Scan: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.

Contents

List of Tables

1 Overview and Scope

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

ThScan service represents the scan functionality of a scanner device. A control point may use this service to initiate a scan operation and receive images that represent the document in the scanner. The control point may pull the images using HTTP/GETs from the Destination URL or the service map push the images using HTTP/POSTs. The service has the following basic functional areas:

Operation Functionality

The *StartScan, Start, Stop, Abort* and *SetConfiguration* actions are used to control the action of the scan service.

Status Functionality

The *GetConfiguration, GetSideInformation, GetDestination* and *GetState* actions are used to query the current state of the scan service.

This service template does not address:

- *Faxing or Copying*

1.1 Change Log for *Scan:1.0*

0.12 – Removed the Key parameters from actions

0.20 – Incorporated inputs from November 9, 2000 review

0.21 – Incorporated inputs from November 27, 2000 review

0.22 – Incorporated inputs from December 6, 2000 review – Note: this is a simplified scanner model

0.23 – Incorporated inputs from December 11, 2000 teleconference review. This includes the following changes:

- Changed the way that we handle the Image and Sheet size and location information. Since the limits change dynamically, I moved the limit information to the return values of an action from the SCPD. See the GetLimits action for more information.
- Added an explanation for each of the following subjects:
  - the relationship between the Image size, Image location variables and the Units variable
  - the relationship between the BaseName variable and the Destination variable
  - the Buffer architecture and use model
  - the use of XHTML-Print as an output format
- Changed the way that we specify *Units* related state variable values. The variables that are affected by *Units* are no longer advertised state variables. Instead, the values and limits are specified in some new actions. This gets around the problem of having to advertise new value limits when the *Units* value is changed.

0.24 – Incorporated inputs from the January 11, 2001 review. This includes the following changes:

- Changed the format of the document to reflect Service Template v1.0
- Added some allowed values to the ErrorCode variable
- Changed Busy state to Finishing to match the state diagrams
- Changed allowed Boolean values to 1 and 0 as per the UPnP Implementor's Guide
- Clarified the behavior of the Abort action with respect to ejecting pages in the feeder
- Clarified the SetConfiguration action with respect to when it can be used.

- Added the SetDefaultConfiguration so that we can get back to default values
- Added the Units variable as an argument to the SetLimits action.
- Clarified the discussion regarding Units, Sizes, Areas, etc. and the coordinate system.
- Added a NotReady state to the state diagram to allow vendors to have a warmup or calibration time during the start of a scan operation. This includes adding a NotReadyTimeout variable, and a StateReason variable.
- Added another possible Scan-To-Print method for discussion.
- Fixed the XML to reflect the changes.

0.40 – Incorporated inputs from the January 22 and 23, 2001 meeting. This includes the following changes:

- Fixed headers to reflect current template and service versions
- Fixed some minor formatting and typographical errors
- Changed the name of the SetDefaultConfiguration action to RestoreDefaultConfiguration to clarify the intended behavior.
- Clarified the behavior of several actions (Start, SetConfiguration, and RestoreDefaultConfiguration) when they are executed while in the wrong state.
- Added NotReadyTimeout as an argument in the syntax of SetConfiguration and GetConfiguration
- Added a description of the ResolutionList variable in the GetLimits action. This includes the ability to give ranges and distinct values. Examples of the value are also given.
- Clarified Section 2.5.1 on page 28 to make sure that Table 39 and Figure 1 are consistent.
- Added a section to describe the behavior of the scanner and feeder when timeouts occur.
- Added the NotReady state to Figure 1.
- Added protocol examples for section 2.5.7.1, section 2.5.7.2, and section 2.5.7.3 in appendicies.
- Added a description of what the XML Service Template is, and how it is used.
- Updated the format of the document to reflect the new Service Template document (v1.01)
- New format includes separating allowed values and ranges into *Required* and *Optional* values. Most state variables were changed to reflect these constraints.

0.41 – Incorporated changes from the February 20, 2001 Teleconference review This includes the following:

- Added a TimeoutNotReady error code (721) to the allowed value for the ErrorCode state variable, and in the discussion on timeouts in section 2.5.6.
- Added Bilevel as a required PixelType value
- Added 1 as a required SideCount value
- Modified the value range for Timeout and NotReadyTimeout to have a vendor specified Max value, and the default value equal to the max value
- Gave the keyword "*buffer" the same case everywhere*
- Combined the GetDestination, and GetSideNumber into one action named GetSideInformation
- Corrected step 4 of the Scanner Operation flow in section 2.5.3 that conflicts with the state conditions for SetConfiguration
- Fixed Table 15 to match the state requirements for the SetConfiguration action.
- Added a new state variable "*embedImages*" to control the XHTML-Print output.
- In the SCPD, the SetConfiguration and GetConfiguration actions had several arguments that were missing the beginning argument tag
- Changed the case of all direction values in the SCPD argument lists to lower case
- The Windows ME UPnP API seems to fail if the direction statement in an argument spec does not come after the relatedStateVariable statement, so I changed them all in the SCPD

0.50 – Fixed some XML errors and updated some links

0.51 – Updated to reflect the changes made during the Scanner meeting in Portland on April 23-24, 2001. This includes the following:

- Added an A_ARG_TYPE_ID state variable and included an argument of this type in the Start, Stop, Abort, RestoreDefaultConfiguration and SetConfiguration actions

- Added an action named StartScan that combines the Start and SetConfiguration actions to provide an atomic action to start a scan
- Fixed the Units value to milli-inches and deleted the SetUnits, GetUnits and GetLimits actions
- Added state variables for Resolution, A_ARG_TYPE_Height, A_ARG_TYPE_Width, A_ARG_TYPE_XOffset, and A_ARG_TYPE_YOffset
- Changed Boolean values to int values with three allowed values –1, 0, and 1 (-1 is use current value)
- Changed all ui4 values to int values and specified a range of –1 to maxint, where –1 means to use the current value
- Added a required value to all string variables "useCurrent" to indicate that the current value should be used.
- Changed the ErrorCode state variable to FailureCode to avoid the name collision with the UPnP ErrorCode values.

0.52 – Updated to reflect the changes made during the teleconference held on 22May2001. This includes the following:

- Use *device-setting* instead of *CurrentValue* for string type state variables
- Renamed A_ARG_TYPE_ID to JobID and changed to have a minimum value of 1
- Changed the names of the A_ARG_TYPE_Height, A_ARG_TYPE_Width, A_ARG_TYPE_XOffset, A_ARG_TYPE_YOffset variables to HeightLimit, WidthLimit, XOffsetLimit, and YOffsetLimit.
- Changed the FailureCode variable type from ui4 to String and changed the allowed values to strings from ui4 values. Enhanced the description of this variable.
- Changed the description of SideCount to have one special value (-1) that means scan all pages. Updated both the description and the XML.
- Updated StartScan and SetConfiguration descriptions to detail how the ImageWidth and ImageHeight values are *clipped* if they extend beyond the HeightLimit and WidthLimit values. and added arguments to return the actual values.
- Deleted the RestoreDefaultConfiguration action
- Deleted the GetDeviceID action and clarified the contents of the DeviceID variable to match the description in the BasicPrint service template.
- Described argument validation and associated error code in the StartScan action.
- Modified the State Transition Diagram to clarify behavior with respect to a Feeder.
- Added a State Transition Table for the Scanner State Transition Diagram (See Table 15 for more information).
- Changed ImageType to ImageFormat in the header of Table 42.
- Added embedImages variable to the SCPD. It was missing in all the actions and the state variable table.
- Clarified the Scan to Print scenarios to match the changes in the multipart MIME proposals and also to explain the limitations in using referenced images.
- Fixed page header and footer to include the standard content for UPnP template documents.
- Added Table 2 to describe the allowed Value List for the Name state variable.
- Added several example scan descriptions (Pull Scan w/Feeder, Push Scan w/Feeder, and Pull Scan w/o Feeder) in the *Theory of Operation*.
- Deleted A_ARG_TYPE_String – it is unused
- Added RegistrationID variable and deleted A_ARG_TYPE_UI4 (RegID was the only use of UI4).
- Changed the name of the *Name* variable to *JobName*.
- Added a table to explain the mapping of PixelType to data format.

0.54 – Updated to reflect the changes from the 21June2001 teleconference review. This includes the following changes:

- Renamed the NotReadyTimeout variable ErrorTimeout. Removed it from the parameter lists of the StartScan and SetConfiguration actions. Changed the description of this variable to explain that this timeout applies to the NotReady, Erred, and Finishing states, and that it is not control point accessible (i.e. a constant).

- Changed ImageNumber to SideNumber in Table 2.
- Updated the DeviceID description to change "Printer Vendor" references to "Scanner Vendor".
- Changed the state transition diagram and table to reflect a *One sheet per job* model for feederless scans.
- Issues:
  - Need to complete a PixelType to Data Format map
  - Need to answer questions about Color Map
  - Need to answer questions about embedded images versus referenced images.

0.70 – updated to reflect the changes from the 31July2001 Scanning Meeting in Toronto Canada. This includes the following changes:

- Reformat the description of state variables to merge the value tables with the variable descriptions
- Remove the description of *Multiple Image transfer using Multi-part MIME* (section 2.5.7.2), *Referenced Images Using Scanner Pull Model* (section 2.5.7.3), and the *embedImages variable*
- Add a path to the Scanner State Transition Diagram, and corresponding entries in the Scanner State Transition Table to add a loop in the Erred state for errors that require intervention.
- Add explanations for Push Scan and Pull Scan models
- Reordered the Scanner Flow examples and added an additional example (Feederless Push Scanning)
- Deprecated Error Code 404
- Added new common Error codes
- Changed from Error Code 710 – Invalid State to the 501 – Action Failed
- Corrected various typographical errors and cut-and-paste errors
- Modified the way that we specify the image characteristics from using PixelType to using ColorType and BitDepth

0.71 – Changes to reflect input from several committee members.

- Corrected references to the PixelType variable in Section 2.2.9 ColorSpace.
- Expanded the definition of the JobID variable in section 2.2.22.
- Removed the reference to the SetDefaultConfiguration action in the Effects on State section of the GetConfiguration action (section 2.4.6.2)
- Removed the allowedValueList from the DeviceID stateVariable definition in the SCPD. allowedValueList values must be strings of less than 31 characters.
- Added one scanner example to explain the simplest scan operation – a feeder-less, button-less Scan with Pull-Image Transfer
- Clarified other scanner examples to include the manipulations of the ExternalActivity service as needed.
- Only string type variables can have an allowedValueList. I changed the type of the BitDepth, UseFeeder, AppendSideNumber, and Resolution variables to string so that they can have allowed value lists. Issue: This will eliminate the ability to have Resolution Ranges. Do we want to do that?
- Changed the handling of the Destination variable as follows:
  - Removed Destination from the evented variable list
  - Added DestinationID variable (i4) as an evented variable
  - Added GetDestination action so that an authorized UCP can get the destination value.
  - Removed the Destination variable from the GetSideInformation action.
  - Changed the Scanner examples to reflect this change.
- Removed the "Multipart MIME Image Transfers" and "Referenced Images in XHTML/Print" sections since those transfer mechanisms were removed in v0.70.
- Updated Multipart MIME references to application/multiplexed references

0.72 – Changes based on input from several committee members. This includes the following:

- Changes Proposed Service Template to Preliminary Design on the first page
- Added a paragraph to describe the RegistrationID variable
- Changed the DestinationChanged paragraph to reflect the real variable name "DestinationID".
- Added GetDestination to the Action list in Table 23

- Changed "Invalid Sate (710)" to "Action Failed (501)" and "Invalid Parameter (404)" to "Invalid Arguments (402)" in several paragraphs that describe the effect of an action.
- Added changes to the Scanning state, On Entry description in Table 40 "Scanner State Transition Table"

0.73 – Changes based on input from committee member reviews

- Updated the reference to the Multipart-related document
- Fixed the FailureCode variable default value in Table 1
- Fixed the BitDepth variable default value in Table 1
- Fixed the SideCount variable default value in Table 1
- Fixed the ScanLength variable range and default value in Table 1
- Fixed various typographical errors
- Clarified the required values in the UseFeeder allowedValueList (paragraph 2.2.10)
- Changed GetSideInformation to GetDestination in the SCPD
- Deleted the A_ARG_TYPE_String variable in the SCPD – it is no longer used
- Changed the type of the DestinationID variable in the SCPD to ui4 to match Table 1
- Clarified the Sheet Size (paragraph 2.5.1) descriptions including:
  - Renamed the XOffsetLimit and YOffsetLimit state variables to be XValueLimit and YValueLimit respectively
  - Updated Table 38 and Figure 1

0.74 – Changes based on reviews

- AppendSideNumber Allowed Value in Table 1 – false is changed to 0
- ImageFormat allowedValueList in Table 4 – text/xhtml-print+xml is changed to application/vnd.pwg-xhtml-print+xml
- Added a column to Table 4 to describe the filename suffix for each image format value.
- Table 41 changed as follows:
  - row 2 and row 3 – added an explanation of the "*Image*" part of the Destination value
  - row 4 and row 5– the Description was changed to indicate that the image is *Posted* to the client's URL
- Removed the *maximumrate* attribute from the ScanLength stateVariable description in the SCPD. This attribute is not supported by the Certification Tool.

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

- Changed the allowed values of the BaseName variable from "buffer" to "pull-absolute" and "pull-relative" to enable a control point to specify whether the Destination URL should be relative to the BaseURL or absolute.
- Add an Error Code "*Invalid Image Specification*" (714) to indicate that the combination of the three variables is not supported to the StartScan and SetConfiguration actions
- The DeviceID variable had a type of 'String' in both Table 1 and in the SCPD. This was changed to 'string'
- Removed several lingering references to the NotReadyTimeout value which was removed previously.
- changed the description of what happens on a timeout (section 2.2.20) to read "transition to Finishing"
- Expanded the explanation of the RegistrationID variable (section 2.2.22) to include descriptions of the interation between the ExternalActivity and the Scanner services.
- Expanded the explanation of the RegID argument in the StartScan action description (section 2.4.1.2)
- Rewrote the "Relationship between the Scanner Service and the ExternalActivity Service" section (2.5.12) to reflect the use models discussed in LA.
- Rewrote the Testing section (4) to reflect the limited Syntax testing that we can actually do in v1.0.
- Added the Reserved state to reflect the interactions with the ExternalActivity service.

0.90 – Changes based on input from the Scanner PlugFest on April 2-3, 2002 in Irvine, CA.

- Changed the document status to *Design Complete*

- Made the JobName state variable required instead of optional since there are required arguments that use it as a relatedStateVariable. Also removed the allowedValueList table from the variable definition section.
- Changed the dataType of XValueLimit and YValueLimit in Table 1 to i4 so that it matches the SCPD definitions.
- Set the minimum value of XValueLimit, YValueLimit, HeightLimit and WidthLimit variables to -1 so that default values could be used in the Certification test scripts.
- Updated the ImageFormat description in section 2.2.5 to reflect the current application/vnd.pwg-xhtml-print+xml name format and the application/vnd.pwg-multiplexed transport format.
- Changed the minimum value for the SideNumber variable to 0 so that it can increment to 1 when the scanner enters the Scanning state.
- Added an *Errors* section for the GetDestination action.
- Clarified section 2.5.9 to better explain the methods used to send an image directly to a printer.
- In the SCPD XML content:
  - Corrected the xhtml allowed value of the ImageFormat state variable
  - Changed *buffer* to *pull-relative* in the BaseName variable default-value
  - Changed the mimimum and defaultValue for the SideNumber variable from 1 to 0
  - Changed the minimum of the HeightLimit, WidthLimit, ImageXOffset and ImageYOffset to -1.

0.91 – Changes needed to satisfy the UPnP Service Template Checklist and based on the 16Apr2002 teleconference:

- Changed the service name from Scanner to Scan
- added an overview description to section 1
- Changed all *nnn* value limits to *vendor-defined*
- Changed table numbers for the state variable descriptions to 1.x
- Added non-evented state variables to the Event list in Table 2
- Changed RegID argument name in StartScan action to RegistrationID
- Changed bUseFeeder argument name in StartScan action to UseFeeder
- Changed argument names in all actions by appending either “In” or “Out” to the name depending upon the direction of the argument in the paragraph header that lists the actions, the argument table, the action description and the SCPD XML description
- Changed the maximum allowedValue of the JobID variable from 0xffffffff to 4294967295
- Clarified the SideNumber Variable

0.92 – Changed to reflect minor changes suggested during the 9May2002 Teleconference

- Changed “somply” to “simply” in paragraph 2.2.14

1.0 – *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 and corrected the name from Scanner to Scan

2 Service Modeling Definitions

2.1 ServiceType

A service that is compliant with this template is identified with the following service type: **urn:schemas-upnp-org:service:***Scan:1*

2.2 State Variables

Table 1: State Variables

Variable Name	Req. or Opt.[1]	Data Type	Allowed Value[2]	Default Value[2]	Eng. Units
JobName	R	string	[device-setting | …]	""	
FailureCode	R	string	[No Error | Jammed | Timeout Reached | ErredTimeout Reached | Destination Not Reachable]	No Error	
State	R	string	[Idle | Reserved | NotReady | Pending | Scanning | Finishing | Erred]	Idle	
StateReason	R	string			
ImageFormat	R	string	[device-setting | image/jpeg]	image/jpeg	
CompressionFactor	R	i4	-1-100	100	n/a
ImageType	R	string	[device-setting | Mixed]	Mixed	
ColorType	R	string	[device-setting | Color | Mono]	Color	
BitDepth	R	string	[device-setting | 8]	8	bits
ColorSpace	R	string	[device-setting | sRGB]	sRGB	n/a
UseFeeder	R	string	[device-setting]	0 (false)	n/a
BaseName	R	string	[device-setting | buffer]	buffer	
AppendSideNumber	R	string	[device-setting|0]	0 (false)	n/a
SideCount	R	i4	[-1-1]	0	n/a
SideNumber	R	i4		1	n/a
Destination	R	string			
Timeout	R	i4	[-1 | 0]		seconds
ErrorTimeout	R	i4		vendor unique	seconds
Resolution	R	string	[device-setting]		Pixels Per Inch
ScanLength	R	i4	0 – vendor-defined	0	milli-inches

Variable Name	Req. or Opt.[1]	Data Type	Allowed Value[2]	Default Value[2]	Eng. Units
DeviceID	R	string			n/a
HeightLimit	R	i4			milli-inches
WidthLimit	R	i4			milli-inches
XValueLimit	R	i4	-1-vendor-defined	vendor-defined	milli-inches
YValueLimit	R	i4	-1-vendor-defined	vendor-defined	milli-inches
RegistrationID	R	ui4			
JobID	R	ui4	1 – vendor-defined		
DestinationID	R	ui4		0	

[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 JobName

This string allows the client to give the job a unique name. This is informational only. It can be used to identify the client or the purpose of the job. If the value is set to *device-setting* in the SetConfiguration or StartScan actions, then the value should not be changed.

2.2.2 FailureCode

A Failure Code indicates a failure, or error, that does not occur as an immediate result of an action and cannot be reported as part of the action response. Error Codes that occur during the execution of an action are not listed as allowed values of the FailureCode unless they can occur outside of the boundary of the action.

Table 1.1: allowedValueList for FailureCode

Value	Req. or Opt.
No Error	R
Jammed	R
Timeout Reached	R
ErredTimeout Reached	R
Destination not Reachable	R
… (Vendor specific error codes)	O

2.2.3 State

The current state of the scanner. The states are defined as follows:

- *Idle* – The scanner is not active

- *Reserved* – The scanner is reserved for use by a specific control point that has the correct RegistrationID. The RegistrationID value came from the ExternalActivity service.
- *NotReady* – The scanning is getting ready to scan.
- *Pending* - The scanner is currently processing a Scan job. It is waiting for client interaction (start action) or feeder interaction (loaded state) to complete
- *Scanning* – The scanner is currently scanning an image to the destination
- *Finishing* – The scanner has completed scanning all available sides and is waiting for the transfer of images to complete
- *Erred* – An error occurred during a scan operation. The scan service is waiting for the client to acknowledge the error by sending an Abort action.

2.2.4 StateReason

The StateReason string is used to augment the information given by the current state. This is especially helpful when in the NotReady state. In this case, the vendor should put information in this variable that tells the client what is going on (i.e. Calibrating, Warming Up, etc.).

2.2.5 ImageFormat

The format of the image to be returned. The list of allowed values may be a subset or superset of the given values. To send images to a printer, the application/vnd.pwg-xhtml-print format MUST be used with JPEG images embedded in an application/vnd.pwg-multiplexed document. See Using XHTML-Print To Send An Image Directly To A Printer on page 38 for more information about using the XHTML image type. If the value is set to *device-setting* in the SetConfiguration or StartScan actions, then the value should not be changed.

NOTE: the ImageFormat value "application/vnd.pwg-xhtml-print+xml" exceeds the recommended maximum length of 31 characters for elements of an allowedValueList. A truncated version (application/vnd.pwg-xhtml-print) is used so that it will fit within that length.

Table 1.2: allowedValueList for ImageFormat

Value	Req. or Opt.	Filename Suffix
device-setting	R	
image/jpeg	R	jpg
application/vnd.pwg-xhtml-print (Scan-to-Print)	O	xml
… (vendor defined)	O	

2.2.6 CompressionFactor

The measure of compression for compressed image formats. If the value is set to –1 (*device-setting)* in the SetConfiguration or StartScan actions, then the value should not be changed.

2.2.7 ImageType

The type of image to be scanned. If the value is set to *device-setting* in the SetConfiguration or StartScan actions, then the value should not be changed.

Table 1.3: allowedValueList for ImageType

Value	Req. or Opt.
device-setting	R
Mixed	R

Photo	O
Text	O
Graphics	O

2.2.8 Color Type and BitDepth

The type of the pixel used to represent the image is very dependent upon the ImageFormat. ColorType and BitDepth are used to describe the basic shape of the pixels. ColorType is used to describe whether the image is monochrome, color, or some sort of extended multi-plane pixel. The BitDepth value is used to describe the size of each part of a pixel. Since JPEG is the only required image format, the values for JPEG are given here. If vendors support additional image formats, then they should extend the allowed values to describe their formats (i.e. PNG-RGB-ALPHA, etc.).

Table 1.4: allowedValueList for ColorType

Value	Req. or Opt.
device-setting	R
Color	O
Mono	O

Note: Either *Color* or *Mono* should be given as an allowed value. Both are allowed, but at least one of them is required. The allowed value list can be extended or a subset of the list given here.

Table 1.5: allowedValueList for BitDepth

Value	Req. or Opt.
device-setting	R
8	R
12	O

The allowed value list may be extended as required.

2.2.8.1 Common Pixel Description Values

The following examples show how to describe most JPEG variations:

Table 1.6: Common Pixel Description Values

Description	ColorType	BitDepth
8-bit Color JPEG	Color	8
8-bit Gray	Mono	8
12-bit Color	Color	12
PNG RGB with Alpha	PNG-RGB-ALPHA	8

2.2.9 ColorSpace

The color space used to describe color images. This value is only valid if the *ColorType* value is *Color*. The allowed values may be a superset of the given values. If the value is set to *device-setting* in the SetConfiguration or StartScan actions, then the value should not be changed.

Table 1.7: allowedValueList for ColorSpace

Value	Req. or Opt.
device-setting	R
sRGB (use ITU-BT.601 specification)	R
YCC	O
… (vendor defined)	O

2.2.10 UseFeeder

Indicates where the sheet for the scan should come from. See Relationship between the Scan Service and the Feeder Service on page 39 for more information on using the feeder. If the value is set to *device-setting* in the SetConfiguration or StartScan actions, then the value should not be changed. Either true (1) or false (0), or both, must be in the allowed value list. It is up to the vendor to specify which they support.

Table 1.8: allowedValueList for UseFeeder

Value	Req. or Opt.
device-setting	R
0 (false)	O
1 (true)	O

2.2.11 BaseName

The name, or URI, for the scanned image files. The values "*pull-relative*" and "*pull-absolute*" are reserved to indicate that the image should be buffered and the client should pull the image from the device. In this case, the *Destination* variable will contain the Image URL, either relative to the BaseURL or absolute. See BaseName, SideNumber, AppendSideNumber and Destination on page 38 for more information. If the value is set to *device-setting* in the SetConfiguration or StartScan actions, then the value should not be changed.

Table 1.9: allowedValueList for BaseName

Value	Req. or Opt.
device-setting	R
pull-relative	R
pull-absolute	R
… (any valid URI)	O

2.2.12 AppendSideNumber

If this value is true, then the side number will be appended to the Basename value to create the destination name (i.e. the destination name will change for each side). See BaseName, SideNumber, AppendSideNumber and Destination on page 38 for more information. If the value is set to *device-setting* in the SetConfiguration or StartScan actions, then the value should not be changed.

Table 1.10: allowedValueList for AppendSideNumber

Value	Req. or Opt.
device-setting	R
0 (false)	R
1 (true)	O

2.2.13 SideCount

The total number of sides remaining to be scanned in the current *Start* or *StartScan* action. This value is set by the *StartScan* and *Start* actions and decremented with each side scanned. A value of –1 means to scan all available sheets. A value of 0 means to scan no pages.

Table 1.11: allowedValueRange for SideCount

Value		Req. or Opt.
Minimum	-1	R
Maximum	VendorUnique (>=1)	R
Step	1	R

2.2.14 SideNumber

The side-number for the current scanning session. The value is reset to 1 each time that the state enters *Idle*. This value is intended to indicate the capacity of a Feeder or other part of the scanner. It is not an error for the SideNumber to reach or exceed the maximum value. Instead, the value should simply wrap around to the minimum value. – Read-Only

Table 1.12: allowedValueList for SideNumber

Value		Req. or Opt.
Minimum	0	R
Maximum	VendorUnique	R
Step	1	R

2.2.15 Destination

The URL that the client should get. Otherwise, this is the name of the location where the device will put the image. This value is generated by the scanner service and cannot be set. See BaseName, SideNumber, AppendSideNumber and Destination on page 38 for more information.

2.2.16 Resolution

The resolution of the image in pixels per inch. The allowed values must be specified by the vendor.

Table 1.13: allowedValueList for Resolution

Value	Req. or Opt.
device-setting	R
Vendor Unique	R

2.2.17 ScanLength

The *ScanLength* variable indicates the total length scanned, in milli-inches, on the current page. This value will be reset to 0 at the beginning of each sheet and will increase as the scan progresses. This variable is intended to serve as a progress indicator. If the control point knows the length of the sheet, then it can indicate progress in percentages. If a vendor does not support this type if indication, then this value should not change.

Table 1.14: allowedValueRange for ScanLength

Value		Req. or Opt.
Minimum	0	R
Maximum	HeightLimit maximum value	R
Step	1	R

2.2.18 DeviceID

The value of this variable MUST exactly match the IEEE 1284-2000 Device ID string, except the length field MUST not be specified.. The value is assigned by the Scanner vendor and MUST NOT be localized by the Scan Service.

The IEEE 1284-2000 Device ID is a length field followed by a case-sensitive string of ASCII characters defining peripheral characteristics and/or capabilities. For the purposes of this specification, the length bytes MUST NOT be included. The Device ID sequence is composed of a series of keys and values of the form:

```
key: value {,value} repeated for each key
```

As indicated, each key will have one value, and MAY have more than one value. The minimum necessary keys (case-sensitive) are MANUFACTURER, COMMAND SET, and MODEL. (These keys MAY be abbreviated as MFG, CMD, and MDL respectively.) Each implementation MUST supply these three keys and possibly additional ones as well. Each key (and each value) is a string of characters. Any characters except colon (:), comma (,), and semi-colon (;) MAY be included as part of the key (or value) string. Any leading or trailing white space (SPACE[x'20'], TAB[x'09'], VTAB[x'0B'], CR[x'0D'], NL[x'0A'], or FF[x'0C']) in the string is ignored by the parsing program (but is still counted as part of the overall length of the sequence).

An example ID String, showing optional comment and active command set keys and their associated values (the text is actually all on one line):

```
MANUFACTURER:ACME Manufacturing;
COMMAND SET:XHTML-Print+xml;
MODEL:Scanner 9;
COMMENT:Anything you like;
ACTIVE COMMAND SET:JPEG;
```

(See IEEE 1284-2000 clause 7.6)

Note: One of the purposes of the DeviceId variable is to select a scanner driver for those UCPs that need a scanner driver. The values of the COMMAND SET key are interpreted by the scanner driver provided by the vendor and so are vendor-defined, rather than being standardized.

2.2.19 HeightLimit, WidthLimit, XValueLimit, YValueLimit

The HeightLimit, WidthLimit, XValueLimit, and YValueLimit values are placeholders. Their values should be ignored. The allowedValueRanges in the *Service Control Protocol Document* (SCPD) are the values of interest. Those values should be used to set limits for the ImageXOffset, ImageYOffset, ImageHeight, and ImageWidth parameters in the SetConfiguration action.

These values are simply placeholders, or constant values. The intent is to specify the limits of these values in the SCPD. The actual values of these variables are not important and should be ignored.

Table 1.15: allowedValueRange for HeightLimit

Value		Req. or Opt.
Minimum	-1	R
Maximum	VendorUnique	R
Step	1	R

Table 1.16: allowedValueRange for WidthLimit

Value		Req. or Opt.
Minimum	-1	R
Maximum	VendorUnique	R
Step	1	R

Table 1.17: allowedValueRange for XValueLimit

Value		Req. or Opt.
Minimum	-1	R
Maximum	VendorUnique	R
Step	1	R

Table 1.18: allowedValueRange for YValueLimit

Value		Req. or Opt.
Minimum	-1	R
Maximum	VendorUnique	R
Step	1	R

2.2.20 Timeout

The inner-side timeout. If SideCount is zero, then the scanner will wait *Timeout* seconds in the Pending state. If the timeout expires, then the state will transition to Finishing. If the value is 0, then timeouts are disabled. If a parameter has a value of –1, then the device should not change the value of the timeout.

Table 1.19: allowedValueRange for Timeout

Value		Req. or Opt.
Minimum	-1	R
Maximum	VendorUnique	R
Step	1	R
Default	Maximum	O

2.2.21 ErrorTimeout

The timeout used to exit the NotReady, Erred, and Finishing states. The scanner will wait *ErrorTimeout* seconds in any of these states. If the timeout expires, then the state will change from NotReady or Finishing to Erred, or from Erred to Idle. The value of this timeout is vendor unique and cannot be changed by the control point.

2.2.22 RegistrationID

The RegistrationID is a placeholder variable that is used to describe the RegistrationID argument from the StartScan action. This value is used to control the interaction between the Scanner service and the ExternalActivity service. If the current RegistrationID value is 0, then any value should be accepted and

sent to the ExternalActivity service. If the current RegistrationID value is not 0, then only that value can be accepted. Any other value will cause an error. The RegistrationID value can be used to control the interactions in two ways.

- ExternalActivity Initiated Operation – The user presses a button associated with the scanner and the ExternalActivity service sets the required RegistrationID value using an unspecified internal mechanism. This reserves the scanner for use by a control point that can supply the required RegistrationID value. The reservation will last until a StartScan action is received with the required RegistrationID value or until a Timeout occurs. If a timeout occurs, then the required RegistrationID value will be set back to 0.

- Control Point Initiated Operation – The user indicates that it wishes to perform a scan. The control point sends a *StartScan* operation with a valid RegistrationID value and typically with UseFeeder = false and SideCount = 0 (the scanner service will wait in Pending state). The scanner state changes as indicated in Figure 2. In addition, the Scanner service forwards the RegistrationID value to the ExternalActivity service using an unspecified internal mechanism. The ExternalActivity service is expected to indicate the registration values associated with the ID as far as it is capable. In this case, the user would walk to the scanner, place the document in the feeder, and press the appropriate button. The control point would see the activity and send a *Start* action to allow the scanner to continue its operation. For more information on the ExternalActivity service behavior, see the ExternalActivity Service Template document.

2.2.23 JobID

A JobID value is unique for the duration of the current job. It is used as an argument to all actions that change the state of the scanner to prevent accidental interference by a third party. The initial value is returned from the *StartScan* action and must be given in most action argument lists.

2.2.24 DestinationID

The Destination variable is used to indicate where an image should be pulled from or sent to. To prevent all UCPs from knowing the Destination value, the DestinationID value is used to indicate that the Destination value has been changed. This evented value is incremented each time that the Destination variable value is set to a new value. The UCP should use the GetDestination action to retrieve the Destination value as needed.

2.3 Eventing and Moderation

Table 2: Event Moderation

Variable Name	Evented	Moderated Event?	Max Event Rate	Logical Combination	Min Delta per Event
			Determined by N Where Rate = (Event)/(N secs)		(N) * (allowed ValueRange Step)
JobName	No				
FailureCode	Yes	No			
State	Yes	No			
StateReason	No				
ImageFormat	No				
CompressionFactor	No				
Imagetype	No				
ColorType	No				
BitDepth	No				
ColorSpace	No				
UseFeeder	No				
BaseName	No				
AppendSideNumber	No				
SideCount	No				
SideNumber	Yes	No			
Destination	No				
Timeout	No				
ErrorTimeout	No				
Resolution	No				
ScanLength	Yes	Yes	1		
DeviceID	No				
HeightLimit	No				
WidthLimit	No				
XValueLimit	No				
YValueLimit	No				
RegistrationID	No				
JobID	No				
DestinationID	Yes	No			

See section 4.4 of the UPnP Architecture document for details regarding event moderation.

2.4 Action Set

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.
StartScan	R
Start	R
Stop	R
Abort	R
SetConfiguration	R
GetConfiguration	R
GetSideInformation	R
GetDestination	R
GetState	R

2.4.1 (Void) StartScan(RegistrationIDIn, UseFeederIn, SideCountIn, JobNameIn, ResolutionIn, ImageXOffsetIn, ImageYOffsetIn, ImageWidthIn, ImageHeightIn, ImageFormatIn, CompressionFactorIn, ImageTypeIn, ColorTypeIn, BitDepthIn, ColorSpaceIn, BaseNameIn, AppendSideNumberIn, TimeoutIn, ActualTimeoutOut, JobIDOut, ActualWidthOut, ActualHeightOut)

2.4.1.1 Arguments

Table 4: Arguments for StartScan

Argument	Direction	RelatedStateVariable
RegistrationIDIn	IN	RegistrationID
UseFeederIn	IN	UseFeeder
SideCountIn	IN	SideCount
JobNameIn	IN	JobName
ResolutionIn	IN	Resolution
ImageXOffsetIn	IN	XValueLimit
ImageYOffsetIn	IN	YValueLimit
ImageWidthIn	IN	WidthLimit
ImageHeightIn	IN	HeightLimit
ImageFormatIn	IN	ImageFormat
CompressionFactorIn	IN	CompressionFactor
ImageTypeIn	IN	ImageType
ColorTypeIn	IN	ColorType
BitDepthIn	IN	BitDepth
ColorSpaceIn	IN	ColorSpace
BaseNameIn	IN	BaseName
AppendSideNumberIn	IN	AppendSideNumber

TimeoutIn	IN	Timeout
ActualTimeoutOut	OUT	Timeout
JobIDOut	OUT	JobID
ActualWidthOut	OUT	WidthLimit
ActualHeightOut	OUT	HeightLimit

2.4.1.2 Effect of Action on State

The RegistrationIDIn value is used to control the interaction between the Scan service and the ExternalActivity service. If the current RegistrationIDIn value is 0, then any valid value should be accepted and sent to the ExternalActivity service. If the current RegistrationIDIn value is not 0, then only that value can be accepted. Any other value must cause an error. The RegistrationIDIn value can be used to control the interactions in two ways.

- ExternalActivity Initiated Operation – The user presses a button associated with the scanner and the ExternalActivity service sets the required RegistrationIDIn value using an unspecified internal mechanism. This reserves the scanner for use by a control point that can supply the required RegistrationIDIn value. The reservation will last until a StartScan action is received with the required RegistrationIDIn value or until a Timeout occurs. If a timeout occurs, then the required RegistrationIDIn value will be set back to 0.

- Control Point Initiated Operation – The user indicates that it wishes to perform a scan. The control point sends a *StartScan* operation with a valid RegistrationIDIn value and typically with UseFeeder = false and SideCount = 0 (the scanner service will wait in Pending state). The scanner state changes as indicated in Figure 2. In addition, the Scan service forwards the RegistrationIDIn value to the ExternalActivity service using an unspecified internal mechanism. The ExternalActivity service is expected to indicate the registration values associated with the ID as far as it is capable. In this case, the user would walk to the scanner, place the document in the feeder, and press the appropriate button. The control point would see the activity and send a *Start* action to allow the scanner to continue its operation. For more information on the ExternalActivity service behavior, see the ExternalActivity Service Template document.

If the RegistrationIDIn value is valid, then the configurations values are set as given and the scan is started *(See section 2.5.12 on page 39 for more information about the RegID).* This action can only be executed while the scanner is in *Idle* state. If the state is not Idle, then the action will be ignored and an *Action Failed (501)* error will be returned. All state variables will be validated before any are changed. If any variable is invalid, then no variables will be changed, and an *Invalid Arguments (402)* error will be returned. If the RegistrationIDIn value is invalid, then the action will be ignored and an *Invalid_ID* error will be returned. See Table 16: Current State vs Service Actions on page 37for details on the results of this action.

NOTE: The resulting values of ImageXOffsetIn, ImageYOffsetIn, ImageWidthIn and ImageHeightIn must combine to fit within the WidthLimit and HeightLimit limits. If the given values do not fit within these limits, then they will be limited to fit.

The value of ImageXOffsetIn + ImageWidthIn must be less than the maximum WidthLimit value. If it is not, then the ImageWidthIn value will be limited, or clipped. The actual value will be returned in ActualWidthOut argument. The value of ImageYOffsetIn + ImageHeightIn must also be less than the maximum HeightLimit value. If it is not, then the ImageHeightIn value will be limited. The actual ImageHeight value is returned in the ActualHeightOut argument.

The JobIDOut value returned must be used in all subsequent actions that change the State of the Scan Service. The algorithm used to generate a unique JobIDOut value is left to the vendor. A simple incrementing algorithm is not recommended because it is too predictable and easy to defeat.

The TimeoutIn value is only advisory. The Scan Service may change the value to an appropriate alternative. The actual timeout value is returned in the ActualTimeoutOut parameter.

The values of the ImageFormatIn, ColorTypeIn, and BitDepthIn are related. Not all file formats will support the same bit depths and color types. If the combination of these three variables are not supported, then an "*Invalid Image Specification*" (714) error will be returned.

NOTE: The scanner will automatically set the configuration to default values when the state returns to *Idle*.

2.4.1.3 Errors

errorCode	errorDescription	Description
501	Action Failed	The Start action can only be performed in the Idle state.
711	Jammed	A sheet has jammed in the scanner or feeder (if present) Clear the paper path and execute the Stop action.
712	Invalid_ID	The given ID value is invalid
714	Invalid Image Specification	The combination of ImageFormat, ColorType and BitDepth are not supported.

2.4.2 (Void) Start*(*JobIDIn, UseFeederIn, SideCountIn*)*

2.4.2.1 Arguments

Table 5: Arguments for Start

Argument	Direction	RelatedStateVariable
JobIDIn	IN	JobID
UseFeederIn	IN	UseFeeder
SideCountIn	IN	SideCount

2.4.2.2 Effect of Action on State

2.4.2.3 ***The UseFeeder, and SideCount variables are set. Initiate the scan of a* SideCount *sides. A SideCount value of 0 means to scan all available pages. See Table 16: Current State vs Service Actions on page 37 for details on the results of this action. This action must only be executed while the scanner is in the* Pending *state. If the state is not in* Pending *state, then the action will be ignored and an* Action Failed (501) *error will be returned. If the JobIDIn value is invalid, then the action will be ignored and an* Invalid_ID *error will be returned.Errors***

errorCode	errorDescription	Description
501	Action Failed	The Start action can only be performed in the Pending state.
711	Jammed	A sheet has jammed in the scanner or feeder (if present) Clear the paper path and execute the Stop action.
712	Invalid_ID	The given ID value is invalid

2.4.3 (Void) Stop(JobIDIn)

2.4.3.1 Arguments

Table 6: Arguments for Stop

Argument	Direction	RelatedStateVariable
JobIDIn	IN	JobID

2.4.3.2 Effect on State

This is used to stop a pending scan operation. Any complete sides that have not been transferred will be transferred before the state returns to Idle. The state will be set to *Finishing*. See Table 16: Current State vs Service Actions on page 37 for more details on this action. This action must only be executed while the scanner is not in the *Erred* state. If the state is *Erred*, then the action will be ignored and an *Action Failed (501)* error will be returned. If the JobIDIn value is invalid, then the action will be ignored and an *Invalid_ID* error will be returned.

2.4.3.3 Errors

errorCode	errorDescription	Description
501	Action Failed	The Stop action cannot be performed in the erred state.
711	Jammed	A sheet has jammed in the scanner or feeder (if present). Clear the paper path and execute the Stop action.
712	Invalid_ID	The given ID value is invalid

2.4.4 (Void) Abort(JobIDIn)

2.4.4.1 Arguments

Table 7: Arguments for Abort

Argument	Direction	RelatedStateVariable
JobIDIn	IN	JobID

2.4.4.2 Effect on State

This is used to abort a pending scan operation. The value of State will be set to Idle. All pending data will be lost. If the *UseFeeder* variable is true (1) then the current page will be ejected, otherwise no page ejection will take place. See Table 16: Current State vs Service Actions on page 37 for more details on this action. If the JobIDIn value is invalid, then the action will be ignored and an *Invalid_ID* error will be returned.

2.4.4.3 Errors

errorCode	errorDescription	Description
712	Invalid_ID	The given ID value is invalid

2.4.5 (Void) *Set*Configuration(JobIDIn, JobNameIn, ResolutionIn, ImageXOffsetIn, ImageYOffsetIn, ImageWidthIn, ImageHeightIn, ImageFormatIn, CompressionFactorIn, ImageTypeIn, ColorTypeIn,

BitDepthIn, ColorSpaceIn, BaseNameIn, AppendSideNumberIn, TimeoutIn, ActualTimeoutOut, ActualWidthOut, ActualHeightOut)

2.4.5.1 Arguments

Table 8: Arguments for SetConfiguration

Action Specification	Direction	RelatedStateVariable
JobIDIn	IN	JobID
JobNameIn	IN	JobName
ResolutionIn	IN	Resolution
ImageXOffsetIn	IN	XValueLimit
ImageYOffsetIn	IN	YValueLimit
ImageWidthIn	IN	WidthLimit
ImageHeightIn	IN	HeightLimit
ImageFormatIn	IN	ImageFormat
CompressionFactorIn	IN	CompressionFactor
ImageTypeIn	IN	ImageType
ColorTypeIn	IN	ColorType
BitDepthIn	IN	BitDepth
ColorSpaceIn	IN	ColorSpace
BaseNameIn	IN	BaseName
AppendSideNumberIn	IN	AppendSideNumber
TimeoutIn	IN	Timeout
ActualTimeoutOut	OUT	Timeout
ActualWidthOut	OUT	WidthLimit
ActualHeightOut	OUT	HeightLimit

2.4.5.2 Effect on State

The SetConfiguration action must only be executed when the state value is Pending. If the state is not Pending, then no variables will be changed and an *Action Failed (501)* error will be returned. All state variables will be validated before any are changed. If any variable is invalid, then no variables will be changed, and an *Invalid Arguments (402)* error will be returned. If the JobIDIn value is invalid, then the action will be ignored and an *Invalid_ID* error will be returned. The state variables will be set as requested if the state is acceptable and the values are all valid. The order of validation of variables is vendor specific. Table 16: Current State vs Service Actions on page 37 for more details on this action.

The value of ImageXOffsetIn + ImageWidthIn must be less than the maximum WidthLimit value. If it is not, then the ImageWidthIn value will be limited, or clipped. The actual value will be returned in ActualWidthOut argument. The value of ImageYOffsetIn + ImageHeightIn must also be less than the maximum HeightLimit value. If it is not, then the ImageHeightIn value will be limited. The actual ImageHeight value is returned in the ActualHeightOut argument.

The TimeoutIn value is only advisory. The Scan Service may change the value to an appropriate alternative. The actual timeout value is returned in the ActualTimeoutOut parameter.

The values of the ImageFormatIn, ColorTypeIn, and BitDepthIn are related. Not all file formats will support the same bit depths and color types. If the combination of these three variables are not supported, then an "*Invalid Image Specification*" (714) error will be returned.

NOTE: The scanner will automatically set the configuration to default values when the state returns to *Idle*.

2.4.5.3 Errors

errorCode	errorDescription	Description
501	Action Failed	The SetConfiguration action must only be performed in the Pending state.
712	Invalid_ID	The given ID value is invalid
714	Invalid Image Specification	The combination of ImageFormat, ColorType and BitDepth are not supported.

2.4.6 (Void) GetConfiguration*(*JobNameOut, ResolutionOut, ImageXOffsetOut, ImageYOffsetOut, ImageWidthOut, ImageHeightOut, ImageFormatOut, CompressionFactorOut, ImageTypeOut, ColorTypeOut, BitDepthOut, ColorSpaceOut, BaseNameOut, AppendSideNumberOut, TimeoutOut*)*

2.4.6.1 Arguments

Table 9: Arguments for GetConfiguration

Argument	Direction	relatedStateVariable
JobNameOut	OUT	JobName
ResolutionOut	OUT	Resolution
ImageXOffsetOut	OUT	XValueLimit
ImageYOffsetOut	OUT	YValueLimit
ImageWidthOut	OUT	WidthLimit
ImageHeightOut	OUT	HeightLimit
ImageFormatOut	OUT	ImageFormat
CompressionFactorOut	OUT	CompressionFactor
ImageTypeOut	OUT	ImageType
ColorTypeOut	OUT	ColorType
BitDepthOut	OUT	BitDepth
ColorSpaceOut	OUT	ColorSpace
BaseNameOut	OUT	BaseName
AppendSideNumberOut	OUT	AppendSideNumber
TimeoutOut	OUT	Timeout

2.4.6.2 Effects on State

None.

This action returns the current values of all of the state variables for scanning.

2.4.7 (Void) GetSideInformation(SideNumberOut, SideCountOut, ScanLengthOut)

2.4.7.1 Arguments

Table 10: Arguments for GetSideInformation

Action Specification	Direction	relatedStateVariable
SideNumberOut	OUT	SideNumber
SideCountOut	OUT	SideCount
ScanLengthOut	OUT	ScanLength

2.4.7.2 Effect on State

None

2.4.8 (Void) GetDestination(JobIDIn, DestinationOut, DestinationIDOut)

2.4.8.1 Arguments

Table 11: Arguments for GetDestination

Action Specification	Direction	relatedStateVariable
JobIDIn	IN	JobID
DestinationOut	OUT	Destination
DestinationIDOut	OUT	DestinationID

2.4.8.2 Effect on State

None

The value of the *Destination* variable is generated based on the values of *BaseName*, *AppendSideNumber*, and *SideCount*. See BaseName, SideNumber, AppendSideNumber and Destination on page 38 for more details. The DestinationID value is incremented each time that Destination is changed.

2.4.8.3 Errors

errorCode	errorDescription	Description
712	Invalid_ID	The given ID value is invalid

2.4.9 (Void) GetState(StateOut, StateReasonOut, FailureCodeOut)

2.4.9.1 Arguments

Table 12: Arguments for GetState

Action Specification		RelatedStateVariable
StateOut	OUT	State
StateReasonOut	OUT	StateReason
FailureCodeOut	OUT	FailureCode

2.4.9.2 Effect on State

None

2.4.10 Common Error Codes

Table 13: 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.

For more information regarding Common Error Codes, see section 3.2 of the UPnP Architecture Document.

2.5 Theory of Operation

2.5.1 Sheet Size and Image Area

The sheet size is defined by the limits of the XValueLimit, YValueLimit, HeightLimit and WidthLimit state variables. These variables are placeholders that define the minimum and maximum values. It is assumed that the scanner does not have the capability to sense the sheet size, so no variables are provided to show that information. The *image area* is defined by the origin of an image and its size. Table 14: Sheet and Image Area Dimensions shows the relationships of the X and Y offset values and the width and height of the image. The image area must fit within the limits of the sheet size. If it does not, then the image area will be clipped to fit.

Table 14: Sheet and Image Area Dimensions

Value Name	Relationships	Description
MaxXValue	XValueLimit	The maximum value of the XValueLimit variable
MinXValue	XValueLimit	The minimum value of the XValueLImit variable
MaxYValue	YValueLimit	The maximum value of the YValueLimit variable
MinYValue	YValueLImit	The minimum value of the YValueLimit variable
MaxImageWidth	WidthLimit	The maximum value of the WidthLimit variable.
MinImageWidth	WidthLimit	The minimum value of the WidthLimit variable.
MaxImageHeight	HeightLimit	The maximum value of the HeightLimit variable.
MinImageHeight	HeightLimit	The minimum value of the HeightLimit variable
ImageXOffset	MinXValue	Defines the X offset of the leading-left corner of the image area shown in Figure 1. This value must meet the following constraints: MinXValue <= ImageXOffset
ImageYOffset	MinYValue	Defines the Y offset of the leading-left corner of the image area shown in Figure 1. This value must meet the following constraints: MinYValue <= ImageYOffset
ImageWidth	MaxImageWidth, MinImageWidth, MaxXValue, ImageXOffset	Defines the width of the image area shown in Figure 1. This value must meet the following constraints: ImageWidth >= MinImageWidth ImageWidth <= MaxImageWidth ImageWidth <= MaxXValue - ImageXOffset
ImageHeight	MaxImageHeight, MinImageHeight, MaxYValue, ImageYOffset	Defines the height of the image area shown in Figure 1. This value must meet the following constraints: ImageHeight >= MinImageHeight ImageHeight <= MaxImageHeight ImageHeight <= MaxYValue – ImageYOffset

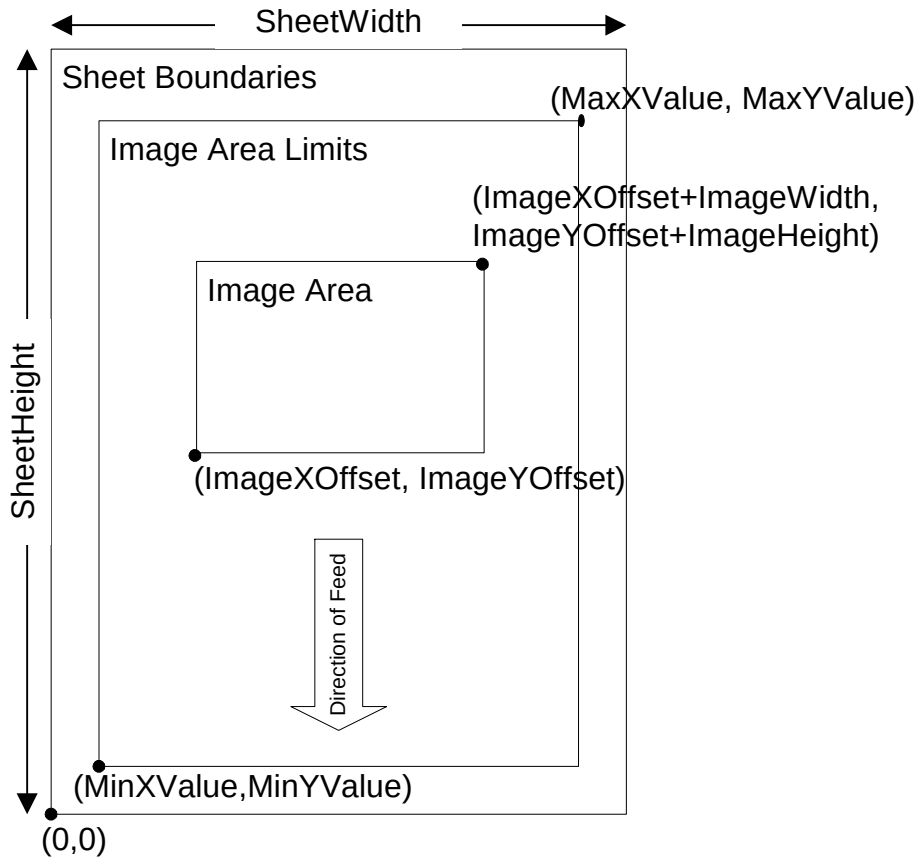

Figure 1: Sheet and Image Areas Illustrated

2.5.2 Flow Example: Feeder-less, Button-less Scan Operation with Pull Image Transfer

The scenario represented in this section involves a scanner without a feeder that is used by a UCP that will pull the images when they are available. The state transition diagram shown in Figure 2 below describes the operation of the Scan Service. A client should follow the flow of events below to perform a scan:

1) The UCP should subscribe with the Scan Service for events.

2) The client starts a scanner operation using the StartScan action with bUseFeeder=false (0), SideCount=1, and BaseName="buffer". All of the parameters will be validated before any settings are changed. If all are valid, then the values are set and the scan operation begins. The order of validation is vendor specific. The state value will change from Idle to Pending (possibly going through the NotReady state in the process).

3) Once the scanner is in Pending state, it will immediately transition to *Scanning* state because UseFeeder is false(0) and SideCount=1.

4) The scanner will generate the *Destination* value and increment the DestinationID value (which will cause an event) and start scanning the first sheet. The UCP should perform a GetDestination action to get the URL for the image and an HTTP/GET operation from the URL to transfer the image. The transfer will probably be chunked (HTTP v1.1). When the sheet is done, the scanner will decrement SideCount, increment SideNumber and transition to Pending because SideCount==0.

5) The UCP should send a Stop action to complete the scan. This will cause the scanner to transition to Finishing.

6) When the state gets to *Finishing*, it will complete any pending transfers and then transition to *Idle*.

7) If the state gets to *Erred*, then the UCP must execute the *Abort* action to clear the error and get back to *Idle* state.

8) When the state enters the *Idle* state, all configurations settings will be set to their default values.

2.5.3 Flow Example: Feeder-less Scan Operation with Pull Image Transfer

The scenario represented in this section involves a scanner without a feeder that is used by a UCP that will pull the images when they are available. The state transition diagram shown in Figure 2 below describes the operation of the Scan Service. A client should follow the flow of events below to perform a scan:

1) The UCP should subscribe with the Scanner and ExternalActivity services for events.

2) The client starts a scanner operation using the StartScan action with bUseFeeder=false (0), SideCount=0, and BaseName="buffer". All of the parameters will be validated before any settings are changed. If all are valid, then the values are set and the scan operation begins. The order of validation is vendor specific. The state value will change from Idle to Pending (possibly going through the NotReady state in the process).

3) Once the scanner is in Pending state, the user should place a document in the scanner and press the button. The button press will be evented to the UCP by the ExternalActivity service. The UCP should scan one sheet using a Start action with bUseFeeder=false(0) and SideCount=1. Since UseFeeder is false (0) and SideCount=1, the scanner will transition to Scanning state.

4) The scanner will generate the *Destination* value and increment the DestinationID value (which will cause an event) and start scanning the first sheet. The UCP should perform a GetDestination action to get the URL for the image and an HTTP/GET operation from the URL to tranfer the image. The transfer will probably be chunked (HTTP v1.1). When the sheet is done, the scanner will decrement SideCount, increment SideNumber and transition to Pending because SideCount==0.

5) If there are more documents to scan, then repeat steps 3 and 4 for each sheet. When there are no more sheets to scan, the UCP should send a Stop action. This will cause the scanner to transition to Finishing.

6) When the state gets to *Finishing*, it will complete any pending transfers and then transition to *Idle*.

7) If the state gets to *Erred*, then the UCP must execute the *Abort* action to clear the error and get back to *Idle* state.

8) When the state enters the *Idle* state, all configurations settings will be set to their default values.

2.5.4 Flow Example: Feeder-less Scan Operation with Push Image Transfer

The scenario represented in this section involves a scanner without a feeder that is used by a UCP that will push the images to a given destination when they are available. The state transition diagram shown in Figure 2 below describes the operation of the Scan Service. A client should follow the flow of events below to perform a scan:

1) The UCP should subscribe with the Scanner and ExternalActivity services for events.

2) The client starts a scanner operation using the StartScan action with bUseFeeder=false (0), SideCount=0, and BaseName="machine:port/path" (i.e. a URL), AppendSheetNumber=true (1). All

of the parameters will be validated before any settings are changed. If all are valid, then the values are set and the scan operation begins. The order of validation is vendor specific. The state value will change from Idle to Pending (possibly going through the NotReady state in the process).

3) Once the scanner is in Pending state, the user should place a document in the scanner and press the button. The button press will be evented to the UCP by the ExternalActivity service. The UCP should scan one sheet using a Start action with bUseFeeder=false(0) and SideCount=1. Since UseFeeder is false (0) and SideCount=1, the scanner will transition to Scanning state.

4) The scanner will generate the *Destination* value and increment the DestinationID value (which will cause an event) and start scanning the first sheet. The destination given by the UCP should be prepared to receive an image. The scanner will perform an HTTP/POST operation to the URL given in the *Destination* variable. The transfer will probably be chunked (HTTP v1.1). When the sheet is done, the scanner will decrement SideCount, increment SideNumber and transition to Pending because SideCount==0.

5) If there are more documents to scan, then repeat steps 3 and 4 for each sheet. When there are no more sheets to scan, the UCP should send a Stop action. This will cause the scanner to transition to Finishing.

6) When the state gets to *Finishing*, it will complete any pending transfers and then transition to *Idle*.

7) If the state gets to *Erred*, then the UCP must execute the *Abort* action to clear the error and get back to *Idle* state.

8) When the state enters the *Idle* state, all configurations settings will be set to their default values.

2.5.5 Flow Example: Scan Operation with Feeder and Pull Image Transfer

The Scan Service is used to operate a scanner device. A scanner device is used to convert a user-supplied document into a digital image. The following example shows a scenario where a scanner with a feeder is used to scan multiple pages. The UCP initiates the scan with UseFeeder set to true (1) and pulls all of the image data from the scanner.The state transition diagram shown in Figure 2 below describes the basic operation of the Scan Service. A client should follow the flow of events below to perform a pull scan:

1) The UCP should subscribe with the Scan Service for events.

2) Assuming that there are documents in the feeder, the client starts a scanner operation using the StartScan action with bUseFeeder=true (1), SideCount > 0, BaseName="buffer". All of the parameters will be validated before any settings are changed. If all are valid, then the values are set and the scan operation begins. The order of validation is vendor specific. The state value will change from Idle to Pending (possibly going through the NotReady state in the process).

3) Once the scanner is in Pending state, it will immediately transition to *Scanning* state and execute Feeder.Load() because UseFeeder is true, Feeder.MorePages is true, and SideCount is non-zero.

4) The scanner will generate the *Destination* value and increment the DestinationID value (which will cause an event) and start scanning the first sheet. The UCP should execute the GetDestination action to get the Destination URL value and an HTTP/GET operation on the URL to transfer the image. The transfer will probably be chunked (HTTP v1.1). When the sheet is done, the scanner will decrement SideCount, increment SideNumber, and do one of the following:
   a) remain in Scanning state (i.e. Repeat Step 4) if Feeder.MorePages is true (1), and SideCount != 0
   b) OR transition to Pending state if:
      i) Feeder.MorePages is false (0)
      ii) OR SideCount == 0

5) When the state transitions to Pending, the scanner will:

a) transition immediately to *Finishing* if Feeder.MorePages is false(0)
b) wait in *Pending* state if Feeder.MorePages is true(1) until:
   i) a Start action is received that sets SideCount > 0 → transition to Scanning
   ii) a Stop action is received → transition to Finishing
   iii) the timeout elapses → transition to Finishing

6) When the state gets to *Finishing*, it will complete any pending transfers and then transition to *Idle*.

7) If the state gets to *Erred*, then the UCP must execute the *Abort* action to clear the error and get back to *Idle* state.

8) When the state enters the *Idle* state, all configurations settings will be set to their default values.

2.5.6 Flow Example: Scan Operation with Feeder and Push Image Transfer

The scenario represented in this section involves a scanner with a feeder that is used by a UCP that wants the images sent when they are available. The state transition diagram shown in Figure 2 below describes the operation of the Scan Service. A client should follow the flow of events below to perform a scan:

1) The UCP should subscribe with the Scan Service for events.

2) The client starts a scanner operation using the StartScan action with bUseFeeder=true (1), SideCount > 0, and BaseName="machine:port/path" (i.e. a URL), AppendSheetNumber=true(1). All of the parameters will be validated before any settings are changed. If all are valid, then the values are set and the scan operation begins. The order of validation is vendor specific. The state value will change from Idle to Pending (possibly going through the NotReady state in the process).

3) Once the scanner is in Pending state, it will immediately transition to *Scanning* state and execute Feeder.Load() because UseFeeder is true, Feeder.MorePages is true, and SideCount is non-zero.

4) The scanner will generate the *Destination* value and increment the DestinationID value (which will cause an event) and start scanning the first sheet. The UCP should be prepared to receive an image. The scanner will perform an HTTP/POST operation to the URL given in the *Destination* variable. The transfer will probably be chunked (HTTP v1.1). When the sheet is done, the scanner will decrement SideCount, increment SideNumber and do one of the following:
   a) remain in Scanning state (i.e. Repeat Step 4) if Feeder.MorePages is true (1), and SideCount != 0
   b) OR transition to Pending state if:
      i) Feeder.MorePages is false (0)
      ii) OR SideCount == 0

5) When the state transitions to Pending, the scanner will:
   a) transition immediately to *Finishing* if Feeder.MorePages is false(0)
   b) wait in *Pending* state if Feeder.MorePages is true(1) until:
      i) a Start action is received that sets SideCount > 0 → transition to Scanning
      ii) a Stop action is received → transition to Finishing
      iii) the timeout elapses → transition to Finishing

6) When the state gets to *Finishing*, it will complete any pending transfers and then transition to *Idle*.

7) If the state gets to *Erred*, then the UCP must execute the *Abort* action to clear the error and get back to *Idle* state.

8) When the state enters the *Idle* state, all configurations settings will be set to their default values.

2.5.7 Scanner Timeouts and Feeder Interactions

There are two defined timeouts for scanning operations. These are:

1. NotReady state Timeout (ErredTimeout variable) – the maximum time that the scanner can spend in the NotReady state.
2. General Timeout (Timeout variable) – the maximum time that the scanner can spend in the Scanning state and/or the Pending state without leaving the Pending state (i.e. starting a new sheet). The general timeout also limits the lifetime of a RegID value while the state is *Idle*.

2.5.7.1 Error Timeout

The error timeout occurs when the scanner spends too much time (more than ErrorTimeout seconds) in the NotReady, Erred, or Finishing state. When this occurs, the FailureCode variable will be set to Error Timeout Reached and the state will be set from NotReady or Finishing to *Erred or from Erred to Idle*.

2.5.7.2 General Timeout

The general timeout value is applied to both a *Registration Timeout* and an *Inner-Page Timeout*. The *Registration Timeout* occurs when an external entity has set the Registration ID value, and the client does not execute a valid *StartScan* action before *Timeout* seconds elapse. An *Inner-Page Timeout* occurs when *Timeout* seconds have elapsed while the scanner has been in any single state except for *Idle* or *Erred*. This can occur for several reasons, including:

- The scanner was unable to transfer the image data to the client (push model)
- The client did not retrieve the image data (pull model)
- The client did not start a scan on the next available sheet

In the first two cases, any remaining data will be flushed from the buffer, the scan operation (if currently active) will be aborted, if *UseFeeder* is true then the sheet will be ejected, the error code will be set to Timeout Reached (720), and the state will be set to *Erred*. In the third case, the state will be set to *Finishing* which will end the scan operation.

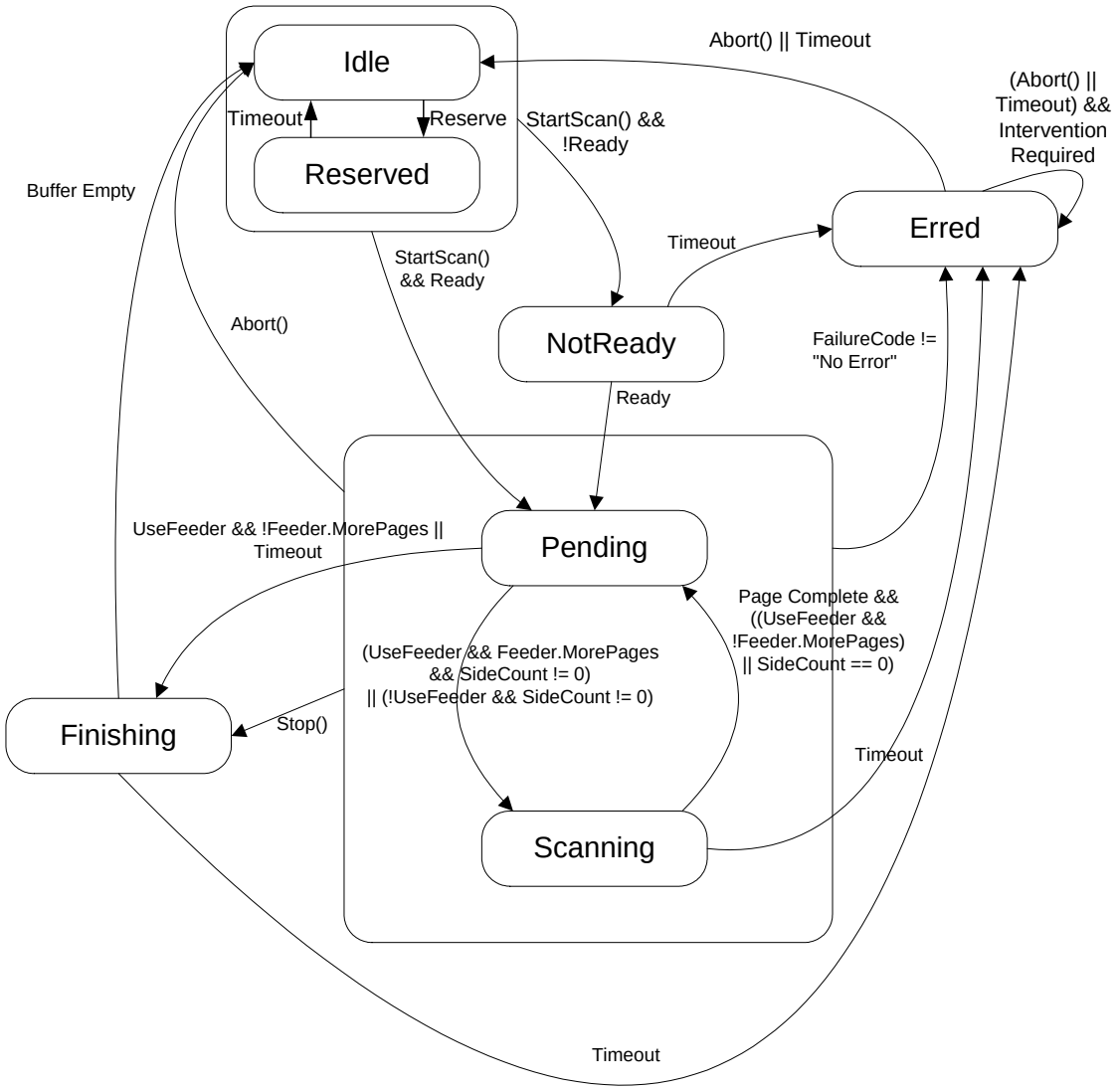

Figure 2 - Scanner State Transition Diagram

Table 15: Scanner State Transition Table

Current State	Actions and Conditions		
Idle	On Entry Reset all variables to default values		
	On Exit		
	Condition	Action	Next State
	StartScan() && Ready		Pending
	StartScan() && !Ready		NotReady
	Reservation from ExternalActivity		Reserved
	Otherwise remain in Idle state		

Current State	Actions and Conditions		
Reserved	On Entry		
	On Exit		
	Condition	Action	Next State
	StartScan() && RegID match && Ready		Pending
	StartScan() && RegID match && !Ready		NotReady
	Timeout		Idle
	Otherwise remain in Reserved state		
NotReady	On Entry Vendor Unique NotReady processing		
	On Exit		
	Condition	Action	Next State
	Ready		Pending
	ErredTimeout		Erred
	Otherwise remain in NotReady state		
Pending	On Entry		
	On Exit		
	Condition	Action	Next State
	Start()	Set SideCount and UseFeeder variables	Pending
	UseFeeder = true (1) && SideCount != 0 && Feeder.MorePages = true		Scanning
	UseFeeder = false (0) && SideCount != 0	[1]SideCount = |SideCount|	Scanning
	Timeout || (UseFeeder = true (1) && Feeder.MorePages = false) || Stop()		Finishing
	Otherwise remain in Pending state – this implies that if the SideCount is 0, and there are more pages in the feeder then remain in Pending state.		
Scanning	On Entry Reset Timeout and ScanLength, increment DestinationID and SideNumber, generate a new Destination value, Scan an image and decrement SideCount		
	On Exit		
	Condition	Action	Next State
	SideCount == 0 || UseFeeder = true && Feeder.MorePages = false		Pending
	Stop()	Stop at end of page	Finishing
	Timeout		Erred
	Otherwise remain in Scanning state (i.e. execute the OnEntry actions again) – this implies that the scanner will repeatedly scan sheets until the SideCount is 0 or there are no pages in the feeder.		

[1] An infinite loop will occur if UseFeeder is false and SideCount is –1. Changing SideCount to the absolute value of SideCount will avoid this condition.

Current State	Actions and Conditions		
Finishing	On Entry Finish data transfers		
	On Exit		
	Condition	Action	Next State
	always		Idle
Erred	On Entry Abort Scanning, Set FailureCode variable		
	On Exit		
	Condition	Action	Next State
	(Abort() || Timeout) && Intervention Required		Erred
	Abort() || Timeout	Clear FailureCode	Idle
	Otherwise remain in Erred state		

Table 16: Current State vs Service Actions

Current State	Service Actions				
	StartScan	Start	Stop	Abort	SetConfiguration
Idle	Start a scan	Return Invalid_State	Ignored	Ignored	Return Invalid_State
Reserved	Start a scan if RegID matches	Return Invalid_State	Ignored	Ignored	Return Invalid_State
Pending	Return Invalid_State	Continue a scan	Next State = Finishing	Next State = Idle	Set a configuration
Scanning		Return Invalid_State			Return Invalid_State
Finishing			Ignored		
NotReady			Return Invalid_State		
Erred					

2.5.8 Scanner Buffer Functionality

The Scan Service model assumes the presence of a data buffer. The operating characteristics of this buffer are as follows:

- The buffer behaves as a FIFO (First-In, First-Out) queue
- Information in the buffer can be unnamed (used in the case where the data is pushed directly to a client supplied URL), or it can be named (used in the case where the client will pull the data from the buffer using an HTTP/GET method)
- Once data is removed from the buffer, it is gone. There is no persistence of the data in the buffer.
- The size of the buffer is unspecified.

The client does not directly interact with the buffer. It specifies the data transfer method by setting the BaseName variable. If the BaseName value is set to "buffer", then the data will be named and the client must *pull* the data from the buffer. The evented *Destination* variable will contain the URL of the buffer. If the BaseName is given a URL, then the data will be pushed through the buffer to the given *data-sink* URL.

2.5.9 Using XHTML-Print To Send An Image Directly To A Printer

One of the base use cases for UPnP Scanning is *Scan to Printer*. This operation requires that the scanner send image and XHTML-Print information to control the printing. *Scan-to-Printer* assumes that a third party acts as a control point for both the Printer and the Scanner services. This third party, which could be within the scanner device, would send a *CreateJob* action to the printer and receive a data-sink URL in return. The third party would then send a *StartScan* action containing the data-sink URL to the scanner to begin the scan. The scanner would begin the scan operation and send the print job data to the printer data-sink URL.

The *ImageFormat* variable must be *application/vnd.pwg-xhtml-print* to send images directly to a printer. In this case, the information sent to the Printer data-sink URL must contain an XHTML-Print document and embedded JPEG images interleaved using an application/multiplexed transfer described in the document found at http://search.ietf.org/internet-drafts/draft-herriot-application-multiplexed-04.txt. The first message of the multiplexed transfer must contain the beginning of an XHTML-Print job stream. The stream must include an object tag for each scanned image. Immediately after the object tag, the current chunk of the mime transfer would end, and a JPEG image, corresponding to the object tag, would follow in one or more chunks. When the JPEG image has been successfully transferred, a new mime chunk, containing the continuing XHTML-Print document, is started. If there is another image to scan, then the new mime chunk contains another object tag. This object tag will be followed by more chunks containing another JPEG Image. This will reoccur for each sheet scanned. When there are no further sheets to be scanned, a final chunk will be sent, which contains the necessary XHTML-Print to complete the print job. See Scan to Print using Multipart MIME on page 53 for a sample document.

2.5.10 BaseName, SideNumber, AppendSideNumber and Destination

The value of the *Destination* state variable is generated by the Scan Service based on the values of the *BaseName*, *SideNumber*, and *AppendSideNumber* variables. The table below defines how these values work together to define the *Destination* value.

Table 17: Destination Names

BaseName	Side Number	Append Side Number	Destination	Description
pull-relative	n/a	False	Image.jpg – where *Image* is a name generated by the Scan Service. This URL is relative to the BaseURL given in the Device Description Document	Client pull transfer from buffer
pull-relative	*nn*	True	Image*nn*.jpg – *Image* is a name generated by the Scan Service. This URL is relative to the BaseURL given in the Device Description Document	Client pull transfer from numbered files in the buffer
pull-absolute	n/a	False	/hostname:port/path/Image.jpg – where *Image* is a name generated by the Scan Service and /hostname:port/path/ are equivalent to the BaseURL given in the Device Description Document	Client pull transfer from buffer

BaseName	Side Number	Append Side Number	Destination	Description
pull-absolute	*nn*	True	/hostname:port/path/Image*nn*.jpg – where *Image* is a name generated by the Scan Service and /hostname:port/path/ are equivalent to the BaseURL given in the Device Description Document	Client pull transfer from numbered files in the buffer
client.xxx.com/filename	*n/a*	False	client.xxx.com/filename.xxx	Post to the client supplied URL. See Table 1.2 for details of the filename suffix.
client.xxx.com/filename	*nn*	True	client.xxx.com/filename*nn*.xxx	Post to the client supplied URL. *nn* is the current value of SideNumber. See Table 1.2 for details of the filename suffix.

2.5.11 Relationship between the Scan Service and the Feeder Service

The Scan Service and the Feeder service were separated to try to keep the complexity of the feeder out of the scan functionality. As a result, the scanner really does not know much about a feeder. It just knows whether it has another page (Feeder.MorePages) and how to Load(), Eject() and Reset() a feeder. The scanner does not know where the feeder gets its next page, or where the pages go when they are ejected, nor does it matter. If a more complex interaction is needed, then the vendor should consider enhancing the feeder or scanner services.

2.5.12 Relationship between the Scan Service and the ExternalActivity Service

The ExternalActivity service is used to represent a user accessible control panel. Two (2) Use-Models can be implemented using interactions between the ExternalActivity service and the Scan Service. The *ExternalActivity Initiated, or Push, model* is used when the user starts the scanner interaction by pressing a button represented in the ExternalActivity service. The *Control Point Initiated, or Pull, model* is used when the control point starts a scan operation and then waits for an ExternalActivity notification to continue it.

2.5.12.1 ExternalActivity Initiated "Push" Operating Model

In this use-model, the user walks up to the scanner, places a document to be scanned into the feeder, or onto the glass, and presses a button associated with the scanner. The ExternalActivity service then sets the required RegistrationID value using an unspecified internal mechanism and sends out a standard UPnP event notification on the Activity variable value. When the EA service sets the RegistrationID value the scanner state is set to *Reserved.* This makes the Scan Service available for exclusive use by a control point that possesses the same RegistrationID value. The reservation lasts until a StartScan action is received

with the required RegistrationID value or until a Timeout occurs. If a timeout occurs, then the required RegistrationID value will be set back to 0 and the scanner state is set to Idle.

2.5.12.2 Control Point Initiated "Pull" Operating Model

In this use model the user starts at a control point that "reserves" the scanner by sending a *StartScan* action with its RegistrationID value, UseFeeder=false and SideCount=0. This action starts a scan operation that will stop in the pending state and wait for something to happen. This action will work if the current registrationID value is 0 and the state is Idle. When these steps occur, the Scan Service will set the current RegistrationID value in the ExternalActivity service to the given value using an unspecified internal mechanism.

2.5.13 Once the Scanner is successfully "reserved", the user may walk to the scanner and place the document in it (either in a feeder or on the glass) and press the button associated with the ExternalActivity service. This button press causes a UPnP event notification for the Activity state variable. The control point must recognize this notification and send an appropriate *Start* action, with the JobID returned by the StartScan action, to continue the scanning operation.Abort Handling

The Abort action is used to immediately terminate an active scan operation. All transfers are immediately terminated. All Scanning is stopped immediately. If there is a page in the scanner, and the UseFeeder variable is true (1), then the page will be ejected unless the current FailureCode is "Jammed".

2.5.14 Extending Scanner Functionality

The Scan Service is not intended to be an all-encompassing template. It does not perform all of the high-level functionality that the committee members could think of. Instead, it performs a simple scan operation and enables some extended features through interaction with a client. Multi-pass scans are possible by using the *Start* action one page at a time. Extended feeders can be used without changing the scanner functionality. Both push (the scanner sends images to a client) and pull (the client gets the images from the scanner) models are supported. No specific buffer size is required. All of these functions can be extended by a vendor if they wish. The additional functionality must be properly described in the *Service Control Protocol Document* and it cannot modify the defined standard actions and state variables.

3 XML Service Template for *Scan:1.0*

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 Scan 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>StartScan</name>
```

```
<argumentList>
  <argument>
    <name>RegistrationIDIn</name>
    <relatedStateVariable>RegistrationID</relatedStateVariable>
    <direction>in</direction>
  </argument>
  <argument>
    <name>UseFeederIn</name>
    <relatedStateVariable>UseFeeder</relatedStateVariable>
    <direction>in</direction>
  </argument>
  <argument>
    <name>SideCountIn</name>
    <relatedStateVariable>SideCount</relatedStateVariable>
    <direction>in</direction>
  </argument>
  <argument>
    <name>JobNameIn</name>
    <relatedStateVariable>JobName</relatedStateVariable>
    <direction>in</direction>
  </argument>
  <argument>
    <name>ResolutionIn</name>
    <relatedStateVariable>Resolution</relatedStateVariable>
    <direction>in</direction>
  </argument>
  <argument>
    <name>ImageXOffsetIn</name>
    <relatedStateVariable>XValueLimit </relatedStateVariable>
    <direction>in</direction>
  </argument>
  <argument>
    <name>ImageYOffsetIn</name>
    <relatedStateVariable>YValueLimit </relatedStateVariable>
    <direction>in</direction>
  </argument>
  <argument>
    <name>ImageWidthIn</name>
    <relatedStateVariable>WidthLimit</relatedStateVariable>
    <direction>in</direction>
  </argument>
  <argument>
    <name>ImageHeightIn</name>
    <relatedStateVariable>HeightLimit</relatedStateVariable>
    <direction>in</direction>
  </argument>
  <argument>
    <name>ImageFormatIn</name>
    <relatedStateVariable>ImageFormat</relatedStateVariable>
    <direction>in</direction>
  </argument>
  <argument>
    <name>CompressionFactorIn</name>
    <relatedStateVariable>CompressionFactor
      </relatedStateVariable>
    <direction>in</direction>
```

```
</argument>
<argument>
  <name>ImageTypeIn</name>
  <relatedStateVariable>ImageType</relatedStateVariable>
  <direction>in</direction>
</argument>
<argument>
  <name>ColorTypeIn</name>
  <relatedStateVariable>ColorType</relatedStateVariable>
  <direction>in</direction>
</argument>
<argument>
  <name>BitDepthIn</name>
  <relatedStateVariable>BitDepth</relatedStateVariable>
  <direction>in</direction>
</argument>
<argument>
  <name>ColorSpaceIn</name>
  <relatedStateVariable>ColorSpace</relatedStateVariable>
  <direction>in</direction>
</argument>
<argument>
  <name>BaseNameIn</name>
  <relatedStateVariable>BaseName</relatedStateVariable>
  <direction>in</direction>
</argument>
<argument>
  <name>AppendSideNumberIn</name>
  <relatedStateVariable>AppendSideNumber
    </relatedStateVariable>
  <direction>in</direction>
</argument>
<argument>
  <name>TimeoutIn</name>
  <relatedStateVariable>Timeout</relatedStateVariable>
  <direction>in</direction>
</argument>
<argument>
  <name>ActualTimeoutOut</name>
  <relatedStateVariable>Timeout</relatedStateVariable>
  <direction>out</direction>
</argument>
<argument>
  <name>JobIDOut</name>
  <relatedStateVariable>JobID</relatedStateVariable>
  <direction>out</direction>
</argument>
<argument>
  <name>ActualWidthOut</name>
  <relatedStateVariable>WidthLimit</relatedStateVariable>
  <direction>out</direction>
</argument>
<argument>
  <name>ActualHeightOut</name>
  <relatedStateVariable>HeightLimit</relatedStateVariable>
  <direction>out</direction>
```

```
      </argument>
    </argumentList>
  </action>
  <action>
    <name>Start</name>
    <argumentList>
      <argument>
        <name>JobIDIn</name>
        <relatedStateVariable>JobID</relatedStateVariable>
        <direction>in</direction>
      </argument>
      <argument>
        <name>UseFeederIn</name>
        <relatedStateVariable>UseFeeder</relatedStateVariable>
        <direction>in</direction>
      </argument>
      <argument>
        <name>SideCountIn</name>
        <relatedStateVariable>SideCount</relatedStateVariable>
        <direction>in</direction>
      </argument>
    </argumentList>
  </action>
  <action>
    <name>Stop</name>
    <argumentList>
      <argument>
        <name>JobIDIn</name>
        <relatedStateVariable>JobID</relatedStateVariable>
        <direction>in</direction>
      </argument>
    </argumentList>
  </action>
  <action>
    <name>Abort</name>
    <argumentList>
      <argument>
        <name>JobIDIn</name>
        <relatedStateVariable>JobID</relatedStateVariable>
        <direction>in</direction>
      </argument>
    </argumentList>
  </action>
  <action>
    <name>SetConfiguration</name>
    <argumentList>
      <argument>
        <name>JobIDIn</name>
        <relatedStateVariable>JobID</relatedStateVariable>
        <direction>in</direction>
      </argument>
      <argument>
        <name>JobNameIn</name>
        <relatedStateVariable>JobName</relatedStateVariable>
        <direction>in</direction>
      </argument>
```

```
<argument>
  <name>ResolutionIn</name>
  <relatedStateVariable>Resolution</relatedStateVariable>
  <direction>in</direction>
</argument>
<argument>
  <name>ImageXOffsetIn</name>
  <relatedStateVariable>XValueLimit</relatedStateVariable>
  <direction>in</direction>
</argument>
<argument>
  <name>ImageYOffsetIn</name>
  <relatedStateVariable>YValueLimit</relatedStateVariable>
  <direction>in</direction>
</argument>
<argument>
  <name>ImageWidthIn</name>
  <relatedStateVariable>WidthLimit</relatedStateVariable>
  <direction>in</direction>
</argument>
<argument>
  <name>ImageHeightIn</name>
  <relatedStateVariable>HeightLimit</relatedStateVariable>
  <direction>in</direction>
</argument>
<argument>
  <name>ImageFormatIn</name>
  <relatedStateVariable>ImageFormat</relatedStateVariable>
  <direction>in</direction>
</argument>
<argument>
  <name>CompressionFactorIn</name>
  <relatedStateVariable>CompressionFactor
    </relatedStateVariable>
  <direction>in</direction>
</argument>
<argument>
  <name>ImageTypeIn</name>
  <relatedStateVariable>ImageType</relatedStateVariable>
  <direction>in</direction>
</argument>
<argument>
  <name>ColorTypeIn</name>
  <relatedStateVariable>ColorType</relatedStateVariable>
  <direction>in</direction>
</argument>
<argument>
  <name>BitDepthIn</name>
  <relatedStateVariable>BitDepth</relatedStateVariable>
  <direction>in</direction>
</argument>
<argument>
  <name>ColorSpaceIn</name>
  <relatedStateVariable>ColorSpace</relatedStateVariable>
  <direction>in</direction>
</argument>
```

```
        <argument>
          <name>BaseNameIn</name>
          <relatedStateVariable>BaseName</relatedStateVariable>
          <direction>in</direction>
        </argument>
        <argument>
          <name>AppendSideNumberIn</name>
          <relatedStateVariable>AppendSideNumber
            </relatedStateVariable>
          <direction>in</direction>
        </argument>
        <argument>
          <name>TimeoutIn</name>
          <relatedStateVariable>Timeout</relatedStateVariable>
          <direction>in</direction>
        </argument>
        <argument>
          <name>ActualTimeoutOut</name>
          <relatedStateVariable>Timeout</relatedStateVariable>
          <direction>out</direction>
        </argument>
        <argument>
          <name>ActualWidthOut</name>
          <relatedStateVariable>WidthLimit</relatedStateVariable>
          <direction>out</direction>
        </argument>
        <argument>
          <name>ActualHeightOut</name>
          <relatedStateVariable>HeightLimit</relatedStateVariable>
          <direction>out</direction>
        </argument>
      </argumentList>
    </action>
    <action>
      <name>GetConfiguration</name>
      <argumentList>
        <argument>
          <name>JobNameOut</name>
          <relatedStateVariable>JobName</relatedStateVariable>
          <direction>out</direction>
        </argument>
        <argument>
          <name>ResolutionOut</name>
          <relatedStateVariable>Resolution</relatedStateVariable>
          <direction>out</direction>
        </argument>
        <argument>
          <name>ImageXOffsetOut</name>
          <relatedStateVariable>XValueLimit</relatedStateVariable>
          <direction>out</direction>
        </argument>
        <argument>
          <name>ImageYOffsetOut</name>
          <relatedStateVariable>YValueLimit</relatedStateVariable>
          <direction>out</direction>
        </argument>
```

```
<argument>
  <name>ImageWidthOut</name>
  <relatedStateVariable>WidthLimit</relatedStateVariable>
  <direction>out</direction>
</argument>
<argument>
  <name>ImageHeightOut</name>
  <relatedStateVariable>HeightLimit</relatedStateVariable>
  <direction>out</direction>
</argument>
<argument>
  <name>ImageFormatOut</name>
  <relatedStateVariable>ImageFormat</relatedStateVariable>
  <direction>out</direction>
</argument>
<argument>
  <name>CompressionFactorOut</name>
  <relatedStateVariable>CompressionFactor
    </relatedStateVariable>
  <direction>out</direction>
</argument>
<argument>
  <name>ImageTypeOut</name>
  <relatedStateVariable>ImageType</relatedStateVariable>
  <direction>out</direction>
</argument>
<argument>
  <name>ColorTypeOut</name>
  <relatedStateVariable>ColorType</relatedStateVariable>
  <direction>out</direction>
</argument>
<argument>
  <name>BitDepthOut</name>
  <relatedStateVariable>BitDepth</relatedStateVariable>
  <direction>out</direction>
</argument>
<argument>
  <name>ColorSpaceOut</name>
  <relatedStateVariable>ColorSpace</relatedStateVariable>
  <direction>out</direction>
</argument>
<argument>
  <name>BaseNameOut</name>
  <relatedStateVariable>BaseName</relatedStateVariable>
  <direction>out</direction>
</argument>
<argument>
  <name>AppendSideNumberOut</name>
  <relatedStateVariable>AppendSideNumber
    </relatedStateVariable>
  <direction>out</direction>
</argument>
<argument>
  <name>TimeoutOut</name>
  <relatedStateVariable>Timeout</relatedStateVariable>
  <direction>out</direction>
```

```
      </argument>
    </argumentList>
  </action>
  <action>
    <name>GetSideInformation</name>
    <argumentList>
      <argument>
        <name>SideNumberOut</name>
        <direction>out</direction>
        <relatedStateVariable>SideNumber</relatedStateVariable>
      </argument>
      <argument>
        <name>SideCountOut</name>
        <direction>out</direction>
        <relatedStateVariable>SideCount</relatedStateVariable>
      </argument>
      <argument>
        <name>ScanLengthOut</name>
        <relatedStateVariable>ScanLength</relatedStateVariable>
        <direction>out</direction>
      </argument>
    </argumentList>
  </action>
  <action>
    <name>GetDestination</name>
    <argumentList>
      <argument>
        <name>JobIDIn</name>
        <relatedStateVariable>JobID</relatedStateVariable>
        <direction>in</direction>
      </argument>
      <argument>
        <name>DestinationOut</name>
        <relatedStateVariable>Destination</relatedStateVariable>
        <direction>out</direction>
      </argument>
      <argument>
        <name>DestinationIDOut</name>
        <relatedStateVariable>DestinationID</relatedStateVariable>
        <direction>out</direction>
      </argument>
    </argumentList>
  </action>
  <action>
    <name>GetState</name>
    <argumentList>
      <argument>
        <name>StateOut</name>
        <relatedStateVariable>State</relatedStateVariable>
        <direction>out</direction>
      </argument>
      <argument>
        <name>StateReasonOut</name>
        <relatedStateVariable>StateReason</relatedStateVariable>
        <direction>out</direction>
      </argument>
```

```
        <argument>
          <name>FailureCodeOut</name>
          <relatedStateVariable>FailureCode</relatedStateVariable>
          <direction>out</direction>
        </argument>
      </argumentList>
    </action>
    Declarations for other actions added by UPnP vendor (if any) go
      here
  </actionList>
  <serviceStateTable>
    <!-- Optional state variable
    this may be removed if not supported -->
    <stateVariable sendEvents="no">
      <name>JobName</name>
      <dataType>string</dataType>
    </stateVariable>
    <stateVariable sendEvents="yes">
      <name>FailureCode</name>
      <dataType>string</dataType>
      <defaultValue>No Error</defaultValue>
      <allowedValueList>
        <allowedValue>No Error</allowedValue>
        <allowedValue>Jammed</allowedValue>
        <allowedValue>Timeout Reached</allowedValue>
        <allowedValue>ErredTimeout Reached</allowedValue>
        <allowedValue>Destination Not Reachable</allowedValue>
        <!-- Other vendor unique error codes should be added here.
        Warning:  These values must be 31 characters or less! -->
        Other allowed values defined by UPnP Forum working committee
          (if
        any) go here
      </allowedValueList>
    </stateVariable>
    <stateVariable sendEvents="yes">
      <name>State</name>
      <dataType>string</dataType>
      <defaultValue>Idle</defaultValue>
      <allowedValueList>
        <allowedValue>Idle</allowedValue>
        <allowedValue>Reserved</allowedValue>
        <allowedValue>NotReady</allowedValue>
        <allowedValue>Pending</allowedValue>
        <allowedValue>Scanning</allowedValue>
        <allowedValue>Finishing</allowedValue>
        <allowedValue>Erred</allowedValue>
      </allowedValueList>
    </stateVariable>
    <stateVariable sendEvents="no">
      <name>StateReason</name>
      <dataType>string</dataType>
    </stateVariable>
    <stateVariable sendEvents="no">
      <name>ImageFormat</name>
      <dataType>string</dataType>
      <defaultValue>image/jpeg</defaultValue>
      <allowedValueList>
```

```
        <allowedValue>device-setting</allowedValue>
        <allowedValue>image/jpeg</allowedValue>
        <!-- Optional defined values
        <allowedValue>application/vnd.pwg-xhtml-print</allowedValue>
        -->
        <!--Other allowed values defined by UPnP Forum working
        committee (if any) go here -->
      </allowedValueList>
    </stateVariable>
    <stateVariable sendEvents="no">
      <name>CompressionFactor</name>
      <dataType>i4</dataType>
      <defaultValue>100</defaultValue>
      <allowedValueRange>
        <minimum>-1</minimum>
        <maximum>100</maximum>
        <step>1</step>
      </allowedValueRange>
    </stateVariable>
    <stateVariable sendEvents="no">
      <name>ImageType</name>
      <dataType>string</dataType>
      <defaultValue>Mixed</defaultValue>
      <allowedValueList>
        <allowedValue>device-setting</allowedValue>
        <allowedValue>Mixed</allowedValue>
        <!-- Optional defined values
        <allowedValue>Photo</allowedValue>
        <allowedValue>Text</allowedValue>
        <allowedValue>Graphics</allowedValue>
        -->
      </allowedValueList>
    </stateVariable>
    <stateVariable sendEvents="no">
      <name>ColorType</name>
      <dataType>string</dataType>
      <defaultValue>Color</defaultValue>
      <allowedValueList>
        <allowedValue>device-setting</allowedValue>
        <!-- Optional defined values – Either Color or Mono must be an
          allowed value
        <allowedValue>Color</allowedValue>
        <allowedValue>Mono</allowedValue>
        -->
      </allowedValueList>
    </stateVariable>
    <stateVariable sendEvents="no">
      <name>BitDepth</name>
      <dataType>string</dataType>
      <defaultValue>8</defaultValue>
      <allowedValueList>
        <allowedValue>device-setting</allowedValue>
        <allowedValue>8</allowedValue>
        <!-- Optional defined values
        <allowedValue>12</allowedValue>
        -->
      </allowedValueList>
```

```
</stateVariable>
<stateVariable sendEvents="no">
  <name>ColorSpace</name>
  <dataType>string</dataType>
  <defaultValue>sRGB</defaultValue>
  <allowedValueList>
    <allowedValue>device-setting</allowedValue>
    <allowedValue>sRGB</allowedValue>
    <!-- Vendor defined values may be added here -->
  </allowedValueList>
</stateVariable>
<stateVariable sendEvents="no">
  <name>UseFeeder</name>
  <dataType>string</dataType>
  <defaultValue>0</defaultValue>
  <allowedValueList>
    <allowedValue>device-setting</allowedValue>
    <allowedValue>0</allowedValue>
    <!-- Optional defined value
    <allowedValue>1</allowedValue>
    -->
  </allowedValueList>
</stateVariable>
<stateVariable sendEvents="no">
  <name>BaseName</name>
  <dataType>string</dataType>
  <defaultValue>pull-relative</defaultValue>
</stateVariable>
<stateVariable sendEvents="no">
  <name>AppendSideNumber</name>
  <dataType>string</dataType>
  <defaultValue>0</defaultValue>
  <allowedValueList>
    <allowedValue>device-setting</allowedValue>
    <allowedValue>0</allowedValue>
    <!-- Optional defined value
    <allowedValue>1</allowedValue>
    -->
  </allowedValueList>
</stateVariable>
<stateVariable sendEvents="no">
  <name>SideCount</name>
  <dataType>i4</dataType>
  <defaultValue>0</defaultValue>
  <allowedValueRange>
    <minimum>-1</minimum><!---1 means all sheets -->
    <maximum>vendor-defined</maximum><!-- Vendor defined range
      value -->
    <step>1</step>
  </allowedValueRange>
</stateVariable>
<stateVariable sendEvents="yes">
  <name>SideNumber</name>
  <dataType>i4</dataType>
  <defaultValue>0</defaultValue>
  <allowedValueRange>
```

```
      <minimum>0</minimum>
      <maximum>vendor-defined</maximum><!-- Vendor defined range
        value -->
      <step>1</step>
    </allowedValueRange>
  </stateVariable>
  <stateVariable sendEvents="no">
    <name>Destination</name>
    <dataType>string</dataType>
  </stateVariable>
  <stateVariable sendEvents="no">
    <name>Timeout</name>
    <dataType>i4</dataType>
    <defaultValue>vendor-defined</defaultValue> <!-- Should be the
      max value -->
    <allowedValueRange>
      <minimum>-1</minimum>
      <maximum>vendor-defined</maximum><!-- Vendor defined range
        value -->
      <step>1</step>
    </allowedValueRange>
  </stateVariable>
  <stateVariable sendEvents="no">
    <name>ErrorTimeout</name>
    <dataType>i4</dataType>
    <defaultValue>vendor-defined</defaultValue>
  </stateVariable>
  <stateVariable sendEvents="no">
    <name>Resolution</name>
    <dataType>string</dataType>
    <defaultValue>vendor-defined</defaultValue> <!-- Should be the
      max value -->
    <!--The vendor may have either an allowedValueList or an
      allowedValueRange for the Resolution variable -->
    <allowedValueList>
      <allowedValue>device-setting</allowedValue><!--The vendor may
        add values as needed -->
    </allowedValueList>
  </stateVariable>
  <stateVariable sendEvents="yes">
    <name>ScanLength</name>
    <dataType>i4</dataType>
    <defaultValue>0</defaultValue> <!-- Should be the max value -->
    <allowedValueRange>
      <minimum>0</minimum>
      <maximum>vendor-defined</maximum><!-- Vendor defined range
        value -->
      <step>1</step>
    </allowedValueRange>
  </stateVariable>
  <stateVariable sendEvents="no">
    <name>DeviceID</name>
    <dataType>string</dataType>
    <defaultValue>vendor-defined</defaultValue> <!--Vendor unique
      value-->
  </stateVariable>
  <stateVariable sendEvents="no">
```

```
    <name>HeightLimit</name>
    <dataType>i4</dataType>
    <defaultValue>vendor-defined</defaultValue> <!-- Should be the
      max value -->
    <allowedValueRange>
      <minimum>-1</minimum>
      <maximum>vendor-defined</maximum><!-- Vendor defined range
        value -->
      <step>1</step>
    </allowedValueRange>
  </stateVariable>
  <stateVariable sendEvents="no">
    <name>WidthLimit</name>
    <dataType>i4</dataType>
    <defaultValue>vendor-defined</defaultValue> <!-- Should be the
      max value -->
    <allowedValueRange>
      <minimum>-1</minimum>
      <maximum>vendor-defined</maximum><!-- Vendor defined range
        value -->
      <step>1</step>
    </allowedValueRange>
  </stateVariable>
  <stateVariable sendEvents="no">
    <name>XValueLimit</name>
    <dataType>i4</dataType>
    <defaultValue>vendor-defined</defaultValue> <!-- Should be the
      max value -->
    <allowedValueRange>
      <minimum>-1</minimum>
      <maximum>vendor-defined</maximum><!-- Vendor defined range
        value -->
      <step>1</step>
    </allowedValueRange>
  </stateVariable>
  <stateVariable sendEvents="no">
    <name>YValueLimit</name>
    <dataType>i4</dataType>
    <defaultValue>vendor-defined</defaultValue> <!-- Should be the
      max value -->
    <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>
  </stateVariable>
  <stateVariable sendEvents="no">
    <name>JobID</name>
    <dataType>ui4</dataType>
    <allowedValueRange>
      <minimum>1</minimum>
      <maximum>4294967295</maximum>
```

```
        <step>1</step>
      </allowedValueRange>
    </stateVariable>
    <stateVariable sendEvents="yes">
      <name>DestinationID</name>
      <dataType>ui4</dataType>
      <defaultValue>0</defaultValue>
      <allowedValueRange>
        <minimum>0</minimum>
        <maximum>4294967295</maximum><!—max-ui4 value -->
        <step>1</step>
      </allowedValueRange>
    </stateVariable>
    Declarations for other state variables added by UPnP vendor (if
      any)
    go here
  </serviceStateTable>
</scpd>
```

4 Testing

4.1 Syntax Testing

4.1.1 Issues

Due to the limited capabilities of the UPnP Certification Tool, only limited syntax testing can be performed at this time. Only actions that can return a successful status using arguments specified at the time the tests are written may be tested. This eliminates any actions that take the JobID as a input argument. This leaves testing of the *StartScan* action.

4.1.2 *StartScan* Syntax Test

This action is tested

5 Appendicies

5.1 Scan to Print using Multipart MIME

The print document represented below shows how an xhtml-print document with two embedded images would be transferred. This assumes the use of the application/multiplexed MIME type described in "The MIME Application/Multiplexed Content-Type" by Robert Herriot which can be found at http://search.ietf.org/internet-drafts/draft-herriot-application-multiplexed-00.txt. Where 00 is the initial revision. Later copies will have an incremented value. This MIME type defines a *chunk header* "CHK x y b" where *x* is the message identifier, *y* is the number of bytes in the chunk, and *b* is a boolean flag that indicates that there are more (Y) chunks or not (N).

Content-type: application/multiplexed;
type="text/xhtml-print+xml:

CHK 1 500 Y
Content-ID <12345.23456xxx@foo.com>
Content-type: text/xhtml-print+xml;charset="utf8"

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/TR/xhtml1">
    <head>
        <title> Testing Page Breaks </title>
        <style>pagebreak { page-break-after: always }</style>
    </head>
    <body>
        <p>Include an image after this!</p>

        <object declare=”declare” height=”20 mm” width=”20 mm” type=”image/jpeg”
            id=”image_1.jpeg”></object>
CHK 2 nnnnnnnn N
Content-Location: image_1.jpeg
Conten-id: <97116092511xyz@foo.bar.net>
Content-type: image/jpeg

...xyz bytes of jpeg binary ... (in this case the whole image)
CHK 1 215 Y
Content-type: text/xhtml-print+xml;charset="utf8"
        <p>This shows the inclusion of a second image in the document</p>

        <object declare=”declare” height=”20 mm” width=”20 mm” type=”image/jpeg”
            id=”image_2.jpeg”></object>

CHK 3 nnnnnnnn N
Content-Location: image_2.jpeg
Conten-id: <97116092511xyz@foo.bar.net>
Content-type: image/jpeg
...xyz bytes of jpeg binary ...
CHK 1 139 N
Content-type text/xhtml-print+xml;charset="utf8"
        <p>This shows the part of the document that comes after the final image.</p>
    </body>
</html>

CHK 0 0 N
```