ExternalActivity:1.0 Service Template Version 1.01

For UPnP™ Version 1.0
Status: ***Standardized DCP***
Date: ***September 11, 2002***

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

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.

Authors	Company
Larry Copp	Hewlett-Packard Co.
Patrick Vine	Microsoft Corporation

Contents

List of Tables

1. Overview and Scope

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

- This service represents a front-panel by registering a control point for a specific user interaction. It allows the service to control its resources by enabling limited registrations. It also interacts with an associated Scan service to address contention issues after a user interaction has occurred.

1.1. Change Log

Front_Panel 0.10 to ExternalAction 0.11 – Included changes from the introduction and review at the 23April2001 meeting and from input received since.

- Changed the name of the service to ExternalActivity
- Changed the name of the RegisterButton to RegisterCommand
- Changed the name of the UnregisterButton to UnregisterCommand
- Added the RenewCommand action
- Added a timeout value to the RegisterCommand action
- Added the CommandStringSize variable to indicate the maximum size of a CommandString value
- Rewrote the Theory of Operation section
- Fixed the XML to match all changes

0.12 – Includes changes based on the May 22, 2001 teleconference review and comments from several members of the working group

- Changed RegisterCommand action to Register
- Changed RenewCommand action to Renew
- Changed UnregisterCommand action to Unregister
- Changed CommandString to DisplayString
- Changed Timeout parameter in Register and Renew to Duration
- Added an *ActualDuration* parameter to the Register and Renew actions.
- Replaced the A_ARG_TYPE_xxx variables with more meaningful variables.
- Added "*device-setting*" value to the allowed value list for ButtonName.
- Changed the Duration variable to an i4 type and added –1 as the "*device-setting*" value.

0.13 – Includes changes based on the June 21, 2001 teleconference review

- Removed the *device-setting* allowed value from the ButtonName variable

0.70 – Updated based on input from the 31 July 2001 meeting in Toronto ON. This includes the following:

- Corrected numerous typographical errors
- Added explanations regarding the meaning of the *ALL* allowed value for the ButtonPress variable.
- Added the standard SCPD explanation header in Section 3.
- Deprecated Error Code 404 – Invalid Var
- Added new Common Error Code values

0.71 – Updated based on input from various team members.

- Clarified the definition of the Activity, DisplayString, and Password variables.
- Corrected various typographical errors
- Removed Fax and Copy from the ButtonPress allowedValueList
- Removed the allowedValueRange from the AvailableRegistrations variable since allowedValueLists are only valid on string types

0.72 – Updated based on input from various team members. This includes the following"

- Changed "*Proposed Service Template*" to "*Preliminary Design*"
- Removed allowed value list from the AvailableRegistrations variable in Table 1.
- Used "Out of Memory (603)" error code instead of CONFIG_NO_RESOURCES in the Register action.
- Used "Invalid Args (402)" instead of CONFIG_TOO_BIG in the Register action.

- Changed the name of the CONFIG_IN_USE error code to DuplicateDisplayName in the Register action.
- Added a specific definition of the Activity variable to clarify the example that is already there.

0.73 – Updated based on implementation issues and reviews

- Removed the *maximumrate attribute from the Activity and AvailableRegistrations variable descriptions in the SCPD. This attribute is not supported by the SCPD*

0.80 – Updated based on input from the 7Feb2002 meeting in Los Angeles

- Removed the SetPassword and all support of passwords
- Several string type variables were of type String – I changed them all to lower case
- The Duration variable in Table 1 was changed from ui4 type to i4.
- Added some information to the Theory of Operations section to explain interactions with an "Associated Service".
- Updated the Testing section (it was empty before and is nearly empty still) to reflect the testing that we implement. The implementation is limited due to Certification Tool capabilities.

0.90 – Updated after the successful completion of the Scanner PlugFest and three sample implementations passed the Certification Test.

- Changed the version to 0.90 and made the status *Design Complete*

0.91 – Updated to reflect changes needed to comply with the Service Template v1.01 and the Service Template Checklist

- Added a simple functional description in section 1
- Added all non-evented variables to Table 2 "Evented Variables"
- Changed the argument names for all actions by appending "In" or "Out" to the existing name based upon the direction of the argument in the Argument tables, action descriptions and the SCPD XML
- Changed *nnn* in variable limits to *vendor-defined*
- Changed the maximum value in the allowedValueRange for RegistrationID variable from 0xffffffff to 4294967295
- *– Document approved by Steering Committee*
- Changed version to 1.0
- Changed status to Standardized DCP
- Changed date to September 11, 2002
- Changed legal disclaimer on the first page to Appendix G of the SC Org and Process Doc.
- Changed all copyrights to "© 1999-2001 Microsoft Corporation. All rights Reserved."
- In section 2.1 "Service Type", changed the version to 1

2. Service Modeling Definitions

2.1. ServiceType

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

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

2.2. Service State Table

Table 1: Service State Table

Variable Name	Req. or Opt.[1]	Data Type	Allowed Value[2]	Default Value [2]	Eng. Units
Activity	*R*	*string*			*n/a*
AvailableRegistrations (Read-Only Value)	*R*	*Boolean*		*<implementation specific>* *Recommended Value: 1*	*N/A*
DisplayString	*R*	*string*			
DisplayStringSize (Read-Only Value)	*R*	*ui4*	*[0|vendor-defined]*		
ButtonName	*R*	*string*	*[All]*	*All*	*n/a*
Duration	*R*	*i4*	*-1 – maxi4*		
RegistrationID	*R*	*ui4*	*1-max ui4*		

[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. Activity

A string that reflects the most recent activity (possibly a button press, an activity such as a motion sensor, etc.). This string contains a concatenation of the ButtonName value, the DisplayString value, and a sequence number as shown in the example below. Vendor specific information may be included after the sequence number if needed.

<ButtonName>;<DisplayString>;<SequenceNumber><;optional vendor specific information>

ScanButton;John Jones;11234;Scanner at NW entrance

The separate parts of the string are each separated by a semicolon (;). The concatenation of the two values solves a race that occurs when two evented variables change simultaneously. UPnP does not specify if the events will arrive together, or if separately, which order they will arrive in. The sequence value is meaningless and it should be ignored. It simply ensures that an event will occur even if the same button is pressed two times sequentially.

2.2.2. AvailableRegistrations

The *AvailableRegistrations* value indicates whether the ExternalActivity service is currently able to accept registrations or not. This value is TRUE (1) when the service is able to successfully execute the Register action.

2.2.3. DisplayString

The DisplayString given in the most recent *Register* action. This value is intended to uniquely identify a user choice in a display (if the device has one). This can be any string value that is unique for an ExternalActivity Service. If the device has a display and uses presents menus to a user, then this string is the message that should be presented for each choice.

2.2.4. DisplayStringSize

The *DisplayStringSize* variable is a constant. It is intended to provide a maximum string length, in characters, for a display. Control points should refer to the SCPD to determine the maximum length of the DisplayString value. The string is specified to be in UTF8 characters. This means that not all characters will be one byte long. They may be up to 4 bytes in length. Vendors should allocate sufficient memory to hold the oversized strings, or at least be able to detect that the string may exceed the available resources and act accordingly. The maximum value is vendor unique.

Table 1.1: DisplayStringSize allowedValueRange

Value		Req. or Opt.
Minimum	0	R
Maximum	Vendor Unique	R
Step	1	R

2.2.5. ButtonName

The name of a button or other activity used in the Register action.

A vendor may extend or subset the allowed values to support any external activity needed. The allowedValueList below contains an optional value of “Scan”, however other values could include concepts like *Red*, *Green*, *TemperatureSensor*, *PhoneRinging*, *ScanTo*, etc. Basically, the ButtonName value can be used to represent a specific type of interaction.

How the ExternalActivity Service Description <defaultValue> and <allowedValueList> elements are configured with these values is implementation specific.

If the value of *All* is given, then the *Register* action should be applied to all supported buttons and activities.

Table 1.2: ButtonPress allowed values

Value	Req. or Opt.
All	R
Scan	O

2.2.6. Duration

Represents the duration of a registration in the Register and Renew actions.

Table 1.3: Duration allowed values

Value		Req. or Opt.
Minimum	-1	R
Maximum	Vendor Unique	R
Step	1	R

A value of –1 means use the current value. A value of 0 means no timeout. Any other value is the number of seconds that must elapse before the registration will timeout.

2.2.7. RegistrationID

Represents the Registration ID value in all of the defined actions.

2.3. Eventing and Moderation

Table 2: Evented Variables

Variable Name	Evented	Moderated Event?	Max Event Rate	Logical Combination	Min Delta per Event
Activity	Yes	Yes	1		N/A
AvailableRegistrations	Yes	Yes	1		N/A
DisplayString	No				
DisplayStringSize	No				
ButtonName	No				
Duration	No				
RegistrationID	No				

2.4. Actions

Immediately following this table is detailed information about these actions, including short descriptions of the actions, the effects of the actions on state variables, and error codes defined by the actions.

Table 3: Actions

Name	Req. or Opt. [1]
Register	*R*
Renew	*R*
Unregister	*R*

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

2.4.1. Register

2.4.1.1. Arguments

Table 4: Arguments for Register

Argument	Direction	relatedStateVariable
ButtonNameIn	*IN*	*ButtonName*
DisplayStringIn	*IN*	*DisplayString*
DurationIn	*IN*	*Duration*
ActualDurationOut	*OUT*	*Duration*
RegistrationIDOut	*OUT*	*RegistrationID*

2.4.1.2. Effect on State

This action registers the given DisplayStringIn with the service. A unique RegistrationIDOut value is returned. The DisplayStringIn must be a unique value to the ExternalActivity Service. The value of the ButtonNameIn is constrained by the action buttons available on the device. For example, if the represented front-panel has two (2) buttons, the values may be limited to "*RedButton*" and "*GreenButton*". If there are not enough resources available for the registration (AvailableRegistrations value is FALSE), then the association will not be made and Out of Memory (603) will be returned. If the DisplayStringIn value is not unique then the action will fail with DuplicateDisplayString (730). If the DisplayStringIn value exceeds the maximum length, indicated in the DisplayStringSize allowedValueRange, or the string contains more bytes than are available, then the action will not be performed and a Invalid Args (402) error will be returned.

When a external activity occurs, the Activity variable changes to reflect the concatenation of the ButtonName variable, DisplayString variable chosen by the external activity and a recently unique sequence value. The RegistrationID value associated with the DisplayString is sent to the associated service (e.g. to the Scanner service). The control point must provide the RegistrationID value, returned from the Register action, when dealing with the associated service after the interaction.

The RegistrationID value will be valid until the Duration has expired. The requested DurationIn is advisory. The service may use a different duration value. The actual duration value used is returned in the ActualDurationOut argument. If the duration expires, then the registration is removed. A control point may prevent the expiration of the registration by using the Renew action to reset the expiration time. There is no notification that the registration has expired.

Note that the control point must subscribe for event notification to receive notification when the state variables change.

2.4.1.3. Errors

errorCode	errorDescription	Description
730	DuplicateDisplayString	The configuration is already set and cannot be changed.

2.4.2. Renew

2.4.2.1. Arguments

Table 5: Arguments for Renew

Argument	Direction	relatedStateVariable
RegistrationIDIn	*IN*	*RegistrationID*
DurationIn	*IN*	*Duration*
ActualDurationOut	*OUT*	*Duration*

2.4.2.2. Effect on State

The *Renew* action extends the expiration time associated with the RegistrationIDIn and DisplayStringIn for an additional *DurationIn* seconds. The requested Duration is advisory. The service may use a different duration value. The actual duration value used is returned in the ActualDurationOut argument. If the RegistrationIDIn value does not refer to an existing ID value, then an Invalid_ID error will be returned.

2.4.2.3. Errors

errorCode	errorDescription	Description
712	Invalid_ID	The given ID value is invalid

2.4.3. Unregister

2.4.3.1. Arguments

Table 6: Arguments for Unregister

Argument	Direction	relatedStateVariable
RegistrationIDIn	*IN*	*RegistrationID*

2.4.3.2. Effect on State

The *Unregister* action removes the given association from the ExternalActivity. This may cause a change in the AvailableRegistrations variable. If the RegistrationIDIn value does not refer to an existing ID value, then an Invalid_ID error will be returned.

2.4.3.3. Errors

errorCode	errorDescription	Description
712	Invalid_ID	An invalid RegistrationID was given.

2.4.4. 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 7: 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.
501	Action Failed	See UPnP Device Architecture section on Control.
600	Argument Value Invalid	The argument value is invalid.
601	Argument Value Out of Range	An argument value is less than the minimum or more than the maximum value of the allowedValueRange, or is not in the allowedValueList.
602	Optional Action Not Implemented	The requested action is optional and is not implemented by the device.
603	Out of Memory	The device does not have sufficient memory available to complete the action. This may be a temporary condition; the control point may choose to retry the unmodified request again later and it may succeed if memory is available.
604 *Proposed*	Human Intervention Required	The device has encountered an error condition which it cannot resolve itself and requires human intervention such as a reset or power cycle. See the device display or documentation for further guidance.

2.5. Theory of Operation

The ExternalActivity service is designed to represent a front-panel device in association with another service. For example, it could represent a simple two- or three-button scanner – in this case, it represents the buttons and is associated with the Scanner service. It could also represent a front-panel with a display,

a NextItem button, a PreviousItem button, and a select button – in this case, command strings would be shown on the display as a user scrolls the list with the NextItem and PreviousItem buttons. The Activity value would be set to a concatenation of the ButtonName of the button pressed, the string shown in the display when the Select button is pressed, and a recently unique sequence value, all separated by semicolons (;). This value is evented, and so interested control points should register for eventing.

The ExternalActivity service provides the Register action so that a control point can register a DisplayString value. Each DisplayString value MUST be unique in the service. The Register action returns a RegistrationID that is associated with the DisplayString. This ID value must be used with all subsequent interactions that relate to the DisplayString.

When the DisplayString is registered, it is given a requested Duration value. The actual duration value is returned in the ActualDuration argument. When the ActualDuration period elapses, the DisplayString is removed from the service. There is no notification that the DisplayString was removed. A control point can renew the DisplayString period by executing the Renew action before the ActualDuration occurs.

The control point should remove the DisplayString value when it no longer needs it. A DisplayString is removed by executing the UnRegister action with the RegistrationID value as an argument.

A service may only be able to store a limited number of registered users simultaneously. Once the service has reached its limits, the AvailableRegistrations variable will change to false (0). Once the service has sufficient resources to accept a new registration, potentially due to a control point calling the UnRegister action or through some other means, the AvailableRegistrations variable changes to true (1).

2.5.1. Interactions with an Associated Service

Typically, the ExternalActivity service is part of a device that combines a front-panel with other functions, such as a scanner. In this combination there is typically another service used to control the other function. This service and the ExternalActivity service can work together through an association that is vendor unique and beyond the scope of this document. The overall behavior of this association is discussed here, but the details of the interactions are left for the vendor to specify and implement.

Interactions with an associated service involve two different use models.

- the ExternalActivity initiated model where the first interaction starts with a user activity
- the externally initiated model where the first interaction starts outside of the ExternalActivity service

2.5.1.1. ExternalActivity Initiated Use Model

This use model starts when a user presses a button or otherwise causes an activity change in the ExternalActivity service. In this case, the Activity variable is changed and event notifications are sent to all subscribers. In addition, the corresponding RegistrationID value is sent to the associated service through an unspecified vendor-specific internal mechanism. The associated service will use the RegistrationID value as specified in its *Service Definition*[2]. No further interaction is needed or specified.

> **A Scanner related example** - The following will occur when the associated service is a Scanner (v1.0) service:
>
> - The control-point subscribes for events with the Scanner and ExternalActivity services
> - The control-point executes the ExternalActivity::Register action with *Button=”Scan”*, *DisplayString=”My Name”*, and *Duration=300* (seconds). The response includes: *ActualDuration=300* and *RegistrationID=123456*.

[2] The UPnP Scanner Service v1.0 is one such associated service. See its definition document for details on how the ExternalActivity service and the Scanner service interact.

- The user walks to the scanner, places a document in it and presses the "Scan" button.
- The *RegistrationID* value is sent to the Scanner service and then the Activity event is sent to all subscribers. The Scanner service uses the *RegistrationID* to restrict access to the *StartScan* action. The restriction is intended to close a race condition where a user presses a scan button and another control-point attempts to start an unrelated scan. Only the control-point that possesses the proper *RegistrationID* value will be allowed to execute the *StartScan* action. The restriction lasts until *StartScan* is successfully executed or until a timeout (Scanner Timeout variable) occurs.
- The control-point executes the *StartScan* action with *RegistrationID=123456* and the scanner service transitions to the Scanning state and the scan proceeds as normal.
- At the end of the scan operation, the control-point unregisters RegistrationID=123456 and unsubscribes from both services.

2.5.1.2. Externally Initiated Use Model

This use model starts when a control-point, or user, interacts with an associated service. In this case, the associated service would use an unspecified, vendor-specific, internal mechanism to set the current RegistrationID value in the ExternalActivity service. If the RegistrationID value is not valid, then it should be ignored. If it is valid, then the ExternalActivity service should select the corresponding registration values. If the front-panel has a display, then the information displayed should show the selected registration values. The selected registration values should be used in the Activity value if the user presses a button, or otherwise causes an activity change.

A Scanner related example – The following is an example with a Scanner (v1.0) service:

- The control-point subscribes with the Scanner and ExternalActivity services
- The control-point executes the ExternalActivity::Register action with *Button="Scan"*, *DisplayString="My Name"*, and *Duration=300* (seconds). The response includes: *ActualDuration=300* and *RegistrationID=123456*.
- The control-point executes the Scanner::StartScan action with *RegistrationID=123456*, *UseFeeder=0* and *SideCount=0*. The response includes: *JobID=4321*. The scanner transitions to the Pending state and waits there for something to happen (either a Start action, a timeout, Stop action or an Abort action). The Scanner service sends the RegistrationID value to the ExternalActivity service, which sets the variables to reflect the corresponding RegistrationID value (*Button="Scan"* and *DisplayString="My Name").*
- The user walks to the scanner, places a document in it and presses the "*Scan*" button. The ExternalActivity sends the RegistrationID (123456) to the Scanner service and sends out the Activity event to all subscribers.
- The control-point recognizes the event and sends a *Start* action with *JobID=4321* to the scanner service to continue the job. The scanner transitions to the *Scanning* state and the scan proceeds as normal.
- When all is complete, the control-point unregisters with RegistrationID=123456 and unsubscribes with both the Scanner and ExternalActivity services.

3. XML Service Description

The XML document below is a sample *Service Control Protocol Document* (SCPD) for a scanner device. It should be modified as needed by a scanner vendor to fully describe the scanner service offered by the scanner device. The scanner device should make the modified document available at the SCPD URL given in the device descriptor. A client will perform an HTTP/GET operation on that URL to get the document. NOTE: The XML comments in this section are for information only and should be omitted in the SCPD.

```
<?xml version="1.0"?>
<scpd xmlns="urn:schemas-upnp-org:service-1-0">
   <specVersion>
```

```
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <actionList>
    <action>
      <name>Register</name>
      <argumentList>
        <argument>
          <name>ButtonNameIn</name>
          <relatedStateVariable>ButtonName</relatedStateVariable>
          <direction>in</direction>
        </argument>
        <argument>
          <name>DisplayStringIn</name>
          <relatedStateVariable>DisplayString </relatedStateVariable>
          <direction>in</direction>
        </argument>
        <argument>
          <name>DurationIn</name>
          <relatedStateVariable>Duration</relatedStateVariable>
          <direction>in</direction>
        </argument>
        <argument>
          <name>ActualDurationOut</name>
          <relatedStateVariable>Duration</relatedStateVariable>
          <direction>out</direction>
        </argument>
        <argument>
          <name>RegistrationIDOut</name>
          <relatedStateVariable>RegistrationID</relatedStateVariable>
          <direction>out</direction>
        </argument>
      </argumentList>
    </action>
    <action>
      <name>Renew</name>
      <argumentList>
        <argument>
          <name>RegistrationIDIn</name>
          <relatedStateVariable>RegistrationID</relatedStateVariable>
          <direction>in</direction>
        </argument>
        <argument>
          <name>DurationIn</name>
          <relatedStateVariable>Duration</relatedStateVariable>
          <direction>in</direction>
        </argument>
        <argument>
          <name>ActualDurationOut</name>
          <relatedStateVariable>Duration</relatedStateVariable>
          <direction>out</direction>
        </argument>
      </argumentList>
    </action>
    <action>
```

```
            <name>Unregister</name>
            <argumentList>
               <argument>
                  <name>RegistrationIDIn</name>
                  <relatedStateVariable>RegistrationID</relatedStateVariable>
                  <direction>in</direction>
               </argument>
            </argumentList>
         </action>
         Declarations for other actions added by UPnP vendor (if any) go here
      </actionList>
      <serviceStateTable>
         <stateVariable sendEvents="yes">
            <name>Activity</name>
            <dataType>string</dataType>
         </stateVariable>
         <stateVariable sendEvents="yes">
            <name>AvailableRegistrations</name>
            <dataType>boolean</dataType>
            <defaultValue>1</defaultValue>
         </stateVariable>
         <stateVariable sendEvents="no">
            <name>DisplayString</name>
            <dataType>string</dataType>
         </stateVariable>
         <stateVariable sendEvents="no">
            <!--  This is a constant value that indicates the size in characters -->
            <name>DisplayStringSize</name>
            <dataType>ui4</dataType>
            <defaultValue>vendor-defined</defaultValue>
            <allowedValueRange>
               <minimum>0</minimum>
               <maximum>vendor-defined</maximum>  <!-- Vendor defined range value -->
               <step>1</step>
            </allowedValueRange>
         </stateVariable>
         <stateVariable sendEvents="no">
            <name>ButtonName</name>
            <dataType>string</dataType>
            <defaultValue>All</defaultValue>
            <allowedValueList>
               <allowedValue>All</allowedValue>
               <!--  Optional values – The vendor may add these values or others as needed
               <allowedValue>Scan</allowedValue>
               <allowedValue>Fax</allowedValue>
               <allowedValue>Copy</allowedValue>
               -->
             </allowedValueList>
         </stateVariable>
         <stateVariable sendEvents="no">
            <name>Duration</name>
            <dataType>i4</dataType>
            <defaultValue>vendor-defined</defaultValue>
            <allowedValueRange>
               <minimum>-1</minimum>
```

```
            <maximum>vendor-defined</maximum>  <!-- Vendor defined range value -->
            <step>1</step>
          </allowedValueRange>
        </stateVariable>
        <stateVariable sendEvents="no">
          <name>RegistrationID</name>
          <dataType>ui4</dataType>
          <defaultValue>vendor-defined</defaultValue>
          <allowedValueRange>
            <minimum>1</minimum>
            <maximum>4294967295</maximum>
            <step>1</step>
          </allowedValueRange>
        </stateVariable>
        Declarations for other state variables added by UPnP vendor (if any) go here
      </serviceStateTable>
    </scpd>
```

4. Testing

4.1. Issues

Numerous issues exist with the Certification Tool at this time. Due to these issues, only limited *Syntax* testing is possible. The testing listed below is limited to actions that will return a successful status with arguments that were specified at the time that the test cases were written. This limits the testing to the *Register* action only.

4.2. Syntax Testing

4.2.1. Register Action Tests

The Register action syntax is tested to make sure that the syntax is valid.

4.2.2. Renew Action Tests

not testable

4.2.3. Unregister Action Tests

not testable