IOT Management and Control Device

For UPnP Version 1.0

Status: Standardized DCP (SDCP)

Date: July 1, 2013

Document Version: 1.0

Service Template Version: 2.00

This Standardized DCP has been adopted as a Standardized DCP by the Steering Committee of the UPnP Forum, pursuant to Section 2.1(c)(ii) of the UPnP Forum Membership Agreement. UPnP Forum Members have rights and licenses defined by Section 3 of the UPnP Forum Membership Agreement to use and reproduce the Standardized DCP in UPnP Compliant Devices. All such use is subject to all of the provisions of the UPnP Forum Membership Agreement.

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.

CONTENTS

1 Scope

This document defines the device for IOT Management and Control, which identifies Level 1 of the device named IOT Management and Control. This Publicly Available Specification is applicable to Standardized DCPs of the UPnP Forum which include this device.

This device definition is compliant with the UPnP Device Architecture version 1.0 [1]. It defines a device type referred to herein as IOT Management and Control device.

This specification defines a general-purpose device that can be used to instantiate a bridge or gateway device that is connected to Sensors and Actuators. This device implements a bridge between the UPnP network and Sensors/Actuators endpoints. It exposes services to check the status, and send and receive data from Sensors. Sensors/Actuators may be connected using a variety of transport protocols including Bluetooth, ZigBee and IEEE 11073.

The device defined herein provides the following capabilities:

1. Access and control of multiple Sensors and Actuators.

2. Expose configuration information from Sensors and Actuators.

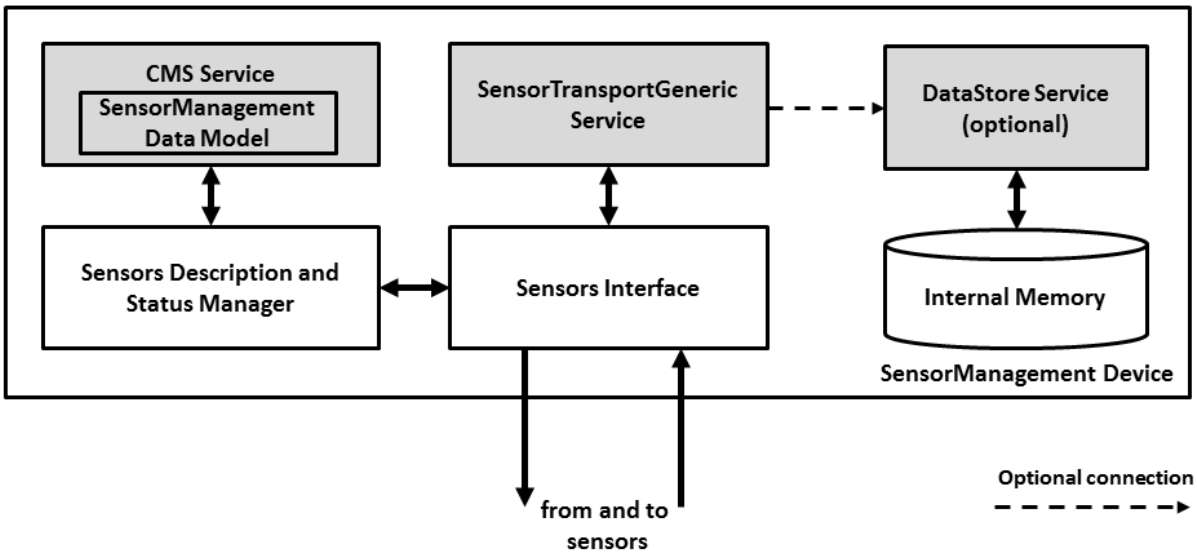

Figure 1 — IOT Management and Control Device Functional Diagram

The shaded boxes represent UPnP services that are contained by the IOT Management and Control device. The un-shaded boxes represent device specific modules that are vendor specific.

2 Normative References

The following referenced documents are indispensable for the application of this document. 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] – UPnP IoT Management and Control Architecture Overview, UPnP Forum, July 1, 2013. Available at: http://www.upnp.org/specs/iotmc/UPnP-iotmc-IoTManagementAndControl-Architecture-Overview-v1-20130701.pdf. Latest version available at:

http://www.upnp.org/specs/iotmc/UPnP-iotmc-IoTManagementAnd Control-Architecture-Overview-v1.pdf.

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

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

[5] – UPnP IoTManagementAndControl DataModel Service, UPnP Forum, July 1, 2013. Available at: http://www.upnp.org/specs/iotmc/UPnP-iotmc-IOTManagementAndControl-DataModel-v1-Service-20130701.pdf. Latest version available at: http://www.upnp.org/specs/iotmc/UPnP-iotmc-IOTManagementAndControl-rDataModel-v1-Service.pdf.

[6] – 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.

[7] – 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.

[8] – 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.

[9] – 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.

[10] – 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.

3 Terms, Definitions and Abbreviations

For the purposes of this document, the terms and definitions given in[1] and [2] apply.

4 Notations and conventions

4.1 Notation

- Strings that are to be taken literally are enclosed in “double quotes”.
- Words that are emphasized are printed in *italic*.
- Keywords that are defined by the UPnP Working Committee are printed using the *forum* character style.
- Keywords that are defined by the UPnP Device Architecture are printed using the **arch** character style.
- A double colon delimiter, “::”, signifies a hierarchical parent-child (parent::child) relationship between the two objects separated by the double colon. This delimiter is used in multiple contexts, for example: Service::Action(), Action()::Argument, parentProperty::childProperty.

4.2 Data Types

This specification uses data type definitions from two different sources. The UPnP Device Architecture defined data types are used to define state variable and action argument data types [1]. The XML Schema namespace is used to define property data types [10].

For UPnP Device Architecture defined Boolean data types, it is strongly RECOMMENDED to use the value "**0**" for false, and the value "**1**" for true. The values "**true**", "**yes**", "**false**", or "**no**" MAY also be used but are NOT RECOMMENDED. The values "**yes**" and "**no**" are deprecated and MUST NOT be sent out by devices but MUST be accepted on input.

For XML Schema defined Boolean data types, it is strongly RECOMMENDED to use the value "*0*" for false, and the value "*1*" for true. The values "*true*", "*yes*", "*false*", or "*no*" MAY also be used but are NOT RECOMMENDED. The values "*yes*" and "*no*" are deprecated and MUST NOT be sent out by devices but MUST be accepted on input.

4.3 Vendor-defined Extensions

Whenever vendors create additional vendor-defined state variables, actions or properties, their assigned names and XML representation MUST follow the naming conventions and XML rules as specified in UPnP Device Architecture [1], Clause 2.5, "Description: Non-standard vendor extensions".

5 Device Definitions

The requirements in this section apply only when *IOT Management and Control* device is used.

5.1 Device Type

The following URN identifies a device that is compliant with this specification:

urn:schemas-upnp-org:device:SensorManagement:1

IOT Management and Control device is used herein to refer to this device type.

5.2 Device Model

Products that expose IOT Management and Control devices must implement the minimum version numbers of all required embedded devices and services specified in the table below. A IOT Management and Control device can be either a *Root* device or can be *Embedded* in another UPnP device. A IOT Management and Control device (*Root* or *Embedded*) can in turn contain other standard or non-standard *Embedded* UPnP devices.

Table 1 — Device Requirements

DeviceType	Root	Req. or Opt. [a]	ServiceType	Req. or Opt. [a]	Service ID [b]
SensorManagement:1	Root or Embedded	R	ConfigurationManagement:2	R	Configuration Management
			SensorTransportGeneric:1	R	SensorTransportGeneric
			DeviceProtection:1	A	DeviceProtection
			DataStore:1	A	DataStore
			Standard non-EHS services defined by UPnP (QoS, Security, etc.) go here.	X	TBD
			Non-standard services embedded by a UPnP vendor go here.	X	TBD
Standard devices embedded by an UPnP vendor go here.	Embedded	O	Services as defined by the corresponding standard UPnP Device Definition go here.		
Non-standard devices embedded by an UPnP vendor go here.	Embedded	X	TBD	TBD	TBD

a R = required, A = allowed, CR = conditionally required, CA = conditionally allowed, X = Non-standard, add -D when deprecated (e.g., R-D, A-D).

b Prefixed by urn:upnp-org:serviceId:

5.2.1 Description of Device Requirements

Any instance of a IOT Management and Control device shall have two services: a single instance of the ConfigurationManagement service [7] and a single instance of the TransportGeneric service [3]. The DeviceProtection service [6] shall be present if Io Management and Control device implementation requires the security and privacy features, as described in the UPnP Iot Management and Control Architecture Overview [2] . An allowed DataStore service [4] may be present if IoT Management and Control device stores sensor's data information.

The ConfigurationManagement service [7] allows control points to configure and check the status of sensors managed by IoT Management and Control device. Additionally, the ConfigurationManagement service allows control points to discover useful information about sensors, such as its type, transport mechanism, etc. See UPnP IoT Management and Control DataModel service [5] for more details about the use of ConfigurationManagement service in the IoT Management and Control device and the UPnP IoT Management and Control Architecture Overview [2]

5.2.2 Relationships Between Services

The IoT Management and Control DataModel service [5] is the start point for sensors management using UPnP. Using the ConfigurationManagement service the control point shall check which transport protocols are supported by the device. The control point, then, shall use the TransportGeneric service [3] to read and write sensors data to a specific sensor using a supported data format.

Sensors managed by the TransportGeneric service [3] can be connected directly to other UPnP services or end-points. For example, a TransportGeneric service from a IoT Management and Control device can be connected to a DataStore service [4] from other

UPnP device using the *SensorTransportGeneric::ConnectDevice()* action. This connection make possible transfer of sensor's data without UPnP action calls or UPnP events. Sensors can be read directly using *SensorsTransportGeneric::ReadSensor()* action. Data can be written to sensors/actuators using *SensorTransportGeneric::WriteSensor()* action.

6 XML Device Description

```
<?xml version="1.0"?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
   <specVersion>
      <major>1</major>
      <minor>0</minor>
   </specVersion>
   <URLBase>base URL for all relative URLs</URLBase>
   <device>
      <deviceType>
         urn:schemas-upnp-org:device:SensorManagement:1
      </deviceType>
      <friendlyName>short user-friendly title</friendlyName>
      <manufacturer>manufacturer name</manufacturer>
      <manufacturerURL>URL to manufacturer site</manufacturerURL>
      <modelDescription>long user-friendly title</modelDescription>
      <modelName>model name</modelName>
      <modelNumber>model number</modelNumber>
      <modelURL>URL to model site</modelURL>
      <serialNumber>manufacturer's serial number</serialNumber>
      <UDN>uuid:UUID</UDN>
      <UPC>Universal Product Code</UPC>
      <iconList>
         <icon>
            <mimetype>image/format</mimetype>
            <width>horizontal pixels</width>
            <height>vertical pixels</height>
            <depth>color depth</depth>
            <url>URL to icon</url>
         </icon>
         XML to declare other icons, if any, go here
      </iconList>
      <serviceList>
         <service>
            <serviceType>
               urn:schemas-upnp-org:service:SensorTransportGeneric:1
            </serviceType>
            <serviceId>
               urn:upnp-org:serviceId:SensorTransportGeneric
            </serviceId>
            <SCPDURL>URL to service description</SCPDURL>
            <controlURL>URL for control</controlURL>
            <eventSubURL>URL for eventing</eventSubURL>
         </service>
         <service>
            <serviceType>
               urn:schemas-upnp-org:service:ConfigurationManagement:2
            </serviceType>
            <serviceId>
               urn:upnp-org:serviceId:ConfigurationManagement
            </serviceId>
            <SCPDURL>URL to service description</SCPDURL>
            <controlURL>URL for control</controlURL>
            <eventSubURL>URL for eventing</eventSubURL>
         </service>
         <service>
            <serviceType>
               urn:schemas-upnp-org:service:DataStore:1
            </serviceType>
            <serviceId>
               urn:upnp-org:serviceId:DataStore
```

```
            </serviceId>
            <SCPDURL>URL to service description</SCPDURL>
            <controlURL>URL for control</controlURL>
            <eventSubURL>URL for eventing</eventSubURL>
         </service>
         <service>
            <serviceType>
               urn:schemas-upnp-org:service:DeviceProtection:1
            </serviceType>
            <serviceId>
               urn:upnp-org:serviceId:DeviceProtection
            </serviceId>
            <SCPDURL>URL to service description</SCPDURL>
            <controlURL>URL for control</controlURL>
            <eventSubURL>URL for eventing</eventSubURL>
         </service>

         Declarations for standard non-EHS services defined by UPnP
         (if any) go here
         Declarations for other services added by UPnP vendor
         (if any) go here
      </serviceList>
      <deviceList>
      Description of embedded devices added by UPnP vendor
      (if any) go here
      </deviceList>
      <presentationURL>URL for presentation</presentationURL>
   </device>
</root>
```

Annex A - Theory of Operation (informative)

A.1 Introduction

A IoT Management and Control device allows a control point to read from and write data to Sensors or Actuators. IoT Management and Control devices can be used in conjunction with one or more UPnP devices to store sensor's data using a DataStore service [4]. The process starts when a control point discovers a IoT Management and Control device. The control point can read and write data directly to IoT Management and Control device using SensorTransportGeneric service [3]. The control point can also connect a IoT Management and Control device to an external storage device using *SensorTransportGeneric::ConnectSensor()* and *SensorTransportGeneric::DisconnectSensor()* actions. The data transfer is performed directly by the IoT Management and Control device and the other device. The data transfer happens independently from the control point. The control point uses UPnP to setup the data transfer, but the transfer is performed using HTTP based transfer protocol.

Therefore, the main operations performed by IoT Management and Control devices are:

- Discovery, configuration and status reading of sensors and actuators.
- Reading and writing of data to sensors or actuators using SensorTransportGeneric service.
- Connecting of IoT Management and Control to external storage devices, where these devices can be UPnP devices or any other type of device supporting HTTP transport.
- Provide allowed access control based on privacy and security settings.

A.2 Device Discovery

Control points can discover IoT Management and Control devices using the standard UPnP SSDP-based device discovery mechanism to search for any device that is a member of the IOT Management and Control device class including *Root* devices and/or *Embedded* devices.

A.3 Sensor Discovery, Configuration and Status Reading

Using the ConfigurationManagement service [7] (CMS), Control Points find description information about Sensors and Actuators managed by IoT Management and Control device. CMS allows Control points to retrieve and manipulate information regarding transport type, sensor IDs, description, status of sensors, etc. For more information about what information is available using ConfigurationManagement service see the IoT Management and Control DataModel service document [5].

Information provided by ConfigurationManagement service shall be used in actions provided by TransportGeneric service [3]. For example, if a Control Point invokes *SensorTransportGeneric::Connect()* action, it is necessary to first have the *SensorTransportGeneric::SensorID* information, which is available as a Parameter in the IoT Management and Control DataModel service [5]. The IoT Management and Control DataModel service also describes what type of Sensors and Actuators are available in the IOT Management and Control device. The type information can be used by Control Points to identify DataItems to be included in the DataRecord provided by the IOT Management and Control device.

A.4 Reading and Writing of Data

Reading and writing of data to Sensors and Actuators is the main function of a IoT Management and Control device. For this function, a Control Point first discovers the *SensorTransportGeneric::SensorID* of the Sensor. The Control Point must use CMS actions to discover and read the parameter */UPnP/SensorMgt/SensorDevice/#/Sensors/#/SensorID*. With the *SensorID* value, the Control Point can invoke *SensorTransportGeneric::ReadSensor()* or *SensorTransportGeneric::WriteSensor()* actions. Figure A.1 illustrates how the reading and writing flow is executed. This flow can be described as follows:

1) The Control Point (CP) uses the IoT Management and Control device's Configuration Management service to get the *SensorID* of the Sensor device that will be used to read and write operations.
2) CP invokes *SensorTransportGeneric::WriteSensor* to write DataRecords in a Sensor device and *SensorTransportGeneric::ReadSensor* to read DataRecords from a Sensor Device.
3) SensorTransportGeneric service interfaces with Sensors device and execute the necessary operations.

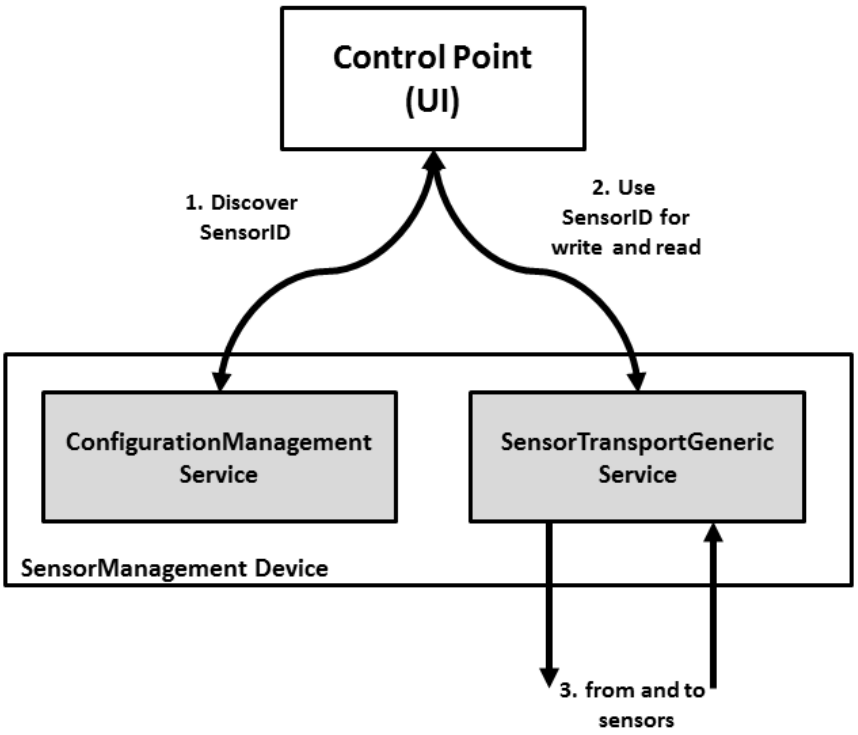

Figure A.1 —Simple Read and Write Flow Diagram.

A.5 Connecting Sensors to Transport Clients

A IoT Management and Control device can connect to transport client(s) in the UPnP network. This includes devices hosting a DataStore service [4] as well as other UPnP transport clients intending to accept sensor data. Figure A.2 illustrates the flow to connect a IOT Management and Control device to an external DataStore service. This flow can be described as follows:

1) The Control Point (CP) uses the IoT Management and Control device's Configuration Management service to get the *SensorID* of the Sensor device that will be connected with the external storage.
2) CP invokes *DataStore::GetDataStoreTransportURL()* to retrieve the URL that shall receive DataRecords from the Sensor.
3) CP invokes *SensorTransportGeneric::Connect()* to connect the Sensor with the DataStore using the *SensorID* and transport URL acquired previously.
4) TransportGeneric service receives new data from the Sensor and creates DataRecords.
5) New DataRecords available for the Sensor are delivered directly to the DataStore using the previously configured transport URL.

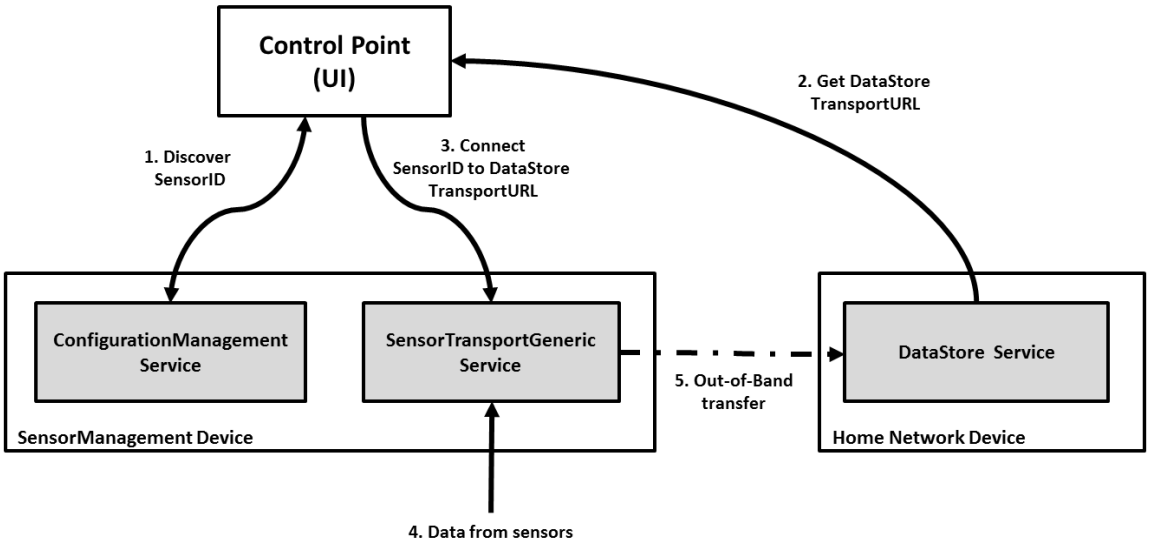

Figure A.2 — Connecting a IoT Management and Control Device to an External Storage.