SmartSpace

Ubisense Health Monitoring Config

For version 3.3

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

Telephone: +44 (0)1223 535170. Website: http://www.ubisense.net.

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

Tel: +44 (0)1223 535170

WWW: *https://www.ubisense.net*

Contents

Overview of Health monitoring

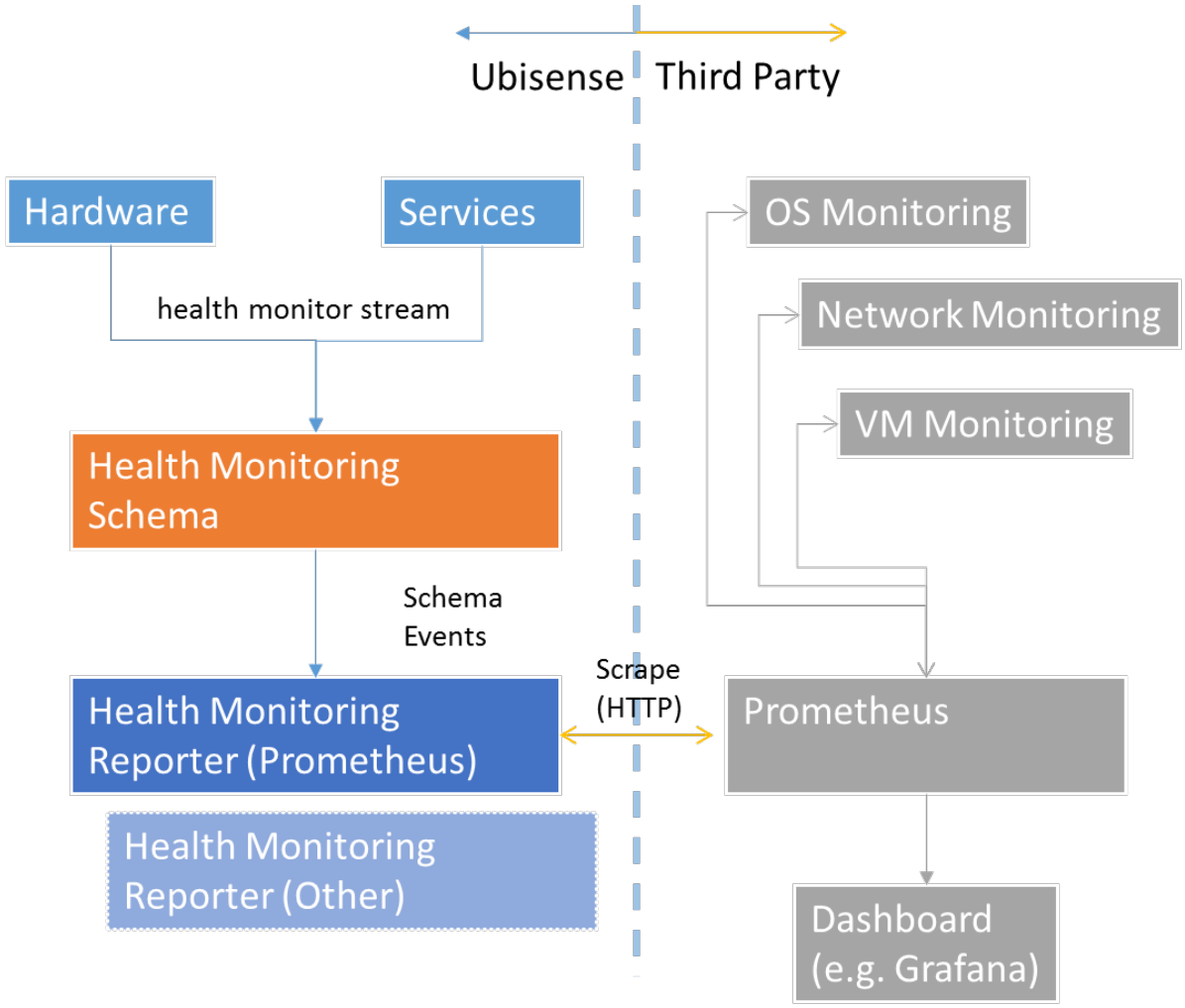
This document describes the features and suggested configuration of health monitoring in SmartSpace 3.3, including DIMENSION4 sensor and tag health. Recording and analyzing system health is often a requirement for enterprise location system deployments. Threshold conditions can be set up to trigger pro-active support before the system fails, and the availability of recorded health data can speed up the diagnosis and correction of issues.

Audience

This guide should be read by those who will be setting up or maintaining a SmartSpace system, and want to provide monitoring of the status and performance of the system over time.

Health Monitoring Architecture

The health monitoring architecture is designed to provide isolation between monitoring and operation of the SmartSpace platform, so that health monitoring has minimal impact on operational resources. It is also designed to be modular so that it composes with the particular product features that the customer has licensed, and to scale to large volumes of recorded data if that is required.



The architecture of health monitoring in SmartSpace

Internally, various components of the SmartSpace system deliver measurements periodically to a central health monitoring service (Ubisense/IT support/Health metrics server), where they are transiently recorded in a schema. The delivery protocol uses UDP to a configured address and port, and is low cost and unreliable – packets can be dropped if the network or server is highly loaded.

Separate services then expose the recorded measurements to be scraped periodically by an external metrics database system. In this release the only format supported is Prometheus. This is provided by the service Ubisense/IT support/Health metrics reporter.

Configuring Health Monitoring

Requirements

Health monitoring requires SmartSpace 3.3.6718 or later, and is also supported by DIMENSION4 1.4 or later (sensor software later than 2586).

SmartSpace and DIMENSION4 Configuration

To configure health monitoring, first configure SmartSpace and DIMENSION4. We recommend this is done before installing the health monitoring software, so no extra service or sensor restarts are required. The following configuration parameters are used:

health_mcast_addr

The address to which measurements are sent. This should either be a multicast address, or should be the unicast address or DNS name of the server on which the service "Ubisense/IT support/Health metrics server" is deployed. Default value: 239.255.255.252

health_mcast_port

The port to which measurements are sent. This must be available on the server on which service "Ubisense/IT support/Health metrics server" is deployed, and must not be blocked by a firewall. Set this to 0 to disable all sending of health measurements. Default value: 49976

health_reporter_port

The port on which the health metrics are exposed for scraping by Prometheus. The service "Ubisense/IT support/Health metrics reporter" listens on this port for HTTP get requests from Prometheus, and responds with the current values of all metrics. Default value: 9494

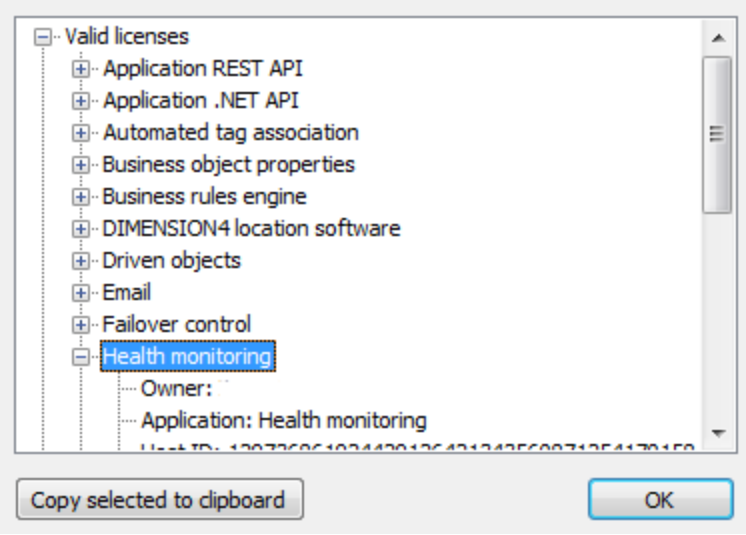
health_reporter_interface

As originally delivered, the health monitoring reporter service would bind to either localhost (in standalone mode) or all external ip addresses (in non-standalone mode). Now by default, it binds to any interface. To bind to a specific interface, set the "health_reporter_interface" to the ip address of the interface you require.

Installation

The IT support component must be licensed for release 3.3 or later. To ensure this is enabled, check **Platform Control/Licenses**, and open Valid licenses. You should see Health monitoring as

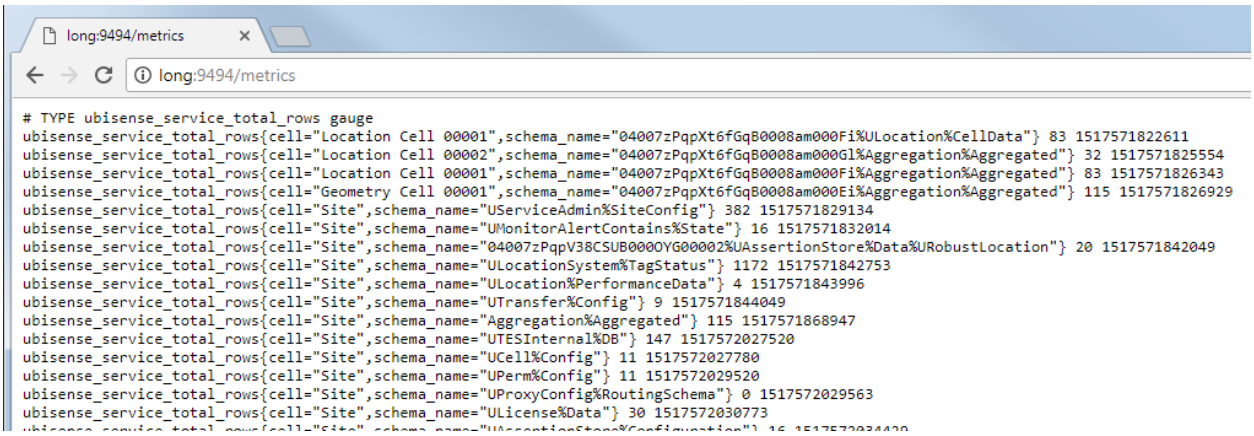
one of the valid licenses.



Now use **Service Manager/INSTALL SERVICE**, and open the SmartSpace distribution packages folder. Select the Health monitoring feature, and click Install.

For DIMENSION4 sensor support, upgrade to a version of the Dimension 4 software that supports health monitoring, and reboot sensors so they are running the correct software version.

Verify that monitoring is working by visiting **http://<server>:9494/metrics** in a browser, where server is the host on which the "Ubisense/IT support/Health monitoring reporter service" is deployed. You should see a text response containing some metric values.



Configuring Third Party Software

The third party software can be run on a separate server from the rest of the Ubisense SmartSpace system, to provide maximum isolation. Prometheus, while normally very efficient, can use a significant amount of memory and disk IO, when large queries are run and on startup, so it is not recommended for a production system to use the same server.

Prometheus

Prometheus is a widely used open source monitoring solution available under the Apache 2 license. There is extensive documentation on installing and configuring Prometheus on the web site *https://prometheus.io/* from which the software can be downloaded for free. In this guide we will describe a simple installation of Prometheus, but for production use we recommend consulting the documentation especially the section on Storage configuration. Prometheus uses disk storage by default, and this may be suitable for production use depending on requirements, but it can also be configured to use remote storage integrations.

The simple Prometheus configuration file, ubisense.yml, looks like this:

```
# global configuration
global:
  # Set the scrape interval to every 15 seconds.
  scrape_interval:     15s
  # Evaluate rules every 15 seconds.
  evaluation_interval: 15s
  # scrape_timeout is set to the global default (10s).

# A scrape configuration for Ubisense SmartSpace, with the health
# monitor reporting service running on "ubicore1".
scrape_configs:
  - job_name: 'ubisense'
    static_configs:
      - targets: ['ubicore1:9494']
```

If you copy this example, ensure you retain the correct indentation in the code. Otherwise the configuration will fail.

To run Prometheus, execute:

```
prometheus.exe --config.file=ubisense.yml
```

Now check that Prometheus is available on its default port by browsing to *http://localhost:9090/*

Note that Prometheus can be used to capture other data such as server CPU/Memory and disk, and these can be incorporated into the monitoring dashboards. See the online help for details.

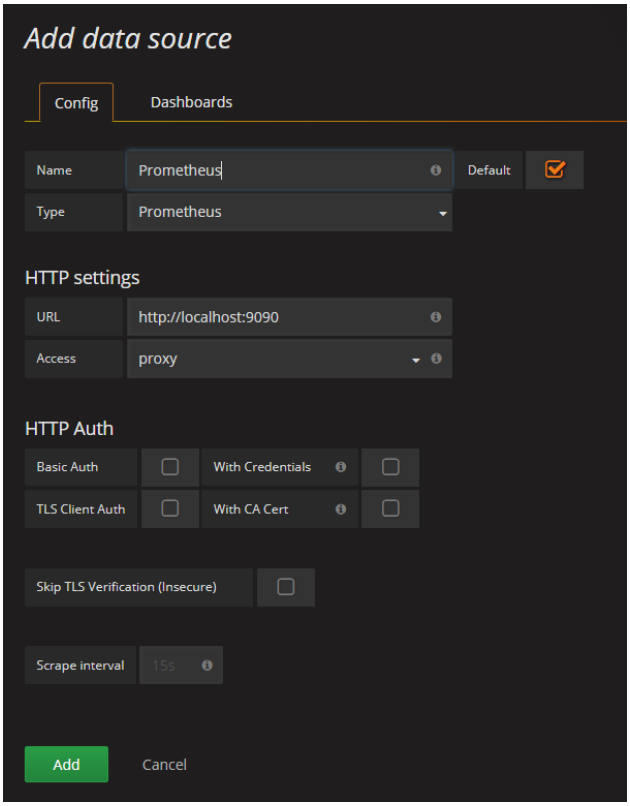
Grafana

Grafana is a widely used time series analytics and dashboard building front-end for Prometheus. It is also open source and is available from *https://grafana.com/* under the Apache 2 license. For production configuration, see the installation documentation. This guide is a simple configuration for getting started.

Download and run Grafana – all configuration can be done inside the web site, which by default is on *http://localhost:3000/*. Login in as admin:admin (the defaults – these can be changed by using a config file).

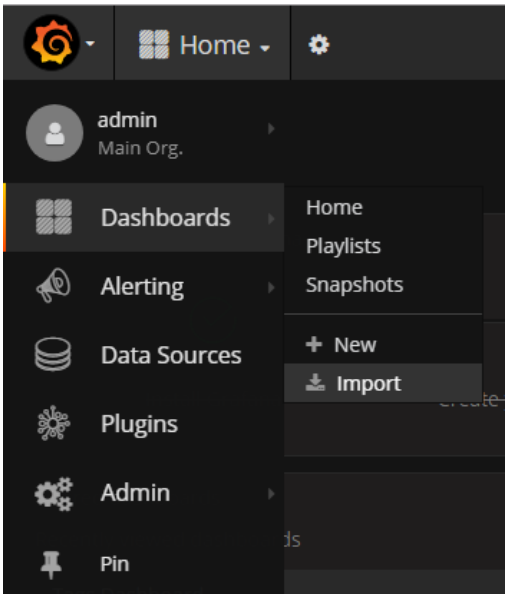
Connect to Prometheus

Add Prometheus as a data source. Select "create your first datasource" from the home page. Call the datasource "Prometheus" (careful to spell this correctly, as it is referenced with this name in the sample dashboards), with Type "Prometheus". Under HTTP settings, set the URL to *http://localhost:9090*, and set Access to "proxy". Click Add.



Load Sample Dashboards

Sample dashboard configurations are available in the **Application Manager**. Go to the **DOWNLOADABLES** task and select IT support/Health metrics sample dashboards. In the Grafana web site, select Dashboards/Import



Now select Upload .json file. Navigate to one of the .json files, and upload it. You should see a working dashboard.

Metrics Provided

The following metrics are provided by Ubisense health monitoring.

- *automatic tag association*
- *location sinks*
- *operations*
- *sensors*
- *servers*
- *services*
- *spatial*
- *tags*
- *thread and disk latency*
- *web map*

In each case the metric name is preceded by "ubisense_".

Metric	Type	Description	Labels
Automatic tag association			
auto_association_bad_battery	counter	Number of attempts to associate a tag with a bad battery	point
auto_association_bad_type	counter	Number of failures to associate due to a tag being of the wrong type	point
auto_association_errors	counter	Number of failures to associate, including bad battery and late data	point
auto_association_late_data	counter	Number of times a tag was detected in the association station the candidate had not been set, resulting in a "waiting for data" warning	point

Metric	Type	Description	Labels
auto_associations	counter	Number of automatic tag associations	point
Operations			
operation_calls	counter	Number of calls to each schema operation	schema_name, operation, cell
operation_max_ duration_seconds	gauge	Maximum time taken to complete each schema operation	schema_name, operation, cell
operation_mean_ duration_seconds	gauge	Mean time taken to complete each schema operation	schema_name, operation, cell
Sensor			
sensor_cable_received	counter	Bytes sensor received via the timing cable	MAC, group
sensor_cable_ recoverable_errors	counter	Recoverable errors in the data from the timing cable	MAC, group
sensor_cable_ unrecoverable_errors	counter	Unrecoverable errors in the data from the timing cable	MAC, group
sensor_cable_ unsynced	counter	Number of times cable was found to be unsynchronised	MAC, group
sensor_dsp_load	gauge	The percentage load of the DSP on each sensor	MAC, group
sensor_dsp_memory_ used	gauge	The percentage of DSP memory used each sensor	MAC, group

Metric	Type	Description	Labels
sensor_late_measurements	counter	The number of times measurements for a tag transmission from another sensor have arrived too late to be included in location calculation	MAC, group
sensor_memory_hwm	gauge	The peak resident memory used on the sensor, in kB	MAC, group
sensor_memory_rss	gauge	The current resident memory used on the sensor, in kB	MAC, group
sensor_network_receive_bytes	counter	Number of bytes received on the ethernet	MAC, group
sensor_network_receive_dropped	counter	Number of dropped packets from ethernet	MAC, group
sensor_network_receive_errors	counter	Number of packet errors from ethernet	MAC, group
sensor_network_receive_packets	counter	Number of packets received from Ethernet	MAC, group
sensor_network_transmit_bytes	counter	Number of bytes sent on ethernet	MAC, group
sensor_network_transmit_dropped	counter	Number of sent packets dropped by ethernet	MAC, group
sensor_network_transmit_errors	counter	Number of transmit errors on ethernet	MAC, group
sensor_network_transmit_packets	counter	Number of packets sent to ethernet	MAC, group
sensor_reboots	counter	Number of times a sensor has rebooted	MAC, group

Metric	Type	Description	Labels
sensor_send_fails	counter	Number of times sending a location message to the sink failed, such as there being no route to the sink address, or other network failure	MAC, group
sensor_send_locations	counter	Number of location messages sent by the sensor	MAC, group
sensor_status	gauge	The status of the sensor, which is one of the following values: 0 : UNKNOWN, 1: INITIALISING, 2: UNSTABLE_TIMING, 3: RUNNING, 4: REBOOTING, 5: UNEXPECTED_CODE, 6: STALE_INITIALISING, 7: STALE_UNSTABLE_TIMING, 8: STALE_RUNNING, 9: STALE_REBOOTING	MAC, group
sensor_tag_count	gauge	The number of tags that have sent a data message to the sensor in the last two minutes. Due to beacon rates and changes in group networking this may not be an accurate measure of the number of tags, but it is a reasonable estimate.	MAC, group
sensor_timing_bytes	counter	Number of bytes received on the timing cable	MAC, group

Metric	Type	Description	Labels
sensor_timing_errors	counter	Number of byte errors on the timing cable	MAC, group
sensor_timing_ unstable	counter	Number of unstable timing cable events	MAC, group
sensor_unexpected_ reboots	counter	Number of times a sensor has rebooted when not instructed to do so, including due to detected network changes and power outages	MAC, group
Servers			
server_ack	counter	Low level RPC protocol acknowledgement packets	schema_name, cell
server_discarded	counter	Low level RPC discarded packets due to server too busy	schema_name, cell
server_handlers	gauge	Low level RPC current number of request handlers	schema_name, cell
server_query_request	counter	Low level RPC queries about progress of request phase	schema_name, cell
server_query_ response	counter	Low level RPC queries about progress of response phase	schema_name, cell
server_received_data	counter	Low level RPC number of received data packets in request	schema_name, cell
server_received_nak	counter	Low level RPC number of replies from server that request packets are missing	schema_name, cell
server_sent_data	counter	Low level RPC number of sent data packets in response	schema_name, cell

Metric	Type	Description	Labels
server_sent_nak	counter	Low level RPC number of replies from client that response packets are missing	schema_name, cell
Services			
build_number	gauge	Service build number deployed at a controller	vendor, package, service, cell
service_max_rows	gauge	Maximum number of rows in a table of the schema	schema_name, cell
service_started	counter	Number of times the controller started the service	vendor, package, service, cell, controller
service_stopped	counter	Number of times the controller stopped the service	vendor, package, service, cell, controller
service_total_rows	gauge	Total number of rows in all tables of the schema	schema_name, cell
service_transactions_total	counter	Number of transactions (changes) for the schema	schema_name, cell
sink_sensor_health_status	counter	Number of sensor health messages received at the location sink service	cell
Location sinks			
sink_status	counter	Number of sensor status messages received at the location sink service	cell

Metric	Type	Description	Labels
sink_tag_status	counter	Number of tag status messages received at the location sink service	cell
sink_tag_type_request	counter	Number of tag type requests received at the location sink service	cell
sink_upstream	counter	Number of upstream messages received at the location sink service	cell
Spatial			
spatial_index_changes	counter	Number of changes to the spatial index required to process spatial containment monitoring	cell
spatial_index_ commits	counter	Number of transactions on the spatial index	cell
spatial_location_ commits	counter	Number of location transactions applied to the spatial monitoring	cell
spatial_location_ establishes	counter	Number of times a location cell state was established at the spatial monitor	cell
spatial_ownership_ commits	counter	Number of transactions on the spatial ownership configuration	cell
spatial_ownership_ establishes	counter	Number of times the spatial ownership configuration was established at the spatial monitor	cell

Metric	Type	Description	Labels
Tags			
tag_battery_state	gauge	The state of the tag battery, which is one of: 0: OK, 1: WARNING, 2: FAILING	tag
tag_energy_used	gauge	The percentage of tag battery capacity currently used	tag
tags_associated	gauge	The number of tags associated with an object	tag
Thread and disk latency			
disk_latency_ milliseconds	gauge	The maximum latency when attempting to perform an operation on a disk file, over the last five minutes, reported when greater than 60ms	operation
thread_scheduling_ events	counter	The number of times a thread was available to run but did not get scheduled to run for more than 30ms	host
thread_scheduling_ max	gauge	The maximum time a thread ready to run waited to be scheduled over five minutes	host

Metric	Type	Description	Labels
thread_scheduling_mean	gauge	The mean time a thread ready to run waited to be scheduled over five minutes	host
Web map			
web_map_thread_count	gauge	The number of threads used by the web map server to process search queries	
web_map_unique_clients	gauge	The number of unique clients of the web map	