FriendlyInfoUpdate:1 Service

For UPnP Version 1.0 and higher

Status: Standardized (DCP)

Date: September 5, 2014

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)(v) 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.

THE UPNP FORUM TAKES NO POSITION AS TO WHETHER ANY INTELLECTUAL PROPERTY RIGHTS EXIST IN THE STANDARDIZED DCPS. THE STANDARDIZED DCPS ARE PROVIDED "AS IS" AND "WITH ALL FAULTS". THE UPNP™ FORUM MAKES NO WARRANTIES, EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE STANDARDIZED DCPS, INCLUDING BUT NOT LIMITED TO ALL IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE, OF REASONABLE CARE OR WORKMANLIKE EFFORT, OR RESULTS OR OF LACK OF NEGLIGENCE.

© 2014, UPnP Forum. All rights reserved.
Contents

Contents ........................................................................................................................................2
List of Tables ................................................................................................................................3
List of Figures .................................................................................................................................4
1 Scope ..........................................................................................................................................4
2 Introduction ...................................................................................................................................4
3 Normative References ................................................................................................................4
4 Terms, definitions, symbols and abbreviated terms .................................................................5
  4.1 Provisioning terms ................................................................................................................5
    4.1.1 conditionally allowed ......................................................................................................5
    4.1.2 conditionally required .....................................................................................................5
    4.1.3 not allowed .....................................................................................................................5
  4.2 Terms .......................................................................................................................................6
    4.2.1 Clean ..............................................................................................................................6
    4.2.2 Pending ............................................................................................................................6
    4.2.3 Friendly ..........................................................................................................................6
    4.2.4 Non-Restrictable .............................................................................................................6
    4.2.5 Restrictable ....................................................................................................................6
  4.3 Abbreviated terms ..................................................................................................................6
### 4.3.1 6
### 4.3.2 6

5 Notations and Conventions

- **5.1.1 Data Types**
- **5.2 Management of XML Namespaces in Standardized DCPs**
- **5.3 Vendor-defined Extensions**

6 Service Modeling Definitions (Normative)

- **6.1 Service Type**
- **6.2 Security Feature**
- **6.2.1 Restrictable and Non-Restricetable actions**
- **6.3 FriendlyInfoUpdate Service Architecture**

6.4 State Variables

- **6.4.1 State Variable Overview**
- **6.4.2 FriendlyNameStatus**
- **6.4.3 FriendlyIconListStatus**
- **6.4.4 A_ARG_TYPE_NewName**
- **6.4.5 A_ARG_TYPE_IconURI**
- **6.4.6 A_ARG_TYPE_UpdateType**
- **6.4.7 A_ARG_TYPE_Token**
- **6.4.8 A_ARG_TYPE_RestoreType**

6.5 Eventing and Moderation

6.6 Actions

- **6.6.1 GetFriendlyName()**
- **6.6.2 GetFriendlyIconList()**
- **6.6.3 SetFriendlyName()**
- **6.6.4 SetFriendlyIconList()**
- **6.6.5 RestoreFriendlyInfo()**

7 Theory of Operation (Informative)

- **7.1 Changing the device <friendlyName>**
- **7.2 Changing the device <iconList>**

8 XML Service Description

---

**List of Tables**

- **Table 6-1:** Assignment of Restrictable/Non-Restricetable Roles .................................................. 8
- **Table 6-2:** State Variables ................................................. 9
- **Table 6-3:** AllowedValueList for **A_ARG_TYPE_UpdateType** .................................................. 14
- **Table 6-4:** AllowedValueList for **A_ARG_TYPE_RestoreType** .................................................. 14
- **Table 6-5:** Eventing and Moderation .................................................. 14
- **Table 6-6:** Actions .................................................. 15
- **Table 6-7:** Arguments for **GetFriendlyName()** .................................................. 15

---

Copyright © 2014, UPnP Forum. All rights Reserved.
1 Scope

This service definition is compliant with the UPnP Device Architecture version 1.0. It defines a service type referred to herein as FriendlyInfoUpdate service. It is scoped to the Device Description Document (DDD) and is a service allowing control points to create orderly updates to the <friendlyName> and <iconList> elements. Once a change has taken place the status of the DDD might not reflect that of the advertised description because the device can not leave the network during ongoing activities. Therefore a state variable is provided to indicate if the DDD contains non-advertised (pending) values. If for any reason the device goes off line, power cycles, or reboots it will most likely advertise its new DDD. If DeviceProtection UPnP DP is implemented on the device then it will support restricting control point actions to control points with specific Roles.

2 Introduction

The FriendlyInfoUpdate service is a UPnP service that enables control points to update specific, friendly, elements of a UPnP device’s DDD. The DDD elements updateable by the service are:

- <friendlyName>
- <iconList>

The service also describes how to reset the value to the original factory setting and in the case of <iconList>, a method for updating the actual device icon binaries using HTTP methods. It is assumed that the device, when supporting the updating of icon binaries, will have the means to manage image binaries, such as determining MIME image type, height, width and color depth, as well as, negating undesired behavior associated with binary transfers involving potential malware.

3 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.


4 Terms, definitions, symbols and abbreviated terms
For the purposes of this document, the terms and definitions given in ISO/IEC 29341-1 and the following apply.

4.1 Provisioning terms

4.1.1 conditionally allowed
CA
The definition or behavior depends on a condition. If the specified condition is met, then the definition or behavior is allowed, otherwise it is not allowed.

4.1.2 conditionally required
CR
The definition or behavior depends on a condition. If the specified condition is met, then the definition or behavior is required, otherwise it is not allowed.

4.1.3 not allowed
The definition or behavior is prohibited by this specification. Opposite of required.

Copyright © 2014, UPnP Forum. All rights Reserved.
4.2 Terms

4.2.1 Clean
A notional device state scoped to this specification in which all internal states related to the DDD are identical to its last advertised Device Description Document.

4.2.2 Pending
The pending state is a notional device state scoped to this specification in which one or more of the internal states related to the DDD are different from its last advertised Device Description Document.

4.2.3 Friendly
DDD element(s) which can be interpreted as human identifiable and not tied directly to the device manufacturer information; specifically the <friendlyName> and device icons, defined in the <iconList> element, are considered friendly.

4.2.4 Non-Restrictable
A category of action, that when the DeviceProtection UPnP DP service is implemented on the device, cannot be blocked according to the presence or absence of a specific Role attached to a Control Point Identity or User Identity. See UPnP CDS4 for further explanation of Role, Control Point Identity and User Identity.

4.2.5 Restrictable
A category of actions that, when the DeviceProtection UPnP DP service is implemented on the device, can be blocked according to the presence or absence of a specific Role attached to a Control Point Identity or User Identity.

4.3 Abbreviated terms

4.3.1 A allowed

4.3.2 DDD device description document

5 Notations and Conventions

- 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.

5.1.1 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 ISO/IEC 29341-1. The XML Schema namespace is used to define property data types XML SCHEMA-2.

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" also allowed to be used but are not recommended. The values "yes" and "no" are deprecated and not allowed to be sent out by devices but shall 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 not allowed to be sent out by devices but shall be accepted on input.

5.2 Management of XML Namespaces in Standardized DCPs
This specification shall follow the conventions of the equivalent section of UPnP CDS4 where applicable.

5.3 Vendor-defined Extensions
Whenever vendors create additional vendor-defined state variables, actions or properties, their assigned names and XML representation shall follow the naming conventions and XML rules as specified in ISO/IEC 29341-1, Section 2.5, "Description: Non-standard vendor extensions".

6 Service Modeling Definitions (Normative)

6.1 Service Type
The following service type identifies a service that is compliant with this specification:

    urn:schemas-upnp-org:service:FriendlyInfoUpdate:1

FriendlyInfoUpdate service is used herein to refer to this service type.

6.2 Security Feature
In the specification, if support for the Security Feature is referenced, this indicates that the device implements the DeviceProtection Service UPnP DP.

6.2.1 Restrictable and Non-Restrictable actions
The FriendlyInfoUpdate service actions defined in this specification have the Restrictable, Non-Restrictable assignments as indicated in Table 6-1.
Table 6-1: Assignment of Restrictable/Non-Restrictable Roles

<table>
<thead>
<tr>
<th>Action Name</th>
<th>Public</th>
<th>Basic</th>
<th>Admin</th>
<th>Vendor Defined</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetFriendlyName()</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>GetFriendlyIconList()</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>SetFriendlyName()</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>SetFriendlyIconList()</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>RestoreFriendlyInfo()</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>YES</td>
</tr>
</tbody>
</table>

A YES value in the table indicates that the action shall be Restrictable when the Security Feature is supported and a control point with only the Role indicated shall not have Action level access, (see UPnP CDS4) and shall receive an error code 606 (see UPnP Device Architecture ISO/IEC 29341-1) in response to the action invocation.

A NO value in the table indicates that the action shall be Non-Restrictable, meaning that even if the Security Feature is supported all control points shall have Action level access when control point invoking the action and shall not receive an error code 606 based on the Security Feature.
6.3 **FriendlyInfoUpdate** Service Architecture

This service is provided by UPnP devices to allow UPnP control points to get (read) and set (write) the value of the `<friendlyName>` and `<iconList>` elements of the UPnP Device Description Document (DDD). The `<friendlyName>` element is required and its value can be found in the UPnP Device Description Document made available in the description phase. The `<iconList>` element is only required if one or more `<icon>` elements are included in the DDD. In most cases, without this service, a UPnP control point is not able to change the device’s `<friendlyName>` or `<iconList>`, however if there are other methods available, this service also defines how interaction with the FriendlyInfoUpdate service state variables is handled.

This service enables control points to get the most current `<friendlyName>` and `<iconList>` and set them to more informative and intuitive versions such as “John’s living room TV” instead of the default “X company Media Renderer model 123”.

The service enforces some basic timing restrictions to prevent race conditions when multiple updates are requested in quick succession or multiple control points are interacting with the service.

6.4 State Variables

*Note: For first-time reader, it might be more insightful to read the theory of operations first and then the action definitions before reading the state variable definitions.*

6.4.1 State Variable Overview

Table 6-2:  
State Variables

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>R/A1</th>
<th>Data Type</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>FriendlyInfoStatus</td>
<td>R</td>
<td>string</td>
<td>See Section 6.4.2</td>
</tr>
<tr>
<td>FriendlyIconListStatus</td>
<td>CR</td>
<td>string</td>
<td>See Section 6.4.3</td>
</tr>
<tr>
<td>A_ARG_TYPE_NewName</td>
<td>R</td>
<td>string</td>
<td>See Section 6.4.4</td>
</tr>
<tr>
<td>A_ARG_TYPE_IconURI</td>
<td>CR</td>
<td>string</td>
<td>See Section 6.4.5</td>
</tr>
<tr>
<td>A_ARG_TYPE_UpdateType</td>
<td>CR</td>
<td>string</td>
<td>See Section 6.4.6</td>
</tr>
<tr>
<td>A_ARG_TYPE_Token</td>
<td>CR</td>
<td>string</td>
<td>See Section 6.4.7</td>
</tr>
<tr>
<td>A_ARG_TYPE_RestoreType</td>
<td>CR</td>
<td>string</td>
<td>See Section 6.4.8</td>
</tr>
</tbody>
</table>

1 R = Required, A = Allowed, CR = Conditionally Required, CA = Conditionally Allowed, X = Non-standard, add -D when deprecated (e.g., R-D, A-D).

6.4.2 **FriendlyInfoStatus**

This required state variable indicates the status of the DDD `<friendlyName>` element value relative to changes incurred via the FriendlyInfoUpdate service or any other method. If the current `<friendlyName>` element value is different from the last `<friendlyName>` element value advertised in the DDD then the `<friendlyName>` `@status` attribute shall have a value of “**DDD**”, otherwise it shall have a value of “**PENDING**”. If the DDD `<friendlyName>` element...
value is changed by any other method prior to it being re-advertised in the DDD then the new value shall also be reflected in the FriendlyNameStatus state variable with the appropriate @status value.

The FriendlyNameStatus state variable is a string that contains a FriendlyNameStatus XML Document. The following example shows a generalized “template” for the FriendlyNameStatus XML Document. The example shows the elements that need to be filled out by individual implementations in the vendor character style. The Schema associated with the FriendlyNameStatus state variable is can be found at XML-FNS-SCHEMA.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<FriendlyNameStatus xmlns="urn:schemas-upnp-org:fd:fns-events"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:schemas-upnp-org:fd:fns-events http://www.upnp.org/schemas/fd/fns-events.xsd">
  <friendlyName status="DDD|PENDING">Describes the current or pending value for the DDD friendlyName</friendlyName>
</FriendlyNameStatus>
```

Allowed. Case sensitive.

Required. <XML>. Shall include a namespace declaration for the FriendlyInfoUpdate service Common Datastructures Schema ("urn:schemas-upnp-org:fd:fns-events"). This is the base wrapper element for the state variable. Shall include one of the following elements:

- `<friendlyName>`
  - Required. xsd:string. Shall appear one time. The value of this element is the same as defined in the `<friendlyName>` element as defined in ISO/IEC 29341-1. The value of this element shall not be empty.
  - @status
    - Required. xsd:string. Shall be one of the following values: “DDD” or “PENDING”. “DDD” indicates that the `<friendlyName>` is the same as advertised in the DDD. “PENDING” indicates that the `<friendlyName>` value has changed and is awaiting re-advertisement of the DDD.

6.4.3 FriendlyIconListStatus

This conditionally required state variable indicates the status of the DDD `<iconList>` element relative to changes incurred via the FriendlyInfoUpdate service or any other method. If the device is capable of advertising an `<iconList>` element in its DDD then it shall support this state variable, otherwise it is not allowed. It shall provide the complete internal `<iconList>` relative to the advertised DDD `<iconList>` and is allowed to expose additional resources for creating new icons via an `<icon>` element `<postToken>` or `<getToken>` element. If the DDD `<iconList>` element value is changed by any other method prior to being re-advertised in the DDD then the new value shall also be reflected in the FriendlyIconListStatus state variable. Note that the FriendlyIconListStatus state variable shall always have an `<iconList>` element present and should have at least one `<icon>` element with `<mimetype>` element value “image/png” since this is recommended in ISO/IEC 29341-1.

The FriendlyIconListStatus state variable is a string that contains a FriendlyIconListStatus XML Document. The following example shows a generalized “template” for the FriendlyIconListStatus XML Document. The example shows the elements that need to be filled out by individual
implementations in the vendor character style. The Schema associated with the FriendlyIconListStatus state variable is can be found at XML-FIS-SCHEMA.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<FriendlyIconListStatus
 xmlns="urn:schemas-upnp-org:fd:fis-events"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:schemas-upnp-org:fd:fis-events"
 http://www.upnp.org/schemas/fd/fis-events.xsd">
   <iconList>
     <!-- Describes current, complete internal device <iconList>. 
      Includes current DDD icons, pending updates and deletions, 
      and resources to create new icons -->
     <icon status="" status="" status="" status="" status=""
          maxbyte="" maxheight="" maxwidth="" maxdepth=""
          url="" url="" url="" url="" url="" url=""
          "" mime="" mime="" mime="" mime="" mime=""
          height="" width="" depth=""
     >
  </icon>
</iconList>
</FriendlyIconListStatus>
```

<xml>
   Allowed. Case sensitive.

   <FriendlyIconListStatus>
      Required. <XML>. Shall include a namespace declaration for the FriendlyInfoUpdate service Common Datastructures Schema ("urn:schemas-upnp-org:fd:fis-events"). This is the base wrapper element for the state variable. Shall include one of the following elements:

   <iconList>
      Required. <XML>. Shall appear one time. The contents of this element shall contain one or more of the following elements:

   <icon>
      Required. <XML>. Shall appear one or more times. Describes an existing icon (either clean or pending) and its associated characteristics, a deleted (pending) icon, or a URI resource where an icon can be created.

      @status
      Required. xsd:string. Indicates the status of the icon. Shall be one of the following values: "DDD", "DELETED", "PENDING", "OPEN".

      "DDD" indicates that the <icon> is part of the advertised DDD and the icon binaries are the same.

Copyright © 2014, UPnP Forum. All rights Reserved.
“DELETED” indicates that the <icon> is currently part of the advertised DDD but the <icon> and its binary are pending deletion.

“OPEN” indicates that the <icon> is a place holder for the addition of a new <icon> with a new icon binary.

“PENDING” indicates that the <icon> has been created from a previously OPEN <icon> slot, its binary is available and it is pending advertisement in the DDD when it is updated.

@maxBytes
Conditionally allowed. xsd:unsignedInt. Shall appear zero or one time.
Indicates the maximum size (in bytes) for a new icon binary. The value should be on a per icon basis especially if the @maxHeight, @maxWidth or @maxDepth attributes are also present. If, on the other hand, it is a device level limitation (and not an individual icon limitation) then it should be uniformly updated across all icons with an @status value of “OPEN” when a binary is uploaded. Shall only be present if the parent <icon> has @status attribute value of “OPEN”, otherwise it is not allowed.

@maxHeight
Conditionally allowed. xsd:unsignedInt. Shall appear zero or one time.
Indicates the maximum vertical size (in pixels) for a new icon binary. Shall only be present if the parent <icon> has @status attribute value of “OPEN”, otherwise it is not allowed.

@maxWidth
Conditionally allowed. xsd:unsignedInt. Shall appear zero or one time.
Indicates the maximum horizontal size (in pixels) for a new icon binary. Shall only be present if the parent <icon> has @status attribute value of “OPEN”, otherwise it is not allowed.

@maxDepth
Conditionally allowed. xsd:unsignedInt. Shall appear zero or one time.
Indicates the maximum color depth (in bits) for a new icon binary. Shall only be present if the parent <icon> has @status attribute value of “OPEN”, otherwise it is not allowed.

<url>
Conditionally allowed. xsd:anyURI. Shall appear zero or one time. Indicates a URI where an icon binary of the MIME image type indicated in the sibling element <mimeType> exists. Shall be present if the icon is part of the current DDD @status of “DDD”), has been created and is waiting advertisement in the DDD @status of “PENDING”), or is present in the current DDD and pending deletion @status of “DELETED”). Shall not be present if the icon is awaiting creation @status of “OPEN”).

<postToken>
Conditionally allowed. xsd:string. Shall appear zero or one time. Provides a device unique token which can be passed to the device by the control point to authorize an HTTP-POST to the device to the URI provided by the postUri@iconToken. The posted binary shall be of the MIME image type indicated in the sibling element <mimeType>. Shall only be present if the device supports the HTTP-POST operation and the @status attribute has a value of “OPEN”.

@postUri
Required. xsd:anyURI. Indicates a resource that can be used to support an HTTP-POST RFC 2616 for a new (“OPEN”) icon binary. Note that full and relative URLs are allowed.

<getToken>
Conditionally Required. xsd:string. Shall appear one time. Provides a device unique token which can be passed to the device by the control point to trigger
an HTTP-GET by the device for the parent <icon>. The acquired icon binary shall be of the MIME image type indicated in the sibling element <mimetype>. Shall be present if the parent <icon> has @status attribute value of "OPEN", otherwise it is not allowed.

<mimetype>
Required. xsd:string. Shall appear one time. Indicates the image MIME image type of the icon binary that shall be associated with the parent <icon> element. If the <icon> element has an @status value of "OPEN" then the value indicates the MIME image type of the image binary that can be uploaded, otherwise the value indicates the MIME image type of an existing icon binary (in the clean or pending state). If multiple MIME image type binaries are supported by the device then the device shall use a different <icon> with the appropriate <mimetype> element for each MIME image type supported.

<width>
Conditionally required. xsd:int. Shall appear zero or one time. Indicates the horizontal dimension (in pixels) of the icon binary associated with the parent <icon> element. Shall be present if the icon @status attribute is "DDD" or "PENDING". Shall not be present if the icon @status attribute is "DELETED" or "OPEN".

<height>
Conditionally required. xsd:int. Shall appear zero or one time. Indicates the vertical dimension (in pixels) of the icon binary associated with the parent <icon> element. Shall be present if the icon @status attribute is "DDD" or "PENDING". Shall not be present if the icon @status attribute is "DELETED" or "OPEN".

<depth>
Conditionally required. xsd:int. Shall appear zero or one time. Indicates the color bits in the icon binary associated with the parent <icon> element. Shall be present if the icon @status attribute is "DDD" or "PENDING". Shall not be present if the icon @status attribute is "DELETED" or "OPEN".

Note that the availability of the <postToken> is dependent on the device supporting the HTTP-POST method. The device shall support the <postToken> resource uniformly on all creatable icons in the FriendlyIconListStatus state variable with the @status attribute of "OPEN", that is, it shall have a <postToken> on all "OPEN" icons, not a mix. Note that this is addressed for <getToken> by its conditionally required status.

6.4.4 A_ARG_TYPE_NewName
This required state variable provides type information for the NewName argument in the SetFriendlyName() action. The data type is string.

6.4.5 A_ARG_TYPE_IconURI
This conditionally required state variable shall be present if the SetFriendlyIconList() action is supported otherwise it is not allowed. It provides type information for the IconURI argument in the SetFriendlyIconList() action. The data type is string.

6.4.6 A_ARG_TYPE_UpdateType
This conditionally required state variable shall be present if the SetFriendlyIconList() action is supported otherwise it is not allowed. It provides type information for the Method argument in the SetFriendlyIconList() action. The data type is string.
### 6.4.7 A_ARG_TYPE_Token

This conditionally required state variable shall be present if the `SetFriendlyIconList()` action is supported otherwise it is not allowed. It provides type information for the `TargetIconToken` input argument in the `SetFriendlyIconList()` action. The data type is `string`.

### 6.4.8 A_ARG_TYPE_RestoreType

This conditionally required state variable shall be present if the `RestoreFriendlyInfo()` action is supported, otherwise it is not allowed. It provides type information for the `RestoreType` argument in the `RestoreFriendlyInfo()` action. The data type is `string`.

<table>
<thead>
<tr>
<th>Value</th>
<th>R/A</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>CR</td>
<td>Indicates that the control point is requesting that all of the device friendly DDD information be restored to its factory settings. Shall be supported if the <code>SetFriendlyIconList</code> action is supported, otherwise it is not allowed.</td>
</tr>
<tr>
<td>FRIENDLYNAME</td>
<td>R</td>
<td>Indicates that the control point is requesting that a device <code>&lt;friendlyName&gt;</code> be restored to its factory setting.</td>
</tr>
<tr>
<td>ICONLIST</td>
<td>CR</td>
<td>Indicates that the control point is requesting that a device <code>&lt;iconList&gt;</code> be restored to its factory setting. Shall be supported if the <code>SetFriendlyIconList</code> action is supported, otherwise it is not allowed.</td>
</tr>
</tbody>
</table>

### 6.5 Eventing and Moderation

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Evented</th>
<th>Moderated</th>
<th>Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>FriendlyNameStatus</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>FriendlyIconListStatus</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>A_ARG_TYPE_NewName</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>
6.6 Actions

Table 6-6: Actions

<table>
<thead>
<tr>
<th>Name</th>
<th>Device R/A&lt;sup&gt;1&lt;/sup&gt;</th>
<th>Control Point R/A&lt;sup&gt;2&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetFriendlyName()</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>GetFriendlyIconList()</td>
<td>CR</td>
<td>A</td>
</tr>
<tr>
<td>SetFriendlyName()</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>SetFriendlyIconList()</td>
<td>CR</td>
<td>A</td>
</tr>
<tr>
<td>RestoreFriendlyInfo()</td>
<td>A</td>
<td>A</td>
</tr>
</tbody>
</table>

<sup>1</sup> For a device this column indicates whether the action shall be implemented or not, where R = Required, A = Allowed, CR = Conditionally Required, CA = Conditionally Allowed, X = Non-standard, add -D when deprecated (e.g., R-D, O-D).

<sup>2</sup> For a control point this column indicates whether a control point shall be capable of invoking this action, where R = Required, A = Allowed, CR = Conditionally Required, CA = Conditionally Allowed, X = Non-standard, add -D when deprecated (e.g., R-D, O-D).

6.6.1 GetFriendlyName()

This required action is used to get the internal value of the `<friendlyName>` element (pending or clean) of the FriendlyNameStatus state variable. This status is returned in the NameStatus output argument which contains a properly escaped FriendlyNameStatus XML Document.

6.6.1.1 Arguments

Table 6-7: Arguments for GetFriendlyName()

<table>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>relatedStateVariable</th>
</tr>
</thead>
<tbody>
<tr>
<td>NameStatus</td>
<td>OUT</td>
<td>FriendlyNameStatus</td>
</tr>
</tbody>
</table>

6.6.1.2 Device Requirements

None.
6.6.1.3 Dependency on State (if any)
Return value is dependent on value of the FriendlyNameStatus state variable. If the Security Feature is supported the action is defined as a Non-Restrictable action and the targeted device is not allowed to restrict its invocation by any control point.

6.6.1.4 Effect on State (if any)
None.

6.6.1.5 Errors
Table 6-8: Error Codes for GetFriendlyName()

<table>
<thead>
<tr>
<th>ErrorCode</th>
<th>errorDescription</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400-499</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>500-599</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>600-699</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>700</td>
<td>Reserved for future extensions.</td>
<td></td>
</tr>
</tbody>
</table>

Note: 800-899 Error Codes are not permitted for standard actions. See UPnP Device Architecture section on Control for more details.

6.6.2 GetFriendlyIconList()
This conditionally required action is used to get any current clean or pending values of the <iconList> element in the DDD via the FriendlyIconListStatus state variable. This status is returned in the IconListStatus output argument which contains the a properly escaped FriendlyIconListStatus XML Document. If the device is capable of advertising an <iconList> element in its DDD then it shall support this action otherwise it is not allowed.

6.6.2.1 Arguments
Table 6-9: Arguments for GetFriendlyIconList()

<table>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>relatedStateVariable</th>
</tr>
</thead>
<tbody>
<tr>
<td>IconListStatus</td>
<td>OUT</td>
<td>FriendlyIconListStatus</td>
</tr>
</tbody>
</table>

6.6.2.2 Device Requirements
None.

6.6.2.3 Dependency on State (if any)
Return value is dependent on value of the FriendlyIconListStatus state variable. If the Security Feature is supported the action is defined as a Non-Restrictable action and the targeted device is not allowed to restrict its invocation by any control point.

6.6.2.4 Effect on State (if any)
None.
6.6.2.5 Errors

Table 6-10: Error Codes for GetFriendlyIconList()

<table>
<thead>
<tr>
<th>ErrorCode</th>
<th>errorDescription</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400-499</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>500-599</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>600-699</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>700</td>
<td>Reserved for future extensions.</td>
<td></td>
</tr>
</tbody>
</table>

Note: 800-899 Error Codes are not permitted for standard actions. See UPnP Device Architecture section on Control for more details.

6.6.3 SetFriendlyName()

This required action is used to set a new value to the <friendlyName> element in the DDD. The NewName input argument identifies the requested new <friendlyName> value. If the NewName input argument value is empty (blank) the the action shall return errorCode 702.

6.6.3.1 Arguments

Table 6-11: Arguments for SetFriendlyName()

<table>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>relatedStateVariable</th>
</tr>
</thead>
<tbody>
<tr>
<td>NewName</td>
<td>IN</td>
<td>A_ARG_TYPE_NewName</td>
</tr>
</tbody>
</table>

6.6.3.2 Device Requirements

If the device supports the Security Feature then this action is defined as a Restrictable action and the targeted device is allowed to restrict its invocation to specific control points which are identifiable or have specific Roles.

6.6.3.3 Dependency on State (if any)

None.

6.6.3.4 Effect on State (if any)

Successful invocation of this action will change the value of the FriendlyNameStatus state variable and cause it to be evented. Unsuccessful invocation of this action, that is the return of any errorCode except 709 shall also cause the FriendlyNameStatus state variable to be evented, however it will be unchanged. This is done to indicate to other control points that the 30 second (or less) lock-out window has expired. If the device is in a state that allows it to leave the network and re-advertise its new friendly values then it should do so, however, it shall wait at least 30 seconds after any invocation of the GetFriendlyName(), GetFriendlyIconList(), SetFriendlyName(), SetFriendlyIconList(), or RestoreFriendlyInfo() actions before initiating the re-advertisement behavior; this is done to provide opportunities for control points to make more than one change to the the DDD without a re-advertisement. The timing for leaving and re-advertising is implementation dependent but shall be conformant to ISO/IEC 29341-1. Once the device re-advertises the value of its DDD <friendlyName> element it should be the same as the value of the FriendlyNameStatus state variable <friendlyName> element immediately prior to the device
leaving the network and the FriendlyNameStatus state variable <friendlyName> element should be in the clean state, that is, its @status value should be "DDD".

6.6.3.5 Errors

Table 6-12: Error Codes for SetFriendlyName()

<table>
<thead>
<tr>
<th>ErrorCode</th>
<th>errorDescription</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400-499</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>500-599</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>600-699</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>700</td>
<td></td>
<td>Reserved for future extensions.</td>
</tr>
<tr>
<td>701</td>
<td>Name too long</td>
<td>The provided friendlyName value is too long.</td>
</tr>
<tr>
<td>702</td>
<td>Empty name not allowed</td>
<td>The friendlyName value shall not have a non-empty (blank) name.</td>
</tr>
</tbody>
</table>

Note: 800-899 Error Codes are not permitted for standard actions. See UPnP Device Architecture section on Control for more details.

6.6.4 SetFriendlyIconList()

This conditionally required action is used to create a new icon or delete an existing clean or pending icon for use in a friendly element <iconList> of the DDD. If the device supports the FriendlyIconListStatus state variable then this action is required otherwise it is not allowed. It acts on information provided in the <iconList> element of FriendlyIconListStatus state variable, which reflects clean, pending, and potentially new icon information.

The UpdateType input argument indicates the direction of the action, that is, whether the actions outcome is to create a new icon or delete an existing icon. The TargetIconToken input argument ties together a specific <icon> element and associated <url> element with a specific HTTP (GET or POST) operation.

In the case of an "CREATE" action, the TargetIconURI input argument indicates where the primary binary resource of the HTTP-GET operation is located. It is allowed be internal or external to the device; for example the device is also a media server and the control point wants to use an exposed image item as an icon.

In the case of a "DELETE" action, the TargetIconURI input argument indicates the internal <url> of the icon binary resource to be deleted.

The behavior of the SetFriendlyIconList() action is further described in Table 6-13 for valid input combinations of the UpdateType, TargetIconToken, and TargetIconURI input arguments. These are the only valid combinations of input arguments that result in a successful icon update.
### Table 6-13: Specific behavior for `SetFriendlyIconList()` upon valid input

<table>
<thead>
<tr>
<th>UpdateType</th>
<th>TargetIconToken</th>
<th>TargetIconURI</th>
<th>Supported Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;CREATE&quot;</td>
<td>Matches an existing <code>FriendlyIconListStatus</code> state variable <code>&lt;icon&gt;</code> element <code>&lt;postToken&gt;</code> element with associated <code>@status</code> value of &quot;OPEN&quot;</td>
<td>Empty</td>
<td>The device shall immediately allow an HTTP-POST to the <code>postUri</code>@<code>postToken</code> associated with the <code>TargetIconURI</code> input for 30 seconds.</td>
</tr>
<tr>
<td></td>
<td>Matches an existing <code>FriendlyIconListStatus</code> state variable <code>&lt;icon&gt;</code> element <code>&lt;getToken&gt;</code> element with associated <code>@status</code> value of &quot;OPEN&quot;</td>
<td>Shall be a valid URI</td>
<td>The device shall attempt and complete an HTTP-GET of <code>TargetIconURI</code> within 30 seconds.</td>
</tr>
<tr>
<td>&quot;DELETE&quot;</td>
<td>Empty</td>
<td>Matches an existing <code>FriendlyIconListStatus</code> state variable <code>&lt;url&gt;</code> element with associated <code>@status</code> value of &quot;PENDING&quot; or &quot;DDD&quot;.</td>
<td>The device should mark for deletion the image binary identified by the <code>TargetIconURI</code> input argument within 30 seconds, unless the device implementation needs to persist the targeted icon.</td>
</tr>
</tbody>
</table>

Upon invocation of the `SetFriendlyIconList()` action, the device shall determine if the input arguments are a valid combination or return an errorCode 703.

Upon invocation of the `SetFriendlyIconList()` action, this action shall complete within 30 seconds, including the requested HTTP-GET or HTTP-POST, or return an errorCode 704, unless one of the other errorCodes are returned prior to the 30 second window.

The device shall prohibit HTTP-POST to any `<postUri>@postToken>` resource advertised in the `FriendlyIconListStatus` state variable unless a `SetFriendlyIconList()` action has been invoked on a specific `<postUri>@postToken>`, then it shall be allowed on that resource for no more than 30 seconds.

The device shall not issue an HTTP-GET for any `<icon>` resources advertised in the `FriendlyIconListStatus` state variable unless a `SetFriendlyIconList()` action has been invoked on the specific `<icon>`, then it shall do so on that resource within 30 seconds.

If the action requests an HTTP-POST or HTTP-GET for any `<icon>` and the MIME image type of the image binary does not match the `<mimetype>` element then the action shall return errorCode 705.

The device shall not expose values associated with the `FriendlyIconListStatus` state variable `<url>`, `<mimetype>`, `<height>`, `<width>`, or `<depth>` elements inconsistent with ISO/IEC 29341-1 (see section 6.6.4.4 for additional detail).

---

1 Upon successful completion, the associated `FriendlyIconListStatus` state variable `<icon>` element shall be updated according to the new icon binaries and the `FriendlyIconListStatus` state variable evented.

2 In some cases the device implementation may want to retain certain icons, at least until a suitable replacement icon has been created, this behavior is allowed.
The device is allowed to fail the action and return errorCode 706 through 709 if no other errorCodes are required as previously indicated.

The device is allowed to add or remove <icon> elements with the @status value of "OPEN" upon any eventing of the FriendlyIconListStatus state variable, that is, the device is allowed to add or remove resources dynamically for creating new icons dependent on its specific capabilities.

6.6.4.1 Arguments

Table 6-14: Arguments for SetFriendlyIconList()

<table>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>relatedStateVariable</th>
</tr>
</thead>
<tbody>
<tr>
<td>UpdateType</td>
<td>IN</td>
<td>A_ARG_TYPE_UpdateType</td>
</tr>
<tr>
<td>TargetIconToken</td>
<td>IN</td>
<td>A_ARG_TYPE_Token</td>
</tr>
<tr>
<td>TargetIconURI</td>
<td>IN</td>
<td>A_ARG_TYPE_IconURI</td>
</tr>
</tbody>
</table>

6.6.4.2 Device Requirements

If the device supports the Security Feature then this action is defined as a Restrictable action and the targeted device is allowed to restrict its invocation to specific control points which are identifiable or have specific Roles.

6.6.4.3 Dependency on State (if any)

If the targeted friendly element has been changed then the execution is dependent on the current FriendlyIconListStatus state variable as described above. Prior to re-advertisement of the DDD, transitions of <icon> @status from the "DDD", "DELETED", "PENDING", and "OPEN" states are restricted as follows:

- An <icon> with @status of "DDD" shall only transition to an @status value of "DELETED". Restoration of the original binary can be achieved in two ways: 1) reloading the original binary in a "OPEN" icon slot, 2) a successful invocation of the RestoreFriendlyInfo() action.
- An <icon> with @status of "OPEN" shall only transition to an @status value of "PENDING".
- An <icon> with @status of "PENDING" shall only transition to an @status value of "OPEN" if the icon binary is deleted.

6.6.4.4 Effect on State (if any)

If the invocation of the SetFriendlyIconList() action creates a binary for the icon associated with the TargetIconToken input argument having an @status value of "OPEN" then the device shall:

1) Update the corresponding <url>, <width>, <height>, <depth>, and <mimetype> values of the FriendlyIconListStatus state variable to correctly indicate the new image characteristics.
2) Change the <icon> @status value to "PENDING".

If the invocation of the SetFriendlyIconList() action deletes the binary of the icon indicated by the TargetIconURI input argument having an @status value of "DDD" or "PENDING" then the device shall update the FriendlyIconListStatus state variable as follows:

1) Remove the associated <width>, <height>, and <depth> elements to correctly indicate the impending lack of an icon binary.
2) Change the associated @status attribute value to "DELETED" if the icon is currently part of the advertised DDD or "OPEN" if it had a previous @status value of "PENDING".
3) Remove the `<url>` element, if the icon transitions to an @status value of "OPEN", and add the `<getToken>`, `<postToken>` or both. The `<postToken>` element shall be added in a uniform manner as previously described.

4) Add the `<maxBytes>`, `<maxHeight>`, `<maxWidth>`, and `<maxDepth>` elements if supported.

The device shall event the FriendlyIconListStatus state variable upon any change. Unsuccessful invocation of this action, that is the return of any errorCode except 709, shall also cause the FriendlyIconListStatus state variable to be evented, however it will be unchanged. This is done to indicate to other control points that the 30 second (or less) lock-out window has expired.

The allowed state transitions are illustrated in Figure 1 along with the FriendlyInfoUpdate service actions (SetFriendlyName() and RestoreFriendlyInfo()) or a DDD re-advertisement that can trigger the state change.

![Diagram of state transitions](image)

If the device is in a state that allows it to leave the network and re-advertise its new friendly values then it should do so however it shall wait at least 30 seconds after any invocation of any GetFriendlyName(), GetFriendlyIconList(), SetFriendlyName(), SetFriendlyIconList() or RestoreFriendlyInfo() actions before initiating the re-advertisement behavior; this is done to provide opportunities for control points to make more than one change to the DDD without a re-advertisement. The timing for leaving and re-advertising is implementation dependent but shall be conformant to ISO/IEC 29341-1.

### 6.6.4.5 Errors

<table>
<thead>
<tr>
<th>ErrorCode</th>
<th>errorDescription</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400-499</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>500-599</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>600-699</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
</tbody>
</table>
## 6.6.5  **RestoreFriendlyInfo()**

This allowed action is used to restore the `<friendlyName>` and, if supported, the `<iconList>` element in the DDD to its factory default settings. The `RestoreType` input argument identifies if the request is to restore the `<friendlyName>` value or the `<iconList>` values or both.

If the `RestoreType` input parameter has a value of "ALL" or "FRIENDLYNAME" then the device shall reset the value of the "FriendlyNameStatus" state variable `<friendlyName>` element value to the factory setting originally advertised in the device DDD.

If the `RestoreType` input parameter has a value of "ALL" or "ICONLIST" and the device supports the SetFriendlyIconList() action then the device shall reset its FriendlyIconListStatus `<iconList>` elements to values indicating the factory setting along with any "OPEN" icon elements that the device allocates resources.

### 6.6.5.1 Arguments

#### Table 6-16: Arguments for **RestoreFriendlyInfo()**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>relatedStateVariable</th>
</tr>
</thead>
<tbody>
<tr>
<td>RestoreType</td>
<td>IN</td>
<td>A_ARG_TYPE_RestoreType</td>
</tr>
</tbody>
</table>

### 6.6.5.2 Device Requirements

If the device supports the Security Feature then this action is defined as a Restrictable action and the targeted device is allowed to restrict its invocation to specific control points which are identifiable or have specific Roles.

### 6.6.5.3 Dependency on State (if any)

None.

---

Copyright © 2014, UPnP Forum. All rights Reserved.
6.6.5.4 Effect on State (if any)

Successful invocation of this action will change the value of the FriendlyNameStatus state variable and cause it to be evented. If the device is in a state that allows it to leave the network and re-advertise its new friendly values then it should do so, however, it shall wait at least 30 seconds after any invocation of any GetFriendlyName(), GetFriendlyIconList(), SetFriendlyName(), SetFriendlyIconList() or RestoreFriendlyInfo() actions before initiating the re-advertisement behavior; this is done to provide opportunities for control points to make more than one change to the DDD without a re-advertisement. The timing for leaving and re-advertising is implementation dependent but shall be conformance to ISO/IEC 29341-1.

Once the device re-advertises the value of its DDD <friendlyName> element it should be the same as the value of the FriendlyNameStatus state variable <friendlyName> element immediately prior to the device leaving the network. Once re-advertised the FriendlyNameStatus state variable <friendlyName> element should be in the clean state.

Once the device re-advertises the value of its DDD <iconList> elements they should reflect the same values as all FriendlyIconListStatus state variable <icon> elements immediately prior to the device leaving the network. Once re-advertised, the FriendlyIconListStatus state variable should be in the clean state. Note that getToken() and postToken() and status@icon are not part of the DDD.

6.6.5.5 Errors

Table 6-17: Error Codes for RestoreFriendlyInfo()

<table>
<thead>
<tr>
<th>ErrorCode</th>
<th>errorDescription</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400-499</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>500-599</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>600-699</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>700</td>
<td>Reserved for future extensions.</td>
<td></td>
</tr>
</tbody>
</table>

Note: 800-899 Error Codes are not permitted for standard actions. See UPnP Device Architecture section on Control for more details.

7 Theory of Operation (Informative)

This service enables UPnP control points to discover devices that allow their friendly DDD - <friendlyName> and <iconList> - information to be updated.

Control points can get (read) the current values, both pending and clean, of those friendly DDD elements using the GetFriendlyName() and GetFriendlyIconList() actions. The GetFriendlyIconList() action can also expose resources for creating new icons. Control points can set (write) the friendly DDD - <friendlyName> and <iconList> - information using the SetFriendlyName() and SetFriendlyIconList() actions respectively. Special timing considerations, to prevent race conditions, are enforced. If supported, a control point can request that the device restore its factory settings using the RestoreFriendlyInfo() action.
7.1 Changing the device <friendlyName>

In this example the control point wants to change the <friendlyName> value of a device it has discovered where the current value of the <friendlyName> elements is <friendlyName>XYZ MediaServer</friendlyName>. It invokes the SetFriendlyName() action as follows:

Request (SOAP):
SetFriendlyName(
  "My Super ACME MediaServer in the Den"
)

As a result, the device grants the control point request to update the DDD <friendlyName>, but in this case, because it is in the process of serving content, it does not immediately leave the network and then re-advertise the new device. Instead, it immediately updates the FriendlyNameStatus state variable from the following value (assuming there are no pending DDD updates, that is, the device is in the clean state with regards to the <friendlyName>)

<?xml version="1.0" encoding="UTF-8"?>
<FriendlyNameStatus
  xmlns="urn:schemas-upnp-org:fd: fName-events"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xsi:schemaLocation="urn:schemas-upnp-org:fd:fns-events
                      http://www.upnp.org/schemas/fd/fns-events.xsd">
  <friendlyName status="DDD">ACME MediaServer</friendlyName>
</FriendlyNameStatus>

and events the change. The device then waits at least 30 seconds to see if there are any additional invocations of the GetFriendlyName(), GetFriendlyIconList(), SetFriendlyName(), SetFriendlyIconList() or RestoreFriendlyInfo() action to see if a re-advertisement is OK. Since it was in the process of serving some content it waits until that session is complete.

7.2 Changing the device <iconList>

In this example, the device indicates in its FriendlyIconListStatus state variable all of its <iconList> information including two existing icons already advertised in the DDD and two open icon slots for posting new icon binaries – one for JPEG and one for PNG.

---

1 In the following examples some tabs, whitespaces, and carriage return/line-feeds have been added within element values for readability and are not intended for exact interpretation.

Copyright © 2014, UPnP Forum. All rights Reserved.
<?xml version="1.0" encoding="UTF-8"?
<FriendlyIconListStatus
xmlns="urn:schemas-upnp-org:fd:fis-events"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:schemas-upnp-org:fd:fis-events http://www.upnp.org/schemas/fd/fis-events.xsd"
><iconList>
<icon status="DDD"
</mime type><getToken>
</postToken>
</icon>
<icon status="OPEN" maxBytes="10000000">
<mime type><getToken>
</postToken>
</icon>
</iconList>
</FriendlyIconListStatus>

The control point invokes a `SetFriendlyIconList()` action targeting a replacement\(^3\) of the “medium” icon with a new “bigger” icon binary located somewhere in the network at URI `http://192.168.1.15:80/iconlocation/zbot720biggerIconBinary.png`. Since the device supports both HTTP-POST and HTTP-GET the control point, uses the `<getToken>` element (not the `<postToken> element) in the `TargetIconToken` input argument to indicate that the device shall use an HTTP-GET to create the new icon binary. The action is as follows:

**Request (SOAP):**
SetFriendlyIconList(
"CREATE",
"get-004-png",
"http://192.168.1.15:80/iconlocation/zbot720biggerIconBinary.png"
)

---

1. In the following examples some tabs, whitespaces, and carriage return/line-feeds have been added within element values for readability and are not intended for exact interpretation.
2. The IP address may be explicit or understood to be relative to the device.
3. The replacement will be accomplished by first creating a new icon and then deleting the existing.
Before returning an action response the device begins execution of an HTTP-GET to
TargetIconURI

Suppose, in the mean time another control point invokes a SetFriendlyIconList() action targeting an
update of the "small" icon with an icon binary that it plans to post as follows:

Request (SOAP):
SetFriendlyIconList(
    "CREATE",
    "post-003-jpg"
)

However, since the device has not completed the outstanding HTTP-GET it returns an errorCode
709.

Within 30 seconds the device completes the original HTTP-GET and immediately updates and
events the FriendlyIconListStatus state variable (as shown below) to reflect the new icon
parameters. The device also adds another open icon slot for additional PNG image uploads. Since
the new icon was 2 megabytes, and in this case the original 10 megabytes the device memory
available is indicated in a new maxBytes value of 8 megabytes.

<?xml version="1.0" encoding="UTF-8"?>
<FriendlyIconListStatus
 xmlns="urn:schemas-upnp-org:fd:fis-events"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
    urn:schemas-upnp-org:fd:fis-events
    http://www.upnp.org/schemas/fd/fis-events.xsd"
><iconList>
    <icon status="DDD"
    <mimetype>image/png</mimetype>
    <height>80</height>
    <width>80</width>
    <depth>24</depth>
    <url>icons/zbot720smallIconBinary.png</url>
</icon>

    <icon status="DDD"
    <mimetype>image/png</mimetype>
    <height>320</height>
    <width>320</width>
    <depth>24</depth>
    <url>icons/zbot720mediumIconBinary.png</url>
</icon>

    <icon status="OPEN" maxBytes="8000000"
    <mimetype>image/jpg</mimetype>
    <getToken>get-003-jpg</getToken>
    <postToken postUri="http://192.168.1.5/icons/jpg?post">
        post-003-jpg
    </postToken>
</icon>

    <icon status="PENDING"
    <mimetype>image/png</mimetype>
    <height>480</height>
    <width>640</width>
    <depth>24</depth>
</icon>
</iconList>
Next, the **DDD** `<icon>` element with `<url>` value of "icons/zbot720mediumIconBinary.png" is targeted for deletion by the control point invoking the following action:

**Request (SOAP):**
SetFriendlyIconList(
"DELETE",
"
"icons/zbot720mediumIconBinary.png")

The device, as a result of a successful invocation of the action marks the icon for deletion and events the updated **FriendlyIconListStatus** state variable as follows:

<?xml version="1.0" encoding="UTF-8"?>
<FriendlyIconListStatus
xmlns="urn:schemas-upnp-org:fd:fis-events"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:schemas-upnp-org:fd:fds-events http://www.upnp.org/schemas/fd/fis-events.xsd">
  <iconList>
    <icon status="DDD">
      <mimetype>image/png</mimetype>
      <height>80</height>
      <width>80</width>
      <depth>24</depth>
      <url>icons/zbot720smallIconBinary.png</url>
    </icon>
    <icon status="DELETED">
      <mimetype>image/png</mimetype>
      <url>icons/zbot720mediumIconBinary.png</url>
    </icon>
    <icon status="OPEN" maxBytes="8000000">
      <mimetype>image/jpg</mimetype>
      <getToken>get-003-jpg</getToken>
      <postToken postUri="http://192.168.1.5/icons/jpg?post">
        post-003-jpg
      </postToken>
    </icon>
    <icon status="PENDING">
      <mimetype>icons/png</mimetype>
      <height>480</height>
      <width>640</width>
      <depth>24</depth>
      <url>icons/zbot720biggerIconBinary.png</url>
    </icon>
    <icon status="OPEN" maxBytes="8000000">
      <mimetype>image/png</mimetype>
      <getToken>get-005-png</getToken>
      <postToken postUri="http://192.168.1.5/icons/png?post">
        post-005-png
      </postToken>
    </icon>
  </iconList>
</FriendlyIconListStatus>
The device then waits 30 seconds to determine if any other FriendlyInfoUpdate service actions are invoked. After 30 seconds have lapsed and no other such actions are invoked, the device is ready to clean its state so it leaves the network and re-advertises itself with the new <friendlyName> and <iconList> values. The FriendlyNameStatus and FriendlyIconListStatus state variable now appear as:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<FriendlyNameStatus
 xmlns="urn:schemas-upnp-org:fd:fns-events"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:urn:schemas-upnp-org:fd:fns-events=
 xsi:schemaLocation="
 http://www.upnp.org/schemas/fd/fns-events.xsd">
  <friendlyName status="DDD">My Super ACME MediaServer in the Den</friendlyName>
</FriendlyNameStatus>
```

and,

```xml
<?xml version="1.0" encoding="UTF-8"?>
<FriendlyIconListStatus
 xmlns="urn:schemas-upnp-org:fd:fis-events"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:urn:schemas-upnp-org:fd:fis-events=
 xsi:schemaLocation="
 http://www.upnp.org/schemas/fd/fis-events.xsd">
  <iconList>
    <icon status="DDD">
      <mimetype>image/png</mimetype>
      <height>80</height>
      <width>80</width>
      <depth>24</depth>
      <url>icons/zbot720smallIconPublic.png</url>
    </icon>
    <icon status="DDD">
      <mimetype>image/png</mimetype>
      <height>480</height>
      <width>640</width>
      <depth>24</depth>
      <url>icons/zbot720biggerIconPublic.png</url>
    </icon>
    <icon status="OPEN" maxBytes="8000000">
      <mimetype>image/jpg</mimetype>
      <getToken>get-003-jpg</getToken>
      <postToken postUri="http://192.168.1.5/icons/jpg?post">
        post-003-jpg
      </postToken>
    </icon>
    <icon status="OPEN" maxBytes="8000000">
      <mimetype>image/png</mimetype>
```

Copyright © 2014, UPnP Forum. All rights Reserved.
respectively. In this case there is no memory gain associated with the deletion of the original DDD icon since it is retained internally for support of the `RestoreFriendlyInfo()` action.

Next, a control point invokes the following action to restore the factory `<friendlyName>` and `<iconList>`:

**Request (SOAP):**
`RestoreFriendlyInfo("ALL")`

The device, after 30 seconds, restores the original DDD friendly information and updates the `FriendlyNameStatus` and `FriendlyIconListStatus` state variables with the original friendly information and events them as follows:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<FriendlyNameStatus
 xmlns="urn:schemas-upnp-org:fd:fis-events"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
  urn:schemas-upnp-org:fd:fis-events
  http://www.upnp.org/schemas/fd/fis-events.xsd">
  <friendlyName status="PENDING">ACME MediaServer</friendlyName>
</FriendlyNameStatus>

and

```xml
<?xml version="1.0" encoding="UTF-8"?>
<FriendlyIconListStatus
 xmlns="urn:schemas-upnp-org:fd:fis-events"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
  urn:schemas-upnp-org:fd:fis-events
  http://www.upnp.org/schemas/fd/fis-events.xsd">
  <icon status="DDD">
    <mimetype>image/png</mimetype>
    <height>80</height>
    <width>80</width>
    <depth>24</depth>
    <url>icons/zbot720smallIconBinary.png</url>
  </icon>
  <icon status="PENDING">
    <mimetype>image/png</mimetype>
    <height>320</height>
    <width>320</width>
    <depth>24</depth>
    <url>icons/zbot720mediumIconBinary.png</url>
  </icon>
  <icon status="OPEN" maxBytes="10000000">
    <mimetype>image/jpg</mimetype>
  </icon>
</FriendlyIconListStatus>
```
FriendlyInfoUpdate:

<getToken>get-003-jpg</getToken>
<postToken postUri="http://192.168.1.5/icons/jpg?post">
    post-003-jpg
</postToken>
</icon>
<icon status="OPEN" maxBytes="10000000">
    <mimetype>image/png</mimetype>
    <height>80</height>
    <width>80</width>
    <depth>24</depth>
    <url>icons/zbot720smallIconBinary.png</url>
</icon>
<icon status="DDD">
    <mimetype>image/png</mimetype>
    <height>80</height>
    <width>80</width>
    <depth>24</depth>
    <url>icons/zbot720smallIconBinary.png</url>
</icon>
<icon status="DDD">
    <mimetype>image/png</mimetype>
    <height>320</height>
    <width>320</width>
    <depth>24</depth>
    <url>icons/zbot720largeIconBinary.png</url>
</icon>
<icon status="OPEN" maxBytes="10000000">
    <url>image/jpg</url>
    <getToken>get-003-jpg</getToken>
    <postToken postUri="http://192.168.1.5/icons/jpg?post">
        post-003-jpg
    </postToken>
</icon>
</iconList>
</FriendlyIconListStatus>

The device then waits 30 seconds to determine if any other FriendlyInfoUpdate service actions are invoked. After 30 seconds have lapsed and no other such actions are invoked, the device is ready to clean its state so it leaves the network and re-advertises itself with the original <friendlyName> and <iconList> values. The FriendlyNameStatus and FriendlyIconListStatus state variable now appear as:

<?xml version="1.0" encoding="UTF-8"?>
<FriendlyNameStatus
    xmlns="urn:schemas-upnp-org:fd:fns-events"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="urn:schemas-upnp-org:fd:fns-events
http://www.upnp.org/schemas/fd/fns-events.xsd">
    <friendlyName status="DDD">ACME MediaServer</friendlyName>
</FriendlyNameStatus>

and,

<?xml version="1.0" encoding="UTF-8"?>
<FriendlyIconListStatus
    xmlns="urn:schemas-upnp-org:fd:fis-events"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="urn:schemas-upnp-org:fd:fis-events
http://www.upnp.org/schemas/fd/fis-events.xsd">
    <iconList>
        <icon status="DDD">
            <mimetype>image/png</mimetype>
            <height>80</height>
            <width>80</width>
            <depth>24</depth>
            <url>icons/zbot720smallIconBinary.png</url>
        </icon>
        <icon status="DDD">
            <mimetype>image/png</mimetype>
            <height>320</height>
            <width>320</width>
            <depth>24</depth>
            <url>icons/zbot720largeIconBinary.png</url>
        </icon>
        <icon status="OPEN" maxBytes="10000000">
            <url>image/jpg</url>
            <getToken>get-003-jpg</getToken>
            <postToken postUri="http://192.168.1.5/icons/jpg?post">
                post-003-jpg
            </postToken>
        </icon>
    </iconList>
</FriendlyIconListStatus>
respectively. Note in this case that the device has internally deleted the non original icons prior to re-advertisement. This behavior is decided by the implementation.

8 XML Service Description

```xml
<?xml version="1.0"?>
<scpd xmlns="urn:schemas-upnp-org:service-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <actionList>
    <action name="GetFriendlyName">
      <argumentList>
        <argument name="NameStatus" direction="out"/>
        <relatedStateVariable>FriendlyNameStatus</relatedStateVariable>
      </argumentList>
    </action>
    <action name="GetFriendlyIconList">
      <argumentList>
        <argument name="IconListStatus" direction="out"/>
        <relatedStateVariable>FriendlyIconListStatus</relatedStateVariable>
      </argumentList>
    </action>
    <action name="SetFriendlyName">
      <argumentList>
        <argument name="NewName" direction="in"/>
        <relatedStateVariable>A_ARG_TYPE_NewName</relatedStateVariable>
      </argumentList>
    </action>
    <action name="SetFriendlyIconList"/>
  </actionList>
</scpd>
```
<argumentList>
    <argument>
        <name>UpdateType</name>
        <direction>in</direction>
        <relatedStateVariable>A_ARG_TYPE_UpdateType</relatedStateVariable>
    </argument>
    <argument>
        <name>TargetIconToken</name>
        <direction>in</direction>
        <relatedStateVariable>A_ARG_TYPE_Token</relatedStateVariable>
    </argument>
    <argument>
        <name>TargetIconURI</name>
        <direction>in</direction>
        <relatedStateVariable>A_ARG_TYPE_IconURI</relatedStateVariable>
    </argument>
</argumentList>

<action>
    <name>RestoreFriendlyInfo</name>
    <argumentList>
        <argument>
            <name>RestoreType</name>
            <direction>in</direction>
            <relatedStateVariable>A_ARG_TYPE_RestoreType</relatedStateVariable>
        </argument>
    </argumentList>
</action>

<serviceStateTable>
    <stateVariable sendEvents="yes">
        <name>FriendlyNameStatus</name>
        <dataType>string</dataType>
    </stateVariable>
    <stateVariable sendEvents="yes">
        <name>FriendlyIconListStatus</name>
        <dataType>string</dataType>
    </stateVariable>
    <stateVariable sendEvents="no">
        <name>A_ARG_TYPE_NewName</name>
        <dataType>string</dataType>
    </stateVariable>
    <stateVariable sendEvents="no">
        <name>A_ARG_TYPE_IconURI</name>
        <dataType>string</dataType>
    </stateVariable>
    <stateVariable sendEvents="no">
        <name>A_ARG_TYPE_UpdateType</name>
        <dataType>string</dataType>
        <allowedValueList>
            <allowedValue>CREATE</allowedValue>
            <allowedValue>DELETE</allowedValue>
        </allowedValueList>
    </stateVariable>
    <stateVariable sendEvents="no">
        <name>A_ARG_TYPE_Token</name>
        <dataType>string</dataType>
    </stateVariable>
</serviceStateTable>
<stateVariable sendEvents="no">
  <name>A_ARG_TYPE_RestoreType</name>
  <dataType>string</dataType>
  <allowedValueList>
    <allowedValue>ALL</allowedValue>
    <allowedValue>FRIENDLYNAME</allowedValue>
    <allowedValue>ICONLIST</allowedValue>
  </allowedValueList>
</stateVariable>
</serviceStateTable>
</scpd>