ANSI E1.17-2015, Architecture for Control Networks

EPI 19.

ACN Discovery on IP Networks

Document number: CP/2013-1018r4

Part of ANSI E1.17-2015

This is a revision of EPI 19 as published in ANSI E1.17-2010.

Abstract

This EPI specifies the Service Location Protocol [SLPv2] implementation requirements for individual components within an E1.17 system necessary to provide discovery on IP networks. It does not provide requirements or recommendations for system wide deployment of SLP such as Scoping or Directory Agent usage.

Foreword – ACN EPIs

ANSI E1.17-2010 is the “Architecture for Control Networks” standard [ACN]. It specifies an architecture – including a suite of protocols and languages that may be configured and combined with other standard protocols in a number of ways to form flexible networked control systems.

E1.17 Profiles for Interoperability (EPIs) are standards documents that specify how conforming implementations are to operate in a particular environment or situation in order to guarantee interoperability. They may specify a single technique, set of parameters or requirement for the various ACN components. They may also specify how other standards (including other EPIs) either defined within ACN or externally are to be used to ensure interoperability.

Note: ESTA became PLASA North America on 1 January 2011. PLASA NA continues ESTA's standards work and continues to support the ESTA protocols.

1. Introduction

Service Location Protocol version 2 provides a generalized mechanism for discovery of services on Internet Protocol (IP) networks. [SLPv2bis] is a simplification of SLPv2 that omits some of the little used and more complex features. In particular predicates use a very simplified form of LDAP search filter and various other features that are difficult to parse or implement have been dropped.

SLP2 is defined in a number of documents. Two main standards specify both a protocol for exchange of discovery information [SLPv2] and Template and Service schemes that define the characteristics and format of that information [SLPtemplate]. An Internet Draft [SLPv2bis] specifies simplifications and revisions that shall apply in ACN systems. Additional documents specify related standards that may be used in E1.17 systems but are not normative to this EPI.

SLPv2bis is used to discover components available within an ACN system. Abstract service types are defined for SDT [SDT] ad-hoc address discovery and DMP device discovery. Additionally ACN components must implement the serviceid scheme that allows discovery of all ACN components by CID and browsing.

This EPI specifies the requirements for implementation of SLP2 by ACN components that enables it to be used to provide all necessary discovery functions.

1.1. SLP Summary

This summary is intended as a guide only and is not normative.

1.1.1. Service Agents, User Agents and Directory Agents

SLP defines Service Agents (SAs), User Agents (UAs) and Directory Agents (DAs).

In ACN terms, Service Agents are components with services to offer e.g., devices. User Agents are components seeking to discover those services e.g., controllers.

Directory Agents are optional servers that cache service information. If one or more Directory Agents are present, then Service Agents register their services with them and User Agents discover services from Directory Agents rather than directly from Service Agents. Directory Agents make SLP scalable to very large networks.

1.1.2. Service Types and URLs

Categories of services are identified by a Service Type string that can become the initial part of a URI (Uniform Resource Identifier). Examples of service type include “service:printer:ipp” or “service:acn.esta”. Service types may be abstract (“service:printer”) or concrete (“service:printer:ipp”) – the latter identifies only printers accessed via Internet Printing Protocol.

In operation, UAs issue service requests for services of a particular type, abstract or concrete. The reply to a service request is some number of service replies, each of which contains a number of URIs identifying services of the requested type.

1.1.3. Attributes and Predicates

As part of the definition of any service type, a number of attributes may be defined. Each attribute associates a name with a value or list of values. The values of these attributes will vary from one instance of the service to another.

Service Requests may include predicates that restrict the service requested, not only to the type specified, but also to instances whose attributes match the predicated rules. Predicates include exact matches (CID=5124409a-e7d7-4142-b565-ac481bb63e98), wildcard matches (fctn=acme lighting*), and comparisons (uptime<200).

Attributes may also be requested from individual services either by using a secondary attribute request exchange once the service has been discovered, or by using an attribute request extension in the initial service request.

1.1.4. Previous Responders

Since service requests can give rise to large numbers of responses, a mechanism is included to allow incremental discovery.

Each Service Request includes a previous responder list (“PRlist”) that identifies by URI all services of the requested type that are already known about. If an SA sees its own URI in the PRlist, it does not reply. Similarly a DA does not include services in the PRlist in its replies.

1.2. ACN’s use of SLP

1.2.1. General

The ACN architecture presents an intermeshed suite of protocols that has the potential to expand. Representing each protocol or feature as a separate service in SLP tends to lead to a messy ad-hoc

development of new service types with difficulties for discovery of particular subsets that User Agents are likely to require.

For this reason, the mechanism presented here groups all ACN components under a single service type and uses simple attributes to indicate the subsets of functionality that a particular component can offer. This allows discovery to be flexible based on quite simple predicates, and extension by addition of extra attributes as required.

In general, to keep predicates simple, it is preferable to define many simple attributes rather than few complex ones: for example, rather than having a single attribute containing a list of features supported, there should be one attribute that may or may not be present for each feature. The addition of extra feature attributes should not present a problem for User Agents that are unaware of them, since those User Agents are unlikely to be able to take advantage of the advertised feature either.

The approach used for ACN is to define a single SLP service type – the ACN component identified by its CID. Specific functionality within components is then handled by defining component level services (called ACN services). An example of an ACN service would be a device exposed by the component and accessed using DMP over SDT. For each ACN service a set of service attributes must be defined that provide the information necessary to access that functionality within the component and which facilitate discovery of components offering that ACN service using simple predicate searches.

This is somewhat in keeping with the spirit of the serviceID scheme proposed in [serviceID] in which a UUID indicates a single service that may be discovered independently of the protocol(s) used to access it and of its address (which may change over time or according to where it is discovered from). However, the exact syntax of the serviceID scheme has been found to be difficult to work with and has not been followed in this Standard.

1.2.2. SLP Version 2 bis

SLP Version 2 is a simplification of Version 1. Version 2 is itself further simplified in an Internet Draft [SLPv2bis] that greatly reduces the complexity of predicates and attribute management.

This EPI Follows all the recommended changes of [SLPv2bis].

2. Protocol Requirements

Refer to SLP protocol definition [SLPv2] as modified by [SLPv2bis].

2.1. Service Agents (SAs)

2.1.1. General

For each ACN service identified, it shall be stated clearly whether discovery of that service using SLP is mandatory, optional or forbidden. For mandatory discovery, all components offering that ACN service shall implement a Service Agent as defined in [SLPv2bis]. For optional discovery, all components offering that ACN service should implement a Service Agent as defined in [SLPv2bis].

2.1.2. Note on LDAP Search Filters (Predicates)

SLP Service Requests include Predicates that are simple filters allowing searches to match particular combinations of attributes. Because attributes defined for ACN services are mostly simple text strings, filters will be simple although they may be long.

Approximate matching (“~=”) is intended to be based on soundex or phonetic matching but is not well defined. If no soundex or phonetic matching is available, approxMatch shall be treated identically to equalityMatch.

In compliance with [SLPv2bis], only single level “&” matching of multiple comparisons is required and spaces are matched as supplied without folding.

E.g., (& (fctn=Acme Lighting Bendilite)(csl-esta.dmp=*))

2.2. User Agents (UAs)

User Agents must be implemented by any host that needs to discover services. Any component that needs to find other components that provide a particular ACN service is required to implement a User Agent as defined in [SLPv2] and [SLPv2bis].

2.3. Directory Agents (DAs)

Directory Agents are an optional feature of SLP2 that provides for cacheing of service information giving improved efficiency and scalability. DAs are not required in ACN systems but provide improved performance and scalability in larger networks. DAs should implement mesh enhancement [MeshSLP]. It is recommended that a DA be provided in any large ACN network that provides centralized management features such as DHCP servers and time servers.

2.4. Use of TCP

SLP2 specifies that TCP (rather than UDP) is required once the size of the PRlist in each request becomes sufficiently large that packets will overflow MTU.

This document provides a mechanism for arbitrarily dividing the search space based on component IDs so that in large systems, discovery can be broken up into smaller pieces. This can prevent the PRlist growing too large and allows implementations with no TCP requirement Section 6, “Division of Searches”.

2.5. Scoping

All services in SLPv2 are scoped. Scoping provides a mechanism to create administrative groupings of services. The default scope for ACN services shall be “ACN-DEFAULT”.

3. Component Name Strings

Each component shall maintain two text identifier strings intended to indicate the function of the component in human readable terms for browsing purposes. These identifiers are not suitable for automated identification of devices since they are not assigned or “policed” by any authority and so cannot be guaranteed unique:

3.1. Fixed Component Type Name (FCTN)

This shall be a UTF-8 [Unicode] string that is assigned during manufacture and is not alterable. It should indicate the type of component e.g., “Acme Lighting Bendilite mkII”. There is no restriction placed on the length of the FCTN string, but components receiving this string, whether by discovery or other methods, may truncate values longer than 63 octets.

3.2. User Assigned Component Name (UACN)

This shall be a UTF-8 [Unicode] string that may be assigned by the user. The UACN shall be maintained in persistent storage and shall be capable of storing at least 63 octets of UTF-8 [Unicode] characters.

The method for the user to assign the UACN is unspecified but methods include, assignment via a DMP property, local assignment (e.g., via a front panel) or assignment via some other remote configuration interface (e.g., a web page).

In the absence of another assigned name, the UACN may be used as a hostname as described below (see:).

4. Service Types (Service: Scheme) and Identifiers

Syntax Notation

This specification uses the Augmented Backus-Naur Form (ABNF) notation of [ABNF] including the following core ABNF syntax rules defined by that specification: ALPHA (letters), CR (carriage return), DIGIT (decimal digits), DQUOTE (double quote), HEXDIG (hexadecimal digits), LF (line feed), and SP (space).

Line Splitting in Examples

In many examples given through this Standard, attribute values, URLs and ACN locators have been split onto multiple lines to facilitate formatting. In many cases, this addition of whitespace characters would not be allowed in practice.

SLP2 services are requested by type and resulting services are returned as a URI. ACN components use the service: scheme as defined in [SLPtemplate], with the service type “acn.esta” and with an ACN component service locator identifying the component, to complete the URL.

The full syntax of the ACN service URI shall also conform to the following syntax:

```
acn-service-URL   =  "service:" component-locator
                        ; see Section 7.2, "ABNF for ACN Locators" below
```

This syntax is consistent with the syntax given for “service: URL” as defined in [SLPtemplate] restricted as follows:

```
service-type    =  "acn.esta"
hostport        =     ;empty - host is discovered by csl- attributes
url-path        =  "/" cid
                        ; see Section 7.2, "ABNF for ACN Locators"
                        ; below for cid
```

Example 1. Example of ACN service: URI

```
service:acn.esta:///7a4def1c-9bdc-11da-988c-000d613667e2
```

The location of the service using a given protocol is provided by other attributes of the service whose definitions must identify a URL format that provides sufficient information to access the service using the given protocol.

4.1. Identification of IP Hosts

The resource location schemes for ACN services allow use of either domain names or literal IP addresses for indicating IP hosts. Implementers should be aware of issues with both these methods.

4.2. Host Names

There is no requirement in ACN systems to support the Domain Name System, so even if Service Agents do have a domain name of their own available, there is no guarantee that a User Agent will be able to resolve this name. Therefore components shall advertise their component services using ip-literal notation. They may advertise the same service using a DNS name as well.

4.3. IP Addresses

Directly specified IP addresses do not need resolution and are therefore more likely to work. But establishing the best IP address to advertise is not easy with some stacks and configurations. This is particularly true when multiple addresses are available and some may be more accessible than others. There is discussion of these issues in [IPv4LL]. Where there is any doubt the service should be advertised multiple times, once for each address.

Literal IP addresses shall be specified in accordance with [URI]. For example, 192.168.99.2 (an IPv4 address); [12ab:0:0:cd30:123:4567:89ab:cdef] (an IPv6 address).

5. Example Implementation

A controller (e.g., a console) comes online, acting as a “SLP User Agent” (UA). It first looks for a Directory Agent (DA) as SLP requires using a service Request for “service:directory-agent”. If a DA is found, the subsequent requests are all unicast to the DA. Otherwise they are multicast.

In either case the controller now issues a service request for “service:acn.esta”. This results in a list of service URIs that are the CIDs of ACN components. The controller can immediately use the CID to identify the specific components it has communicated with previously (if it remembers such things).

For each CID discovered, the controller then issues an attribute request for that CID. The “csl-esta.dmp” attribute returned identifies a list of locators for DMP access, including device and/or controller capability and the root level DCIDs.

Additional attributes identify the FCTN and UACN and other information. The console can use the DCID reported to determine the type of device (note that a component may change the device it exposes from session to session so the DCID advertised should always be examined). The console can use that type to determine if it wants to communicate with the device. If the DCID reported is not one the console is familiar with, then the console can request the device description language (DDL) definition of that type from the device to learn about the device.

An alternative strategy used by another controller might be to search immediately for those components it has already been configured to control using the full service URI for each. If a component is missing this controller then searches specifically for other components of identical type using a predicate to specify the DCID of the device type it is seeking e.g. (csl-esta.dmp=*d:28ed9ffc-c40c-435c-a3c5-a722797d766f*).

6. Division of Searches

As well as being a part of the service URI, the CID of a component advertising itself is a required attribute. By predicating searches on a partial CID using a wildcard, the search space may be divided up. In the presence of large numbers of devices, this technique allows discovery to be partitioned in large systems to always ensure that the PRlist in service requests does not overflow MTU and means that use of TCP can be avoided.

For example, if searches only on the type “service:acn.esta” are generating too many responses, the search may be divided by adding a predicate matching (cid=0*), (cid=1*) etc. this breaks the search into 16 sub-searches. Predicating on the first two characters of the CID (cid=00*) divides the search space into 256 sub-searches and so on.

To facilitate this mechanism ACN services shall use the format specified in [UUID] for string representation of UUIDs.

7. Component Service Locators for ACN

Each ACN component service advertised requires one or more locators specifying sufficient information to access the service or part of the service. For resources accessed using ACN protocols the following locator structure is defined. Note that these locators are defined in an ad-hoc manner for specific uses and while their format is often similar to URIs in common with [newURI], these locators do not necessarily have generic applicability or comply with Internet standards outside of the SLP discovery context.

Because ACN protocols commence above the UDP layer but potentially include multiple layers above this, the URL scheme traces a path through the protocol stack including as much detail as is required to access and understand the next protocol layer at each step. Each protocol in turn, starting with the lowest ACN protocol above the root layer protocol (e.g. SDT) is listed by its PLASA registered name [ESTA-IDs], and is followed by sufficient information to access it. each protocol section is separated using a semicolon ";" from preceding sections and the protocol name is followed by access information as necessary using "/". Note that access information may incorporate further "/" characters but should not incorporate ";".

An absolute resource locator must begin by identifying the ACN scheme and the component ID as above. Following this, the first protocol_name shall be the lowest ACN protocol above the root layer protocol e.g. "esta.sdt".

Subsequent protocol_names and access information shall be defined as required for each protocol.

All protocol names within the ACN architecture shall use the name as defined in [ESTA-IDs].

Locators for ACN services accessed using other protocols shall use established URL format appropriate to that protocol.

Example 2. example of absolute ACN resource locator

(This locator has been folded by the addition of newlines.)

```
acn.esta:///7a4def1c-9bdc-11da-988c-000d613667e2;
  esta.sdt/10.0.0.10:12324;
  esta.dmp/cd:28ed9ffc-c40c-435c-a3c5-a722797d766f
```

Note

In the "acn.esta" scheme the naming authority "esta" comes last because it is governed by the rules of [SLPv2], while in ACN protocol names, "esta" comes first because it is governed by the rules of [ESTA-IDs].

7.1. Relative Locators

Within a known context (for example where the ACN context is established and the CID is known), relative locators may be formed by splitting the locator at any ";" separator and omitting the text up to and including that separator.

7.2. ABNF for ACN Locators

ACN URLs shall fit the following ABNF using the syntax of [ABNF]. Portions of this ABNF correspond identically to [URI].

```
ACN-locator        =   absolute-locator / relative-locator
absolute-locator   =   component-locator [ ";" relative-locator ]
component-locator  =   acn-scheme scheme-delimiter cid
acn-scheme         =   "acn.esta"
```

```
scheme-delimiter   =   ":///"
  ; Used for compatibility with open SLP
cid                =   UUID
  ; cid is the UUID of the ACN component
relative-locator   =   protocol-spec [ ";" relative-locator ]
protocol-spec      =   ProtocolName [ "/" ProtocolAccess ]
protocol-access    =   *( unreserved / reserved / pct-encoded )
; the following productions are externally defined
; see  for information
unreserved   =  ; unreserved characters
reserved     =  ; The reserved delimiter characters may have
                ; specific meanings delimiting structural
                ; components of locators. Note that in
                ; accordance with [SLPv2] some of these
                ; characters may need to be escaped when they
                ; occur in attribute values and queries.
                ; In particular see [SLPv2] section 5.
pct-encoded     ; Percent encoding allows use of arbitrary unicode
                ; characters using UTF-8 encodings. It also allows
                ; escaping of characters which would otherwise
                ; have special meanings
UUID            ; A unique identifier
ALPHA           ; letters
DIGIT           ; digits
HEXDIG          ; hexadecimal digits
ProtocolName    ; an ACN protocol name
```

7.3. Specific Protocol Locator Formats

The following resources are located using the “acn.esta:” scheme and correspond to the defined service types. Protocols within the ACN architecture shall be identified using the protocol names assigned directly or indirectly by PLASA in accordance with [ESTA-IDs].

7.3.1. SDT

7.3.1.1. General

The URL identifies an SDT ad-hoc address that may be used for SDT Join and other ad-hoc requests. The <host_location> field shall include both a network host and a port since there is no standardized port for SDT ad-hoc connections. No further access information is required. For example:

```
esta.sdt/10.0.0.20:2345           ;an IPv4 SDT ad-hoc address
```

7.3.1.2. ABNF for SDT Service Locator

```
; This ABNF extends Section 7.2, "ABNF for ACN Locators"
; Note for portability, all components are case insensitive
sdt-protocol-spec      =  sdt-ProtocolName "/" sdt-protocol-access
sdt-ProtocolName       =  "esta.sdt"
sdt-protocol-access    =  host-location
host-location          =  host [ ":" port ]
; the following productions are externally defined
; see Appendix A, ABNF for External Definitions for information
```

```
host                ; a network host
port                ; a TCP or UDP port
```

7.3.2. DMP

7.3.2.1. General

DMP components may have two distinct functionalities. Devices expose properties, respond to get/set property messages and potentially generate events; Controllers generate get/set property messages and subscribe to events. Many components have both functionalities, but this is not a requirement. The DMP URL may indicate which functionalities are implemented or referred to by a “C” and or “D” character in the protocol_access string. A DMP device URL may identify the DCID of the single root level device exposed by the component. A fragment identifier on a DMP URL identifies a property or property group within the abstract device model of the device as expressed in its device description.

Examples of DMP URLs

```
esta.dmp/CD:28ed9ffc-c40c-435c-a3c5-a722797d766f
```

A component with controller capability and that exposes a root device with DCID 28ed...

```
esta.dmp/c
```

A DMP controller – no device is exposed.

7.3.2.2. ABNF for DMP Service Locator

```
; This ABNF extends Section
; Note for portability, all components are case insensitive
dmp-ProtocolName       =  "esta.dmp"
;
; A DMP component must have either
; controller or device functionality or both.
;
dmp-protocol-access    =  ctl-access
                        / dev-access
                        / ( ctl-access dev-access )
ctl-access             =  "c" / "C"
dev-access             =  ("d" / "D") ddl-access
ddl-access             =  ":" root-dcid
root-dcid              =  UUID
; the following productions are externally defined
; see  for information
UUID                 ; a Universally Unique Identifier
```

8. Rules for ACN Component Services

Within a component each top level (application layer) protocol constitutes a component service and is expected to introduce its own attributes and resource locator format. Intermediate level protocols such as SDT do not generally need to be declared separately but will appear within the locators for the higher layer protocols they support. For each component service rules shall be provided stating when components are required, allowed or forbidden from advertising that service and what information must or should be provided.

Component service advertisements shall identify the location(s) of the service in the “csl-X” attributes, where X is the name of the application layer protocol e.g. “csl-esta.dmp”.

The value of a “csl-X” attribute shall be a list of one or more locators specifying sufficient information to access the service or part of the service. For resources accessed using ACN protocols the locator structure shall be a relative locator as defined in commencing at the first (outermost) protocol but with the ACN scheme and CID omitted (these are clearly established by the context of the service advertisement).

8.1. All Components

All ACN components that are required to advertise using SLP shall provide the following attributes:

cid

The Component Identifier. This is the same as in the service advertisement but allows it to be used as a predicate when searching for services.

acn-services

This is a list of component services in the form of application layer protocol names. For each protocol name listed within this attribute, there shall be a corresponding “csl-...” attribute detailing the access methods for that service.

acn-fctn, acn-uacn

These are the component name strings as defined above Section 3, “Component Name Strings”. Their primary intention within SLP is for browsing and network management.

In addition, all ACN components may expose the following attribute:

version

This is a string indicating the software version of the ACN component.

8.2. Device Management Protocol (DMP)

8.2.1. Component Service location Attribute

Components using DMP can implement two distinct functionalities controller and device. Many components will implement both but this is not a requirement. Both controller and device functionality are advertised using a single csl-esta.dmp attribute.

A device is defined as a controllable entity composed of a hierarchy of properties as describable by a device description in [DDL]. Within the ACN architecture a component may expose a single root level device together with all its sub-devices that constitutes the entire functionality of the component that is controllable using the property metaphor of DDL.

Note
A component can implement a device that has no actual DMP properties, provided that it declares and serves a device description in accordance with [DDL] and [DDLretrieval], (for example, a component could declare a set of fixed DDL properties such as manufacturer, model, version and other information in its description). Such a component would however be required to respond to control messages such as get-property with appropriate failure responses.

All ACN components containing DMP devices that fall within the scope of this EPI shall advertise their devices as described here.

A controller is essentially opaque as it is normally the component that seeks devices to control. It issues get/set-property messages and subscribes to events.

Note
A controller does not provide any service that can be accessed in the normal sense – by definition, attempting to establish a DMP connection to a controller would either be refused or would actually connect to a device within the same component. However, discovery of controllers does have uses in network and system management and diagnostics (including querying them at the SDT level using the get-sessions message).

All ACN components containing DMP controllers that fall within the scope of this EPI shall advertise their controller functionality as described here. Any ACN component that advertises device functionality and that also implements controller functionality shall advertise that controller capability as part of the same advertisement.

Example 3. DMP service advertisement

```
Service URI
   = "service:acn.esta:///6A6DCA18-87A5-4403-8A42-F1B1BA979F67"
Attributes:
   (csl-esta.dmp=esta.sdt/10.0.1.234:54321;
                 esta.dmp/cd:379a882e-01d4-4979-b1f2-028bbf345b80)
```

The component with CID = 6A6DCA18-87A5-4403-8A42-F1B1BA979F67 has both controller and device functionality and the device's root DCID is 379a882e-01d4-4979-b1f2-028bbf345b80.

8.2.1.1. Component Service Location Values attributes

Components may include the **csl-esta.dmp.values** attribute in order to expose the string equivalents of values contained in individual DMP properties. This attribute shall consist of one or more key:value pairs of the form <DMP address>:value, delimited by commas.

Example 4. DMP value exposition

```
Attributes:
        (csl-esta.dmp.values=1337:ACME_ROX,2:10,82:19)
```

Note
Any change to the attributes exposed over SLP requires that the component re-advertise. Because of this, a level of thoughtfulness is required when choosing DMP properties to be added to the discovery string. Care must be taken not to compromise the stability of SLP with constantly oscillating values, or with overly delayed reporting of value changes.

8.2.2. Description Location attribute

The attribute "device-description" enables components to advertise the location and method to retrieve the DDL description(s) of their devices (root and sub-devices). Specific protocols may impose requirements on how descriptions must be made available, for example DMP devices within the scope of [DDLretrieval] are required to make their descriptions available using TFTP, but this attribute provides a more general way to advertise descriptions and to provide information on alternative locations or methods for access.

All ACN components exposing devices accessible via DMP shall provide a device-description attribute whose value(s) shall declare all mandatory methods by which the DDL can be retrieved and may declare additional methods for retrieval.

Each value of this attribute specifies a location and method to access one or more descriptions using the syntax: `<DCID>:<URL>` where `<DCID>` is the UUID of the DDL module and `<URL>` specifies a locator for the module.

For example a value

```
e5949c4f-3c70-49d5-b7bc-044061e0c9d8:
    tftp://85.74.63.52/e5949c4f-3c70-49d5-b7bc-044061e0c9d8.ddl
```

indicates that the description for device with DCID “e5949c4f-3c70-49d5-b7bc-044061e0c9d8” is accessible at the location given. The same device description might also be advertised using

```
e5949c4f-3c70-49d5-b7bc-044061e0c9d8:
    http://www.example.com/ddl/mydevice/root.xml
```

which shows that the description may also be retrieved by HTTP from a generic Internet address.

8.2.3. Symbolic values for DCIDs

Since a component may contain many devices whose DCIDs are long strings and those strings are often repeated in the URL part as well as the DCID part of the “device-description” attribute, the symbol “$” may be used to represent a UUID string as follows:

- When used in the `<DCID>` part of an attribute value, “$” shall mean the DCID of any DDL module that the component makes available (including all those it is required to serve under [DDLretrieval]).
- When used in the `<URL>` part of an attribute value, “$” shall mean the same DCID that is specified in the DCID part.

Example 5. Examples of Use of $ Symbol

```
e5949c4f-3c70-49d5-b7bc-044061e0c9d8:
ftp://85.74.63.52:10021/subdevices/dev-$.xml
```

The DDL module “e5949c4f-3c70-49d5-b7bc-044061e0c9d8” may be retrieved by FTP from 85.74.63.52 port 10021 using the path and filename “/subdevices/dev-e5949c4f-3c70-49d5-b7bc-044061e0c9d8.xml”

```
$:tftp://85.74.63.52/$.ddl
```

This declaration indicates that the description for any sub-device, languageset or behaviorset within the component may be retrieved by TFTP using the filename “dcid.ddl” where dcid is the DCID of the device. Assuming that 85.74.63.52 is the component’s own IP address, this corresponds exactly with the requirements of [DDLretrieval].

8.2.4. Subdevices

An ACN component that contains subdevices may optionally express the presence of any or all of them by using the **csl-esta.dmp.subdevices** attribute. Subdevices shall be presented as a comma-delimited list of the form `<DCID>:<DMP start address>`. If multiple subdevices of the same type are to be exposed, they may continue to be appended to the DCID with colons. To avoid ambiguity, the “$” shall not be used in this attribute.

*Example 6.Use of **csl-esta.dmp.subdevices***

```
Attributes:
   (csl-esta.dmp.subdevices=AACBA029-F87A-6292-B1BF-62CF00D62351:1,
                            499A2C0D-07F3-47FA-8E80-1300BC7D0052:101)
```

9. Accessibility Issues and Link-local Addresses

9.1. SLP With Link-local Addresses

When Link-local addresses [IPv4LL] are used (as referenced in EPI13 and EPI29), all requirements of [IPv4LL] regarding advertising of Link-local addresses apply to service advertisements using SLP as specified in this EPI.

This means that if a host conforming to this EPI has advertised services using a Link-local address and a routable address has become available, in addition to transitioning to using that routable address, the host must de-register any services that use the Link-local address and that were previously registered either with the local service Service Agent (SA) or with a Directory Agent (DA), and re-register as necessary using the routable address. See [IPv4LL] for detailed consideration on transition between addresses.

Despite this, it is possible that an SA using a Link-local address advertises to a DA on the same link that then passes that advertisement to hosts on other networks. Those implementing DA code should be aware of this issue and may choose not to forward service advertisements that were received from Link-local hosts, to off-link User Agents.

A User Agent (UA) must be aware that not only may services using Link-local addresses (or private network addresses) be inaccessible, but that there may, by coincidence, be similar services in the local context using the same address. A UA can guard against this case by checking whether the DA is on the same link as itself (for example, by examining its routing tables to see if the DA needs to be accessed via a router). When the DA is on a different link, the UA should ignore any advertisement from it that specifies a Link Local address.

9.2. General Accessibility Problems

In any network – particularly more complex routed ones, a discovery search for ACN components may find ones that are not actually reachable. Reasons include asymmetric routing, access policies (firewalls, authentication), Network Address Translation, and many kinds of software failure. The only truly failsafe test is to attempt to connect to a discovered component and check for correct CID (e.g. by attempting to join it to an SDT session).

10. Example Device Advertisement

In this example, the same component is advertising a root device accessible via DMP. This device is also accessible via TCP/IP using the fictitious protocol “acme_ltg.aop” at port 666. The DCID is the same in both cases indicating that they share a common device description for both protocols. The descriptions are available by tftp but may also be retrieved using ftp from the Internet.

```
Service type requested = service:acn.esta
Service URI = service:acn.esta:6A6DCA18-87A5-4403-8A42-F1B1BA979F67
Attributes =
  (cid=6A6DCA18-87A5-4403-8A42-F1B1BA979F67),
  (acn-fctn=FooBar inc, Baz mk II),
  (acn-uacn=The Baz in the cellar),
  (acn-services=esta.dmp,acme_ltg.aop),
  (csl-esta.dmp=esta.sdt/10.0.0.20:2146;
               esta.dmp/cd:8F594875-FDA6-496a-9DE2-8349E9C91DA7),
  (device-description=
    $:tftp://10.0.0.20/U$.ddl,
```

```
    $:ftp://ftp.foobar.com/public/ddl/modules/mod-$.xml)
  (csl-acme_ltg.aop=
    acme_ltg.aop://10.0.0.20:666/8F594875-FDA6-496a-9DE2-8349E9C91DA7),
  (csl-esta.dmp.values=1337:ACME_ROX,2:10,82:19),
  (csl-esta.dmp.subdevices=AACBA029-F87A-6292-B1BF-62CF00D62351:1,
                           499A2C0D-07F3-47FA-8E80-1300BC7D0052:101),
  (version=1.2.0.9.0.9)
```

11. SLP Template for acn.esta

```
Name of submitter: "PLASA Technical Standards Manager"
<standards@plasa.org>
Language of service template: en
Security Considerations:
For security discussion see
ANSI ESTA E1.17-2010 Architecture for Control Networks
Template Text:
-------------------------template begins here----------------------
template-type=acn.esta

template-version=1.0

template-description=
This is a concrete service type. The purpose of the acn.esta
service type is to identify ACN components and their subsidiary
services as defined by the ACN Architecture.

template-url-syntax=
url-path            =  cid
cid                 =  8 HEXDIG "-" 4 HEXDIG "-"
                       4 HEXDIG "-" 4 HEXDIG "-" 12 HEXDIG
# A cid is a UUID with strict text encoding.
# this is specified in RFC4122

cid=string L
# This is a text rendering of the unique identifier.
# It contains the same value that appears in the
# "service:acn.esta" URI.

acn-fctn=string
# The Fixed Component Type Name
# See ANSI E1.17: EPI-19. Discovery on IP networks

acn-uacn=string L
# The User Assigned Component Name
# See ANSI E1.17: EPI-19. Discovery on IP networks

acn-services=string MLX
# A list of application layer ACN protocol names for component
# services being advertised. For each item "xxx" in this list there
# shall be a corresponding "csl-xxx" attribute
```

```
# the following apply for components advertising the DMP protocol

version=string LO
# An identifier for the version of the component

csl-esta.dmp=string ML
# The acn-resource locator(s) required to access any DMP device which
# is exposed by the component. A component may only expose zero or
# one devices but that device may be accessible via multiple
# protocols or multiple addresses therefore this is a locator list.

csl-esta.dmp.values=string MLO
# An optional list of key:value pairs that each indicate an absolute DMP
# address and a string representation of the value that property takes.
# This may contain one or more entries, but all DMP properties need not be
# exposed in this manner.

device-description=string ML
# This is a list of URLs indicating where the device
# Description Language description of devices may be
# retrieved. The rules for device access under different protocols
# may indicate specific requirements for how descriptions must
# be made available. This attribute must reflect such rules
# but may also indicate availability by other means or from other
# places.

csl-esta.dmp.subdevices=string MLO
# An optional list of key:value pairs that each contain the DCID of an ACN
# subdevice and its associated DMP start addresses. Multiple DMP addresses
# may be used where there are multiple occurrences of the same
# subdevice. This list need not include all subdevices present on the
# device.
# The format of each string vlaue is:
# subdevice-string = DCID ":" address-list
# DCID             = UUID
# address-list     = address / address ":" address-list
# address          = * DIGIT
------------------template ends here-------------------------
```

Annex A ABNF for External Definitions

The following ABNF productions are defined externally. They are reproduced here for informative reference only, the normative reference being the original documents. The versions here may have been modified in a few cases to give consistency of notation.

A.1 Productions from [ABNF]

```
ALPHA          =  %x41-5A / %x61-7A ; A-Z / a-z
DIGIT          =  %x30-39 ; 0-9
HEXDIG         =  DIGIT / "A" / "B" / "C" / "D" / "E" / "F"
; NOTE: since ABNF strings are case-insensitive so is HEXDIG
```

A.2 Productions from [URI]

```
host          =  IP-literal / IPv4address / reg-name
IP-literal    =  "[" ( IPv6address / IPvFuture ) "]"
IPvFuture     =  "v" 1*HEXDIG "." 1*( unreserved / sub-delims / ":" )
IPv6address   =                            6( h16 ":" ) ls32
              /                         "::" 5( h16 ":" ) ls32
              /  [               h16 ] "::" 4( h16 ":" ) ls32
              /  [ *1( h16 ":" ) h16 ] "::" 3( h16 ":" ) ls32
              /  [ *2( h16 ":" ) h16 ] "::" 2( h16 ":" ) ls32
              /  [ *3( h16 ":" ) h16 ] "::"    h16 ":"   ls32
              /  [ *4( h16 ":" ) h16 ] "::"              ls32
              /  [ *5( h16 ":" ) h16 ] "::"              h16
              /  [ *6( h16 ":" ) h16 ] "::"
  ls32        =   ( h16 ":" h16 ) / IPv4address
              ;   least-significant 32 bits of address
  h16         =   1*4HEXDIG
              ;   16 bits of address represented in hexadecimal
IPv4address   =   dec-octet "." dec-octet "." dec-octet "." dec-octet
  dec-octet   =   DIGIT                 ; 0-9
              /  %x31-39 DIGIT        ; 10-99
              /  "1" 2DIGIT           ; 100-199
              /  "2" %x30-34 DIGIT    ; 200-249
              /  "25" %x30-35         ; 250-255
reg-name      =   *( unreserved / pct-encoded / sub-delims )
port          =   *DIGIT
unreserved    =   ALPHA / DIGIT / "-" / "." / "_" / "~"
reserved      =   gen-delims / sub-delims
gen-delims    =   ":" / "/" / "?" / "#" / "[" / "]" / "@"
sub-delims    =   "!" / "$" / "&" / "'" / "(" / ")"
                       / "*" / "+" / "," / ";" / "="
pct-encoded   =   "%" HEXDIG HEXDIG
```

A.3 Productions from [UUID]

```
UUID                   =  time-low "-" time-mid "-"
                          time-high-and-version "-"
                          clock-seq-and-reserved
                          clock-seq-low "-" node
time-low               =  4hexOctet
time-mid               =  2hexOctet
time-high-and-version  =  2hexOctet
clock-seq-and-reserved =  hexOctet
clock-seq-low          =  hexOctet
node                   =  6hexOctet
hexOctet               =  HEXDIG HEXDIG
```

A.4 Productions from [ESTA-IDs]

```
OrgName                =  nameStartChar *NameChar
nameStartChar          =  ALPHA / "_"
NameChar               =  nameStartChar / DIGIT / ":" / "-"
```

```
ProtocolName            =  OrgName "." OAPN
OAPN                    =  nameStartChar *(NameChar / ".")
```

Annex B Normative References

[ACN] Entertainment Services and Technology Association, since 1 January 2011 "PLASA North America" [http://tsp.plasa.org/]. ANSI E1.17 - 2010, Entertainment Technology - Architecture for Control Networks. The edition current when this Standard is approved.

[SDT] Entertainment Services and Technology Association, since 1 January 2011 "PLASA North America" [http://tsp.plasa.org/]. ANSI E1.17 - 2010, Entertainment Technology - Architecture for Control Networks. Session Data Transport Protocol. The edition current when this Standard is approved.

[DMP] Entertainment Services and Technology Association, since 1 January 2011 "PLASA North America" [http://tsp.plasa.org/]. ANSI E1.17 - 2010, Entertainment Technology - Architecture for Control Networks. Device Management Protocol. The edition current when this Standard is approved.

[DDL] Entertainment Services and Technology Association, since 1 January 2011 "PLASA North America" [http://tsp.plasa.org/]. ANSI E1.17 - 2010, Entertainment Technology - Architecture for Control Networks. Device Description Language. The edition current when this Standard is approved.

[DDLretrieval] Entertainment Services and Technology Association, since 1 January 2011 "PLASA North America" [http://tsp.plasa.org/]. ANSI E1.17 - 2010, Entertainment Technology - Architecture for Control Networks. EPI 11. Retrieval of Device Descriptions from DMP Devices on IPv4 networks. The edition current when this Standard is approved.

[UDP] Internet Engineering Task Force (IETF) [http://ietf.org/]. RFC 768 [http://ietf.org/rfc/rfc0768.txt]. Postel. User Datagram Protocol. 1980.

[SLPv2] Internet Engineering Task Force (IETF) [http://ietf.org/]. RFC 2608 [http://ietf.org/rfc/rfc2608.txt]. Guttman, Perkins, Veizades, and Day. Service Location Protocol, Version 2. 1999.

[SLPv2bis] Internet Engineering Task Force (IETF) [http://ietf.org/]. guttman-svrloc-slpv2bis-01.txt [http://www.watersprings.org/pub/id/draft-guttman-svrloc-slpv2bis-01.txt]. Erik Guttman and James Kempf. Internet Draft: Proposed Modifications to the Service Location Protocol, Version 2. 2000.

[SLPtemplate] Internet Engineering Task Force (IETF) [http://ietf.org/]. RFC 2609 [http://ietf.org/rfc/rfc2609.txt]. Guttman, Perkins, and Kempf. Service Templates and Service: Schemes. 1999.

[serviceID] Internet Engineering Task Force (IETF) [http://ietf.org/]. draft-guttman-svrloc-serviceid-02.txt [http://www.watersprings.org/pub/id/draft-guttman-svrloc-serviceid-02.txt]. Erik Guttman. Internet Draft: The serviceid: URI scheme for Service Location.. 2002.

[IPv4LL] Internet Engineering Task Force (IETF) [http://ietf.org/]. RFC3927 [http://ietf.org/rfc/rfc3927.txt]. Stuart Cheshire, Bernard Aboba, and Erik Guttman. Dynamic Configuration of IPv4 Link-Local Addresses. 2005.

[MeshSLP] Internet Engineering Task Force (IETF) [http://ietf.org/]. RFC3528 [http://ietf.org/rfc/rfc3528.txt]. W. Zhao, H. Schulzrinne, and Erik Guttman. Mesh-enhanced Service Location Protocol (mSLP). April 2003.

[UUID] Internet Engineering Task Force (IETF) [http://ietf.org/]. RFC 4122 [http://ietf.org/rfc/rfc4122.txt]. P. Leach, M. Mealling, and R. Salz. A Universally Unique IDentifier (UUID) URN Namespace. July 2005.

[ESTA-IDs] Entertainment Services and Technology Association, since 1 January 2011 "PLASA North America" [http://tsp.plasa.org/]. ANSI E1.17 - 2010. Entertainment Technology - Architecture for Control Networks. EPI-16 ESTA Registered Names and Identifiers – Format and Procedure for Registration. 2006-10-19.

[ABNF] Internet Engineering Task Force (IETF) [http://ietf.org/]. RFC 5234 [http://ietf.org/rfc/rfc5234.txt]. D. Crocker and P. Overell. Augmented BNF for Syntax Specifications: ABNF.January 2008.

[URI] Internet Engineering Task Force (IETF) [http://ietf.org/]. RFC 3986 [http://ietf.org/rfc/rfc3986.txt]. T. Berners-Lee, R. Fielding, and L. Masinter. Uniform Resource Identifier (URI): Generic Syntax. January 2005.

[newURI] InternetEngineering Task Force (IETF) [http://ietf.org/]. RFC 4395. [http://ietf.org/rfc/rfc4395.txt]. T. Hansen, T. Hardie, and L. Masinter. Guidelines and Registration Procedures for New URI Schemes. February 2006.

[Unicode] The Unicode Consortium [http://www.unicode.org/consortium/consort.html]. The Unicode Standard, [http://www.unicode.org/versions/Unicode6.3.0] Version 6.3.0, (Mountain View, CA: The Unicode Consortium, 2013. ISBN 978-1-936213-08-5).