IoT Management and Control Architecture Overview

For UPnP Version 1.0

Status: Standardized DCP (SDCP)

Date: July 1, 2013

Document Version: 1.0

Service Template Version: 2.00

Authors [a]	**Company**
Clarke Stevens	Cablelabs
Jangwook Park (Vice-Chair)	LGE
Paul Jeon (Vice-Chair)	LGE
Russell Berkoff (Chair)	Samsung Electronics
Danilo Santos	Signove
Gerhard Mekenkamp	TPVision

[a] The UPnP forum in no way guarantees the accuracy or completeness of this author list and in no way implies any rights for or support from those members listed. This list is not the specifications' contributor list that is kept on the UPnP Forum's website.

NOTE: While the name of this document is IoT Management and Control to make clear that it provides many of the interfaces needed for support of a complete IoT environment implemented using UPnP technology, the name used in normative sections is SensorManagement. SensorManagement continues to be supported at this level for historical reasons.

CONTENTS

1 Scope

1.1 Introduction

This document describes the overall UPnP IoTManagementAndControl Architecture, which forms the foundation for the UPnP IoTManagementAndControl device [11] and UPnP DataStore service [13] specifications. The IoTManagementAndControl device hosts services to bridge sensor devices connected to both UPnP networks as well as non-UPnP based networks. The DataStore service provides persistent retention and distribution of both sensor data as well as data from mobile devices which may leave the UPnP network at any time. This service can be hosted within the UPnP IoTManagementAndControl device as well as within other UPnP compliant devices.

1.2 Goals

The UPnP IoTManagementAndControl Architecture was explicitly defined to meet the following goals:

- Describe sensors and actuators residing on both UPnP and non-UPnP networks.
- Provide data transport services for sensors and actuators to UPnP network clients.
- Define a service to describe, retain and distribute data received from sensors as well as other non-persistent data sources.

- Define an allowed device protection model for both the sensor and data retention components.

1.3 Non-Goals

The following are not initial goals of the IoTManagementAndControl architecture:

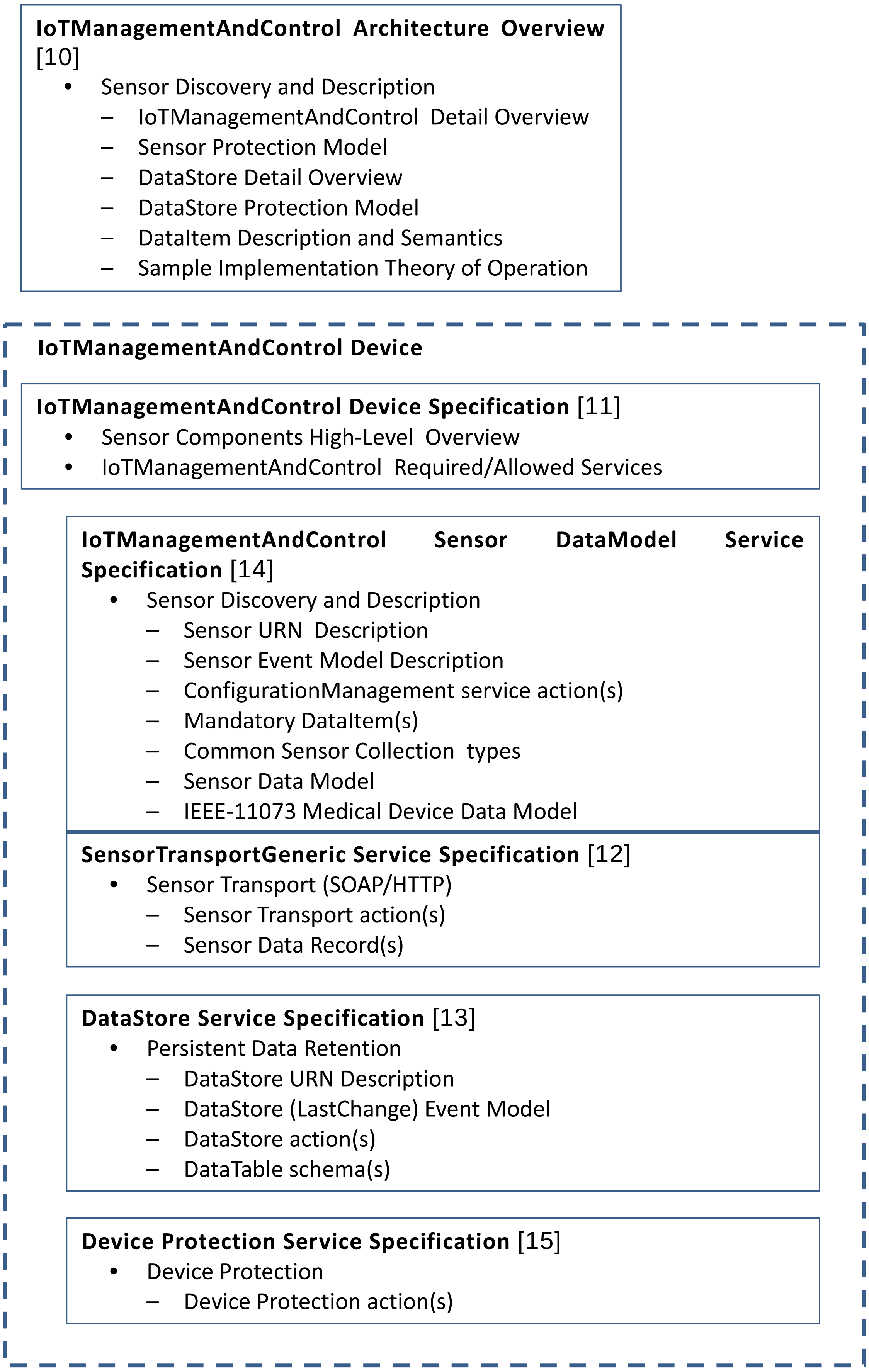
- Low-level control of bridged networks

  The initial version of UPnP IoTManagementAndControl treats Sensors and Actuators as abstract data sources and sinks and does not expose details or provide direct access to bridging network protocols. Low-level control of selected bridged network protocols will be considered in subsequent versions of the architecture.

- Low-latency control of sensors and actuators

  The initial version of UPnP IoTManagementAndControl treats sensors and actuators as autonomous objects requiring relatively infrequent supervision from home-network clients. Closed loop control of sensor and actuator pairs is better accomplished directly within the internal vendor-device sensor/actuator architecture with UPnP home-network clients providing overall supervision. However, UPnP IoTManagementAndControl does support sensors which have substantial throughput requirements using transport connections.

1.4 IoTManagementAndControl and DataStore Specification Map



IoTManagementAndControl Architecture Overview [10]

- Sensor Discovery and Description
  - IoTManagementAndControl Detail Overview
  - Sensor Protection Model
  - DataStore Detail Overview
  - DataStore Protection Model
  - DataItem Description and Semantics
  - Sample Implementation Theory of Operation

IoTManagementAndControl Device

IoTManagementAndControl Device Specification [11]

- Sensor Components High-Level Overview
- IoTManagementAndControl Required/Allowed Services

IoTManagementAndControl Sensor DataModel Service Specification [14]

- Sensor Discovery and Description
  - Sensor URN Description
  - Sensor Event Model Description
  - ConfigurationManagement service action(s)
  - Mandatory DataItem(s)
  - Common Sensor Collection types
  - Sensor Data Model
  - IEEE-11073 Medical Device Data Model

SensorTransportGeneric Service Specification [12]

- Sensor Transport (SOAP/HTTP)
  - Sensor Transport action(s)
  - Sensor Data Record(s)

DataStore Service Specification [13]

- Persistent Data Retention
  - DataStore URN Description
  - DataStore (LastChange) Event Model
  - DataStore action(s)
  - DataTable schema(s)

Device Protection Service Specification [15]

- Device Protection
  - Device Protection action(s)

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

[1] UPnP Device Architecture, version 1.0, UPnP Forum, June 13, 2000. Available at: http://upnp.org/specs/arch/UPnPDA10_20000613.pdf. Latest version available at: http://upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.0.pdf.

[2] ISO 8601 Data elements and interchange formats – Information interchange -- Representation of dates and times, International Standards Organization, December 21, 2000. Available at: ISO 8601:2000.

[3] IETF RFC 2119, Key words for use in RFCs to Indicate Requirement Levels, S. Bradner, 1997. Available at: http://www.faqs.org/rfcs/rfc2119.html.

[4] HyperText Transport Protocol – HTTP/1.1, R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, June 1999. Available at: http://www.ietf.org/rfc/rfc2616.txt.

[5] IETF RFC 3339, Date and Time on the Internet: Timestamps, G. Klyne, Clearswift Corporation, C. Newman, Sun Microsystems, July 2002. Available at: http://www.ietf.org/rfc/rfc3339.txt.

[6] Extensible Markup Language (XML) 1.0 (Third Edition), François Yergeau, Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, eds., W3C Recommendation, February 4, 2004. Available at: http://www.w3.org/TR/2004/REC-xml-20040204.

[7] XML Schema Part 2: Data Types, Second Edition, Paul V. Biron, Ashok Malhotra, W3C Recommendation, 28 October 2004. Available at: http://www.w3.org/TR/2004/REC-xmlschema-2-20041028.

[8] XML Path Language (XPATH) Version 1.0. James Clark, Steve DeRose, W3C Recommendation 16 November 1999. Available from: http://www.w3.org/TR/1999/REC-xpath-19991116.

[9] ISO/IEEE-11073-20601 Health informatics - Personal health device communication - Part 20601: Application Profile - Optimized exchange protocol, 2010. Available at: http://www.iso.org/iso/search.htm?qt=11073&searchSubmit=Search&sort=rel&type=simple&published=true

[10] UPnP Sensor and DataStore Architecture Overview, UPnP Forum, July 1, 2013. Available at: http://www.upnp.org/specs/smgt/UPnP-smgt-IoTManagementAndControlArchitectureOverview-v1-20130701.pdf. Latest version available at: http://www.upnp.org/specs/smgt/UPnP-smgt-IoTManagementAndControlArchitectureOverview-v1.pdf.

[11] UPnP IoTManagementAndControl:1 Device, UPnP Forum July 1, 2013. Available at: http://www.upnp.org/specs/smgt/UPnP-smgt-IoTManagementAndControl-v1-Device-20130701.pdf. Latest version available at: http://www.upnp.org/specs/smgt/UPnP-smgt-IoTManagementAndControl-v1-Device.pdf.

[12] UPnP SensorTransportGeneric:1 Service, UPnP Forum July 1, 2013. Available at: http://www.upnp.org/specs/smgt/UPnP-smgt-SensorTransportGeneric-v1-Service-20130701.pdf. Latest version available at: http://www.upnp.org/specs/smgt/UPnP-smgt-SensorTransportGeneric-v1-Service.pdf.

[13] UPnP DataStore:1 Service, UPnP Forum, July 1, 2013. Available at: http://www.upnp.org/specs/smgt/UPnP-smgt-DataStore-v1-Service-20130701.pdf. Latest version available at: http://www.upnp.org/specs/smgt/UPnP-smgt-DataStore-v1-Service.pdf.

[14] UPnP IoTManagementAndControl Sensor DataModel Service, UPnP Forum, July 1, 2013. Available at: http://www.upnp.org/specs/smgt/UPnP-smgt-SensorDataModel-v1-Service-20130701.pdf. Latest version available at: http://www.upnp.org/specs/smgt/UPnP-smgt-SensorDataModel-v1-Service.pdf.

[15] UPnP DeviceProtection:1 Service, UPnP Forum, February 24, 2011. Available at: http://www.upnp.org/specs/gw/UPnP-gw-DeviceProtection-v1-Service-20110224.pdf. Latest version available at: http://www.upnp.org/specs/gw/UPnP-gw-DeviceProtection-v1-Service.pdf.

[16] UPnP ConfigurationManagement:2 Service, UPnP Forum, February 16, 2012. Available at: http://www.upnp.org/specs/dm/UPnP-dm-ConfigurationManagement-v2-Service-20120216.pdf. Latest version available at: http://www.upnp.org/specs/dm/UPnP-dm-ConfigurationManagement-v2-Service.pdf.

[17] XML Schema DataStore LastChange Eventing, UPnP Forum, July 1, 2013. Available at: http://www.upnp.org/schemas/ds/dsevent-v1-20130701.xsd. Latest version available at: http://www.upnp.org/schemas/ds/dsevent.xsd.

[18] XML Schema UPnP DataStore DataStoreInfo, UPnP Forum, July 1, 2013. Available at: http://www.upnp.org/schemas/ds/dsinfo-v1-20130701.xsd. Latest version available at: http://www.upnp.org/schemas/dsinfo.xsd.

[19] XML Schema UPnP DataStore DataTableInfo, UPnP Forum, July 1, 2013. Available at: http://www.upnp.org/schemas/ds/dtinfo-v1-20130701.xsd. Latest version available at: http://www.upnp.org/schemas/ds/dtinfo.xsd.

[20] XML Schema UPnP DataStore DataStoreGroups, UPnP Forum, July 1, 2013. Available at: http://www.upnp.org/schemas/ds/dsgroups-v1-20130701.xsd. Latest version available at: http://www.upnp.org/schemas/ds/dsgroups.xsd.

[21] XML Schema UPnP DataStore DataRecord, UPnP Forum, July 1, 2013. Available at: http://www.upnp.org/schemas/ds/drecs-v1-20130701.xsd. Latest version available at: http://www.upnp.org/schemas/ds/drecs.xsd.

[22] XML Schema UPnP DataStore DataRecordFilter, UPnP Forum, July 1, 2013. Available at: http://www.upnp.org/schemas/ds/drecfilter-v1-20130701.xsd. Latest version available at: http://www.upnp.org/schemas/ds/drecfilter.xsd.

[23] XML Schema UPnP DataStore DataRecord Status, UPnP Forum, July 1, 2013. Available at: http://www.upnp.org/schemas/ds/drecstatus-v1-20130701.xsd. Latest version available at: http://www.upnp.org/schemas/ds/drecstatus.xsd.

[24] XML Schema UPnP IoTManagementAndControl DataRecord Information, UPnP Forum, July 1, 2013. Available at: http://www.upnp.org/schemas/smgt/srecinfo-v1-20130701.xsd. Latest version available at: http://www.upnp.org/schemas/smgt/srecinfo.xsd.

[25] XML Schema UPnP IoTManagementAndControl Sensor DataModel DataItem Description, UPnP Forum, July 1, 2013. Available at: http://www.upnp.org/schemas/smgt/sdmdid-v1-20130701.xsd. Latest version available at: http://www.upnp.org/schemas/smgt/sdmdid.xsd.

3 Terms, definitions and abbreviations

For initial readers of this document, it may be helpful to proceed to subclause 4, "Architectural Overview" and refer to this section for terms and definitions as needed.

3.1 IoTManagementAndControl terms and definitions

The following terms apply to the IoTManagementAndControl Device [11], IoTManagementAndControl Sensor DataModel Service [14], and IoTManagementAndControl SensorTransportGeneric Service [12] specifications.

3.1.1 Sensor

A Sensor defines sets of named DataItem(s) (see subclause 3.2) which can be read and/or written via the SensorTransportGeneric service [12]. Access to Sensor DataItem(s) is provided by UPnP SOAP actions or via HTTP sensor transport connections. Discovery, control and status of Sensor(s) are provided through parameter(s) defined by the IoTManagementAndControl Sensor DataModel service [14]. Depending on the type of Sensor, additional sensor control and status parameters may also be defined by additional sensor ecosystem specific specifications.

3.1.2 SensorCollection

A SensorCollection contains zero or more Sensors. The Sensors enumerated within a SensorCollection typically relate to a real world device. In some instances the Sensors may be directly related to the operation of the device such as an IEEE-11073 Medical Device System [9], while in other instances, Sensors may indirectly assist in performing monitoring and possibly control operations on the device; such as sensors within a refrigerator or home thermostat. SensorCollections may identify other UPnP Devices which can provide additional command and control functionality.

3.1.3 SensorGroup

Sensors can be grouped using SensorGroup(s). An individual Sensor may participate (be a member of) zero or more SensorGroup(s). SensorGroup(s) may include Sensors from multiple SensorCollection(s). In addition to identifying related Sensors, SensorGroup(s) also can be assigned DeviceProtection permissions which can then be used as DeviceProtection Roles.

3.1.4 SensorID

A SensorID is an identifier which uniquely identifies a Sensor to SOAP actions implemented by the SensorTransportGeneric service [12]. SensorID values are provided by the IoTManagementAndControl Sensor DataModel service [14] *SensorID* parameter node.

3.1.5 SensorNode

A SensorNode is a set of IoTManagementAndControl Sensor DataModel [14] nodes which describe a Sensor and the DataItem(s) it supports. The required and allowed SensorNode parameters and corresponding syntax and semantics are detailed by IoTManagementAndControl Sensor DataModel service specification.

3.1.6 SensorPath

A PartialPath (see subclause 3.3) to a SensorNode.

3.1.7 SensorURN

A SensorURN(s) identifies sets of required and allowed DataItem(s) that a Sensor provides. The syntax and semantics of SensorURN(s) is detailed in the IoTManagementAndControl Sensor DataModel service specification [14], and subclause 5.2.6, "Sensor Normative Type Identifiers".

3.2 DataStore terms and definitions

The following terms apply to the DataStore Service specification [13],

3.2.1 DataItem

A DataItem identifies an output from a Sensor (or input to an Actuator). A DataItem is defined by a name, data type and encoding. See subclause 4.3, "DataItem Semantics" for a detailed description of DataItem(s).

3.2.2 DataRecord

A DataRecord consists of a defined set of DataItem(s). DataRecord(s) may contain DataItem(s) from multiple sensor types as well as containing DataItem(s) from multiple sensors of the same type.

3.2.3 DataStore

A DataStore is a collection of DataTable(s).

3.2.4 DataStore Group

DataTable(s) within a DataStore shall be a member of one or more groups. Membership in a group indicates the participating DataTable(s) are related in some way. A given DataTable may participate in multiple groups. The DataStore service however, only defines membership in a group and access rights to a group. Additional ecosystem specific requirements may provide further rules and meaning attached to group membership.

3.2.5 DataTable

A DataTable consists of a collection of DataRecord(s) and a key/value Dictionary managed by a DataStore service. All DataRecord(s) in a DataTable share a common DataRecord format.

3.2.6 DataTable Dictionary

A DataTable provides a key/value data structure containing information about the DataTable. DataItem(s) stored in the DataTable records may refer to entries stored in the DataTable's Dictionary.

3.2.7 DataTable GUID

A globally unique identifier of a DataTable.

3.2.8 DataTableID

A UPnP action argument identifying a specific DataTable (see DataTable GUID).

3.2.9 DataTable URN

A uniform resource name (URN) which identifies the contents stored in a DataTable. See DataStore service [13], subclause 5.3.4, "DataTable URN" for further information.

3.3 Configuration Management terms and definitions

The following terms apply to the ConfigurationManagement Service [16]. The IoTManagementAndControl Sensor DataModel service [14] defines a profile of ConfigurationManagement service.

3.3.1 Leaf

A leaf element in the data model’s logical tree.

3.3.2 Node

An element in the logical tree that represents the hierarchical structure of the data model.

3.3.3 Path

A string representation of the sequence of Node(s) starting with a Root Node and ending at the Node of interest. Specifically it’s the concatenation of the Node names.

3.3.4
PartialPath
A Path from the Root to a Node in the data model tree which is not a Leaf.

3.3.5
ParameterInitializationPath
A sequence of Node(s) starting from SingleInstance Node and ending to the Leaf Node.

3.3.6
Root
The root element of the data model's logical tree.

3.4 Device Protection terms and definitions

The following terms apply to the DeviceProtection service [15].

3.4.1
Device Protection Roles
The UPnP DeviceProtection service associates control points with device protection roles which define allowable operations a control point may perform on a UPnP service. UPnP DeviceProtection service roles are a union of basic roles defined by the UPnP DeviceProtection service and device protection roles defined by the UPnP service subject to device protection. See UPnP DeviceProtection service [15] and subclause 4.1.9, "Sensor Collection Protection Model" and subclause 4.2.6, "DataStore Protection Model" for further details.

3.4.2
Control Point Identity
The UPnP DeviceProtection service associates control point identities with device protection roles. A UPnP control point may have its own assigned roles as well as roles resulting from particular end-users being associated with the control point.

4 Architectural Overview

The following figure shows the high-level relationship of various components within the UPnP Sensor Management ecosystem. On the left side of the diagram are various Home Automation, Personal Health and Mobile/Lifestyle devices which can support non-UPnP remote network protocols. The UPnP IoTManagementAndControl device near the center of this figure provides services to discover and connect these devices to clients on the UPnP Home-Network. The UPnP DataStore service which can be hosted in the IoTManagementAndControl device and/or in one or more UPnP Home-Network devices can retain data from both UPnP IoTManagementAndControl devices as well as other UPnP home-network clients.

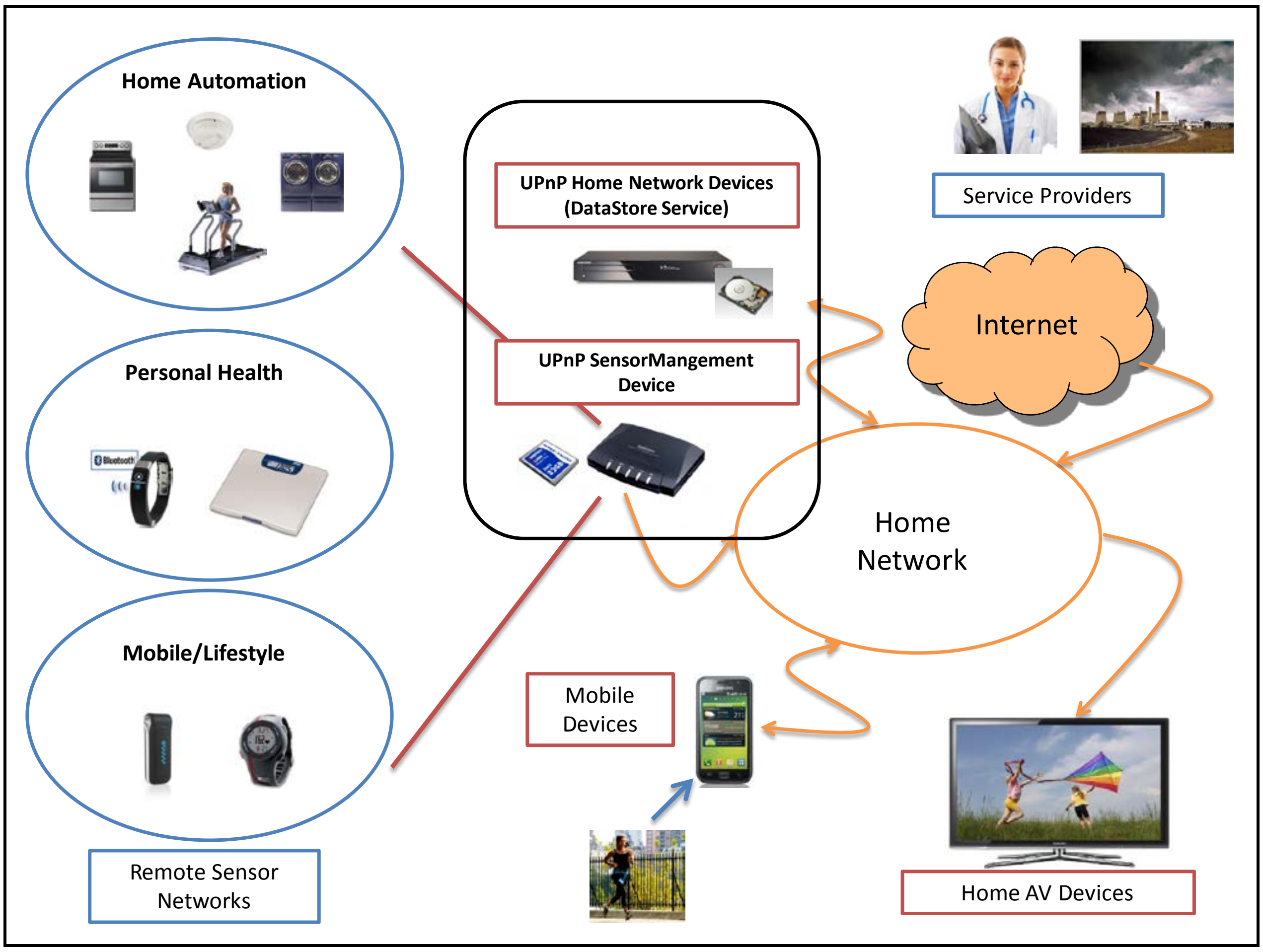


Figure 1 — IoTManagementAndControl Architectural Components

4.1 UPnP IoTManagementAndControl Device

The IoTManagementAndControl device provides the following services:

- Sensor Description
- Sensor Data Transport
- Device Protection (opt)
- Vendor defined services (opt)

The following figure shows component services contained in the UPnP IoTManagementAndControl device:

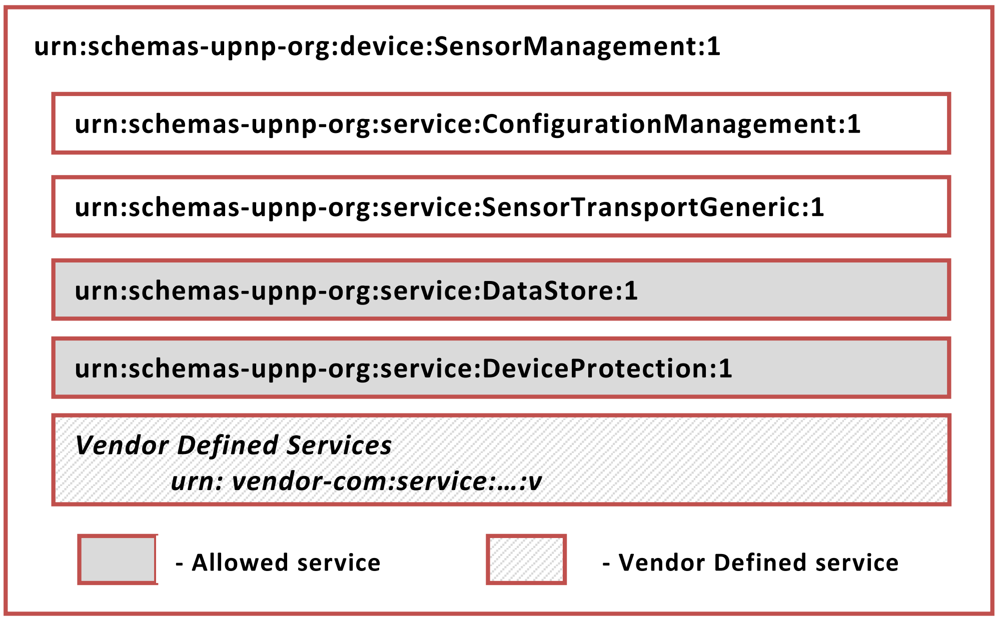

urn:schemas-upnp-org:device:SensorManagement:1
urn:schemas-upnp-org:service:ConfigurationManagement:1
urn:schemas-upnp-org:service:SensorTransportGeneric:1
urn:schemas-upnp-org:service:DataStore:1
urn:schemas-upnp-org:service:DeviceProtection:1
Vendor Defined Services
urn: vendor-com:service:…:v
- Allowed service
- Vendor Defined service

Figure 2 — IoTManagementAndControl Device services

4.1.1 Sensor Data Model

The UPnP IoTManagementAndControl device includes an instance of the UPnP ConfigurationManagement service [16]. The UPnP IoTManagementAndControl Sensor DataModel service specification [14] defines the IoTManagementAndControl Sensor DataModel service and a profile for hosting the ConfigurationManagement service within the IoTManagementAndControl device. The IoTManagementAndControl Sensor DataModel service provides the primary method for UPnP home-network clients to identify sensors and actuators supported by the IoTManagementAndControl device.

4.1.2 Sensor Collections

UPnP IoTManagementAndControl groups Sens*ors* within SensorCollections(s) which provide both physical and logical context for otherwise unrelated sensors. A simple example would be a refrigerator. This device would likely support a number of different types of sensors which perform various functions such as compartment temperature control and monitoring, power usage and door status. However, given the myriad of real-world uses of sensors, UPnP does not typically attempt to define a standard sensor model for refrigerators. Instead, UPnP IoTManagementAndControl advertises the type of the collection, allowing UPnP control points which recognize the advertised collection type to provide user-interface and control services.

4.1.3 Sensor Control

Sensors managed by UPnP IoTManagementAndControl are generally expected to operate without direct supervision by UPnP home-network clients. However, the IoTManagementAndControl device does provide basic facilities to enable automatic polling of sensors which require it. These facilities are controlled by parameter nodes in the UPnP IoTManagementAndControl Sensor DataModel service [14]. In addition, sensor types which require more detailed controls may define additional control parameters under the *CollectionSpecific* and *SensorSpecific* nodes of the UPnP IoTManagementAndControl Sensor DataModel service.

4.1.4 Sensors DataItem(s)

UPnP treats Sensor(s) as sources of data. UPnP defines the term DataItem as a data source (or sink) with a "name", an associated "type" and "encoding". The data type associated with a Sensor DataItem may range from simple scalar quantities to an XML document depending on sensor's parent ecosystem. Each Sensor may define multiple DataItem(s) which are grouped under a *SensorURN* node.

4.1.5 Sensor DataItem Description Document

Many sensors generate simple data and in these instances a text-based description of the sensor accompanied by information on the method of sampling, (average, instantaneous) and the units of the measurement are sufficient to interpret the information provided by a DataItem. To facilitate the description of DataItem(s), UPnP IoTManagementAndControl defines DataItem Descriptive XML schema [25]. This XML document is provided by the Sensor's DataItem's *Description* parameter node (see subclause 4.4, "DataItem Description XML Document").

4.1.6 Sensors DataRecord(s)

A UPnP IoTManagementAndControl device can advertise sensors which may generate one or more DataItem(s). A UPnP client can then inspect the DataItem(s) advertised for the target Sensor and can request generation of DataRecord(s) consisting of a requested subset of DataItem(s) provided by the Sensor.

4.1.7 Sensors Data Transport

A UPnP IoTManagementAndControl devices supports two methods for obtaining sensor data:

- HTTP/HTTPS based transport connections
- SOAP based transport

When using HTTP transport connections, a IoTManagementAndControl control point provides a transport URL that can accept DataRecord(s) produced by the IoTManagementAndControl device. The control point then issues SOAP actions to the IoTManagementAndControl device providing the transport URL and the DataItem(s) to be included in the DataRecord(s) provided. Once established, the IoTManagementAndControl device will connect directly to the HTTP server identified by the transport URL and will deliver DataRecord directly without further control point interaction. This method is suited to cases where a sensor may either generate large amounts of data as well as sensors which generate infrequent data requests. In addition, this method allows the IoTManagementAndControl device to support multiple independent transport connections allowing sensor data to be replicated to multiple UPnP clients.

When using SOAP based transport, the IoTManagementAndControl device generates an event indicating one or more Sensor(s) has data available to be read. A UPnP IoTManagementAndControl control point can then issue a SOAP action to request DataRecord(s) from the Sensor IoTManagementAndControl device SensorGenericTransport service.

4.1.8 Sensors Groups

While SensorCollections relate sensors within a particular device, additional sensor relationships are useful both for implementing DeviceProtection as well as for context based discovery of related sensors. For example temperature measurement sensors may appear in a number of contexts such as inside of a refrigerator as well as within a room thermostat or hot-water system. Therefore SensorGroup(s) support relating sensor(s) which participate in one or more context-based groups, such a group named "HomeTemperatureControl" allow UPnP clients to locate sets of sensors that serve a particular context.

4.1.9 Sensor Protection Model

SensorGroup(s) may also be used to construct DeviceProtection roles for Sensors. In this usage a DeviceProtection role is constructed from a SensorPermission and a SensorGroup name as follows:

```
[SensorRole]::=[SensorPermission]#[SensorGroup]
```

Table 1 — Sensor Permissions

Permission	Description
smgt:ReadSensor	A control point is permitted to issue ReadSensor() actions to the corresponding Sensor.
smgt:WriteSensor	A control point is permitted to issue WriteSensor() actions to the corresponding Sensor.
smgt:ConnectSensor	A control point is permitted to issue ConnectSensor() and DisconnectSensor() actions to the corresponding Sensor.
smgt:CommandSensor	A control point is permitted to modify IoTManagementAndControl properties in the DataModel.
smgt:ViewSensor	A control point is permitted to view IoTManagementAndControl properties for this Sensor.

When the DeviceProtection feature is implemented the IoTManagementAndControl Sensor DataModel service shall support the following roles:

- Admin – A control point with the Admin role can view/read/write/connect/command any sensor.
- Public – A control point with the Public role can read or write specific Sensors which permit this access.
- Basic – A control point with the Basic role can read or write specific Sensors which permit this access.

See the following IoTManagementAndControl Sensor DataModel service parameter for further details on the implementation of default DeviceProtection roles:

/UPnP/SensorMgt/SensorCollections/#/Sensors/#/SensorDefaultPermissions/

In addition, the IoTManagementAndControl Sensor DataModel service which implements the DeviceProtection feature shall support the following group roles:

- smgt:ReadSensor#[SensorGroup] - A control point with this role is permitted to issue ReadSensor() actions to Sensor(s) belonging to the corresponding SensorGroup.
- smgt:WriteSensor#[SensorGroup] - A control point with this role is permitted to issue WriteSensor() actions to Sensor(s) belonging to the corresponding SensorGroup.
- smgt:CommandSensor#[SensorGroup] - A control point with this role is permitted to write IoTManagementAndControl Sensor DataModel service parameter values to Sensor(s) belonging to the corresponding SensorGroup.
- smgt:ViewSensor#[SensorGroup] - A control point with this role is permitted to view IoTManagementAndControl properties for Sensor(s) belonging to the corresponding SensorGroup.

4.2 UPnP DataStore Service

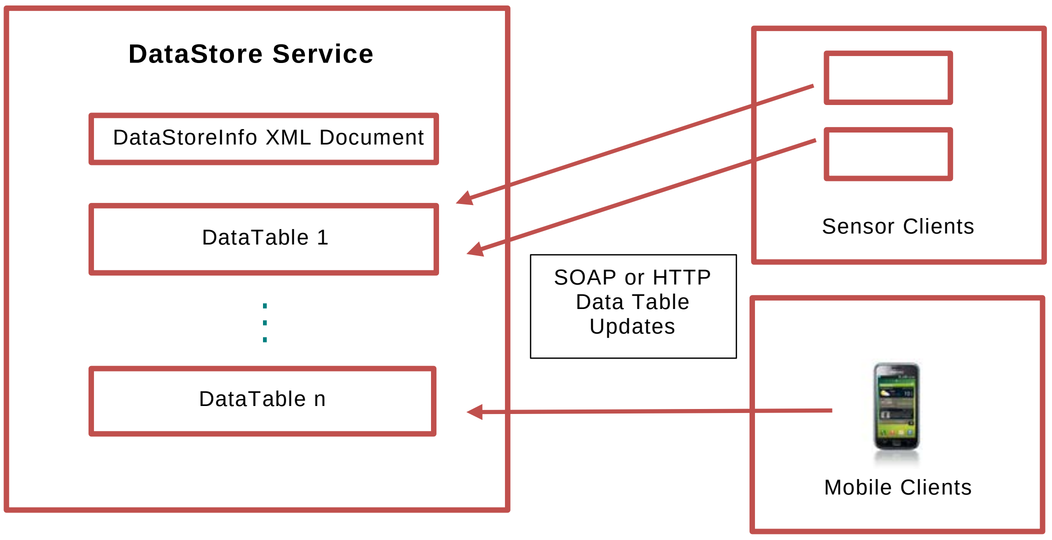


Figure 3 — DataStore service

The DataStore service provides the ability to acquire and persistently store information for later access. This service allows UPnP devices such as mobile phones and sensors to make information available for subsequent retrieval. This increase the flexibility of the UPnP ecosystem by eliminating requirements to have an immediate nexus between information sources and sinks on the UPnP network. The *DataStore* service additionally allows UPnP devices with limited or temporary storage capabilities to persist information for subsequent retrieval. The *DataStore* service constructs are intended to be modelled after and compatible with well-established database models.

The service defined herein provides the following functionality:

- Methods to define, create and delete tables of data records.
- Methods to define and identify the contents a data records.
- Methods to accept data records from both streaming and programmed sources.
- Methods to select and retrieve data record contents.

4.2.1 DataTable(s)

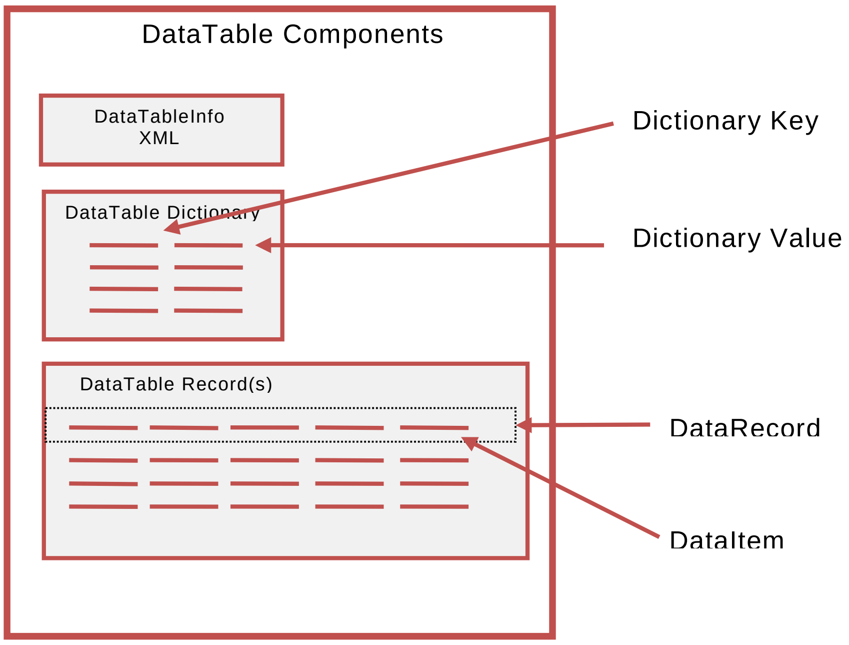


Figure 4 — DataStore service - DataTable Components

A DataTable consists of identifying metadata, a set of DataRecord(s) and a Dictionary. The DataStore service also is allowed to provide access control to DataTable(s) via the UPnP DeviceProtection service. The device hosting the DataStore service may create pre-defined DataTable(s) or the DataStore service may allow control points to dynamically create DataTable(s).

4.2.2 DataTable DataRecord(s)

When a DataTable is created (or pre-defined) a DataRecord format is established for the table. Information about each DataTable is conveyed by an XML document including identifying information, DeviceProtection roles, and DataRecord contents (see XML Schema UPnP DataStore DataStoreInfo [19]). The recorded information in a DataTable is conveyed via DataRecord(s) consisting of fields each identifying a DataItem with an associated name and value (see XML Schema UPnP IoTManagementAndControl DataRecord Information [21]).

4.2.3 DataTable Dictionary

Each DataTable has an associated Dictionary. The DataTable's Dictionary is organized as a key-value structure. The values stored in the Dictionary are always of type utf-8, although the string values may contain encoded data types such as XML or Base64. DataTable record field(s) can refer to key(s) in the DataTable’s Dictionary. When a DataTable record is read the DataStore service can automatically resolve Dictionary references by substituting the corresponding the Dictionary value for the Dictionary key contained in the stored DataRecord.

4.2.4 DataItem types

The DataStore service always conveys stored DataItem(s) as strings. However, the contents of the stored strings may be XML documents, integer(s) or binary content (conveyed as Base64 encoded strings). The DataTable's DataRecord format indicates the data type for each DataItem. For data types of simple or moderate complexity, the DataItem's XML description document (see XML Schema UPnP IoTManagementAndControl Sensor DataModel service DataItem Description [25]) may be stored in the DataTable's Dictionary for each named DataItem. Alternatively, definitions associated the DataTable's URN may be used to identify sets of definitions for abstract data types.

4.2.5 DataTable Operations

DataTable(s) are created and deleted via SOAP actions. In addition to DataTable creation, SOAP clients may either directly write DataTable records via SOAP actions or may arrange transport connections via the DataStore service allowing asynchronous writing of DataTable records. DataTable records are retrieved via SOAP actions. DataTable Dictionary entries are read and written via SOAP actions.

4.2.6 DataStore Protection Model

The DataStore service [13] is allowed to restrict control point access to DataTable(s) using the DeviceProtection service [15]. When the DeviceProtection feature is implemented the DataStore service shall support the following roles:

- *Admin* – A control point with the *Admin* role can create/read/write/delete any DataStore Table and can create or remove any DataStore group.
- *Public* – A control point with the *Public* role can read or write specific DataStore tables which permit this access.
- *Basic* – A control point with the *Basic* role can read or write specific DataStore tables which permit this access.

In addition the DataStore service [13] which implements the DeviceProtection feature shall support the following group roles:

- *ds:Master#[GroupName]* - A control point with a *ds:Master#[GroupName]* identity for the indicated DataStore group may create or delete the corresponding group and may create or delete DataTable(s) belonging to that DataStore group. If a created or deleted DataTable participates in multiple DataStore groups, then the control point is required to have corresponding *ds:Master* identities for all groups the target DataTable references.
- *ds:Reader#[GroupName]* - A control point with a *ds:Reader#[GroupName]* identity for the indicated DataStore group may read DataTable(s) which are a member of the identified group.
- *ds:Writer#[GroupName]* - A control point with a *ds:Writer#[GroupName]* identity for the indicated DataStore group may write DataTable(s) which are a member of the identified group.

4.3 DataItem Semantics

The IoTManagementAndControl Sensor DataModel service and DataStore service use DataItem(s) to information. These DataItem(s) may include simple scalars, CSVs, abstract data types or XML documents. IoTManagementAndControl uses four components to describe each DataItem. These components may be conveyed differently:

For example:

The IoTManagementAndControl Sensor DataModel service [14] defines these components as Parameter nodes:

```
DataItems/#/Name
DataItems/#/Type
DataItems/#/Encoding
DataItems/#/Description
```

See UPnP Sensor Sensor DataModel service [14], Annex A.1.1.34, "...#/SensorURNs/#/DataItems/" for further details.

While the DataStore service provides these components as <field> element attribute values:

```
<field
   name="..."
   type="..."
   encoding="..." />
```

See UPnP DataStore:1 Service [13], subclause 5.5.12, "A_ARG_TYPE_DataTableInfo" for further details.

The following clause(s) discuss the underlying information conveyed so the services can interoperate correctly.

4.3.1 DataItem Name

Each DataItem shall have a name which identifies a specific occurrence of a data item.

DataItem names may have additional syntax requirements defined by the underlying sensor ecosystem.

Examples of DataItem names:

```
RoomThermostat
FreezerCompartmentTemperature
2b88f740-b151-11e2-9e96-0800200c9a66
$1-Obs
```

4.3.1.1 DataItem Name Requirements

DataItem name(s) shall conform to the following requirements:

- A set of DataItems described by a SensorURN shall each have a distinct name. DataItem name(s) while being distinct are not required to be globally unique.
- DataItem name(s) shall be restricted to the following ranges of allowed characters unless additional characters are specifically defined by the SensorURN.

```
   "A" - "Z" (U+0041 - U+005A)
   "a" - "z" (U+0061 - U+007A)
   "0" - "9" (U+0030 - U+0039)
   "_" (U+005F)
   "-" (U+002D)
```

- DataItem name(s) shall not include the following characters:

```
    "[" (U+005B)
    "]" (U+005D)
    ":" (U+003A)
```

- DataItem name(s) shall conform to any additional ecosystem requirement defined by the SensorURN.

4.3.2 DataItem Prefix

A DataItem prefix is a set of characters appended to a DataItem during transmission to make the DataItem name unique.

For example:

A household may deploy identical thermostat device(s) in each room. Although a Sensor Management device would consider these devices as distinct Sensors, it would likely use the same set of named DataItem(s) to describe each Sensor. To enable the DataStore service to aggregate measurements from these Sensors in a single DataTable, the IoTManagementAndControl Sensor Transport Generic service may be configured to append a prefix string to each reported DataItem on a per-Connection basis.

```
    [LivingRoom]Thermostat
    [Bedroom]Thermostat
    [Kitchen]Thermostat
```

4.3.2.1 DataItem Prefix Requirements

DataItem prefix(es) shall conform to the following requirements:

- DataItem prefix(es) shall be restricted to the following ranges of allowed characters unless additional characters are specifically defined by SensorURN naming conventions.

```
    "A" - "Z" (U+0041 - U+005A)
    "a" - "z" (U+0061 - U+007A)
    "0" - "9" (U+0030 - U+0039)
    "_" (U+005F)
    "-" (U+002D)
```

- DataItem prefix(es) shall not be included in DataItem name(s) reported by the IoTManagementAndControl Sensor DataModel service. DataItem prefix(es) are selected on a per-connection basis when a transport connection to a Sensor is requested. The requested prefix is only attached the DataItem prior to transmission.
- DataItem prefix(es) appended to the DataItem name shall be enclosed in the "[" and "]" characters.
- DataItem prefix(es) conform to any additional ecosystem requirement defined by the SensorURN.

4.3.3 DataItem Type

Each DataItem shall have a type which identifies the underlying data type of the data item.

DataItem type(s) rely on existing data type ontologies. The contents of the DataItem type field shall be used to identify the data type ontology and a specific data type defined by that ontology.

The following are examples of DataItem types:

```
uda:ui4
xsd:duration
xml:http://vendor.com/schemas/myschema.xsd:MyRootElementNS
```

```
    upnp:urn:schemas-upnp-org:service:AVTransport:3:A_ARG_TYPE_SeekMode
```

A DataItem composed of multiple values shall be conveyed in CSV (Comma Separated Variable) format. The corresponding DataItem type shall provide a CSV of the data type(s) for the sub-components of each CSV item. The corresponding DataItem Description Document may provide additional context information for the sub-components of each CSV item.

4.3.3.1 DataItem Type Requirements

DataItem type values shall conform to the following requirements:

- A DataItem type shall have the following format:

```
    [DataItem type]::=
       [identifier] [":" [Information field(s)]] ":" [Type field] ["," [DataItem
type]]
```

The following table defines type

Identifier	Information field	Data type	Description
uda	None	UDA defined type	A data type is defined specifically by UPnP Device Architecture.
xsd	None	XSD built-in type(s)	A built-in data type as defined by http://www.w3.org/2001/XMLSchema.
xml	Schema Location	Root element namespace	Namespace of an XML document.
upnp	UPnP Service URN	UPnP service state variable name	A variable defined by a UPnP SCPD.
mds	Medical Device System (IEEE-11073)		
<other>	Per ecosystem SensorURN	Per ecosystem SensorURN	Type defined by specific sensor ecosystem.

Note: A UPnP Sensor Transport client (such as a DataStore service is permitted to accept DataItem(s) with types it does not directly support; provided it can correctly store and forward the received type information to the next destination.

4.3.4 DataItem Encoding

Each DataItem shall have an encoding which identifies the format of the original DataItem.

4.3.4.1 DataItem Encoding Requirements

The following DataItem encoding values shall be supported:

Encoding	Description
ascii	DataItem consists of ASCII characters in the range (U+0000 – U+007F)
utf-8	DataItem shall conform to UTF-8 encoding
base64	DataItem is binary data encoded as Base64

4.3.5 DataItem Description Document

Each DataItem is allowed to provide an XML document that provides additional information about the DataItem. See the following section.

4.4 DataItem Description XML Document

4.4.1 Introduction

DataItem(s) may provide an informational XML document which describes the characteristics of each DataItem.

4.4.2 DataItem Categories

Information provided by DataItem(s) may be grouped under the following categories. DataItem(s) in CSV format may contain multiple categories.

- Measurement

  This DataItem (or DataItem CSV component) is a measured quantity. The DataItem description document provides information concerning the measurement units and the measurement method, i.e. average, cumulative, or current.

- Limit

  This DataItem (or DataItem CSV component) serves as a limit for a measured quantity. This item is typically related to an Alarm which indicates when the measured quantity has exceeded the specified limit.

- Interval

  This DataItem (or DataItem CSV component) provides a measurement interval.

- Setting

  This DataItem (or DataItem CSV component) represents a setting such as a thermostat setting, or may be an actuator input.

- Alarm

  This DataItem (or DataItem CSV component) represents an Alarm indicating an unusual condition. The Alarm may be set as the result of other subsystem alarms or may be directly related to a measured quantity or limit.

4.4.3 DataItem Description Document Elements

If a DataItem provides a simple scalar quantity then one of the above categories will likely apply and the DataItem Description document shall contain a corresponding descriptive element such as `<measurement>`, `<limit>`, `<interval>`, `<alarm>`, or `<setting>`. If the DataItem is related to other named DataItems then the `<relateditem>` element should be present to describe the relationships to peer DataItems. The DataItem may also be a CSV containing multiple categories such as a measurement, a limiting value and an alarm. In this case, multiple category elements are included to describe to component values of the CSV. If a DataItem provides an XML document; the corresponding DataItem description XML document can use an XPath [8] expression to identify XML elements within the target DataItem document. The XPath expression is supplied by the `path=` attribute of the descriptive element within the DataItem description XML document.

4.4.3.1 DataItem Description Document Template

The DataItem Description parameter value shall be set to an XML document conforming to the XML Schema UPnP IoTManagementAndControl Sensor DataModel service DataItem Description [25] and as described below:

.

```
<?xml version="1.0" encoding="UTF-8"?>
<DataItemDescription
  xmlns="urn:schemas-upnp-org:smgt:sdmdid"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="
```

```
    urn:schemas-upnp-org:smgt:sdmdid.xsd
    http://www.upnp.org/schemas/smgt/sdmdid-v1.xsd"

  itemname="Name parameter for this DataItem"
  access="read/read-write/write access for this DataItem">

  <description>
    Description of this DataItem (suitable for end-user display)
  </description>

  <measurement
    units="Units of measurement"
    treatment="Treatment for this measurement (current,average)"
    accumulation="Accumulation timeframe for this measurement"
    access="read/read-write/write access for this quantity"
    path="For XML based DataItem(s) XPath spec to element/attr">

    <relateditem
      itemname="Name parameter of DataItem related to this DataItem"
      relationtype="Type of relationship" />

  </measurement>

  <limit
    units="Units of measurement"
    type="Type of limit (high,low,timeout)"
    access="read/read-write/write access for this quantity"
    path="For XML based DataItem(s) XPath spec to element/attr">

    <relateditem
      itemname="Name parameter of DataItem related to this DataItem"
      relationtype="Type of relationship" />

  </limit>

  <setting
    units="Units of setting"
    access="read/read-write/write access for this quantity"
    path="For XML based DataItem(s) XPath spec to element/attr">

    <relateditem
      itemname="Name parameter of DataItem related to this DataItem"
      relationtype="Type of relationship" />

  </setting>

  <interval
    units="Units of measurement"
    access="read/read-write/write access for this quantity"
    path="For XML based DataItem(s) XPath spec to element/attr">

    <relateditem
      itemname="Name parameter of DataItem related to this DataItem"
      relationtype="Type of relationship" />

  </interval>

  <alarm
    access="read/read-write/write access for this quantity"
    path="For XML based DataItem(s) XPath spec to element/attr">

    <relateditem
      itemname="Name parameter of DataItem related to this DataItem"
      relationtype="Type of relationship" />

  </alarm>

</DataItemDescription>
```

`<?xml>`

Allowed. Case sensitive.

`<DataItemDescription>`

Required, XML. Shall include a namespace declaration for the DataItem Description Schema ("urn:schemas-upnp-org:smgt:sdmdid"). Shall include zero or more of the following elements:

`itemname`

Required, xsd:string. Shall be equal to the name of the DataItem being described by this XML document.

`access`

Required. xsd:string. Shall identify whether any components of the DataItem can be read and/or written. The following table provides the allowed values for the `access=` attribute:

Table 2 — `access=` attribute allowed values

Allowed Values	Description
ro	DataItem can be read and has no writable components.
rw	DataItem can be read and has one or more writable components.
wo	DataItem can only be written.

`<description>`

Required. xsd:string. Shall provide a description of the corresponding DataItem (suitable for display to end-users).

`<measurement>`

Allowed. XML. This element shall be present zero or more times. This element provides information when the corresponding DataItem (or DataItem CSV component) provides a measured scalar quantity.

`units`

Required, xsd:string. This attribute provides the units for the measured quantity. See subclause 4.5, "DataItem Description Units of Measurement" for allowed values for this attribute. Support of sensors which report in arbitrary units is not allowed unless otherwise such behaviour is specifically allowed by the corresponding SensorURN definition.

`treatment`

Required. xsd:string. This attribute describes the treatment of the measured quantity. The following table provides allowed values for this attribute

Table 3 — `treatment=` attribute allowed values (measurement)

Allowed Values	Description
current	Measurement is the result from the last time the sensor was read.
average	Measurement is an averaged value. The `accumulation=` value indicates if the average was for the current interval or cumulative.
high	Measurement is the highest value detected. The `accumulation=` value indicates if the sampling period was for the current interval or cumulative.
low	Measurement is the lowest value detected. The `accumulation=` value indicates if the sampling period was for the current interval or cumulative.
delta	Measurement is a change from the previous measurement.
pct-on	Measurement is for percentage-on. The `accumulation=` value indicates if the sampling period was for the current

	interval or cumulative.
pct-off	Measurement is for percentage-off. The `accumulation=` value indicates if the sampling period was for the current interval or cumulative.
count	Measurement is a simple (event) counter. The `accumulation=` value indicates if the sampling period was for the current interval or cumulative.

`accumulation`

Required. xsd:string. This attribute describes the timeframe which applies to the measured quantity. The following table provides allowed values for this attribute:

Table 4 — `accumulation=` attribute allowed values (measurement)

Allowed Values	Description
current	This measurement is not accumulated. The measurement provided is the last value obtained from for this DataItem.
interval	This measurement was accumulated over an interval. The interval value is provided by the `<relateditem>` or `<interval>` element.
cumulative	This measurement was accumulated, and is not reset at interval boundaries. If the corresponding access attribute for this measurement is "rw", then a write shall set the initial accumulated value.

`access`

Required. xsd:string. Access for measurement component. See Table 2, "`access=` attribute allowed values".

`path`

Allowed. xsd:string. For DataItems which are defined as XML documents, this attribute provides an XPath [8] expression to locate the corresponding element or attribute with the DataItem’s XML document.

`<relateditem>`

Allowed. XML. This allowed element may appear zero or more times. Each instance provides information about a separate DataItem which is related to the DataItem being described. If there are multiple related DataItems, then this element may be repeated to describe the related DataItems. If there are no applicable DataItems or the related fields included within this DataItem, then this element shall be omitted.

`itemname`

Required, xsd:string. This attribute provides the name of the related DataItem. The DataItem name shall correspond to another named peer DataItem.

`relationtype`

Required, xsd:string. This attribute shall identify how the DataItem identified by the `itemname=` attribute is related this DataItem. The following allowed values describe the relationship to the separate peer DataItem identified.

Table 5 — `relationtype=` allowed values (measurement)

Allowed Value	Description
limit	The related DataItem provides a limiting value for the measured DataItem.
alarm	The related DataItem is an Alarm for the measured DataItem.
setting	The related DataItem provides a setting value for the

	measured DataItem
Interval	The related DataItem provides a measurement interval for the measured DataItem.

`<alarm>`

Allowed. XML. This element may be present when the corresponding DataItem (or DataItem component) describes an alarm condition. The corresponding DataItem may provide a boolean value indicating whether an alarm condition is present or may provide a text string indicating the type of alarm.

`access`

Required, xsd:string. Access for an alarm component may be "*rw*" or "*ro*" depending if the Sensor anticipates the UPnP IoTManagementAndControl control point will explicitly reset the alarm.

`path`

Allowed, xsd:string. For DataItems which are defined as XML documents, this attribute provides an XPath [8] expression to locate the corresponding element or attribute with the DataItem's XML document.

`<relateditem>`

Allowed. XML. This allowed element may appear zero or more times. Each instance provides information about a separate DataItem which is related to the DataItem being described. If there are multiple related DataItems, then this element may be repeated to describe the related DataItems.

`itemname`

Required, xsd:string. This attribute provides the name of the related DataItem. The DataItem name shall correspond to another named peer DataItem.

`relationtype`

Required, xsd:string. This attribute shall identify how the DataItem identified by the `itemname=` attribute is related this DataItem. The following allowed values describe the relationship to the separate peer DataItem identified.

Table 6 — `relationtype=` attribute (alarm)

Allowed Value	Description
source	The related DataItem is a potential source for the alarm. The referred DataItem may be a measurement, limit or another alarm.
reset_interval	Amount of time the alarm will remain active before being automatically reset.

`<limit>`

Allowed. XML. This element may be present zero or more times. This element provides information when the corresponding DataItem (or DataItem component) provides a limiting value for a measured quantity.

`units`

Required, xsd:string. This attribute provides the units for the limiting quantity. See subclause 4.5, "DataItem Description Units of Measurement" for allowed values for this attribute. Support of sensors which report in arbitrary units is prohibited unless otherwise such behaviour is specifically allowed by the corresponding SensorURN definition. In these cases the SensorURN shall provide the necessary scaling information to correctly interpret the sensor measurements.

`limittype`

Required. xsd:string. This attribute describes the type of limit to be applied to the DataItem. The following allowed values describe the types of limits:

Table 7 — `limittype=` attribute (limit)

Allowed Value	Description
High	Limiting value is the highest acceptable value for the measured quantity
Low	Limiting value is the lowest acceptable value for the measured quantity
Timeout	Limiting value is a timeout.

`access`

Required, xsd:string. Access for a limit component may be "rw" or "ro" depending control point is be permitted to modify the limit value.

`path`

Allowed, xsd:string. For DataItems which are defined as XML documents, this attribute provides an XPath [8] expression to locate the corresponding element or attribute with the DataItem's XML document.

`<relateditem>`

Allowed. XML. This allowed element may appear zero or more times. Each instance provides information about a separate DataItem which is related to the DataItem being described. If there are multiple related DataItem(s), then this element may be repeated to describe the related DataItems.

`itemname`

Required, xsd:string. This attribute provides the name of the related DataItem. The DataItem name shall correspond to another named DataItem available under the same DataItems Multiinstance node as this DataItem.

`relationtype`

Required. xsd:string. This attribute shall identify how the DataItem identified by the `itemname=` attribute is related this DataItem. The following allowed values describe the relationship to the separate DataItem identified.

Table 8 — `relationtype=` attribute allowed values (limit)

Allowed Value	Description
measurement	The related DataItem is a measurement to which this Limit applies.
alarm	The related DataItem is an Alarm for the measured DataItem, for example a master alarm which is set based on a subsystem alarm status.

`<setting>`

Allowed. XML. This element may be present zero or more times. This element provides information when the corresponding DataItem (or DataItem component) provides a measured scalar quantity.

`units`

Required, xsd:string. This attribute provides the units for the setting quantity. See subclause 4.5, "DataItem Description Units of Measurement" for allowed values for this attribute.

`access`

Required, xsd:string. Access for a setting component may be "rw" or "ro" depending if the Sensor anticipates the UPnP IoTManagementAndControl control point be permitted to modify the setting value.

`path`

Allowed, xsd:string. For DataItems which are defined as XML documents, this attribute provides an XPath [8] expression to locate the corresponding element or attribute with the DataItem's XML document.

`<relateditem>`

Allowed, XML. This allowed element may appear zero or more times. Each instance provides information about a separate DataItem which is related to the DataItem being described. If there are multiple related DataItems, then this element may be repeated to describe the related DataItems.

`itemname`

Required, xsd:string. This attribute provides the name of the related DataItem. The DataItem name shall correspond to another named peer DataItem.

`relationtype`

Required, xsd:string. This attribute shall identify how the DataItem identified by the `itemname` attribute is related this DataItem. The following allowed values describe the relationship to the peer DataItem .

Table 9 — `relationtype=` attribute allowed values (setting)

Allowed Value	Description
measurement	The related DataItem is a measurement directly related to the setting contained in this DataItem. For example:a temperature measurement related to a current thermostat setting,
monitor	The related DataItem is a measurement indirectly related to the setting contained in this DataItem. Such as measurement of heat-on time as related to a temperature setting.

`<interval>`

Allowed. XML. This element shall be present zero or more times. This DataItem provides a measurement interval for other related DataItem(s) (or DataItem components).

`units`

Required, xsd:string. This attribute provides the units for the measurement interval. The following table provides allowed values for this attribute.

Table 10 — `units=` attribute allowed values (interval)

Allowed Value	Description
duration	An interval expressed per ISO-8601 duration notation
s	An interval expressed in seconds.

`access`

Required. xsd:string. Access for a setting component may be "*rw*" or "*ro*" depending if the Sensor anticipates the UPnP IoTManagementAndControl control point be permitted to modify the interval value.

`path`

Allowed, xsd:string. For DataItems which are defined as XML documents, this attribute provides an XPath [8] expression to locate the corresponding element or attribute with the DataItem's XML document.

`<relateditem>`

Allowed, XML. This allowed element may appear zero or more times. Each instance provides information about a separate DataItem which is related to the DataItem being described. If there are multiple related DataItem(s), then this element may be repeated to describe the related DataItems.

`itemname`

Required, xsd:string. This attribute provides the name of the related DataItem. The DataItem name shall correspond to another named peer DataItem.

`relationtype`

Required, xsd:string. This attribute shall identify how the DataItem identified by the `itemname=` attribute is related this DataItem. The following allowed values describe the relationship to the separate DataItem identified.

Table 11 — `relationtype=` attribute allowed values (interval)

Allowed Value	Description
measurement	The interval applies to the related measurement DataItem.
alarm	The interval applies to the related alarm, indicating a timeout condition.

4.5 DataItem Description Units of Measurement

4.5.1 SI Prefixes

Prefix	Factor
Y	10^24 - yotta
Z	10^21 - zetta
E	10^18 - exa
P	10^15 - peta
T	10^12 - tera
G	10^9 - giga
M	10^6 - mega
K	10^3 - kilo
H	10^2 - hecto
Da	10^1 - deca
D	10^-1 - deci
C	10^-2 - centi
M	10^-3 - mili
U	10^-6 - micro
n	10^-9 - nano
p	10^-12 - pico
f	10^-15 - femto
a	10^-18 - atto
z	10^-21 - zepto
y	10^-24 - yocto

4.5.2 SI Base Units

Unit Symbol	Definition	Category
m	meters	length
kg	kilogram	mass
s	seconds	time
A	ampere	electric current
mol	mole	amount
K	degrees Kelvin	temperature
cd	candela	luminous intensity

4.5.3 SI Derived Units

Unit Symbol	Definition	Category
N	newton	force
Pa	pascal	pressure
J	joules	work/energy

W	watts	power
Hz	hertz	frequency
C	coulomb	electric charge
F	farad	capacitance
V	volt	electric potential
O	ohm	electric resistance
H	henry	inductance
S	siemens	conductivity
T	tesla	magnetic field strength
Wb	weber	magnetic flux
lm	lumen	luminous flux
rad	radian	angle
sr	steradian	solid angle
lx	lux	illuminance
C	degrees Celsius	temperature
Bq	Becquerel	radioactivity (decay per unit time)
Gy	gray	radioactivity (adsorbed dose)
Sv	sievert	radioactivity (equivalent dose)

4.5.4 Non-SI in common use

Unit Symbol	Definition	Category
g	grams	mass
l	litre	volume
cal	calorie	energy
Cal	1000 calories	energy
kcal	1000 calories	energy
degC	degrees Celsius	temperature

4.5.5 British units in common use

Unit Symbol	Definition	Category
in	inch	length
ft	foot	length
yd	yard	length
mi	mile	length
oz	ounce	weight
lb	pound	weight
t	ton	weight
psi	pounds per square inch	psi
Btu	british thermal units	energy
degF	degrees Fahrenheit	temperature

4.5.6 Coordinates

Unit Symbol	Definition	Category

coord-dms	Coordinates CSV(lat:deg,min,sec,lon:deg,min,sec)	location
coord-gps	Coordinates CSV(lat:deg-float, lon:deg-float)	location

4.5.7 Duration

Unit Symbol	Definition	Category
duration	Duration per ISO-8601	time

4.5.8 Encoding

The SI and Non-SI unit symbols listed in subclause 4.5.2 and subclause 4.5.4 may be used directly or with the scaling prefixes listed in subclause 4.5.1.

For example:

- "m" - meters
- "km" - kilometer
- "ml" - milliliter
- "kW" - kilowatt
- "nm" - nanometer

British units may be used directly. For example:

- "oz" - ounce
- "mi" - mile
- "lb" – pound

Other units may be derived by combination of the listed units:

- "m/s^2" - acceleration
- "kW-h" - energy
- "lb-ft" - torque
- "N-m" - torque

4.6 Theory of Operation (informative)

4.6.1 IoTManagementAndControl Device

The following examples show discovery and connection to sensors managed by a IoTManagementAndControl device.

4.6.1.1 Discovering of the Data Model

After detecting a UPnP IoTManagementAndControl Device a control point will typically check the supported data model. The GetSupportedDatamodels() action returns an XML document containing the currently supported Data Model definitions. The data model supported by the IoTManagementAndControl Device contains the following URI: urn:upnp-org:smgt:1

```
Request:
GetSupportedDataModels()

Response:
GetSupportedDataModels(
"<?xml version="1.0" encoding="UTF-8"?>
<cms:SupportedDataModels
   xmlns:cms="urn:schemas-upnp-org:dm:cms"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation=
      "urn: schemas-upnp-org:dm:cms http://www.upnp.org/schemas/dm/cms.xsd">
   <SubTree>
      <URI>
         urn:upnp-org:smgt:1
      </URI>
      <Location>
         /UPnP/SensorMgt
      </Location>
      <URL>
      http://www.upnp.org/specs/smgt/UPnP-smgt-SensorDataModel-v1-Service.pdf
      </URL>
      <Description>
         ... device vendor descriptive text ...
      </Description>
   </SubTree>
</cms:SupportedDataModels>" )
```

Next a control point calls the GetSupportedParameters() action using "/UPnP/SensorMgt/" as starting Node with SearchDepth set to ("0"), which indicates that the control point wants to read the entire sub-tree. Using this information the control point can determine which parameters are supported, including allowed or eco-system specific parameters.

```
Request:
GetSupportedParameters(
   "/UPnP/SensorMgt",
   0)

Response:
GetSupportedParameters(
   "<?xml version="1.0" encoding="UTF-8"?>
    <cms:StructurePathList
      xmlns:cms="urn:schemas-upnp-org:dm:cms"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation=
         "urn:schemas-upnp-org:dm:cms http://www.upnp.org/schemas/dm/cms.xsd">
   <StructurePath>
      /UPnP/SensorMgt/
   </StructurePath>
   <StructurePath>
      /UPnP/SensorMgt/SensorCollectionsNumberofEntries
   </StructurePath>
   <StructurePath>
      /UPnP/SensorMgt/SensorCollections/#/
   </StructurePath>
   <StructurePath>
      /UPnP/SensorMgt/SensorCollections/#/CollectionType
   </StructurePath>
   <StructurePath>
      /UPnP/SensorMgt/SensorCollections/#/CollectionFriendlyName
   </StructurePath>
   <StructurePath>
      /UPnP/SensorMgt/SensorCollections/#/CollectionInformation
   </StructurePath>
   <StructurePath>
      /UPnP/SensorMgt/SensorCollections/#/CollectionUniqueIdentifier
   </StructurePath>
   <StructurePath>
      /UPnP/SensorMgt/SensorCollections/#/SensorsNumberOfEntries
   </StructurePath>
   <StructurePath>
      /UPnP/SensorMgt/SensorCollections/#/Sensors/#/
   </StructurePath>
   <StructurePath>
      /UPnP/SensorMgt/SensorCollections/#/Sensors/#/SensorID
   </StructurePath>
   <StructurePath>
      /UPnP/SensorMgt/SensorCollections/#/Sensors/#/SensorType
   </StructurePath>
   <StructurePath>
      /UPnP/SensorMgt/SensorCollections/#/Sensors/#/SensorUniqueIdentifier
   </StructurePath>
   <StructurePath>
      /UPnP/SensorMgt/SensorCollections/#/Sensors/#/SensorURNsNumberOfEntries
   </StructurePath>
   <StructurePath>
      /UPnP/SensorMgt/SensorCollections/#/Sensors/#/SensorURNs/#/
   </StructurePath>
   <StructurePath>
      /UPnP/SensorMgt/SensorCollections/#/Sensors/#/SensorURNs/#/SensorURN
   </StructurePath>
   <StructurePath>
   /UPnP/SensorMgt/SensorCollections/#/Sensors/#/SensorURNs/#/DataItemsNumberOfEntr
ies
   </StructurePath>
   <StructurePath>
      /UPnP/SensorMgt/SensorCollections/#/Sensors/#/SensorURNs/#/DataItems/#/
   </StructurePath>
   <StructurePath>
      /UPnP/SensorMgt/SensorCollections/#/Sensors/#/SensorURNs/#/DataItems/#/Name
   </StructurePath>
   <StructurePath>
      /UPnP/SensorMgt/SensorCollections/#/Sensors/#/SensorURNs/#/DataItems/#/Type
   </StructurePath>
   <StructurePath>
   /UPnP/SensorMgt/SensorCollections/#/Sensors/#/SensorURNs/#/DataItems/#/Encoding
   </StructurePath>
   </cms:StructurePathList>")
```

Note: *GetSupportedParameters()* refers to *ConfigurationManagement::GetSupportedParameters()*

4.6.1.2 Obtaining SensorCollection and Sensor Information

As a next step a control point typically queries the data model to determine which SensorCollections and Sensors are represented by the IoTManagementAndControl Device. A control point may, for example, provide a list of available SensorCollections to the user, and allow the user to select a SensorCollection to obtain additional information. Alternatively, a control point may look for a specific type of sensor.

To determine which SensorCollections and Sensors are represented by the data model, the control point invokes the *GetInstances()* action. This action returns an XML formatted list of Instance Paths. Whereas the data model describes which type of information is available, instance paths refer to the actual number of instances present for each multi-instance node. The Fridge example, defined in the Annex A, "Sample Device Illustration" contains one SensorCollection, two Sensors, two SensorURN and ten DataItem instances in total.

To obtain a list of SensorCollection instances the control point calls *GetInstances(*"/UPnP/SensorMgt/SensorCollections/",1*)* . The first argument defines the starting node, in this case the SensorCollections; setting the second argument, limits the SearchDepth to 1. Therefore, only SensorCollection instances will be returned by this call. The XML document returned by this call will contain a list of all SensorCollection instances. In the Fridge example, only one instance is present. The corresponding XML output is shown below.

```
Request:
GetInstances(
   "/UPnP/SensorMgt",
   1)

Response:
GetInstances(
   "<?xml version="1.0" encoding="UTF-8"?>
    <cms:InstancePathList
      xmlns:cms="urn:schemas-upnp-org:dm:cms"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation=
         "urn:schemas-upnp-org:dm:cms http://www.upnp.org/schemas/dm/cms.xsd">
      <InstancePath>
           /UPnP/SensorMgt/SensorCollectionsNumberOfEntries
      </InstancePath>
      <InstancePath>
         /UPnP/SensorMgt/SensorCollections/1/
      </InstancePath>
   </cms:InstancePathList>")
```

Note: *GetInstances()* refers to *ConfigurationManagement::GetInstances()*

Using the SensorCollection instances, the control point can construct a path to query which Sensor instances are available in a particular SensorCollection. For example, *GetInstances(* "/UPnP/SensorMgt/SensorCollections/1/Sensors/",1 *)* will return an XML document providing a list of Sensor instances for SensorCollection instance 1.

```
Request:
GetInstances(
   "/UPnP/SensorMgt/SensorCollections/1/Sensors/",
   1)

Response:
GetInstances(
   "<?xml version="1.0" encoding="UTF-8"?>
    <cms:InstancePathList
      xmlns:cms="urn:schemas-upnp-org:dm:cms"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation=
         "urn:schemas-upnp-org:dm:cms http://www.upnp.org/schemas/dm/cms.xsd">
      <InstancePath>
         /UPnP/SensorMgt/SensorCollections/1/SensorNumberOfEntries
      </InstancePath>
      <InstancePath>
         /UPnP/SensorMgt/SensorCollections/1/Sensor/1
      </InstancePath>
   </cms:InstancePathList>")
```

Note: *GetInstances()* refers to *ConfigurationManagement::GetInstances()*

Alternatively, a control point implementation may request SensorCollections and Sensor instances in a combined list by invoking *GetInstances(* "/UPnP/SensorMgt/SensorCollections/",3 *)*.

Request:
```
GetInstances(
   "/UPnP/SensorMgt/SensorCollections/1/Sensors/",
   3)
```

Response:
```
GetInstances(
   "<?xml version="1.0" encoding="UTF-8"?>
    <cms:InstancePathList
      xmlns:cms="urn:schemas-upnp-org:dm:cms"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation=
         "urn:schemas-upnp-org:dm:cms http://www.upnp.org/schemas/dm/cms.xsd">
      <InstancePath>
         /UPnP/SensorMgt/SensorCollections/1/SensorNumberOfEntries
      </InstancePath>
      <InstancePath>
         /UPnP/SensorMgt/SensorCollections/1/Sensor/1
      </InstancePath>
      <InstancePath>
         /UPnP/SensorMgt/SensorCollections/1/Sensor/1/SensorID
      </InstancePath>
      <InstancePath>
         /UPnP/SensorMgt/SensorCollections/1/Sensor/1/SensorType
      </InstancePath>
      <InstancePath>
         /UPnP/SensorMgt/SensorCollections/1/Sensor/1/SensorURNsNumberOfEntries
      </InstancePath>
      <InstancePath>
         /UPnP/SensorMgt/SensorCollections/1/Sensor/1/SensorURNs/1
      </InstancePath>
      </cms:InstancePathList>")
```

Ultimately, a control point may obtain all instances represented by the sensor management device by using GetInstances("/UPnP/SensorMgt/",0) as *SearchDepth* 0 indicates unlimited search depth. It must be noted, however, that a IoTManagementAndControl device may represent a large number of Sensors.

Control points can request the current value of a parameter by combining the information obtained through the *GetParameters()* and the *GetInstances()* action. Using the *GetValues()* action control points can request information on specific parameter, or request all parameters corresponding to a certain instance. The *ContentPathList* input parameter of the *GetValues()* action is used to specify a number of Instance paths for which values are requested. The result of the *GetValues()* action is an XML document which contains a set of parameter paths and corresponding values.

Invoking the *GetValues()* action using the XML document below requests the current value of the CollectionType and the CollectionFriendlyName for SensorCollection instance 1. In case of the Fridge example, the following XML document, which provides the values for the CollectionType and CollectionFriendlyName, is returned.

Request:
```
GetValues(
   "<?xml version="1.0" encoding="UTF-8"?>
    <cms:ContentPathList xmlns:cms="urn:schemas-upnp-org:dm:cms"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation=
         "urn:schemas-upnp-org:dm:cms http://www.upnp.org/schemas/dm/cms.xsd" >
      <ContentPath>
         /UPnP/SensorMgt/SensorCollections/1/CollectionType
      </ContentPath>
      <ContentPath>
         /UPnP/SensorMgt/SensorCollections/1/CollectionFriendlyName
      </ContentPath>
   </cms:ContentPathList>" )
```

Response:

```
GetValues(
   "<?xml version="1.0" encoding="UTF-8"?>
   <cms:ParameterValueList
      xmlns:cms="urn:schemas-upnp-org:dm:cms"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation=
         "urn:schemas-upnp-org:dm:cms http://www.upnp.org/schemas/dm/cms.xsd">
      <Parameter>
         <ParameterPath>
            /UPnP/SensorMgt/SensorCollections/1/CollectionType
         </ParameterPath>
         <Value>
            urn:upnp-org:smgt-sct:refrigerator:AcmeSensorsCorp-com
            :AcmeIntegratedController:FrigidaireCorp:rf217acrs
         </Value>
      </Parameter>
      <Parameter>
         <ParameterPath>
            /UPnP/SensorMgt/SensorCollections/1/CollectionFriendlyName
         </ParameterPath>
         <Value>
            Your Refrigerator
         </Value>
      </Parameter>
   </cms:ParameterValueList>" )
```

Note: *GetValues()* refers to *ConfigurationManagement::GetValues()*

In a similar fashion parameters can be modified by a control point using the *SetValues()* action. The access attribute determines if a parameter can be written or is read-only. In the latter case the *SetValues()* action will return error 706 "Read-only violation".

4.6.1.3 Reading and writing sensor data

The previous section detailed the reading and writing of data model parameter values. These parameter values provide information about the sensors and the type of data these sensors can produce. The actual data values provided by a sensor are accessed using the SensorTransportGeneric service.

The SensorTransportGeneric service offers two approaches to read data from or write data to a sensor.

In the first case a control point can setup a connection to a specific sensor identified by its *SensorID* and *SensorURN* by invoking the *ConnectSensor()* action. The *SensorID*, and *SensorURN* input arguments can be obtained for a specific sensor instance using the *GetInstances()* and *GetValues()* actions described in the previous sections.

Since a SensorURN can provide multiple types of sensor data as described by the corresponding DataItem(s), the *SensorRecordInfo* argument is used to select which data to report.

4.6.1.4 Obtaining sensor data via transport connection

To setup an HTTP connection to read the energy consumption of the Fridge described in the Annex A, "Sample Device Illustration", the ConnectSensor() action is invoked in the following example. The action returns the TransportConnectionID output argument. Subsequently the control point can receive data items on the URL provided by the TransportURL argument.

```
Request:
ConnectSensor(
   "Sensor0001",
   "MyControlPoint",
   "upnp-org:smgt-surn:refrigerator:AcmeSensorsCorp-com
    :AcmeIntegratedController:FrigidaireCorp:rf217acrs:monitor",
   "<?xml version="1.0" encoding="UTF-8"?>
    <SensorRecordInfo xmlns="urn:schemas-upnp-org:smgt:srecinfo"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation=
         "urn:schemas-upnp-org:smgt:srecinfo
            http://www.upnp.org/schemas/smgt/srecinfo.xsd" >
      <sensorrecord>
         <field name="ClientID" />
         <field name="AccumulatedPowerUsed" />
      </sensorrecord>
    </SensorRecordInfo>",
   0,
   "http://192.168.1.111:49153/SensorMgt/" );
```

```
Response:
ConnectSensor(
   "1ec4b930-b57b-11e2-9e96-0800200c9a66" )
```

Note: *ConnectSensor()* refers to *SensorTransportGeneric::ConnectSensor()*

In this example the *SensorDataTypeEnable* argument is set to ("0"), which indicates that the provided data records would not include type, encoding and namespace attributes. The ClientID argument set to "MyControlPoint" is used to set the value corresponding DataItem of the indicated Sensor. Using the transport URL provided, the Sensor will provide DataRecord(s) as shown below.

```
HTTP:(HTTP-POST to transport URL)
POST /SensorMgt HTTP/1.1
Host: example.com
Content-Type: text/xml
Content-Length: 417
<?xml version="1.0" encoding="UTF-8"?>
<DataRecords xmlns="urn:schemas-upnp-org:ds:drecs"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:schemas-upnp-org:ds:drecs
      http://www.upnp.org/schemas/ds/drecs-v1.xsd">
   <datarecord>
      <field name="ClientID">
         MyControlPoint
      </field>
      <field name="AccumulatedPowerUsed">
         42
      </field>
   </datarecord>
</DataRecords>
```

4.6.1.5 Obtaining sensor data via SOAP action

To read data from or write data to a sensor using the second alternative, the *ReadSensor()* and *WriteSensor()* actions are available. Instead of setting up a connection, a single set of data items is read or written in each call. To read the energy consumption of the Fridge, the *ReadSensor()* action is called using the following input arguments.

```
Request:
ReadSensor(
   "Sensor0001",
   "MyControlPoint",
   "urn:upnp-org:smgt-surn:refrigerator:AcmeSensorsCorp-com
   :AcmeIntegratedController:FrigidaireCorp:rf217acrs:monitor",
   "<?xml version="1.0" encoding="UTF-8"?>
   <SensorRecordInfo xmlns:cms="urn:schemas-upnp-org:dm:cms"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation=
         "urn:schemas-upnp-org:smgt:srecinfo
            http://www.upnp.org/schemas/smtg/srecinfo.xsd" >
      <sensorrecord>
         <field name="AccumulatedPowerUsed" />
      </sensorrecord>
   </SensorRecordInfo>",
   0,
   1)
```

```
Response:
ReadSensor(
   "<?xml version="1.0" encoding="UTF-8"?>
    <DataRecords xmlns="urn:schemas-upnp-org:ds:drecs"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation=
         "urn:schemas-upnp-org:ds:drecs
            http://www.upnp.org/schemas/ds/drecs-v1.xsd">
      <datarecord>
         <field name="AccumulatedPowerUsed">42</field>
      </datarecord>
      </DataRecords>",
      ""
)
```

Note: ReadSensor() refers to SensorTransportGeneric::ReadSensor()

4.6.2 Configuring the DataStore service

In this example configuring the DataStore service to accept data from a sensor connected to a IoTManagementAndControl device is shown.

4.6.2.1 Obtaining Sensor Information

This example continues the sample discussed in subclause 4.6.1, "IoTManagementAndControl Device".

Request:
```
GetValues(
   "<?xml version="1.0" encoding="UTF-8"?>
    <cms:ContentPathList xmlns:cms="urn:schemas-upnp-org:dm:cms"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation=
         "urn:schemas-upnp-org:dm:cms http://www.upnp.org/schemas/dm/cms.xsd" >
      <ContentPath>
         /UPnP/SensorMgt/SensorCollections/1/Sensor/1/SensorURNs/1/SensorURN
      </ContentPath>
      <!-- Following paths are repeated for each DataItem -->
      <ContentPath>
         /UPnP/SensorMgt/SensorCollections/1/Sensor/1/SensorURNs/1/SensorURN/1
         /DataItems/1/Name
      </ContentPath>
      <ContentPath>
         /UPnP/SensorMgt/SensorCollections/1/Sensor/1/SensorURNs/1/SensorURN/1
         /DataItems/1/Type
      </ContentPath>
      <ContentPath>
         /UPnP/SensorMgt/SensorCollections/1/Sensor/1/SensorURNs/1/SensorURN/1
         /DataItems/1/Encoding
      </ContentPath>
      <ContentPath>
         /UPnP/SensorMgt/SensorCollections/1/Sensor/1/SensorURNs/1/SensorURN/1
         /DataItems/1/Description
      </ContentPath>
   </cms:ContentPathList>" )
```

Response:
```
GetValues(
   "<?xml version="1.0" encoding="UTF-8"?>
    <cms:ParameterValueList
      xmlns="urn:schemas-upnp-org:dm:cms"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation=
         "urn:schemas-upnp-org:dm:cms http://www.upnp.org/schemas/dm/cms.xsd">
      <Parameter>
         <ParameterPath>
            /UPnP/SensorMgt/SensorCollections/1/Sensor/1/SensorURNs/1/SensorURN
         </ParameterPath>
         <Value>
            urn:upnp-org:smgt-surn:refrigerator:AcmeSensorsCorp-com
            :AcmeIntegratedController:FrigidaireCorp:rf217acrs:monitor
         </Value>
      </Parameter>
      <Parameter>
         <ParameterPath>
            /UPnP/SensorMgt/SensorCollections/1/Sensor/1/SensorURNs/1/SensorURN/1
            /DataItems/1/Name
         </ParameterPath>
         <Value>
            AccumulatedPowerUsed
         </Value>
      </Parameter>
      <Parameter>
         <ParameterPath>
            /UPnP/SensorMgt/SensorCollections/1/Sensor/1/SensorURNs/1/SensorURN/1
            /DataItems/1/Type
         </ParameterPath>
         <Value>
            uda:ui4
         </Value>
      </Parameter>
      <Parameter>
         <ParameterPath>
            /UPnP/SensorMgt/SensorCollections/1/Sensor/1/SensorURNs/1/SensorURN/1
            /DataItems/1/Encoding
         </ParameterPath>
         <Value>
            string
         </Value>
      </Parameter>
      <Parameter>
         <ParameterPath>
            /UPnP/SensorMgt/SensorCollections/1/Sensor/1/SensorURNs/1/SensorURN/1
            /DataItems/1/Description
         </ParameterPath>
         <Value>
            <!-- Note this document would be XML-escaped prior to insertion -->
            <!-- into this document -->
            <?xml version="1.0" encoding="UTF-8"?>
            <DataItemDescription
               xmlns="urn:schemas-upnp-org:ds:sdmdid"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:schemaLocation="
                  urn:schemas-upnp-org:smgt:sdmdid.xsd
                     http://www.upnp.org/schemas/ds/sdmdid.xsd"

               itemname="AccumulatedPowerUsed"
               access="rw">

               <description>
                  Accumulated Power Consumption (KWh)
               </description>

               <measurement
                  units="kW-h"
                  access="rw"
                  treatment="current"
                  accumulation="cumulative" />
            </DataItemDescription>
         </Value>
      </Parameter>
   </cms:ParameterValueList>" )
```

Note: GetValues() refers to ConfigrationManagement::GetValues()

4.6.2.2 Creating a DataStore table

Once information about a sensor is collected a DataTable can be created in the DataStore service based on this information.

Request:
```
CreateDataStoreTable(
   "<DataStoreTableInfo
    <?xml version="1.0" encoding="UTF-8"?>
    <DataTableInfo xmlns="urn:schemas-upnp-org:ds:dtinfo"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="urn:schemas-upnp-org:ds:dtinfo
            http://www.upnp.org/schemas/ds/dtinfo.xsd"

         tableGUID=""
         tableURN=
            "urn:upnp-org:smgt-surn:refrigerator:AcmeSensorsCorp-com
             :AcmeIntegratedController:FrigidaireCorp:rf217acrs:monitor"
         updateID=""
      >
      <datatableretain  duration="P6M" />
      <datarecord>
         <field name="ReceiveTimestamp"
                type="uda:dateTime"
                encoding="ascii"
                required="0" />
         <field name="AccumulatedPowerUsed"
                type="uda:ui4"
                encoding="ascii"
                required="1" />
         <field name="FreezerTemp"
                type="uda:i4"
                encoding="ascii"
                required="1" />
         <field name="GroceryTemp"
                type="uda:i4"
                encoding="ascii"
                required="1" />
         <field name="VegetableTemp"
                type="uda:i4"
                encoding="ascii"
                required="1" />
         <field name="DoorAlarm"
                type="uda:i4"
                encoding="ascii"
                required="1" />
         <field name="PowerFaultAlarm"
                type="uda:i4"
                encoding="ascii"
                required="1" />
         <field name="StatusInterval-Stored"
                type="xsd:duration"
                encoding="ascii"
                tableprop="1"
                required="0" />
         </datarecord>
      </DataTableInfo>" )
```

Response:
```
CreateDataStoreTable(
   "1c8a54c0-b5d7-11e2-9e96-0800200c9a66" )
```

Note: *CreateDataStoreTable()* refers to *DataStore::CreateDataStoreTable()*

The DataItem Description document (for each DataItem) can be stored in the DataTable Dictionary as follows:

```
Request:
SetDataStoreTableKeyValue(
   "1c8a54c0-b5d7-11e2-9e96-0800200c9a66",
   "ItemDescription-AccumulatedPowerUsed"
   "<?xml version="1.0" encoding="UTF-8"?>
    <DataItemDescription
      xmlns="urn:schemas-upnp-org:smgt:sdmdid"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="
         urn:schemas-upnp-org:smgt:sdmdid.xsd
         http://www.upnp.org/schemas/ds/sdmdid.xsd"

      itemname="AccumulatedPowerUsed"
      access="rw">

      <description>
         Accumulated Power Consumption (KWh)
      </description>

      <measurement
         units="kW-h"
         access="rw"
         treatment="current"
         accumulation="cumulative" />
   </DataItemDescription>" )

Response:
SetDataStoreTableKeyValue()
```

Note: *SetDataStoreTableKeyValue()* refers to *DataStore::SetDataStoreKeyValue()*

In addition the StatusInterval DataItem item is stored in the DataTable Dictionary since it is unlikely to change. This item is supplied by the DataTable Dictionary when read.

```
Request:
SetDataStoreTableKeyValue(
   "1c8a54c0-b5d7-11e2-9e96-0800200c9a66",
   “StatusInterval-Stored"
   "PT300S" )

Response:
SetDataStoreTableKeyValue()
```

4.6.2.3 Getting information about a DataStore table

A summary of DataTable(s) available in the DataStore service can be obtained as shown following example:

```
Request:
GetDataStoreInfo()

Response:
GetDataStoreInfo(
   "<?xml version="1.0" encoding="UTF-8"?>
    <DataStoreInfo xmlns="urn:schemas-upnp-org:ds:dsinfo"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="urn:schemas-upnp-org:ds:dsinfo
         http://www.upnp.org/schemas/ds/dsinfo.xsd">

      <datastoretables>
         <datastoretable
```

```
            tableGUID=
               "urn:upnp-org:smgt-surn:refrigerator:AcmeSensorsCorp-com
               :AcmeIntegratedController:FrigidaireCorp:rf217acrs:monitor"
            tableURN="1c8a54c0-b5d7-11e2-9e96-0800200c9a66"
            updateID="7">
         </datastoretable>
      </datastoretables>
   </DataStoreInfo>
```

Note: GetDataStoreInfo() refers to DataStore::GetDataStoreInfo()

Information about a specific DataTable can be obtained as shown following example:

Request:

```
GetDataStoreTableInfo(
   "1c8a54c0-b5d7-11e2-9e96-0800200c9a66",
```

Response:

```
GetDataStoreTableInfo(
   "<?xml version="1.0" encoding="UTF-8"?>
    <DataTableInfo xmlns="urn:schemas-upnp-org:ds:dtinfo"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="urn:schemas-upnp-org:ds:dtinfo
            http://www.upnp.org/schemas/ds/dtinfo.xsd"

         tableGUID="1c8a54c0-b5d7-11e2-9e96-0800200c9a66"
         tableURN=
            "urn:upnp-org:smgt-surn:refrigerator:AcmeSensorsCorp-com
             :AcmeIntegratedController:FrigidaireCorp:rf217acrs:monitor"
         updateID="7">
         <datatableretain duration="P6M" />
         <datarecord>
            <field   name="ReceivedTimestamp"
                     type="uda:dateTime"
                     encoding="ascii"
                     required="0" />
            <field   name="AccumulatedPowerUsed"
                     type="uda:ui4"
                     encoding="ascii"
                     required="1" />
            <field   name="FreezerTemp"
                     type="uda:i4"
                     encoding="ascii"
                     required="1" />
            <field   name="GroceryTemp"
                     type="uda:i4"
                     encoding="ascii"
                     required="1" />
            <field   name="VegetableTemp"
                     type="uda:i4"
                     encoding="ascii"
                     required="1" />
            <field   name="DoorAlarm"
                     type="uda:i4"
                     encoding="ascii"
                     required="1" />
            <field   name="PowerFaultAlarm"
                     type="uda:i4"
                     encoding="ascii"
                     required="1" />
            <field   name="StatusInterval-Stored"
                     type="xsd:duration"
                     encoding="ascii"
```

```
                      tableprop="1"
                      required="0" />
            </datarecord>
         </DataTableInfo>" )
```

Note: *GetDataStoreTableInfo()* refers to *DataStore::GetDataStoreTableInfo()*

4.6.2.4 Establishing a connection to a DataStore table

Once a DataTable is created, a URL to allow IoTManagementAndControl sensors to access the DataTable can be obtained by providing the *DataTableID* as follows:

Request:
```
GetDataStoreTransportURL(
   "1c8a54c0-b5d7-11e2-9e96-0800200c9a66" )
```

Response:
```
GetDataStoreTransportURL(
   "http://192.168.1.60/DataStoreTables/1c8a54c0-b5d7-11e2-9e96-0800200c9a66/1/" )
```

Note: *GetDataStoreTransportURL()* refers to *DataStore::GetDataStoreTransportURL()*

The DataStore transport URL is provided to the IoTManagementAndControl device SensorTransportGeneric service as follows. A *TransportConnectionID* is returned to allow for termination of this specific transport connection.

Request:
```
ConnectSensor(
   "Sensor0001",
   "SensorConnection_1",
   "urn:upnp-org:smgt-surn:refrigerator:AcmeSensorsCorp-com
    :AcmeIntegratedController:FrigidaireCorp:rf217acrs:monitor",
   "<?xml version="1.0" encoding="utf-8"?>
    <SensorRecordInfo
      xmlns="urn:schemas-upnp-org:smgt:srecinfo"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="urn:schemas-upnp-org:smgt:srecinfo
         http://www.upnp.org/schemas/smgt/srecinfo-v1.xsd">
      <sensorrecord>
         <field name="ReceiveTimestamp" />
         <field name="AccumulatedPowerUsed" />
         <field name="FreezerTemp" />
         <field name="GroceryTemp" />
         <field name="VegetableTemp" />
         <field name="DoorAlarm" />
         <field name="PowerFaultAlarm" />
      </sensorrecord>
   </SensorRecordInfo>"
   0,
   "http://192.168.1.60/DataStoreTables/1c8a54c0-b5d7-11e2-9e96-0800200c9a66/1/" )
```

Response:
```
ConnectSensor(
   "6f306300-b5e0-11e2-9e96-0800200c9a66" )
```

Note: *ConnectSensor()* refers to *SensorTransportGeneric::ConnectSensor()*

Once the *ConnectSensor()* action completes, the IoTManagementAndControl device can connect to the DataStore and issue HTTP-POST request to deliver DataRecord(s) directly to the connected DataTable.

```
HTTP-POST to:
   http://192.168.1.60/DataStoreTables/1c8a54c0-b5d7-11e2-9e96-0800200c9a66/1/)

POST DataStoreTables/1c8a54c0-b5d7-11e2-9e96-0800200c9a66/1/ HTTP/1.1
Host: example.com
Content-Type: text/xml
Content-Length: 417
<?xml version="1.0" encoding="UTF-8"?>
<DataRecords xmlns="urn:schemas-upnp-org:ds:drecs"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:schemas-upnp-org:ds:drecs
      http://www.upnp.org/schemas/ds/drecs-v1.xsd">
   <datarecord>
      <field name="ReceivedTimestamp">2013-07-01T10:00:00</field>
      <field name="AccumulatedPowerUsed">42</field>
      <field name="FreezerTemp">-10</field>
      <field name="GroceryTemp">5</field>
      <field name="VegetableTemp">10</field>
      <field name="DoorAlarm">0</field>
      <field name="PowerFaultAlarm">0</field>
   </datarecord>
   <datarecord>
      <field name="ReceivedTimestamp">2013-07-01T10:05:00</field>
      <field name="AccumulatedPowerUsed">65</field>
      <field name="FreezerTemp">-10</field>
      <field name="GroceryTemp">5</field>
      <field name="VegetableTemp">10</field>
      <field name="DoorAlarm">1</field>
      <field name="PowerFaultAlarm">0</field>
   </datarecord>
</DataRecords>
```

4.6.2.5 Retrieving information from a DataStore table

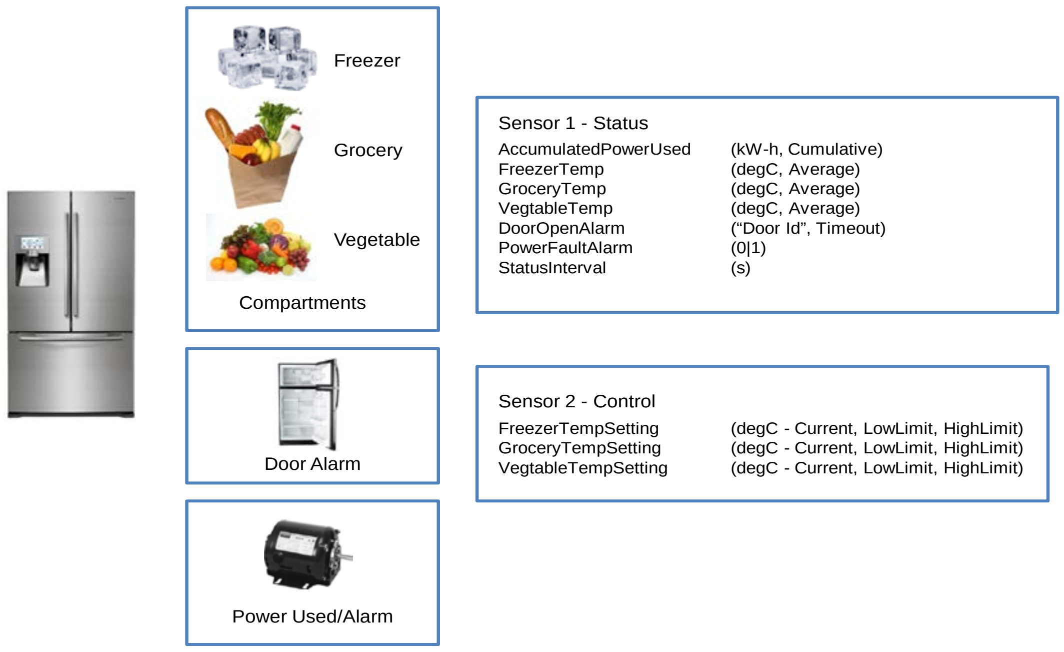
The following example show reading the DataTable records using a defined time range:

```
Request:
ReadDataStoreTableRecords(
   "1c8a54c0-b5d7-11e2-9e96-0800200c9a66",
   "<?xml version="1.0" encoding="UTF-8"?>
    <DataRecordFilter xmlns="urn:schemas-upnp-org:ds:dsfilter"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="urn:schemas-upnp-org:ds:dsfilter
         http://www.upnp.org/schemas/ds/dsfilter.xsd">
      <filterset>
         <filter condition="ReceiveTimestamp > 2013-07-01T00:00:00" />
         <filter condition="ReceiveTimestamp < 2013-07-02T00:00:00" />
      </filterset>
    </DataRecordFilter>"
   "0",
   0,
   1 )

Response:
ReadDataStoreTableRecords(
   "<?xml version="1.0" encoding="UTF-8"?>
    <DataRecords xmlns="urn:schemas-upnp-org:ds:drecs"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="urn:schemas-upnp-org:ds:drecs
         http://www.upnp.org/schemas/ds/drecs-v1.xsd">
      <datarecord>
         <field name="ReceivedTimestamp">2013-07-01T10:00:00</field>
         <field name="AccumulatedPowerUsed">42</field>
         <field name="FreezerTemp">-10</field>
         <field name="GroceryTemp">5</field>
         <field name="VegetableTemp">10</field>
         <field name="DoorAlarm">0</field>
         <field name="PowerFaultAlarm">0</field>
         <field name="StatusInterval-Stored">PT300S</field>
      </datarecord>
      <datarecord>
         <field name="ReceivedTimestamp">2013-07-01T10:05:00</field>
         <field name="AccumulatedPowerUsed">65</field>
         <field name="FreezerTemp">-10</field>
         <field name="GroceryTemp">5</field>
         <field name="VegetableTemp">10</field>
         <field name="DoorAlarm">1</field>
         <field name="PowerFaultAlarm">0</field>
         <field name="StatusInterval-Stored">PT300S</field>
      </datarecord>
      ... Additional DataRecord(s) for timerange requested ...
   </DataRecords>",
   "0" )
```

Note: Element values and attributes containing XML markup character(s) such as the DataRecord `<field>` element(s) or the `condition=` attribute of the `<filter>` element need to be included in their XML escaped representation.

Note: *ReadDataStoreTableRecords()* refers to *DataStore::ReadDataStoreTableRecords()*

Annex A Sample Device Illustration (informative)



Sensor 1 - Status

AccumulatedPowerUsed	(kW-h, Cumulative)
FreezerTemp	(degC, Average)
GroceryTemp	(degC, Average)
VegtableTemp	(degC, Average)
DoorOpenAlarm	(“Door Id”, Timeout)
PowerFaultAlarm	(0|1)
StatusInterval	(s)

Sensor 2 - Control

FreezerTempSetting	(degC - Current, LowLimit, HighLimit)
GroceryTempSetting	(degC - Current, LowLimit, HighLimit)
VegtableTempSetting	(degC - Current, LowLimit, HighLimit)

Figure A.1 — Sample Device

Parameters	Value
/UPnP/SensorMgt	
SensorCollectionsNumberofEntries	`1`
SensorCollections/	
1/CollectionID	`Collection0001`
1/CollectionType	`urn:upnp-org:smgt-sct:refrigerator:AcmeSensorsCorp-com:AcmeIntegratedController:FrigidaireCorp:rf217acrs`
1/CollectionFriendlyName	`"Your Refrigerator"`
1/CollectionInformation	`"Vendor Refrigerator Model RF217ACRS"`
1/CollectionUniqueIdentifier	`"123456789"`
1/CollectionSpecific	
1/SensorsNumberofEntries	`2`
1/Sensors/	
1/SensorID	`Sensor0001`
1/SensorType	`urn:upnp-org:smgt-st:refrigerator:AcmeSensorsCorp-com:AcmeIntegratedController:FrigidaireCorp:rf217acrs:monitor`
1/SensorUpdateRequest	`0`
1/SensorPollingInterval	`0`
1/SensorReportChangeOnly	`0`
1/SensorsRelatedNumberofEntries	`1`
1/SensorGroupsNumberofEntries	`1`

1/SensorPermissionsNumberOfEntries	`1`
1/SensorsRelated/	
1/SensorPath	`SensorCollections/1/Sensor/2`
1/SensorGroups	
1/SensorGroup	`ApplianceStatus`
1/SensorDefaultPermissions/	
1/SensorDefaultRole	`Basic`
1/SensorDefaultPermissions	`smgt:ViewSensor,smgt:ReadSensor,smgt:ConnectSensor`
1/SensorSpecific	
1/SensorURNsNumberOfEntries	`1`
1/SensorURNs	
1/SensorURN	`urn:upnp-org:smgt-surn:refrigerator:AcmeSensorsCorp-com:AcmeIntegratedController:FrigidaireCorp:rf217acrs:monitor`
1/DataItemsNumberOfEntries	`9`
1/DataItems/	
1/Name	`AccumulatedPowerUsed`
1/Type	`uda:ui4`
1/Encoding	`ascii`
1/Description	See Annex A.1.1.1
2/Name	`FreezerTemp`
2/Type	`uda:i4`
2/Encoding	`ascii`
2/Description	See Annex A.1.1.2
3/Name	`GroceryTemp`
3/Type	`uda:i4`
3/Encoding	`ascii`
3/Description	See Annex A.1.1.2
4/Name	`VegetableTemp`
4/Type	`uda:i4`
4/Encoding	`ascii`
4/Description	See Annex A.1.1.2
5/Name	`DoorOpenAlarm`
5/Type	`uda:boolean`
5/Encoding	`ascii`
5/Description	See Annex A.1.1.4
6/Name	`PowerFaultAlarm`
6/Type	`uda:boolean`
6/Encoding	`ascii`
6/Description	See Annex A.1.1.5
7/Name	`StatusInterval`
7/Type	`xsd:duration`
7/Encoding	`ascii`
7/Description	See Annex A.1.1.3
8/Name	`ReceivedTimestamp`

8/Type	xsd:dateTime
8/Encoding	ascii
8/Description	
9/Name	ClientID
9/Type	xsd:string
9/Encoding	utf-8
9/Description	
2/SensorID	Sensor0002
2/SensorType	urn:upnp-org:smgt-surn:refrigerator:AcmeSensorsCorp-com:AcmeIntegratedController:FrigidaireCorp:rf217acrs:setting
2/SensorUpdateRequest	0
2/SensorPollingInterval	0
2/SensorReportChangeOnly	0
2/SensorsRelated/	
1/SensorPath	SensorCollections/1/Sensor/1
2/SensorGroups	
1/SensorGroup	ApplianceControl
2/SensorDefaultPermissions/	
1/SensorDefaultRole	
1/SensorDefaultPermissions	
2/SensorSpecific	
2/SensorURNsNumberOfEntries	1
2/SensorURNs	
1/SensorURN	urn:upnp-org:smgt-surn:refrigerator:vendor-com:rf217acrs:control
1/DataItemsNumberOfEntries	3
1/DataItems/	
1/Name	FreezerTempSetting
1/Type	uda:i4,uda:i4,uda:i4
1/Encoding	Ascii
1/Description	See Annex A.1.1.6
2/Name	GroceryTempSetting
2/Type	uda:i4,uda:i4,uda:i4
2/Encoding	Ascii
2/Description	See Annex A.1.1.6
3/Name	VegetableTempSetting
3/Type	uda:i4,uda:i4,uda:i4
3/Encoding	ascii
3/Description	See Annex A.1.1.6

A.1.1 Sample DataItem Descriptions Documents

A.1.1.1 AccumulatedPowerUsed

```
<?xml version="1.0" encoding="UTF-8"?>
<DataItemDescription
  xmlns="urn:schemas-upnp-org:smgt:sdmdid"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="
    urn:schemas-upnp-org:smgt:sdmdid
    http://www.upnp.org/schemas/smgt/sdmdid.xsd"

  itemname="AccumulatedPowerUsed"
  access="rw">

  <description>
    Accumulated Power Consumption (KWh)
  </description>

  <measurement
    units="kW-h"
    access="rw"
    treatment="current"
    accumulation="cumulative" />

</DataItemDescription>
```

A.1.1.2 FreezerTemp (also GroceryTemp, VegetableTemp)

```
<?xml version="1.0" encoding="UTF-8"?>
<DataItemDescription
  xmlns="urn:schemas-upnp-org:smgt:sdmdid"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="
    urn:schemas-upnp-org:smgt:sdmdid
    http://www.upnp.org/schemas/smgt/sdmdid.xsd”

  itemname="FreezerTemp"
  access="ro">

  <description>
    Freezer (Avg) Compartment Temperature
  </description>

  <measurement
    units="degC"
    access="ro"
    treatment="average"
    accumulation="interval">

    <relateditem
      itemname="StatusInterval"
      relationtype="interval" />

  </measurement>

</DataItemDescription>
```

A.1.1.3 StatusInterval

```
<?xml version="1.0" encoding="UTF-8"?>
<DataItemDescription
  xmlns="urn:schemas-upnp-org:smgt:sdmdid"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="
    urn:schemas-upnp-org:smgt:sdmdid
    http://www.upnp.org/smgt/sdmdid.xsd"
  itemname="StatusInterval"
  access="ro">

  <description>
    Status Monitoring Interval (sec)
  </description>

  <interval
    units="duration"
    access="ro">

    <relateditem
      itemname="FreezerTemp"
      relationtype="measurement" />
    <relateditem
      itemname="GroceryTemp"
      relationtype="measurement" />
    <relateditem
      itemname="VegetableTemp"
      relationtype="measurement" />

  </interval>

</DataItemDescription>
```

A.1.1.4 DoorOpenAlarm

```
<?xml version="1.0" encoding="UTF-8"?>
<DataItemDescription
  xmlns="urn:schemas-upnp-org:smgt:sdmdid"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="
    urn:schemas-upnp-org:smgt:sdmdid
    http://www.upnp.org/schemas/smgt/sdmdid.xsd"

  itemname="DoorOpenAlarm"
  access="ro">

  <description>
    Door Open Alarm (Door Name, Timeout)
  </description>

  <alarm />
  <limit units="s"
         limittype="timeout"
         access="ro" />

</DataItemDescription>
```

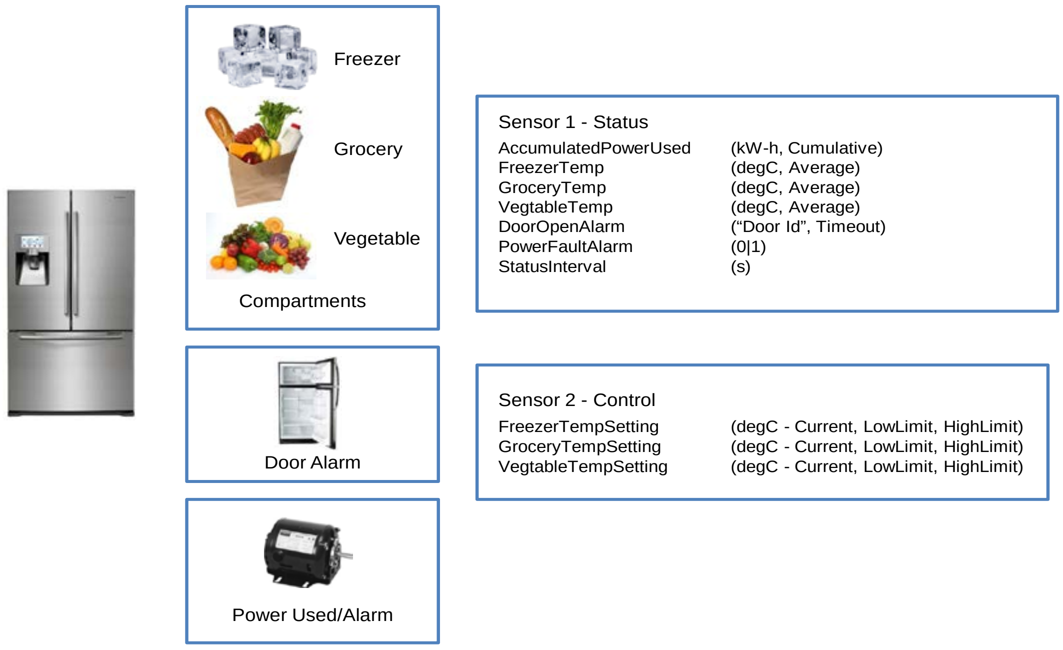
A.1.1.5 PowerFaultAlarm

```
<?xml version="1.0" encoding="UTF-8"?>
<DataItemDescription
  xmlns="urn:schemas-upnp-org:smgt:sdmdid"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="
    urn:schemas-upnp-org:smgt:sdmdid
    http://www.upnp.org/schemas/smgt/sdmdid.xsd"

  itemname="PowerFaultAlarm"
  access="ro">

  <description>
    Power Fault Alarm
  </description>

  <alarm>

    <relateditem
      itemname="StatusInterval"
      relationtype="reset_interval" />

  </alarm>

</DataItemDescription>
```

A.1.1.6 FreezerTempSetting, GroceryTempSetting, VegtableTempSetting

```
<?xml version="1.0" encoding="UTF-8"?>
<DataItemDescription
  xmlns="urn:schemas-upnp-org:smgt:sdmdid"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="
    urn:schemas-upnp-org:smgt:sdmdid
    http://www.upnp.org/schemas/smgt/sdmdid.xsd"

  itemname="FreezerTempSetting"
  access="rw">

  <description>
    Freezer Temperature Control
  </description>

  <setting units="degC"
           access="rw" />

  <measurement
    units="degC"
    treatment="average"
    accumulation="interval"
    access="ro" />

  <limit units="degC"
         limittype="low"
         access="ro" />

  <limit units="degC"
         limittype="high"
         access="ro" />

</DataItemDescription>
```

Annex B Using IoTManagementAndControl in the Internet of Things (informative)



Sensor 1 - Status

AccumulatedPowerUsed	(kW-h, Cumulative)
FreezerTemp	(degC, Average)
GroceryTemp	(degC, Average)
VegtableTemp	(degC, Average)
DoorOpenAlarm	(“Door Id”, Timeout)
PowerFaultAlarm	(0|1)
StatusInterval	(s)

Sensor 2 - Control

FreezerTempSetting	(degC - Current, LowLimit, HighLimit)
GroceryTempSetting	(degC - Current, LowLimit, HighLimit)
VegtableTempSetting	(degC - Current, LowLimit, HighLimit)

B.1.1 Internet of Things Architecture using UPnP IoTManagementAndControl

To be added.

B.1.2 Using the IoT Data Model

To be added.

B.1.3 Adding a New IoT Device (example)

To be added.

B.1.4 Other UPnP Specifications Supporting the Complete UPnP IoT Architecture

To be added.