

Nokia Research Center / Helsinki
Dana Pavel, Dirk TrossenFINAL
24.10.06**ABSTRACTION MODEL IN THE N-RSA SYSTEM ARCHITECTURE**

Authors: Dana Pavel, Dirk Trossen
Nokia Research Center / Helsinki

Editor: Elena Balandina
Nokia Research Center / Helsinki

ABSTRACT

This document outlines the abstraction model for the Nokia Remote Sensing Architecture (N-RSA), defining the used language for describing the sensor data within the system but also the query language that is used to define discovery and acquisition queries for sensor data. The document outlines requirements for the abstraction model, the integration into the overall architecture and discusses two available choices, SensorML and TinyML, out of which the latter is used within the system.

Version:	1.0
Authors:	Dana Pavel, Dirk Trossen

Nokia Research Center / Helsinki
Dana Pavel, Dirk Trossen

FINAL
24.10.06

GLOSSARY

AS	Application Server
CR	Code Repository
IG	Intermediary Gateway
N-RSA	Nokia Remote Sensing Architecture
W3C	WWW Consortium
XML	Extensible Markup Language

Nokia Research Center / Helsinki
Dana Pavel, Dirk TrossenFINAL
24.10.06**TABLE OF CONTENTS**

GLOSSARY	2
1. INTRODUCTION.....	4
2. USE CASES	4
2.1 EXAMPLE 1: "WEATHER" SENSOR BOX	4
2.2 EXAMPLE 2: DROP-OFF BOXES SENSOR MODULES	5
3. REQUIREMENTS	5
4. DATA REPRESENTATION: SENSOR AND AGGREGATION ABSTRACTIONS	6
4.1 METHODOLOGY OF SENSOR AND AGGREGATION REPRESENTATION	6
4.2 INTEGRATION IN THE N-RSA SYSTEM ARCHITECTURE.....	6
4.3 CHOICE FOR SENSOR & AGGREGATION REPRESENTATION LANGUAGE	7
5. QUERY LANGUAGE	10
6. REFERENCES.....	11
ANNEX A HIGH LEVEL DIAGRAM OF SENSOR ML.....	12
ANNEX B ELEMENTS OF TINY ML.....	12

Nokia Research Center / Helsinki
Dana Pavel, Dirk TrossenFINAL
24.10.06

1. INTRODUCTION

One of the main characteristics of the system architecture designed in N-RSA [1][2] is that part of the intelligence required in information processing happens closer to the sensors, namely in the actual collecting point we call *Intermediary Gateway*. The main goal is to reduce the amount of sensor data that is transmitted to the end system by sending data only when necessary and even in already preprocessed form (e.g., aggregated or fused). For this, the IG has to have the capability of performing certain reasoning, and reasoning requires proper sensor and sensor data abstractions as well as capabilities of combining more simple sensor data into more complex one.

For this, we assume the existence of a proper abstraction model for sensor data and aggregated sensor information (based on the defined requirements in [1]). Such abstraction model will be the basis for the query language that is used for acquiring sensor data and aggregated data within the N-RSA. It will further serve as a basis for the code repository functionality of the N-RSA that allows for extending the aggregation functionality of the system.

This report will define abovementioned abstraction model and the query language that will be used within the N-RSA. Interoperability and flexibility are the main factors in the selection of appropriate means to define such model and language.

2. USE CASES

Given the vast space of applications that could benefit from remote sensing and the large number of sensors available, the following use cases show illustrative examples of remote sensing.

2.1 Example 1: “Weather” sensor box

We can imagine the sensor box that includes sensors that can measure temperature, pressure, and humidity. In the prototypes created in N-RSA we use the similar “weather” sensor box together with a Series 60 phone application that reads and displays the sensor data.

Since the only function in the current prototype is to simply poll the sensor data remotely, there is no need for creating sensor data abstractions. However, one of the main goals of N-RSA project was to allow for event-based sensor data delivery, which would require that the gateway is capable of reasoning based on the sensor data read from the “weather” sensor box. We can imagine scenarios when the gateway is supposed to send a notification to an application when the temperature is close to 0 degrees Celsius and the pressure is over a certain limit. We can also imagine scenarios in which the application wants to be notified when it’s getting cold, which would require a certain interpretation of what “cold” means.

Nokia Research Center / Helsinki
Dana Pavel, Dirk Trossen

FINAL
24.10.06

Therefore, means are required to abstract from the particular sensors used as well as to express the queries for acquisition properly. In cases of aggregation, a proper semantic interpretation of particular labels, such as “cold” has to be provided.

2.2 Example 2: Drop-off boxes sensor modules

We consider a drop-off box that could be used for a multitude of sensors. First of all, it could include load sensors, that would be used for giving an indication how much load has been put into the box. However, we can also imagine that the drop-off box would be a good place for other sensors, like temperature, wind speed, air quality, and hazmat. The hazmat sensors are especially interesting, since they would allow detecting when contaminated load has been put into the box, without infecting other content further.

Here, extendibility of sensor descriptions is key, together with the possibility to formulate appropriate queries for the different sensors and possible sensor aggregations.

3. REQUIREMENTS

[1] defines high-level system requirements for the N-RSA, which also included requirements for the abstraction model. These requirements are:

- The abstraction model should enable gathering of sensor data at any place, i.e., it should support local and remote sensors
- The abstraction model should support simple and aggregated sensor information. *Simple* sensor information refers to the actually sensed information, while *aggregated* refers to operations performed on a collection of sensor data, possibly from different sources, in order to derive another (possibly new type of) sensor information.
- The abstraction model must provide a metadata description of sensors and aggregated data, referring to means of describing sensor data.
- The abstraction model must allow for inclusion of new sensors of any type, requiring the extendibility of the abstraction model.
- A query language is required to formulate the desired acquisition of sensed data from the application's side. This query language is based on the defined abstraction model.

These requirements will be the basis for the following description of the data representation and query language within the N-RSA.

In addition to the functional requirements as defined in [1], one of the major selection criteria for a representation model should be interoperability in order to enable usage of sensors from different sensor vendors. Hence, we need to strongly consider existing efforts and trends in sensor representation.

Nokia Research Center / Helsinki
Dana Pavel, Dirk Trossen

FINAL
24.10.06

4. DATA REPRESENTATION: SENSOR AND AGGREGATION ABSTRACTIONS

The following section defines the abstraction model used in the N-RSA. For that, we first outline the methodology of sensor and aggregation representation that will be used in N-RSA. We then explain the integration into the N-RSA, before presenting the actual candidates for the representation model and the final choice for the N-RSA.

4.1 Methodology of Sensor and Aggregation Representation

The used methodology of representing sensors and aggregations of sensors within N-RSA is following a widely accepted principle, namely a tree-based representation of sensor data and performed aggregations, leading to more and more complex forms of sensor aggregations based on aggregations and/or sensor data.

Within this tree, the leaves represent sensors as the smallest, undividable unit, while the nodes represent possible aggregations of different types of sensors. It is further possible to abstract specific instances of sensors of the same type with a generic abstraction for this particular type. For instance, different location sensors (even providing different accuracies of sensor data) can be abstracted as a generic location type sensor. Such abstraction is helpful, for instance, in the sensor discovery process but also for acquisition.

The aggregation combines one or more sensors into a new type of information. For instance, temperature and wind speed data can be used to derive wind chill. This derivation can then again be used for input into other aggregation. With that methodology, one is able to build a tree of sensor and aggregation representation.

In this, the “labeling” of aggregation (but also to some extent of sensors) poses a problem with respect to ambiguity of such labels. Certain labels, such as “warm” or “cold” can have different meanings and can be highly subjective. For this, the notion of *ontologies* is helpful. These ontologies represent a collection of semantic descriptions of particular labels. A solution for N-RSA should support such notion.

4.2 Integration in the N-RSA System Architecture

This section is looking at how to integrate the data representation into the N-RSA with respect to the involved components.

An application that is interested in a particular piece of data, either in directly sensed form or derived through a particular aggregation, formulates a query (see Section 5 for the discussion on the used query language). Such query is based on the data representation model defined in this document.

Upon reception at the gateway, the Acquisition component forwards the query to the Query Resolver component in order to perform a resolving operation on this query. This operation determines the involved sensors in the query. It further determines the aggregation labels. The sensor data acquisition is handled locally through instantiating appropriate local procedures (outside the scope of this document). As for the aggregation labels, the Query

Nokia Research Center / Helsinki
Dana Pavel, Dirk Trossen

FINAL
24.10.06

Resolver determines whether or not the semantics of the particular label is known, i.e., whether or not the particular operation of this aggregation exists at the gateway. If so, appropriate procedures for this aggregation are instantiated locally. If the aggregation operation is unknown, the Code Repository component [2] is queried. This component determines, based on a particular ontology, the semantics of the unknown label and whether or not the aggregation operation according to this semantic is supported. If so, the appropriate operation is handed back to the gateway.

This process is repeated for all labels in the query to ensure that all necessary sensor data and aggregations are available to execute the given query.

4.3 Choice for Sensor & Aggregation Representation Language

When designing a remote monitoring system architecture we have to take into consideration that sensors are not going to be provided by the same manufacturer. Also, the system has to be able to cope with sensors that appear and disappear, without requiring manual reconfiguration every time this happens.

Most of the sensor representations work is done in a totally proprietary manner, either by various research groups or by manufacturers. However, due to the potential heterogeneity of the sensors in the system, a standard-based representation is most desirable in order to assure widespread adoption of the sensor descriptions. Hence, we refrained from defining an own format for N-RSA. Instead, we evaluated existing approaches with respect to their applicability for our architecture. In this evaluation, two approaches seemed the most promising, namely *SensorML* [3] and *TinyML* [5].

Though not yet a standard for this area, SensorML is probably the most advanced work in this direction. It is an XML-based sensor specification (schema), currently under standardization by the Open GIS consortium, and also considered as one of the most promising sensor description work by the Sensor Web [4] community¹.

The SensorML can be used for providing general information about sensors and sensor groups, and it enables discovery of sensors as well as processing and analysis of sensor measurements. Although the SensorML effort was mainly designed to deal with geospatial sensors, their authors tried to make it general enough so that it would be able to deal with any type of sensors. The sensor description is very detailed and includes information like sensor identification, location, constraints, platform attached to, coordinate reference system, sensor description, and measurement characteristics.

¹ The Sensor Web is an evolving effort that includes work performed in various research areas and places. The vision of the Sensor Web architecture is to allow for connecting distributed, heterogeneous sensors in an interoperable, intelligent, dynamic and flexible way. For this, sensor representation standards are required as well as proper system architectures that allow for hierarchies, dealing with dynamic sensor environments, easy to extend, capable of performing certain operations on sensor data even before it reaches the remote application.

Nokia Research Center / Helsinki
Dana Pavel, Dirk Trossen

FINAL
24.10.06

SensorML also defines a *SensorGroup* concept, which can be a sensor package or a sensor array. A sensor package is composed of multiple sensors that operate together to provide a collective observation or related group of observations. For example, a collection of sensors can be used for measuring wave velocity and direction. This could probably be used when describing the multi-sensor modules. A sensor array is a set of sensors of the same type but in different locations. The locations can be either on the same platform or on different platforms. A sensor array produces observations that are used to build a spatial coverage.

The following shows an example representation for a wind speed sensor in SensorML (see also ANNEX A for the high level diagram for SensorML).

```
<response id=ysi_wss_0001>
  <GeneralPropertyModel>
    <dynamicRange>
      <minimum>
        <Quantity observable type=#windSpeedunitOfMeasure=#mph>0
      </Quantity>
      </minimum>
      <maximum>
        <Quantity observable type=#windSpeedunitOfMeasure=#mph>134
      </Quantity>
      </maximum>
    </dynamicRange>
    <threshold>
      <Quantity bservableType=#windSpeedunitOfMeasure=#mph>2.2
    </Quantity>
    </threshold>
    <survivableRange>
      <maximum>
        <Quantity observableType=#windSpeedunitOfMeasure=#mph>220
      </Quantity>
      </maximum>
    </survivableRange>
    <operationalRange>
      <minimum>
        <Quantity
          observableType=#airTemeratureunitOfMeasure=#Celsius>-40
        </Quantity>
      </minimum>
      <maximum>
        <Quantity
          observableType=#airTemeratureunitOfMeasure=#Celsius>40
        </Quantity>
      </maximum>
    </operationalRange>
  </GeneralPropertyModel>
</response>
```

Nokia Research Center / Helsinki
Dana Pavel, Dirk Trossen

FINAL
24.10.06

Another work we consider as being very interesting is TinyML [5], an Intel Research project for sensor representation, currently being integrated in the TinyOS efforts². TinyML actually follows some of the ideas in SensorML, but the authors tried to reduce the amount of sensor metadata while adding some missing concepts. TinyML considers that sensors are potentially attached to *platforms*. A platform has a basic infrastructure (which includes a processor, an energy source and a communication device), sensors and/or actuators that the remote user can control in order to implement certain actions regarding the sensors.

In addition to simply grouping sensors, TinyML, unlike other efforts, allows for describing *virtual sensors*, where a virtual sensor can either be an interpreted output of another sensor, or a collection of multiple sensor outputs. With this, aggregation is enabled by defining intrinsic functions (identified through labels) for each of these virtual sensors. The usage of labels in this concept allows the linkage to ontologies **Error! Reference source not found.** as to semantically describing these labels (as described in Section 4.2). With this feature, TinyML is well suited to support hierarchical aggregations of sensed data (see also ANNEX B for the elements of the TinyML schema).

With the above in mind, we conclude that TinyML is the appropriate choice for the sensor and aggregation representation language in N-RSA, because:

- The concepts are similar to SensorML (the OpenGIS standard approach)
- It is simpler than SensorML, e.g., most of the geospatial-specific metadata has been removed
- It supports aggregation of sensor data through virtual sensor concept
- It supports ontologies through labeling intrinsic functions for virtual sensors
- It is integrated in TinyOS efforts, i.e., the SmartDust system, which is one of the main efforts in smart sensors, due to Intel's aggressive support.

Since interoperability is a key element in our decision-making, it would be desirable to finally come to an interoperable and standardized language that supports sensor and aggregation representation. Hence, it is worthwhile to closely follow the SensorML standardization efforts in order to see whether or not SensorML will be developing in this direction.

² TinyOS is used within the UC Berkeley and Intel Research research sensor platform *SmartDust*.

Nokia Research Center / Helsinki
Dana Pavel, Dirk Trossen

FINAL
24.10.06

5. QUERY LANGUAGE

The query language within the N-RSA is needed in order to formulate an acquisition request regarding data to be delivered back to the sender as part of the notifications sent by the intermediary gateway. The queries are formed based on the representation model that was introduced in Section 4, i.e., on TinyML.

Similar to choosing the representation model for N-RSA, interoperability of our chosen query language is a key element. Hence, we do not intend to invent our own query language if there does exist a standard-based approach.

Since our chosen sensor and aggregation representation language TinyML is an XML-based data representation, we decided to choose XQuery [7] as the query language in N-RSA. At the time of writing this document, XQuery was in the last call for becoming a W3C standard, which addresses our interoperability aspect when making our choice.

XQuery offers complex constructions and operations for querying nodes within an XML-based representation model. These nodes represent either the actual sensors or the intrinsic functions, i.e., the aggregation operations, within TinyML.

The integration within the N-RSA (see also Section 4.2 for a more general discussion) happens as follows, illustrated in Figure 1.

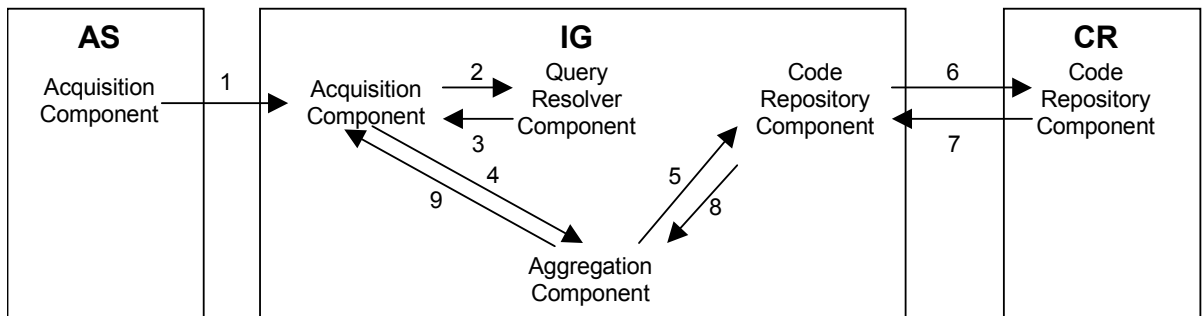


Figure 1 Query Handling in N-RSA

Upon receiving the AS' acquisition request (step 1, via the A_C interface [2]), the IG's Acquisition component forwards the query to the Query Resolver component (step 2, via the $A_{Q(local)}$ interface [2]). The Query Resolver verifies the validity of the query and extracts the sensor and aggregation labels of the query, returned to the Acquisition component in step 3.

For all sensor labels, the Acquisition component instantiates appropriate local sensor acquisition modules. The aggregation labels are forwarded to the Aggregation component (step 4 via the $A_{A(local)}$ interface [2]).

Nokia Research Center / Helsinki
Dana Pavel, Dirk Trossen

FINAL
24.10.06

For unknown labels, the Aggregation component requests the aggregation from the Code Repository component (step 5, via the $C_{A(local)}$ interface [2]), which contacts the Code Repository component at the Code Repository server [2] (step 6, via the C_R interface [2]).

The Code Repository server then provides an implementation based on the semantics of the particular label (using a particular ontology for these semantics, such as Anduril **Error! Reference source not found.**). Such implementation can consist of code modules that are instantiated at the intermediary gateway. For simple aggregations though, such as “maximum temperature”, the particular label could potentially be expressed as an XQuery *function* [7]. In this case, the appropriate XQuery function definition is delivered back to the Code Repository at the IG (step 7, via the C_R interface [2]), which forwards the result to the Query Resolver (step 8 via the $C_{A(local)}$ interface [2]) to be included in the original query as a replacement for the unknown label.

The Aggregation component finally responds in step 9, via the $A_{A(local)}$ interface [2], by returning either the object pointers for the particular (executable-based) aggregation implementation or the XQuery function code (returned as an octetstream).

The Acquisition component then instantiates the XQuery execution by replacing the labels with the current measurements and aggregations, according to the determined semantics.

6. REFERENCES

- [1] D. Trossen, D. Pavel, “N-RSA High-Level System Architecture”, N-RSA 2004 project report, January 2004
- [2] D. Trossen, D. Pavel, “N-RSA High-Level System Architecture: Functionality & Interface Description”, N-RSA project report, February 2004
- [3] SensorML @ Open GIS - <http://www.opengis.org/functional/?page=swe>
- [4] Vincent Tao, “The Smart Sensor Web”, GEO World, Sept. 2003
- [5] Nathan Ota and William T.C. Kramer, “TinyML: Meta-data for Wireless Networks”, <http://kingkong.me.berkeley.edu/~nota/research/TinyML/project-paper-1.pdf>
- [6] J. Hyryläinen, I.Jantunen “SSI Protocol Specification V1.0”, April 2005, available at http://ssi-protocol.net/SSI%20protocol%20specification_10a-2.pdf
- [7] D. Draper et al., “XQuery 1.0 and XPath 2.0 Formal Semantics”, W3C Work Draft, February 2004, <http://www.w3.org/TR/xquery-semantics/>

Nokia Research Center / Helsinki
Dana Pavel, Dirk Trossen

FINAL
24.10.06

ANNEX A HIGH LEVEL DIAGRAM OF SENSOR ML

The following picture shows the high level diagram of Sensor ML [3]:

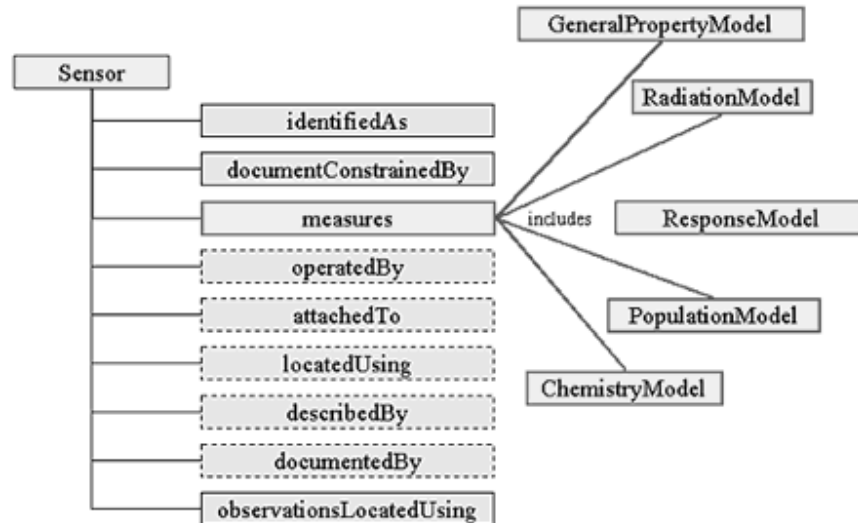


Figure 2 SensorML: High Level Diagram

ANNEX B ELEMENTS OF TINY ML

The following list shows the elements of the TinyML schema [5]:

superSensorField A collection of sensor fields.

SensorField A basic sensor network – consisting of platforms, sensors and actuators and reference data

Platform The basic physical platform in a sensor field, which contains processing and communication elements. It may also contain on or more sensors and actuators. An example is the MICA ‘mote’

BasicSensor A basic physical sensor that is attached to a platform. It is identified by type.

BasicActuator A basic physical actuator that is attached to a platform. It is identified by type.

FieldReference Appearing once in a Sensor Field, this is location information that may be used to connect the sensor information to the real world reference. An example would be a node with GPS location ability.

Nokia Research Center / Helsinki
Dana Pavel, Dirk Trossen

FINAL
24.10.06

FieldDescription A description of a sensor field consisting of routing description, a field description and other information

FieldSensor This is a virtual sensor associated with a set of sensors on different platforms in a sensor field. These could be basic sensors or virtual sensors. An example is getting the average temperature value for an entire sensor field.

FieldActuator This is a virtual actuator associated with a set of actuators on different platforms in a sensor field. These could be basic actuators or virtual actuators.

VirtualSensor A virtual physical sensor that is associated with one or more basic sensors on a platform. The virtual sensor performs a transformation on physical sensor data according to a function. An example is a virtual sensor that transforms a thermistor value (with calibration) into a reading in Celsius.

VirtualActuator A virtual physical actuator that is associated with one or more basic actuators on a platform. The virtual actuator performs a transformation on physical actuator according to a function.

Query A complex flag that is associated with a number of elements. If set to TRUE, then the fields associated with the flag are returned. This allows any element in the sensor field to be queried. Query has an optional start and duration time.

SetFlag A complex Boolean flag that is associated with a number of elements. If set to TRUE, and the element is settable, then the fields associated with the flag are set to the provided value(s). Examples of use are setting the function associated with a virtual sensor or setting the values of an actuator. SetFlag also has a start time and duration.

Type A general element the definition for the type of a component or element. This could be a manufacturers part number or a specification that indicates the type of a photo sensor.

Vendor A general element describing the vendor of the component. This element consists of Vendor Description and a Universal Resource Indicator (URI).

Range A description of the range of a sensor or actuator. It is composed of a distance measure and the units used.

Accuracy Describes the accuracy of a component.

Location The location of the parent element. It can be either absolute or relative coordinates.

Orientation The orientation description for a platform, sensor or actuator. It is in one of several formats

Nokia Research Center / Helsinki
Dana Pavel, Dirk Trossen

FINAL
24.10.06

Hardware An element that describes the hardware of a platform. It could consist of descriptions for processor, radio and energy source.

Software The software associated with a platform. This includes the OS description and the version.

Function The function specification for a virtual sensor or actuator, including field sensors and actuators. This function can be either preset or specified by a user if allowed.

SpanofMembers A list that is used to describe a virtual sensor actuators and other composite components.

Coordinates An element that describes coordinates for several other elements. It consists of an X and Y values, and possibly a Z and timestamp.

Datavalues The element for any data values from a sensor or actuator. Data values can be integer, floating point or string.

CurrentSettings The current setting of an actuator.

Timestamp A timestamp for data value readings and settings and for query responses

EnergySource The energy source of a platform. Components are the Power level and Source Description ID The ID field for a component. This is used by sensor fields and platforms primarily.

ModelDescription The platform model as part of the description of the platform.

Absolute One of three types of orientation specifications. This is in absolute X, Y, Z values

Quaternion Quaternion extends the concept of rotation in three dimensions to rotation in four dimensions. This avoids the problem of "gimbal-lock" and allows for the implementation of smooth and continuous rotation. In effect, they may be considered to add a additional rotation angle to spherical coordinates, i.e., Longitude, Latitude and Rotation angles that are calculated from the combination of the three coordinates of the rotation axis and the rotation angle.

Euler An orientation specification in Eulian angles – Pitch, Roll and Yaw

Xvalue A value representing an X value

Yvalue A value representing an Y value

Zvalue A value representing an Z value

Nokia Research Center / Helsinki
Dana Pavel, Dirk Trossen

FINAL
24.10.06

Svalue orientation	A value representing an S value for a quaterion description of the orientation
Pitch (wings)	The angle relative to the X axis (e.g. on an aircraft the axis defined by the wings)
Roll (tail to nose)	The angle relative to the Y axis (e.g. on an aircraft the axis defined from tail to nose)
Yaw (pointing up)	The angle relative to the Z axis (e.g. on an aircraft the axis defined by pointing up)
Value-int	An integer value – a sub element of Datavalues
Value-flt	A floating point value – a sub element of Datavalues
Value-string	A string value – a sub element of Datavalues
Processor	An element describing a processor on a platform – consisting of a description and speed specification
ProcessorSpeed	The clock rate of a processor
ProcessorDescription	The description of a processor
Radio	An element to describe a radio on a platform
Routing	A description of the routing of a sensor field
Reference	A reference description of a sensor field
OtherInformation	A description of the other information related to a sensor field
Units	A string describing the units of a range or value.
PowerLevel	The power level of an energy source on a platform
SourceDescription	A description of an energy source. Examples include battery or photo
OSDescription	The operating systems description.
Version	A string description of the version (for the software, hardware, etc.
RelativeLocation	The relative location of a platform or other component
AbsoluteLocation	An absolute location of a platform or other component

Nokia Research Center / Helsinki
Dana Pavel, Dirk TrossenFINAL
24.10.06

Distance	The distance of a range specification
Qflag	The Boolean flag associated with a query Request
StartTime	The time a query is supposed to start (for a request) or time it did start (for a response).
Duration	The duration of a query request
Date	The date of a time stamp
Time	The time of a time stamp
VendorDescription	The string description of a vendor
URI	A Universal Resource Indicator – the XML version of a URL.