HouseStatus:1 Service Template

For UPnP™ Device Architecture V 1.0

Status: Standardized DCP
Date: May 13th, 2003

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 Membership Agreement. UPnP Forum Members have rights and licenses defined by Section 3 of the UPnP 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 Membership Agreement.

Authors	Company
Andrew Fiddian-Green	Siemens Building Technologies Ltd.
Larry Stickler	Honeywell Inc.

Contents

List of Tables

1. Overview and Scope

This service definition is compliant with the UPnP Device Architecture version 1.0.

This service-type provides an indication about house occupancy status and operational mode. It is commonly used as a mechanism for influencing the state of Devices and/or Control Points depending upon whether people are in the house. Typical applications are: e.g. switching on or off lights, air-conditioning etc.

Occupancy status can be derived i) directly from an occupant via a user interface, or ii) indirectly by algorithms such as a calendar program, or iii) indirectly by heuristics that determine the status from subsystem or device activity. That is to say: this service type would be implemented in two types of occupancy "detector" devices:

- A physical switch (e.g. a home/away push button on a device).
- A "virtual" switch that uses some kind of algorithm or heuristics to work out if the house is occupied (e.g. a calendar or a predictive algorithm).

In case a) the physical detection device would incorporate this occupancy service, but in case b) the MMI of the device that contains the algorithm would incorporate this occupancy service.

This service is a “source” of UPnP event messages. Control Points that are interested to be updated about the occupancy state of the house should subscribe to receive events from this service type. (However, Control Points are also permitted to “poll” the service from time to time in order to enquire about the current occupancy state).

This service template does not address:

- Presence detection for security alarm purposes.
- Actual number of persons in the building (or building part) e.g. for demand controlled ventilation in (say) a conference room.

1.1. Change Log

[30 Aug 2000]	v0.1	Document created.
[18 Dec 2000]	v0.7	Inputs from Larry Stickler regarding HPnP concepts; in particular the issue of synchronizing between multiple instances of the service.
[10 April 2001]	v0.71	Scope reduced from general occupancy detection to simple “house state”. Service name changed from Occupancy to HouseStatus. Migrated to template v1.01
[11 April 2001]	v0.72	ActivityLevel added. Also some tweaks.
[15 May 2001]	v0.73	DormancyLevel state variable added. “Set” actions added. NOTE: since it is now possible to set the state from outside the device, the DeterminationMethod is no longer meaningful; therefore it is deleted also…
[20 July 2001]	v0.8	PetsAtHome state added. Moved to Template Design Complete
[6 Aug 2001]	v0.81	Corrected according to checklist review by Hans Langels
[31 May 2002]	v0.9	Revision marks removed. Test chapter added.

[13 May 2003] v1.0 Converted to Approved Standard.

2. Service Modeling Definitions

2.1. Service Type

The following service type identifies a service that is compliant with this template:

urn:schemas-upnp-org:service:*HouseStatus:1*

2.2. State Variables

Table 1: State Variables

Variable Name	Req. or Opt.[1]	Data Type	Allowed Value [2]	Default Value [2]	Eng. Units
OccupancyState	R	string	Occupied, Unoccupied, Indeterminate	Occupied	none
ActivityLevel	O	string	See table 2	See table 3	none
DormancyLevel	O	string	See table 4	See table 5	none
Non-standard state variables implemented by an UPnP vendor go here.	*X*	*TBD*	*TBD*	*TBD*	*TBD*

[1] R = Required, O = Optional, X = Non-standard.

[2] Values listed in this column are required. To specify standard optional values or to delegate assignment of values to the vendor, you must reference a specific instance of an appropriate table below.

2.2.1. OccupancyState

This is a read only variable that represents the occupancy status of the house, whereby:

- **Occupied** = People in the house
- **Unoccupied** = No people in the house
- **Indeterminate** = The service is unable to determine if the house is occupied or not.

2.2.2. ActivityLevel

This is an optional read only variable that acts as a qualifier to provide an extra level of detail concerning the occupancy status of the house. It indicates the level of activity of the occupants. Whereby:

- **Regular** = Indicates that the house is in a neutral/normal state of occupancy. Note: ActivityLevel is optional, so in case of a service where ActivityLevel is NOT implemented, a Control Point should assume that the activity level is regular.

- **Asleep** = Means that although the house is occupied, the occupants are asleep – meaning that the degree of activity is lower than "Regular". (example: this could be used to adjust the temperature or switch off lights).
- **HighActivity =** Means that the house is occupied with a degree of activity that is higher than "Regular" – e.g. for a party. (example: this could be used to increase the speed of a ventilation fan).

Table 2: AllowedValueList for ActivityLevel

Value	Req. or Opt.
Regular	R
Asleep	R
HighActivity	R
Vendor-defined	O

[1] R = Required, O = Optional

Table 3: DefaultValue for ActivityLevel

Value	Req. or Opt.
Regular	R

[1] R = Required, O = Optional

2.2.3. DormancyLevel

This is an optional read only variable that acts as a qualifier to provide an extra level of detail concerning the un-occupied status of the house. It indicates the expected duration of the absence. Whereby:

- **Vacation** = Means that the duration of the unoccupied period is expected to be, (but not guaranteed to be), longer than 1 day.
- **Regular =** Indicates that the house is in a neutral/normal state of un-occupancy. Note: DormancyLevel is optional, so in case of a service where DormancyLevel is NOT implemented, a Control Point should assume that the level is regular.
- **PetsAtHome** = The house is not occupied by humans, but there may be pets moving around inside the house. (i.e. the security motion detectors inside the house should not be armed).

Table 4: AllowedValueList for DormancyLevel

Value	Req. or Opt.
Vacation	R
Regular	R
PetsAtHome	R
Vendor-defined	O

[1] R = Required, O = Optional

Table 5: DefaultValue for DormancyLevel

Value	Req. or Opt.
Regular	R

[1] R = Required, O = Optional

2.2.4. Relationships Between State Variables

None.

2.3. Eventing and Moderation

For the HouseStatus service, an example event subscription would be as follows:

a) A thermostat (say) implements the HouseStatus service in the physical form of a home / away button on the front.

b) The Lights, Answering machine, Oven, Lawn Sprinkler (or whatever) do an M-SEARCH for an HouseStatus service, and when they find one, they subscribe to receive events from that service using SUBSCRIBE. (In principle, the thermostat could also use a subscription to its own HouseStatus service too, but in practice since it is in the same device, it would probably use an internal signal instead).

c) When the homeowner presses the home / away button on the thermostat, the service publishes (sources) the corresponding OccupancyState change event NOTIFY message.

d) The Lights, Answering machine, Oven, Lawn Sprinkler (or whatever) and thermostat all receive (sink) this notification and switch over into their "Occupied" respectively "Unoccupied" mode accordingly.

Table 6: Event Moderation

Variable Name	Evented	Moderated Event?	Max Event Rate [1]	Logical Combination	Min Delta per Event [2]
OccupancyState	Yes	Yes	30	no	See note 3
ActivityLevel	Yes	Yes	30	no	See note 3
DormancyLevel	Yes	Yes	30	no	See note 3
Non-standard state variables implemented by an UPnP vendor go here.	*TBD*	*TBD*	*TBD*	*TBD*	*TBD*

[1] Determined by N, where Rate = (Event)/(N secs).
[2] (N) * (allowedValueRange Step).

Note 3: Never send events if the value of the state variable has not changed! This is important for avoiding race conditions. See section 2.6

2.3.1. Event Model

2.3.1.1. Delayed Response

Depending on the physical implementation of the containing device, a state change of HouseStatus may be triggered in a number of ways: e.g. “going out” button on the front of a security panel, “standby” button on a room thermostat etc. Especially, in the case of human operated buttons, etc. it is obvious that the person will still remain in the building for a certain period even AFTER the button has been set to Unoccupied mode. Therefore any Control Point that subscribes to HouseStatus events must make allowances for this.

Example: After receiving an OccupancyState=Unoccupied event, a lighting circuit should perhaps wait for a certain period of time before turning off the lights. However, in the case of receiving an OccupancyState=Occupied event, it should probably turn on the lights immediately.

2.4. Actions

Table 7: Actions

Name	Req. or Opt. [1]
GetOccupancyState	R
GetActivityLevel	O
GetDormancyLevel	O
SetOccupancyState	R
SetActivityLevel	O
SetDormancyLevel	O
Non-standard actions implemented by an UPnP vendor go here.	X

[1] R = Required, O = Optional, X = Non-standard.

2.4.1. GetOccupancyState

Reads the current value of OccupancyState

2.4.1.1. Arguments

Table 8: Arguments for GetOccupancyState

Argument	Direction	relatedStateVariable
CurrentOccupancyState	OUT [R]	OccupancyState

[R] = Return Value (RETVAL)

2.4.1.2. Effect on State

None.

2.4.1.3. Errors

errorCode	errorDescription	Description
402	Invalid Args	See UPnP Device Architecture section on Control.
501	Action Failed	See UPnP Device Architecture section on Control.
800-899	*TBD*	*(Specified by UPnP vendor.)*

2.4.2. GetActivityLevel

Reads the current value of ActivityLevel

2.4.2.1. Arguments

Table 9: Arguments for GetActivityLevel

Argument	Direction	relatedStateVariable
CurrentActivityLevel	OUT [R]	ActivityLevel

[R] = Return Value (RETVAL)

2.4.2.2. Effect on State

None.

2.4.2.3. Errors

errorCode	errorDescription	Description
402	Invalid Args	See UPnP Device Architecture section on Control.
501	Action Failed	See UPnP Device Architecture section on Control.
800-899	*TBD*	*(Specified by UPnP vendor.)*

2.4.3. GetDormancyLevel

Reads the current value of DormancyLevel

2.4.3.1. Arguments

Table 10: Arguments for GetDormancyLevel

Argument	Direction	relatedStateVariable
CurrentDormancyLevel	OUT [R]	DormancyLevel

[R] = Return Value (RETVAL)

2.4.3.2. Effect on State

None.

2.4.3.3. Errors

errorCode	errorDescription	Description
402	Invalid Args	See UPnP Device Architecture section on Control.
501	Action Failed	See UPnP Device Architecture section on Control.
800-899	*TBD*	*(Specified by UPnP vendor.)*

2.4.4. SetOccupancyState

Changes the value of OccupancyState.

2.4.4.1. Arguments

Table 11: Arguments for SetOccupancyState

Argument	Direction	relatedStateVariable
NewOccupancyState	IN	OccupancyState

R = Return Value (RETVAL)

2.4.4.2. Effect on State

Changes the value of OccupancyState.

2.4.4.3. Errors

errorCode	errorDescription	Description
402	Invalid Args	See UPnP Device Architecture section on Control.
501	Action Failed	See UPnP Device Architecture section on Control.
800-899	*TBD*	*(Specified by UPnP vendor.)*

2.4.5. SetActivityLevel

Changes the value of ActivityLevel

2.4.5.1. Arguments

Table 12: Arguments for SetActivityLevel

Argument	Direction	relatedStateVariable
NewActivityLevel	IN	ActivityLevel

R = Return Value (RETVAL)

2.4.5.2. Effect on State

Changes the value of ActivityLevel

2.4.5.3. Errors

errorCode	errorDescription	Description
402	Invalid Args	See UPnP Device Architecture section on Control.
501	Action Failed	See UPnP Device Architecture section on Control.
800-899	*TBD*	*(Specified by UPnP vendor.)*

2.4.6. SetDormancyLevel

Changes the value of DormancyLevel.

2.4.6.1. Arguments

Table 13: Arguments for SetDormancyLevel

Argument	Direction	relatedStateVariable
NewDormancyLevel	IN	DormancyLevel

[R] = Return Value (RETVAL)

2.4.6.2. Effect on State

Changes the value of DormancyLevel.

2.4.6.3. Errors

errorCode	errorDescription	Description
402	Invalid Args	See UPnP Device Architecture section on Control.
501	Action Failed	See UPnP Device Architecture section on Control.
800-899	*TBD*	*(Specified by UPnP vendor.)*

2.4.7. Non-Standard Actions Implemented by a UPnP Vendor

To facilitate certification, non-standard actions implemented by UPnP vendors should be included in this service template. The UPnP Device Architecture lists naming requirements for non-standard actions (see the section on Description).

2.4.8. Common Error Codes

The following table lists error codes common to actions for this service type. If an action results in multiple errors, the most specific error should be returned.

Table 14: Common Error Codes

ErrorCode	errorDescription	Description
401	Invalid Action	See UPnP Device Architecture section on Control.
402	Invalid Args	See UPnP Device Architecture section on Control.
404	Invalid Var	See UPnP Device Architecture section on Control.
501	Action Failed	See UPnP Device Architecture section on Control.

ErrorCode	errorDescription	Description
600-699	TBD	Common action errors. Defined by UPnP Forum Technical Committee.
701-799		Common action errors defined by the UPnP Forum working committees.
800-899	*TBD*	*(Specified by UPnP vendor.)*

2.5. Theory of Operation

This service-type provides an indication about house occupancy status and operational mode. It is commonly used as a mechanism for influencing the state of Control Points depending upon whether people are in the house. For example, when the house is unoccupied, the environmental system may use a different heating or cooling goal, or the lighting system may turn unneeded lights off.

Typically such a Control Point may SUBSCRIBE to receive occupied / unoccupied events from the OccupancyState variable; whenever it receives the respective event notifications, it will adjust it's state accordingly. Alternatively a Control Point may poll this service using the GetOccupancyState function.

It is the responsibility of each device vendor to select a behavior --a strategy-- that is appropriate for best supporting the user's desire for a given state reported by the HouseStatus service. This mechanism should provide a simple, coordinated way for users to operate their homes.

Occupancy State	Activity Level	Dormancy Level	Remarks
Occupied	Regular		There are people in the house. There is a "normal" (i.e. regular) amount of occupant activity.
	Asleep		There are people in the house. There is a ***below*** "normal" amount of occupant activity – e.g. the occupants are sleeping.
	HighActivity		There are people in the house. There is an ***above*** "normal" amount of occupant activity – e.g. the occupants are partying.
Unoccupied		Regular	There are no people in the house (i.e. the house is "dormant"). The level of dormancy is "normal" – e.g. the occupants are out at work, and they are expected to return in the evening.
		Vacation	There are no people in the house. The level of dormancy is ***above*** "normal" – e.g. the occupants are away on vacation (say > 48 hours absence).
		PetsAtHome	There are no people in the house. The level of dormancy is ***below*** "normal" – e.g. the

Occupancy State	Activity Level	Dormancy Level	Remarks
Indeterminate			The service is currently unable to ascertain the state of the house,

2.6. Synchronization of Multiple Instances

It is conceivable that a home might contain more than one instance of the HouseStatus service (e.g. a home / away button on the thermostat, and another one on the security panel). This could lead to potentially confusing situations where one device might indicate an occupied status and another might indicate an unoccupied status. In order to resolve this issue, vendors MUST to do the following:

i) Each device that implements the HouseStatus service, MUST also implement a Control Point.

ii) On initialization, (and from time to time thereafter), this Control Point must search for all other instances of HouseStatus.

iii) This Control Point must subscribe to receive event notifications from all such other instances of HouseStatus that it finds.

iv) Whenever this Control Point receives an occupancy state change notification from another instance of the HouseStatus service, the service must update the value of its OccupancyState, ActivityLevel or DormancyLevel variables to the same value(s) as that received in the notification message.

v) If (and only if) the value of its own OccupancyState, ActivityLevel or DormancyLevel variables change, the service shall in turn notify all other Control points that have subscribed to it. NB the HouseStatus service should NEVER send state change notifications if the value of its OccupancyState, ActivityLevel or DormancyLevel variables have not changed – since otherwise race conditions could develop with two or more instances of the HouseStatus service notifying each other in a circular fashion.

3. XML Service Description

```
<?xml version="1.0"?>
<scpd xmlns="urn:schemas-upnp-org:service-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <actionList>
   <action>
    <name>GetOccupancyState</name>
      <argumentList>
        <argument>
          <name>CurrentOccupancyState</name>
          <direction>out</direction>
          <retval />
          <relatedStateVariable>OccupancyState</relatedStateVariable>
        </argument>
      </argumentList>
   </action>
   <action>
    <name>GetActivityLevel</name>
      <argumentList>
        <argument>
          <name>CurrentActivityLevel</name>
          <direction>out</direction>
          <retval />
          <relatedStateVariable>ActivityLevel</relatedStateVariable>
        </argument>
      </argumentList>
   </action>
   <action>
    <name>GetDormancyLevel</name>
      <argumentList>
        <argument>
          <name>CurrentDormancyLevel</name>
          <direction>out</direction>
          <retval />
          <relatedStateVariable>DormancyLevel</relatedStateVariable>
        </argument>
      </argumentList>
   </action>
   <action>
    <name>SetOccupancyState</name>
      <argumentList>
        <argument>
          <name>NewOccupancyState</name>
          <direction>in</direction>
          <relatedStateVariable>OccupancyState</relatedStateVariable>
        </argument>
      </argumentList>
   </action>
   <action>
    <name>SetActivityLevel</name>
```

```
      <argumentList>
        <argument>
          <name>NewActivityLevel</name>
          <direction>in</direction>
          <relatedStateVariable>ActivityLevel</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
    <action>
     <name>SetDormancyLevel</name>
      <argumentList>
        <argument>
          <name>NewDormancyLevel</name>
          <direction>in</direction>
          <relatedStateVariable>DormancyLevel</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
Declarations for other actions added by UPnP vendor (if any) go here
  </actionList>
  <serviceStateTable>
    <stateVariable sendEvents="yes">
      <name>OccupancyState</name>
      <dataType>string</dataType>
      <defaultValue>Occupied</defaultValue>
      <allowedValueList>
        <allowedValue>Occupied</allowedValue>
        <allowedValue>Unoccupied</allowedValue>
        <allowedValue>Indeterminate</allowedValue>
      </allowedValueList>
    </stateVariable>
    <stateVariable sendEvents="yes">
      <name>ActivityLevel</name>
      <dataType>string</dataType>
      <defaultValue>Regular</defaultValue>
      <allowedValueList>
        <allowedValue>Regular</allowedValue>
        <allowedValue>Asleep</allowedValue>
        <allowedValue>HighActivity</allowedValue>
      </allowedValueList>
    </stateVariable>
    </stateVariable>
    <stateVariable sendEvents="yes">
      <name>DormancyLevel</name>
      <dataType>string</dataType>
      <defaultValue>Regular</defaultValue>
      <allowedValueList>
        <allowedValue>Regular</allowedValue>
        <allowedValue>Vacation</allowedValue>
        <allowedValue>PetsAtHome</allowedValue>
      </allowedValueList>
    </stateVariable>
Declarations for other state variables added by UPnP vendor (if any)
    go here
```

```
  </serviceStateTable>
</scpd>
```

4. Test

Testing of the UPnP functions Addressing, Discovery, Description, Control (Syntax) and Eventing are performed by the UPnP Test Tool v1.1 based on the following documents:

- UPnP Device Architecture v1.0
- The Service Definitions in chapter 2 of this document
- The XML Service Description in chapter 3 of this document
- The UPnP Test Tool service template test file: ***HouseStatus1.xml***
- The UPnP Test Tool service template test file: ***HouseStatus1.SyntaxTests.xml***

The test suite does not include tests for Control Semantics, since it is felt that such tests would not provide a higher level of interoperability.