# UPnP FanSpeed:1 Service Template Version 1.01

**For UPnP Device Architecture 1.0**

**Status: Standardized DCP**
**Date: September 21st, 2007**





| Authors | Company |
|---|---|
| Larry Stickler | Honeywell |
| Andrew Fiddian-Green | Siemens Building Technologies |

# Contents



# List of Tables

# 1. Overview and Scope

This service definition is compliant with the UPnP Device Architecture version 1.0. It defines a service type referred to herein as FanSpeed:1

FanSpeed:1 provides programmatic control and status information for air fans used in Heating Ventilation and Air-Conditioning (HVAC) applications. It allows a control point to command the speed of the fan by means of a continuous 0% to 100% control variable. Fans which are On/ Off or three speed (Off/ Low/ Medium/ High) respond by mapping the continuous control variable to specific vendor dependant switching points. It provides optional functionality for dual direction reversible fans.

FanSpeed:1 enables the following functions:

- Control of the speed of an air-conditioning or ventilation fan.
- Reversible fans

# 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:*FanSpeed:1***

The shorthand FanSpeed:1 is used herein to refer to this service type.

## 2.2. State Variables

Defines the state variables for the target running speed of the fan and its actual speed. Additionally defines optional state variables for “forward” and “reverse” operation.

NOTE: (Explanation of the meaning of speed): Table 1 below describes Allowed Value ranges of 0 to 100 which signify a fan speed in the range of 0% to 100%. In all such cases, a value of 0% corresponds to a FULLY STOPPED physical condition, and a value of 100% corresponds to the FULL SPEED physical condition. For values between 0% and 100% the physical condition of the fan is mapped as closely as possible to the 0% to 100% control variable. In particular for fans with discrete speeds (e.g. Off/ Low/ Medium/ High) the mapping takes the form of a “staircase”. The exact mapping is left to the vendor’s discretion.

**Table 1: State Variables**

| Variable Name | Req. or Opt.[1] | Data Type | Allowed Value[2] | Default Value[2] | Eng. Units |
|---|---|---|---|---|---|
| FanSpeedTarget | R | ui1 | >= 0, <= 100, += 1 | 0 | Percent |
| FanSpeedStatus | R | ui1 | >= 0, <= 100, += 1 | 0 | Percent |
| DirectionTarget | O | boolean | 0 = “Forward”,<br>1 = “Reverse” | 0 | n/a |
| DirectionStatus | O | boolean | 0 = “Forward”,<br>1 = “Reverse” | 0 | n/a |
| *Non-standard state variables implemented* | *X* | *TBD* | *TBD* | *TBD* | *TBD* |

| Variable Name | Req. or Opt.[1] | Data Type | Allowed Value[2] | Default Value[2] | Eng. Units |
|---|---|---|---|---|---|
| *by an UPnP vendor go here.* | | | | | |

[1] R = Required, O = Optional, X = Non-standard.
[2] Values listed in this column are (all) required.

### 2.2.1. FanSpeedTarget

Determines the target speed for the fan. (See the above note "Explanation of the meaning of speed").

### 2.2.2. FanSpeedStatus

Represents the actual speed for the fan. (See the above note "Explanation of the meaning of speed").

### 2.2.3. DirectionTarget

Determines the target running direction for the fan. This is an optional state variable; in the case of fans that do not implement this state variable, they must behave as if DirectionTarget were equal to 0 i.e. "Forward".

### 2.2.4. DirectionStatus

Represents the actual running direction for the fan. This is an optional state variable; in the case of fans that do not implement this state variable, a control point must behave as if DirectionStatus were equal to 0 i.e. "Forward".

### 2.2.5. Relationships Between State Variables

Whenever the value of FanSpeedTarget changes, the actual physical fan speed should start to change toward the value of FanSpeedTarget according to the mapping illustrated in the examples below. Due to the physical inertia of the fan, this process will take a certain period of time that depends on the vendor's implementation. The value of the FanSpeedStatus state variable should correspond to the actual physical fan speed according to the mapping illustrated in the examples below.

FanSpeedTarget and FanSpeedStatus are integers with the range 0% to 100%. Depending on the actual type of fan employed (e.g. three speed fan, modulating fan etc.), the 0…100% range should map to the actual physical fan speed according to the following principles.

Two common examples are given below for guidance, but actual implementation is at the discretion of the vendor:

**Table 2: Modulating Fan Example**

| Input of Setting of FanSpeedTarget | Resulting Actual Physical Speed | Resulting value of FanSpeedStatus |
|---|---|---|
| 0% | Off ("hard" off) | 0% |
| 1…Minimum Speed (i.e. Stalling Speed)% | Off ("soft" off) | 1% |
| Min. Stall Speed…100% | Linear mapping according to the value of FanSpeedTarget | Actual speed:<br>(Min. Stall Speed … 100%) |

**Table 3: Three-Speed Fan Example**

| Input of Setting of FanSpeedTarget | Resulting Actual Physical Speed | Resulting value of FanSpeedStatus |
|---|---|---|
| 0% | Off (“hard” off) | 0% |
| 1…25% | Off (“soft” off) | Same mapping as FanSpeedTarget |
| 26…50% | Low | ditto |
| 51…75% | Medium | ditto |
| 76…100% | High | ditto |

*NOTE: To facilitate certification, UPnP vendors should include their own version of the mapping table illustrated above.*

Whenever the value of DirectionTarget changes, the actual physical fan direction should start to change toward the value of DirectionTarget. Due to the physical inertia of the fan, this process will take a period of time that depends on the vendor’s implementation. The corresponding value of the DirectionStatus state variable should in turn reflect the actual physical fan direction.

**NOTES:**

i) If the actual physical fan speed or direction deviates from what is expected in FanSpeedTarget or DirectionTarget, then the corresponding xxxStatus state variable should reflect the real physical fan status and NOT the xxxTarget values.

ii) Vendors that implement control point strategies should bear in mind that due to friction, inertia, hysteresis and numerical rounding it is quite possible that the xxxStatus variables will take an indeterminate time to reach the value of the corresponding xxxTarget variables. Indeed (especially in the case of the fan speed), it is quite likely that the xxxStatus variable might *never* achieve exactly the same value as the xxxTarget variable.

*Relationships between standard state variable(s) defined herein and any non-standard state variable(s) is TBD.*

## 2.3. Eventing and Moderation

**Table 4: Event Moderation**

| Variable Name | Evented | Moderated Event | Max Event Rate[1] | Logical Combination | Min Delta per Event[2] |
|---|---|---|---|---|---|
| FanSpeedTarget | no | | | | |
| FanSpeedStatus | yes | yes | 30 | OR | 10 * (Step) |
| DirectionTarget | no | | | | |
| DirectionStatus | yes | no | | | |
| *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).

## 2.4. Actions

**Table 5: Actions**

| Name | Req. or Opt. [1] |
|---|---|
| SetFanSpeed | R |
| GetFanSpeed | R |
| GetFanSpeedTarget | R |
| SetFanDirection | O |
| GetFanDirection | O |
| GetFanDirectionTarget | O |
| *Non-standard actions implemented by an UPnP vendor go here.* | X |

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

### 2.4.1. SetFanSpeed

Sets the new value of FanSpeedTarget.

#### *2.4.1.1. Arguments*

**Table 6: Arguments for SetFanSpeed**

| Argument | Direction | relatedStateVariable |
|---|---|---|
| NewFanSpeedTarget | IN | FanSpeedTarget |

#### *2.4.1.2. Dependency on State*

None.

#### *2.4.1.3. Effect on State*

Sets the new value of FanSpeedTarget. The actual physical fan speed, (and thus the value of FanSpeedStatus), should map to FanSpeedTarget according to section 2.2.

#### *2.4.1.4. 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. GetFanSpeed

Returns the current value of FanSpeedStatus.

#### *2.4.2.1. Arguments*

**Table 7: Arguments for GetFanSpeed**

| Argument | Direction | relatedStateVariable |
|---|---|---|
| CurrentFanSpeedStatus | OUT [R] | FanSpeedStatus |

[R] = Return Value (RETVAL)

#### *2.4.2.2. Dependency on State*

Returns the current value of FanSpeedStatus.

#### *2.4.2.3. Effect on State*

None.

#### *2.4.2.4. 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. GetFanSpeedTarget

Returns the current value of FanSpeedTarget.

#### *2.4.3.1. Arguments*

**Table 8: Arguments for GetFanSpeedTarget**

| Argument | Direction | relatedStateVariable |
|---|---|---|
| CurrentFanSpeedTarget | OUT [R] | FanSpeedTarget |

[R] = Return Value (RETVAL)

#### *2.4.3.2. Dependency on State*

Returns the current value of FanSpeedTarget.

#### *2.4.3.3. Effect on State*

None.

#### *2.4.3.4. 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. SetFanDirection

Sets the new value of DirectionTarget.

### *2.4.4.1. Arguments*

**Table 9: Arguments for SetDirection**

| Argument | Direction | RelatedStateVariable |
|---|---|---|
| NewDirectionTarget | IN | DirectionTarget |

### *2.4.4.2. Dependency on State*

None.

### *2.4.4.3. Effect on State*

Sets the new value of DirectionTarget. The actual physical fan direction, (and thus the value of DirectionStatus), should follow DirectionTarget.

### *2.4.4.4. 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. GetFanDirection

Returns the current value of DirectionStatus.

### *2.4.5.1. Arguments*

**Table 10: Arguments for GetDirection**

| Argument | Direction | RelatedStateVariable |
|---|---|---|
| CurrentDirectionStatus | OUT [R] | DirectionStatus |

[R] = Return Value (RETVAL)

### *2.4.5.2. Dependency on State*

Returns the current value of DirectionStatus.

### *2.4.5.3. Effect on State*

None.

### *2.4.5.4. 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. GetFanDirectionTarget

Returns the current value of DirectionTarget.

#### *2.4.6.1. Arguments*

**Table 11: Arguments for GetDirectionTarget**

| Argument | Direction | RelatedStateVariable |
|---|---|---|
| CurrentDirectionTarget | OUT [R] | DirectionTarget |

[R] = Return Value (RETVAL)

#### *2.4.6.2. Dependency on State*

Returns the current value of DirectionTarget.

#### *2.4.6.3. Effect on State*

None.

#### *2.4.6.4. 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 an UPnP Vendor

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

### 2.4.8. Relationships Between Actions

The actions defined herein may be called in any order.

*Relationships between standard action(s) defined herein and any non-standard action(s) is TBD.*

### 2.4.9. Common Action 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.

| 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. |
| 600-699 | TBD | Common action errors. Defined by UPnP Forum Technical Committee. |
| *800-899* | *TBD* | *(Specified by UPnP vendor.)* |

## 2.5. Theory of Operation

Control Points will use SetFanSpeed to set the value of FanSpeedTarget; this in turn determines the running speed of the fan. Depending on the type of fan, it must adjust its actual physical speed to a value matching as closely as possible to the FanSpeedTarget – some examples of possible mappings are given in section 2.2. Due to the physical inertia of the fan, the physical fan speed and hence the value of FanSpeedStatus will take a period of time to "catch up" with FanSpeedTarget.

Control Points may interrogate the actual fan speed by calling GetFanSpeed. This function reads the value of FanSpeedStatus. In normal operation conditions, in the steady state, FanSpeedStatus will return +/- the same value as FanSpeedTarget. However, in the case of faults, or external overrides, the actual fan speed may differ from that requested by FanSpeedTarget. In such cases, FanSpeedStatus must return the actual physical speed in accordance with the mapping examples in section 2.2.

Similarly, Control Points will use SetFanDirection to set the value of DirectionTarget; this in turn determines the running direction of the fan. Depending on the type of fan, it must adjust its actual physical direction to the DirectionTarget. Due to the physical inertia of the fan, the physical fan speed and hence the value of DirectionStatus will take a period of time to "catch up" with DirectionTarget.

Control Points may interrogate the actual fan direction by calling GetFanDirection. This function reads the value of DirectionStatus. In normal operation conditions, in the steady state, DirectionStatus will return the same value as DirectionTarget. However, in the case of faults, or external overrides, the actual fan direction may differ from that requested by DirectionTarget. In such cases, DirectionStatus must return the actual physical fan direction.

**NOTE:** It is possible that a Control Point could issue a series of SetFanSpeed or SetFanDirection commands in rapid succession. The vendor is responsible for ensuring that in all cases, the fan responds safely, smoothly and without damage to itself. E.g. if a fan is running at (say) 100% speed "forward", and a control point switches the value of DirectionTarget to 1 "reverse", then it is the responsibility of the vendor to ensure that the fan transitions gradually from 100% "forward" to 0% "stopped" to 100% "reverse".

# 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>SetFanSpeed</name>
      <argumentList>
        <argument>
          <name>NewFanSpeedTarget</name>
          <direction>in</direction>
          <relatedStateVariable>FanSpeedTarget</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
    <action>
    <name>GetFanSpeed</name>
      <argumentList>
        <argument>
          <name>CurrentFanSpeedStatus</name>
          <direction>out</direction>
          <retval />
          <relatedStateVariable>FanSpeedStatus</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
    <action>
    <name>GetFanSpeedTarget</name>
      <argumentList>
        <argument>
          <name>CurrentFanSpeedTarget</name>
          <direction>out</direction>
          <retval />
          <relatedStateVariable>FanSpeedTarget</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
    <action>
    <name>SetFanDirection</name>
      <argumentList>
        <argument>
          <name>NewDirectionTarget</name>
          <direction>in</direction>
          <relatedStateVariable>DirectionTarget</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
    <action>
    <name>GetFanDirection</name>
      <argumentList>
        <argument>
          <name>CurrentDirectionStatus</name>
          <direction>out</direction>
          <retval />
```

```
          <relatedStateVariable>DirectionStatus</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
    <action>
    <name>GetFanDirectionTarget</name>
      <argumentList>
        <argument>
          <name>CurrentDirectionTarget</name>
          <direction>out</direction>
          <retval />
          <relatedStateVariable>DirectionTarget</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
    Declarations for other actions added by UPnP vendor (if any) go
here
  </actionList>
  <serviceStateTable>
    <stateVariable sendEvents="no">
      <name>FanSpeedTarget</name>
      <dataType>ui1</dataType>
      <defaultValue>0</defaultValue>
      <allowedValueRange>
        <minimum>0</minimum>
        <maximum>100</maximum>
        <step>1</step>
      </allowedValueRange>
    </stateVariable>
    <stateVariable sendEvents="yes">
      <name>FanSpeedStatus</name>
      <dataType>ui1</dataType>
      <defaultValue>0</defaultValue>
      <allowedValueRange>
        <minimum>0</minimum>
        <maximum>100</maximum>
        <step>1</step>
      </allowedValueRange>
    </stateVariable>
    <stateVariable sendEvents="no">
      <name>DirectionTarget</name>
      <dataType>boolean</dataType>
      <defaultValue>0</defaultValue>
    </stateVariable>
    <stateVariable sendEvents="yes">
      <name>DirectionStatus</name>
      <dataType>boolean</dataType>
      <defaultValue>0</defaultValue>
    </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: ***FanSpeed1.xml***
- The UPnP Test Tool service template test file: ***FanSpeed1.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.