MultiScreenArchitecture:1

For UPnP Version 1.0
Status: Standardized DCP (SDCP)
Date: September 30, 2014
Service Template Version 3.0

Authors	Company
Clarke Stevens	CableLabs
Wouter van der Beek	Cisco Systems Inc.
Seung R. Yang (Chair)	LG Electronics
Anders Klemets	Microsoft
Nicholas Frame	TP Vision
Note: The UPnP Forum in no way guarantees the accuracy or completeness of this author list and in no way implies any rights for or support from those members listed. This list is not the specifications' contributor list that is kept on the UPnP Forum's website.	

CONTENTS

1 Scope

Today multi-screen/second-screen solutions are proliferating. However, each is a proprietary vertical for particular vendor(s). Therefore users expectations aren’t being met:

- Seamless interoperability across vendors.
- Ability for second screen integrated usages rather than 100s of different apps.

The UPnP Multi-Screen Device Control Protocols (DCPs) provide an open interface to enable this interoperability between devices and applications, i.e. enable time-sensitive and interactive services, including implementation-specific applications, among various display devices.

This document describes the overall Multi-Screen Architecture, which forms the foundation for the UPnP Multi-Screen Device and Service templates. The Multi-Screen Architecture defines the general interaction between UPnP control points and UPnP devices defined by the UPnP Multi-Screen DCPs.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

[1] – *UPnP Device Architecture, version 1.0*, UPnP Forum, October 15, 2008.
Available at: http://www.upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.0-20081015.pdf.
Latest version available at: http://www.upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.0.pdf.

[2] – *ApplicationManagement:1*, UPnP Forum, September 30, 2014.
Available at: http://www.upnp.org/specs/ms/UPnP-ms-ApplicationManagement-v1-Service-20140930.pdf.
Latest version available at: http://www.upnp.org/specs/ms/UPnP-ms-ApplicationManagement-v1-Service.pdf.

[3] – *ApplicationManagement:2*, UPnP Forum, September 30, 2014.
Available at: http://www.upnp.org/specs/ms/UPnP-ms-ApplicationManagement-v2- Service-20140930.pdf.
Latest version available at: http://www.upnp.org/specs/ms/UPnP-ms-ApplicationManagement-v2-Service.pdf.

[4] – *ScreenDevice:1*, UPnP Forum, September 30, 2014.
Available at: http://www.upnp.org/specs/ms/UPnP-ms-ScreenDevice-v1-Device-20140930.pdf.
Latest version available at: http://www.upnp.org/specs/ms/UPnP-ms-ScreenDevice-v1-Device.pdf.

[5] – *ScreenDevice:2*, UPnP Forum, September 30, 2014.
Available at: http://www.upnp.org/specs/ms/UPnP-ms-ScreenDevice-v2-Device-20140930.pdf.
Latest version available at: http://www.upnp.org/specs/ms/UPnP-ms-ScreenDevice-v2-Device.pdf.

[6] – *XML Schema Part 1: Structures, Second Edition*, Henry S. Thompson, David Beech, Murray Maloney, Noah Mendelsohn, W3C Recommendation, 28 October 2004.
Available at: http://www.w3.org/TR/2004/REC-xmlschema-1-20041028.

[7] – *XML Schema Part 2: Data Types, Second Edition*, Paul V. Biron, Ashok Malhotra, W3C Recommendation, 28 October 2004.
Available at: http://www.w3.org/TR/2004/REC-xmlschema-2-20041028.

[8] – *The “xml:” Namespace*, November 3, 2004.
Available at: http://www.w3.org/XML/1998/namespace.

[9] – *Namespaces in XML*, Tim Bray, Dave Hollander, Andrew Layman, eds., W3C Recommendation, January 14, 1999.
Available at: http://www.w3.org/TR/1999/REC-xml-names-19990114.

[10] – *XML Schema for FeatureList XML Structures*, UPnP Forum, September 30, 2014.
Available at: http://www.upnp.org/ schemas/ms/FeatureList-v1-20140930.xsd.
Latest version available at: http://www.upnp.org/ schemas/ms/FeatureList.xsd.

[11] – *XML Schema for AppInfoList XML Structures*, UPnP Forum, September 30, 2014.
Available at: http://www.upnp.org/ schemas/ms/AppInfoList-v2-20140930.xsd.
Latest version available at: http://www.upnp.org/ schemas/ms/AppInfoList.xsd.

3 Terms, definitions, symbols and abbreviations

For the purposes of this document, the terms and definitions given in [1] and the following apply.

3.1 Provisioning terms (abbreviated forms are used *only* in tables)

3.1.1
allowed
A
The definition or behavior is allowed.

3.1.2
conditionally allowed
CA
The definition or behavior depends on a condition. If the specified condition is met, then the definition or behavior is allowed, otherwise it is not allowed.

3.1.3
conditionally required
CR
The definition or behavior depends on a condition. If the specified condition is met, then the definition or behavior is required. Otherwise the definition or behavior is allowed as default unless specifically defined as not allowed.

3.1.4
required
R
The definition or behavior is required.

3.1.5
R/A
Used in a table column heading to indicate that each abbreviated entry in the column declares the provisioning status of the item named in the entry's row.

3.1.6
X
Vendor-defined, non-standard.

3.1.7
-D
Declares that the item referred to is deprecated, when it is appended to any of the other abbreviated provisioning terms.

3.1.8
Screen Device
A UPnP component used to provide various interactive services with other display device(s) which is (are) implemented with Screen Control Point(s). Designed to be controlled by and interact with Screen Control Point(s). See subclause 5.1 for details.

3.1.9
Screen Control Point
A UPnP component used to provide various interactive services with other display device(s) which is (are) implemented with Screen Device(s). Designed to control and interact with Screen Device(s) with direct input from end-users. See subclause 5.2 for details.

3.1.10
Multi-Screen Service

Time-sensitive and interactive services, including implementation-specific applications, among various display devices. The display devices can be categorized into the main screen device and companion screen device by the roles and usages of the specific applications.

3.1.11

Main screen device :

Usually the main screen device is assumed as a lean-back display device such as a TV or set-top box which is controlled by companion screen devices. But any display device such as a smart phone, tablet, etc. can be a main screen device depending on usage scenarios.

3.1.12

Companion screen device

Usually the companion screen device is assumed as a lean-forward & handheld display device such as a smart phone or tablet which controls main screen devices. But any display device such as a TV or set-top box, etc. can be a main screen device depending on usage scenarios.

3.1.13
CSV list (or CSV)
comma separated value list
list—or one-dimensional array—of values contained in a string and separated by commas

3.2 Symbols

3.2.1
::
signifies a hierarchical parent-child (parent::child) relationship between the two objects separated by the double colon. This delimiter is used in multiple contexts, for example: Service::Action(), Action()::Argument, parentXMLElement::childXMLElement.

4 Notations and Conventions

4.1 Notation

- UPnP interface names defined in the UPnP Device Architecture specification [1] are styled in **green bold underlined** text.
- UPnP interface names defined outside of the UPnP Device Architecture specification [1] are styled in *red italic underlined* text.
- Some additional non-interface names and terms are styled in *italic* text.
- Words that are emphasized are also styled in *italic* text. The difference between italic terms and italics for emphasis will be apparent by context.
- Strings that are to be taken literally are enclosed in “double quotes”.

4.1.1 Data Types

Data type definitions come from three sources:

- All state variable and action argument data types are defined in the UPnP Device Architecture specification [1].
- Basic data types for properties are defined in the XML Schema Part 2: Data Types [7].
- Additional data types are defined in the XML schema(s) (see [10], [11]) for their associated service(s).

For UPnP Device Architecture defined **boolean** data types, it is strongly recommended to use the value “**0**” for false, and the value “**1**” for true. However, when used as input arguments, the values “**false**”, “**no**”, “**true**”, “**yes**” may also be encountered and shall be accepted. Nevertheless, it is strongly recommended that all **boolean** state variables and output arguments be represented as “**0**” and “**1**”.

For XML Schema defined Boolean data types, it is strongly recommended to use the value "*0*" for false, and the value "*1*" for true. However, when used as input properties, the values "*false*", "*true*" may also be encountered and shall be accepted. Nevertheless, it is strongly recommended that all Boolean properties be represented as "*0*" and "*1*".

4.1.2 Strings Embedded in Other Strings

Some string variables and arguments described in this document contain substrings that shall be independently identifiable and extractable for other processing. This requires the definition of appropriate substring delimiters and an escaping mechanism so that these delimiters can also appear as ordinary characters in the string and/or its independent substrings. This document uses embedded strings in two contexts – Comma Separated Value (CSV) lists (see subclause 4.2.1) and property values in search criteria strings. Escaping conventions use the backslash character, "\" (character code U+005C), as follows:

a) Backslash ("\") is represented as "\\" in both contexts.

b) Comma (",") is

   1) represented as "\," in individual substring entries in CSV lists

   2) not escaped in search strings

c) Double quote ("\"") is

   1) not escaped in CSV lists

   2) not escaped in search strings when it appears as the start or end delimiter of a property value

   3) represented as "\"" in search strings when it appears as a character that is part of the property value

4.2 Derived Data Types

This subclause defines a derived data type that is represented as a string data type with special syntax. This specification uses string data type definitions that originate from two different sources. The UPnP Device Architecture [1] defined **string** data type is used to define state variable and action argument **string** data types. The XML Schema namespace is used to define property xsd:string data types. The following definition applies to both string data types.

4.2.1 CSV Lists

The UPnP Multi-Screen DCPs use state variables, action arguments and properties that represent lists – or one-dimensional arrays – of values. The UPnP Device Architecture, Version 1.0 [1], does not provide for either an array type or a list type, so a list type is defined here. Lists may either be homogeneous (all values are the same type) or heterogeneous (all values can be of different types). Lists may also consist of repeated occurrences of homogeneous or heterogeneous subsequences, all of which have the same syntax and semantics (same number of values, same value types and in the same order). The data type of a homogeneous list is **string** or xsd:string and denoted by CSV (*x*), where *x* is the type of the individual values. The data type of a heterogeneous list is also **string** or xsd:string and denoted by CSV (*x, y, z*), where *x*, *y* and *z* are the types of the individual values. If the number of values in the heterogeneous list is too large to show each type individually, that variable type is represented as CSV (heterogeneous), and the variable description includes additional information as to the expected sequence of values appearing in the list and their corresponding types. The data type of a repeated subsequence list is **string** or xsd:string and denoted by CSV ({a,b,c},{*x*, *y*, *z*}), where a, b, c, *x*, *y* and *z* are the types of the individual values in the subsequence and the subsequences may be repeated zero or more times.

- A list is represented as a **string** type (for state variables and action arguments) or xsd:string type (for properties).
- Commas separate values within a list.

- Integer values are represented in CSVs with the same syntax as the integer data type specified in [1] (that is: allowed leading sign, allowed leading zeroes, numeric US-ASCII)
- Boolean values are represented in state variable and action argument CSVs as either "0" for false or "1" for true. These values are a subset of the defined boolean data type values specified in [1]: 0, false, no, 1, true, yes.
- Escaping conventions for the comma and backslash characters are defined in 4.1.2.
- White space before, after, or interior to any numeric data type is not allowed.
- White space before, after, or interior to any other data type is part of the value.

Table 1 — CSV Examples

Type refinement of string	Value	Comments
CSV (int) or CSV (xsd:integer)	"1,-5,006,0,+7"	List of 5 integers.
CSV (boolean) or CSV (xsd:Boolean)	"0,1,1,0"	List of 4 booleans
CSV (string) or CSV (xsd:string)	"multi,screen"	List of 2 strings
CSV (string) or CSV (xsd:string)	"Smith\, Fred,Jones\, Davey"	List of 2 names, "Smith, Fred" and "Jones, Davey"
CSV (i4,string,ui2) or CSV (xsd:int, xsd:string, xsd:unsignedShort)	"-29837, string with leading blanks,0"	Note that the second value is " string with leading blanks"
CSV (i4) or CSV (xsd:int)	"3, 4"	Illegal CSV. White space is not allowed as part of an integer value.
CSV (string) or CSV (xsd:string)	",,"	List of 3 empty string values
CSV (heterogeneous)	"Alice,Marketing,5,Sue,R&D,21,Dave,Finance,7"	List of unspecified number of people and associated attributes. Each person is described by 3 elements: a name string, a department string and years-of-service ui2 or a name xsd:string, a department xsd:string and years-of-service xsd:unsignedShort.

4.2.2 XML Document

An XML document is a string that represents a valid XML 1.0 document according to a specific schema. Every occurrence of the phrase "*XML Document*" is italicized and preceded by the document's root element name (also italicized), as listed in column 3, "Valid Root Element(s)" of Table 3.

For example, the phrase *AppInfoList XML Document* refers to a valid XML 1.0 document according to the XML Schema for AppInfoList [11]. Such a document comprises a single <*AppInfoList* …> root element, and it is allowed to be preceded by the XML declaration `<?xml version="1.0" …?>`.

This string will therefore be of one of the following two forms:

"<*AppInfoList* …>…</*AppInfoList*>"

`or`

"<?xml …?><AppInfoList …>…</AppInfoList>"

4.2.3 XML Fragment

An XML fragment is a sequence of XML elements that are valid direct or indirect child elements of the root element according to a specific schema. Every occurrence of the phrase "*XML Fragment*" is italicized and preceded by the document's root element name (also italicized), as listed in column 3, "Valid Root Element(s)" of Table 3, " — Schema-related Information".

The following are examples of *AppInfoList XML Fragments*:

"<runningStatus …>…</runningStatus>"

or

"<apptoAppInfo…>…</apptoAppInfo >"

or

"< friendlyName>Sunrise</friendlyName>"

4.3 Management of XML Namespaces in Standardized DCPs

UPnP specifications make extensive use of XML namespaces. This enables separate DCPs, and even separate components of an individual DCP, to be designed independently and still avoid name collisions when they share XML documents. Every name in an XML document belongs to exactly one namespace. In documents, XML names appear in one of two forms: qualified or unqualified. An unqualified name (or no-colon-name) contains no colon (":") characters. An unqualified name belongs to the document's default namespace. A qualified name is two no-colon-names separated by one colon character. The no-colon-name before the colon is the qualified name's namespace prefix, the no-colon-name after the colon is the qualified name's "local" name (meaning local to the namespace identified by the namespace prefix). Similarly, the unqualified name is a local name in the default namespace.

The formal name of a namespace is a URI. The namespace prefix used in an XML document is *not* the name of the namespace. The namespace name shall be globally unique. It has a single definition that is accessible to anyone who uses the namespace. It has the same meaning anywhere that it is used, both inside and outside XML documents. The namespace prefix, however, in formal XML usage, is defined only in an XML document. It shall be locally unique to the document. Any valid XML no-colon-name may be used. And, in formal XML usage, different XML documents may use different namespace prefixes to refer to the same namespace. The creation and use of the namespace prefix was standardized by the W3C XML Committee in [9] strictly as a convenient local shorthand replacement for the full URI name of a namespace in individual documents.

All of the namespaces used in this specification are listed in Table 2 and Table 3. For each such namespace, Table 2 gives a brief description of it, its name (a URI) and its defined "standard" prefix name. Some namespaces included in these tables are not directly used or referenced in this document. They are included for completeness to accommodate those situations where this specification is used in conjunction with other UPnP specifications to construct a complete system of devices and services. For example, since the Scheduled Recording Service depends on and refers to the ContentDirectory service, the predefined "srs:" namespace prefix is included. The individual specifications in such collections all use the same standard prefix. The standard prefixes are also used in Table 3 to cross-reference additional namespace information. This second table includes each namespace's valid XML document root element(s) (if any), its schema file name, versioning information (to be discussed in more detail below), and a link to the entry in Clause 2 for its associated schema.

The normative definitions for these namespaces are the documents referenced in Table 2. The schemas are designed to support these definitions for both human understanding and as test tools. However, limitations of the XML Schema language itself make it difficult for the UPnP-defined schemas to accurately represent all details of the namespace definitions. As a result, the schemas will validate many XML documents that are not valid according to the specifications.

The Working Committee expects to continue refining these schemas after specification release to reduce the number of documents that are validated by the schemas while violating the specifications, but the schemas will still be informative, supporting documents. Some schemas might become normative in future versions of the specifications.

Table 2 — Namespace Definitions

Standard Name-space Prefix	Namespace Name	Namespace Description	Normative Definition Document Reference
Multi-Screen Working Committee defined namespaces			
featurelist	urn:schemas-upnp-org:ms:FeatureList	*FeatureList* state variable for ApplicationManagement	[10]
appinfolist	urn:schemas-upnp-org:ms:AppInfoList	*AppInfoList* state variable for ApplicationManagement	[11]
Externally defined namespaces			
xsd	http://www.w3.org/2001/XMLSchema	XML Schema Language 1.0	[6], [7]
xsi	http://www.w3.org/2001/XMLSchema-instance	XML Schema Instance Document schema	2.6 & 3.2.7 in [6]
xml	http://www.w3.org/XML/1998/namespace	The "xml:" Namespace	[8]

Table 3 — Schema-related Information

Standard Name-space Prefix	Relative URI and File Name [a] • Form 1, Form 2, Form3	Valid Root Element(s)	Schema Reference
Multi-Screen Working Committee Defined Namespaces			
featurelist	FeatureList-v*n*-*yyyymmdd*.xsd FeatureList -v*n*.xsd FeatureList.xsd	<*FeatureList*>	[10]
appinfolist	AppInfoList-v*n*-*yyyymmdd*.xsd AppInfoList -v*n*.xsd AppInfoList.xsd	<*AppInfoList*>	[11]
Externally Defined Namespaces			
xsd	*n/a*	`<schema>`	[6], [7]
xsi	*n/a*		2.6 & 3.2.7 in [6]
xml	*n/a*		[8]

a Absolute URIs are generated by prefixing the relative URIs with "http://www.upnp.org/schemas/av/"

4.3.1 Namespace Prefix Requirements

There are many occurrences in this specification of string data types that contain XML names (property names). These XML names in strings will not be processed under namespace-aware conditions. Therefore, all occurrences in instance documents of XML names in strings shall use the standard namespace prefixes as declared in Table 2. In order to properly process the XML documents described herein, control points and devices shall use namespace-aware XML processors [9] for both reading and writing. As allowed by [9], the namespace prefixes used in an instance document are at the sole discretion of the document creator. Therefore, the declared prefix for a namespace in a document may be different from the standard prefix. All devices shall be able to correctly process any valid XML instance document, even when it uses a non-standard prefix for ordinary XML names. However, it is strongly recommended that all devices use these standard prefixes for all instance documents to avoid confusion on the part of both human and machine readers. These standard prefixes are used in all descriptive text and all XML examples in this and related UPnP specifications. However, each individual specification may assume a default namespace for its descriptive text. In that case, names from that namespace may appear with no prefix.

The assumed default namespace, if any, for each UPnP Multi-Screen specification is given in Table 4.

Note: all UPnP Multi-Screen schemas declare attributes to be "unqualified", so namespace prefixes are never used with Multi-Screen Working Committee defined attribute names.

Table 4 — Default Namespaces for the Multi-Screen Specifications

Multi-Screen Specification Name	Default Namespace Prefix
ApplicationManagement	appinfolist

4.3.2 Namespace Names, Namespace Versioning and Schema Versioning

The UPnP Multi-Screen service specifications define several data structures (such as state variables and action arguments) whose format is an XML instance document that complies with one or more specific XML schemas, which define XML namespaces. Each namespace is uniquely identified by an assigned namespace name. The namespace names that are defined by the Multi-Screen Working Committee are URNs. See Table 2 for a current list of namespace names. Additionally, each namespace corresponds to an XML schema document that provides a machine-readable representation of the associated namespace to enable automated validation of the XML (state variable or action parameter) instance documents.

Within an XML schema and XML instance document, the name of each corresponding namespace appears as the value of an `xmlns` attribute within the root element. Each `xmlns` attribute also includes a namespace prefix that is associated with that namespace in order to qualify and disambiguate element and attribute names that are defined within different namespaces. The schemas that correspond to the listed namespaces are identified by URI values that are listed in the `schemaLocation` attribute also within the root element (see 4.3.3).

In order to enable both forward and backward compatibility, namespace names are permanently assigned and shall not change even when a new version of a specification changes the definition of a namespace. However, all changes to a namespace definition shall be backward-compatible. In other words, the updated definition of a namespace shall not invalidate any XML documents that comply with an earlier definition of that same namespace. This means, for example, that a namespace shall not be changed so that a new element or attribute becomes required in a conforming instance document. Although namespace names shall not change, namespaces still have version numbers that reflect a specific set of definitional changes. Each time the definition of a namespace is changed, the namespace's version number is incremented by one.

Whenever a new namespace version is created, a new XML schema document (.xsd) is created and published so that the new namespace definition is represented in a machine-readable form. Since a XML schema document is just a representation of a namespace definition, translation errors can occur. Therefore, it is sometime necessary to re-release a published schema in order to correct typos or other namespace representation errors. In order to easily identify the potential multiplicity of schema releases for the same namespace, the URI of each released schema shall conform to the following format (called Form 1):

Form 1: "http://www.upnp.org/schemas/ms/" ***schema-root-name*** "-v" ***ver*** "-" ***yyyymmdd*** where

- ***schema-root-name*** is the name of the root element of the namespace that this schema represents.
- ***ver*** corresponds to the version number of the namespace that is represented by the schema.

- ***yyyymmdd*** is the year, month and day (in the Gregorian calendar) that this schema was released.

Table 3 identifies the URI formats for each of the namespaces that are currently defined by the UPnP Multi-Screen Working Committee.

As an example, the original schema URI for the "rcs-event" namespace (that was released with the original publication of the UPnP AV service specifications in the year 2002) was "http://www.upnp.org/schemas/av/rcs-event-v1-20020625.xsd". When the UPnP AV service specifications were subsequently updated in the year 2006, the URI for the updated version of the "rcs-event" namespace was "http://www.upnp.org/schemas/av/rcs-event-v2-20060531.xsd". However, in 2006, the schema URI for the newly created "srs-event" namespace was "http://www.upnp.org/schemas/av/srs-event-v1-20060531.xsd". Note the version field for the "srs-event" schema is "v1" since it was first version of that namespace whereas the version field for the "rcs-event" schema is "v2" since it was the second version of that namespace.

In addition to the dated schema URIs that are associated with each namespace, each namepace also has a set of undated schema URIs. These undated schema URIs have two distinct formats with slightly different meanings:

Form 2: "http://www.upnp.org/schemas/ms/" *schema-root-name* "-v" ***ver***

where ***ver*** is described above.

Form 3: "http://www.upnp.org/schemas/ms/" *schema-root-name*

Form 2 of the undated schema URI is always linked to the most recent release of the schema that represents the version of the namespace indicated by ***ver***. For example, the undated URI ".../av/rcs-event-v2.xsd" is linked to the most recent schema release of version 2 of the "rcs-event" namespace. Therefore, on May 31, 2006 (20060531), the undated schema URI was linked to the schema that is otherwise known as ".../av/rcs-event-v2-20060531.xsd". Furthermore, if the schema for version 2 of the "rcs-event" namespace was ever re-released, for example to fix a typo in the 20060531 schema, then the same undated schema URI (".../av/rcs-event-v2.xsd") would automatically be updated to link to the updated version 2 schema for the "rcs-event" namespace.

Form 3 of the undated schema URI is always linked to the most recent release of the schema that represents the highest version of the namespace that has been published. For example, on June 25, 2002 (20020625), the undated schema URI ".../av/rcs-event.xsd" was linked to the schema that is otherwise known as ".../av/rcs-event-v1-20020625.xsd". However, on May 31, 2006 (20060531), that same undated schema URI was linked to the schema that is otherwise known as ".../av/rcs-event-v2-20060531.xsd".

When referencing a schema URI within an XML instance document or a referencing XML schema document, the following usage rules apply:

- All instance documents, whether generated by a service or a control point, shall use Form 3.
- All UPnP Multi-Screen published schemas that reference other UPnP Multi-Screen schemas shall also use Form 3.

Within an XML instance document, the definition for the `schemaLocation` attribute comes from the XML Schema namespace "http://www.w3.org/2002/XMLSchema-instance". A single occurrence of the attribute can declare the location of one or more schemas. The `schemaLocation` attribute value consists of a whitespace separated list of values that is interpreted as a namespace name followed by its schema location URL. This pair-sequence is repeated as necessary for the schemas that need to be located for this instance document.

In addition to the schema URI naming and usage rules described above, each released schema shall contain a `version` attribute in the `<schema>` root element. Its value shall correspond to the format:

ver "-" ***yyyymmdd*** where ***ver*** and ***yyyymmdd*** are described above.

The `version` attribute provides self-identification of the namespace version and release date of the schema itself. For example, within the original schema released for the "rcs-event" namespace (…/rcs-event-v2-20020625.xsd), the `<schema>` root element contains the following attribute: `version="2-20020625"`.

4.3.3 Namespace Usage Examples

The `schemaLocation` attribute for XML instance documents comes from the XML Schema instance namespace "http://www.w3.org/2002/XMLSchema-instance". A single occurrence of the attribute can declare the location of one or more schemas. The `schemaLocation` attribute value consists of a whitespace separated list of values: namespace name followed by its schema location URL. This pair-sequence is repeated as necessary for the schemas that need to be located for this instance document.

Example 1:

Sample *DIDL-Lite XML Instance Document*. Note that the references to the UPnP AV schemas do not contain any version or release date information. In other words, the references follow Form 3 from above. Consequently, this example is valid for all releases of the UPnP AV service specifications.

```
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite
   xmlns:dc="http://purl.org/dc/elements/1.1/"
   xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
   xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="
      urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/
         http://www.upnp.org/schemas/av/didl-lite.xsd
      urn:schemas-upnp-org:metadata-1-0/upnp/
         http://www.upnp.org/schemas/av/upnp.xsd">
   <item id="18" parentID="13" restricted="0">
      ...
   </item>
</DIDL-Lite>
```

4.4 Vendor-defined Extensions

Whenever vendors create additional vendor-defined state variables, actions or properties, their assigned names and XML representation shall follow the naming conventions and XML rules as specified below.

4.4.1 Vendor-defined Action Names

Vendor-defined action names shall begin with "X_". Additionally, it should be followed by an ICANN assigned domain name owned by the vendor followed by the underscore character ("_"). It shall then be followed by the vendor-assigned action name. The vendor-assigned action name shall not contain a hyphen character ("-", 2D Hex in UTF-8) nor a hash character ("#", 23 Hex in UTF-8). Vendor-assigned action names are case sensitive. The first character of the name shall be a US-ASCII letter ("A"-"Z", "a"-"z"), US-ASCII digit ("0"-"9"), an underscore ("_"), or a non-experimental Unicode letter or digit greater than U+007F. Succeeding characters shall be a US-ASCII letter ("A"-"Z", "a"-"z"), US-ASCII digit ("0"-"9"), an underscore ("_"), a period ("."), a Unicode combiningchar, an extender, or a non-experimental Unicode letter or digit greater than U+007F. The first three letters shall not be "XML" in any combination of case.

4.4.2 Vendor-defined State Variable Names

Vendor-defined state variable names shall begin with "X_". Additionally, it should be followed by an ICANN assigned domain name owned by the vendor, followed by the underscore character ("_"). It shall then be followed by the vendor-assigned state variable name. The vendor-assigned state variable name shall not contain a hyphen character ("-", 2D Hex in UTF-8). Vendor-assigned action names are case sensitive. The first character of the name shall be a US-ASCII letter ("A"-"Z", "a"-"z"), US-ASCII digit ("0"-"9"), an underscore ("_"), or a non-experimental Unicode letter or digit greater than U+007F. Succeeding characters shall be a US-ASCII letter (“A”-“Z”, “a”-“z”), US-ASCII digit (“0”-“9”), an underscore (“_”), a period (“.”), a Unicode combiningchar, an extender, or a non-experimental Unicode letter or digit greater than U+007F. The first three letters shall not be “XML” in any combination of case.

4.4.3 Vendor-defined XML Elements and attributes

UPnP vendors may add non-standard elements and attributes to a UPnP standard XML document, such as a device or service description. Each addition shall be scoped by a vendor-owned XML namespace. Arbitrary XML shall be enclosed in an element that begins with “X_,” and this element shall be a sub element of a standard complex type. Non-standard attributes may be added to standard elements provided these attributes are scoped by a vendor-owned XML namespace and begin with “X_”.

5 Multi-Screen Architectural Overview

In most UPnP scenarios, a control point controls the operation of one or more UPnP devices in order to accomplish the desired behavior. Although the control point is managing multiple devices, all interactions occur in isolation between the control point and each device. The control point coordinates the operation of each device to achieve an overall, synchronized, end-user effect. The individual devices do not interact directly with each another. All of the coordination between the devices is performed by the control point and not the devices themselves.

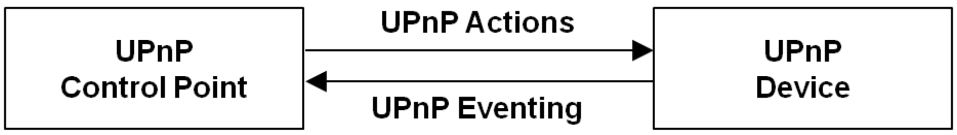

Figure 1 – Typical UPnP Device Interaction Model

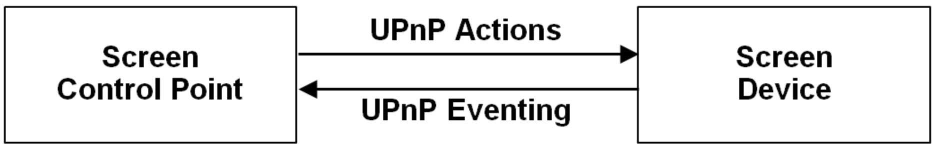

Figure 2 – UPnP Multi-Screen Interaction Model: SD and SCP

5.1 Screen Device

A UPnP component used to provide various interactive services with other display device(s) which is (are) implemented with Screen Control Point(s). Designed to be controlled by and interact with Screen Control Point(s) as Figure 2.

In order to support basic multi-screen services, it is encouraged to be implemented for a *Main screen device*, e.g., a lean-back display device such as a TV or set-top box which is controlled by *Companion screen devices* as Figure 3.

In order to support extended multi-screen services which require more sophisticated interactions, it is encouraged to be implemented also for a *Companion screen device*, e.g., a lean-forward & handheld display device such as a smart phone or tablet which controls *Main screen devices* as Figure 4.

See [2], [3], [4] and [5] for details of the associated services and the requirements.

5.2 Screen Control Point

A UPnP component used to provide various interactive services with other display device(s) which is (are) implemented with Screen Device(s). Designed to control and interact with Screen Device(s) with direct input from end-users as Figure 2.

In order to support basic multi-screen services, it is encouraged to be implemented for a *Companion screen device*, e.g., a lean-forward & handheld display device such as a smart phone or tablet which controls *Main screen devices* as Figure3.

In order to support extended multi-screen services which require more sophisticated interactions, it is encouraged to be implemented also for a *Main screen device*, e.g., a lean-back display device such as a TV or set-top box which is controlled by *Companion screen devices* as Figure 4.

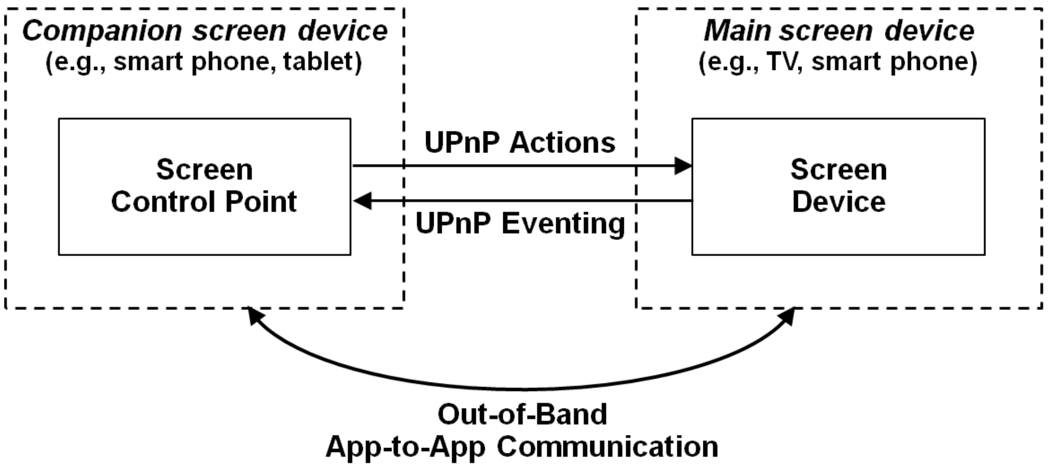

Figure 3 – Basic Multi-Screen Architecture

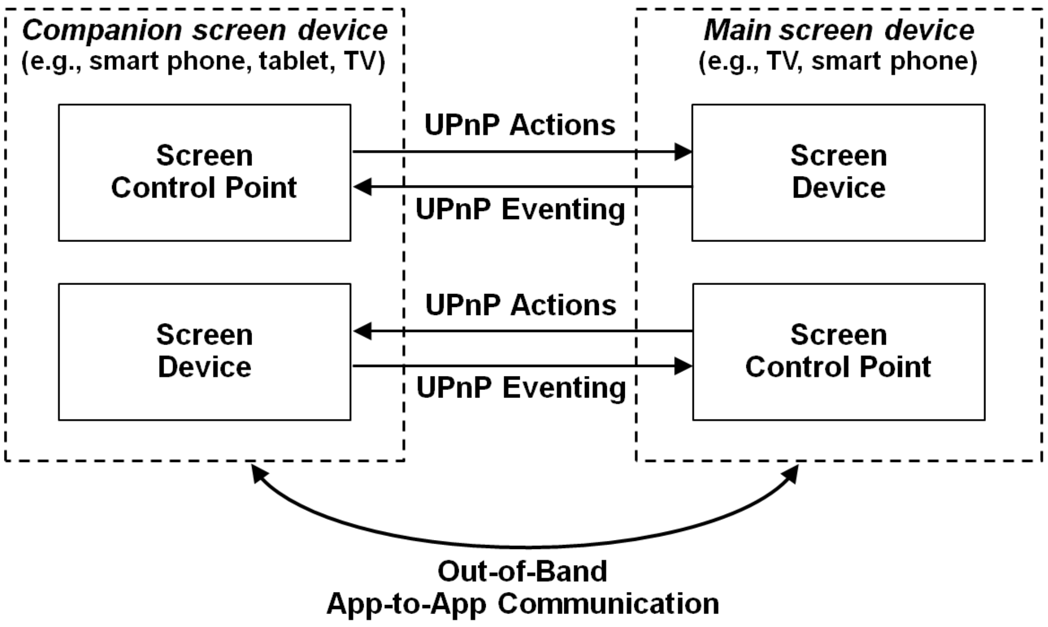

Figure 4 – Extended Multi-Screen Architecture