queries can then be used to populate tables and charts, and the charts built into reports with user-entered parameters.



Reports can then be assigned to specific roles so that they can be run by users that are members of those roles.

The editor pages can be accessed from a wide variety of client operating systems and do not require installation. The editors are easiest to use on devices with tablet- or desktop-sized screens.