DDL demystified

BY PHILIP NYE

Device Description Language is the tool that differentiates ACN from other protocols

DEVICE DESCRIPTION LANGUAGE (DDL) is one of the least understood parts of the ANSI E1.17 ACN networking architecture, yet it is DDL that sets ACN apart from other protocols. The core concepts and structure of DDL are actually quite simple and are not hard to grasp. However, features such as multi-lingual labeling resources and parameterized modules can distract from the core ideas, and the sheer volume of information that can be expressed about even the most simple device makes for information overload—especially if trying to read DDL in its raw Extensible Markup Language (XML) form (see sidebar page 27). This article introduces and explains those core concepts to assist people reading or writing device descriptions.

ACN background

As its name suggests, ACN—Architecture for Control Networks—is not simply a protocol, but a framework for control networking designed to encompass a wide range of requirements. It includes several protocols, including Device Management Protocol (DMP) as well as DDL. This architecture allows for new protocols or methods to be added in harmony with those already in use.

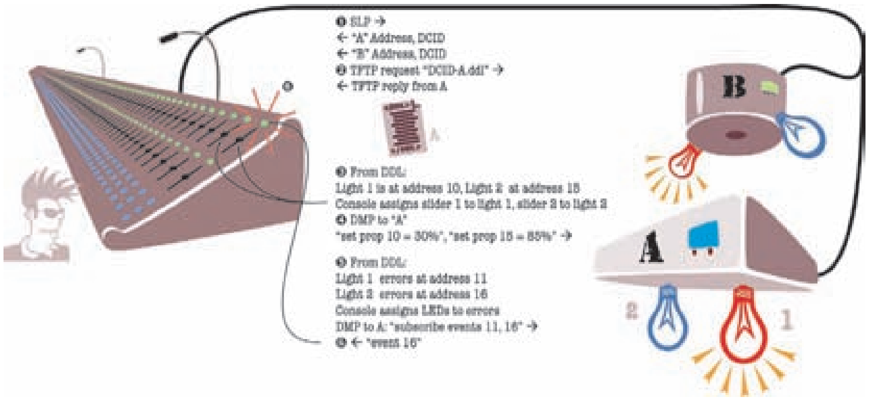
DMP is the main ACN protocol by which a controller talks to a device it wants to monitor or control. It does this, much as DMX512 does, by "stuffing values into pigeon holes" within a device. If the right value goes to the right pigeon hole, the device does what is required, but the controller (e.g., a lighting console) needs to be correctly configured to know what value to put into which hole. In the world of DMX512 the pigeon holes are called "slots," and this problem is generally solved by controllers having a library of device personalities that tell the controller which DMX512 slot does what.

DMP is considerably more flexible. Each device can have billions of "properties" (the DMP name for a pigeon hole) and, as well as being written to, these properties can be read, or configured to spontaneously publish their value when it changes. Properties need not be 8-bit quantities; they can be any number of bytes or be variable length (e.g., a text string). Their values can be encoded as integers, floating-point numbers, strings, or specialized formats such as dates and times. Some properties may be written frequently in order to control a device, whilst others may be read rarely for configuration purposes.

The task of DDL is to provide a language for expressing device personalities in the new and rich environment provided by ACN. It must be capable of describing, not just DMP devices, but also those devices controlled by E1.31 (DMX512-type data sent via ACN), and be capable of being extended to protocols not yet introduced. These device personalities must be in a language that a controller can be programmed to read and understand, and must provide the information that a controller needs to control the device.

Because different controllers have vastly differing needs, a device description may provide information that is of interest to one controller but not to another. Each controller must be able to identify and ignore irrelevant information without missing the data it does require. The process of loading and parsing device descriptions should be as automatic as possible.

It is important to bear in mind that there can be many controllers and lots of different types of devices in an ACN network, and a controller may not be expected to control all the devices it finds. This can be for reasons of capability (it does not have the necessary functionality to control this device), or policy (e.g., the system administrator does not want the moving light controller to control conventional dimmers). The controller should be able to filter the devices it finds on both these criteria during discovery.


1 SLP →
← "A" Address, DCID
← "B" Address, DCID
2 TFTP request "DCID-A.ddl" →
← TFTP reply from A
A
3 From DDL:
Light 1 is at address 10, Light 2 at address 15
Console assigns slider 1 to light 1, slider 2 to light 2
4 DMP to "A"
"set prop 10 = 30%", "set prop 15 = 85%" →
5 From DDL:
Light 1 errors at address 11
Light 2 errors at address 16
Console assigns LEDs to errors
DMP to A: "subscribe events 11, 16" →
6 ← "event 16"
B
A
2
1

A quick summary of discovery in ACN

The picture illustrates a typical discovery sequence in an ACN system:

1. The controller uses Service Location Protocol (SLP) to find the network addresses and device types of all the devices.
2. The controller requests the DDL description from device A if it needs it. It can do the same for device B.
3. The DDL description tells the controller how to control the device. The controller configures its control surface for that device.
4. The controller can now connect to the device and control it.
5. The DDL tells the controller about error status properties. The controller subscribes to those properties and configures its control surface.
6. When an error occurs, the device immediately tells all subscribers. The controller alerts the operator.

System discovery in detail, online and offline

Online discovery consists of finding the network addresses of accessible devices and basic information on the types of those devices. On a Windows computer, this process takes place when you browse for printers: the PC finds other nodes that talk back to it, and then filters them to show just those that are printers. On ACN networks, SLP fulfils this role (Apple Macs also use SLP for finding printers) and provides rich information on the sorts of node it finds on the network including, if they contain an ACN device, the Device Class Identifier (DCID), a number that identifies the type of device.

Lighting folks often want to configure a controller offline whilst not connected to a system. This can be done manually or by a file import. In either case, the controller ends up with a list of device types making up the system it will control. Later when the controller connects to the network, this configuration must be mapped to the devices that are found—often with more help from the operator.

Once the controller has its system of devices of known types, it has to be configured for the capabilities of each device. For DMX512, this means loading, or writing the personality for each fixture. In ACN environments, the controller can load and read the DDL description of each device.

What DDL tells the controller

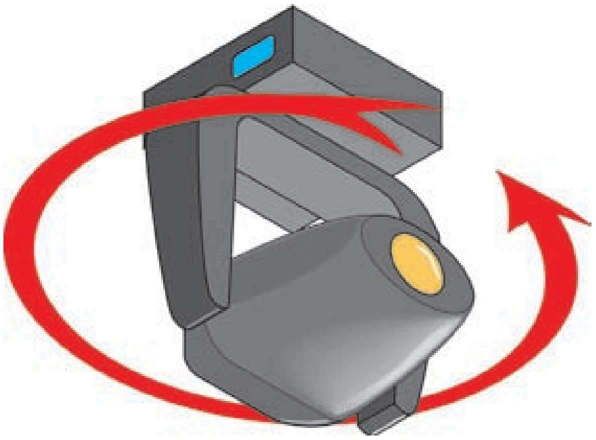
The device description provides the controller, with:

- A structured list of the properties of the device,
- Where and how to get or set the values of those properties,
- For each property, one or more behaviors that say how the property works,
- Relationships between the properties,
- Additional fixed values for various characteristics of the fixture.

A controller can get the device description in several ways. Recall that system discovery provides a DCID that identifies the device type. All devices of the same type (e.g. "Bendilite mk2") have the same DCID and the same device description applies to all of them; other devices that have a different description will have a different DCID. ACN requires that DMP devices store a copy of their device description so the controller can download it if needed, using TFTP. Since all Bendilite mk2s have the same description and the same DCID, once it has obtained the Device Description, the controller can apply it to every Bendilite mk2 in the rig (identifying them by DCID). Some controllers save a copy in a library in case they encounter another one in the future. For those who wish to pre-configure their controllers offline, fixture manufacturers can make their device descriptions available by a wide range of methods, e.g., by download from the web or published on CDROM.

Creating the DDL description, a top-down approach

Using the pan axis as an example of a typical feature: Pan is a *property* of the moving light—note that in DDL *property* has a slightly different (but closely related) meaning than DMP. We build up the pan property from the top down.



```
<property>...<property>
```

property "pan"	

(**Note:** In all these examples, the DDL is shown on the left with a tabular representation of the same structure on the right. The new items in each successive chart are emphasized in bold or brighter yellow.)

Any property (and some other elements) can have a *label*:

```
<property>
  <label>pan axis</label>
  ...
<property>
```

property		
	label	"pan axis"

A property may have sub-properties. It may also have a *value*—and the necessary details depend on how the value is accessed—in this case the value is accessible on the network. For DMP there are lots of possibilities that are expressed as attributes of a DMP specific `<propref_DMP>` element.

```
<property>
  <label>Pan axis</label>
  <property valuetype="network">
    <label>
      Pan axis target position
    </label>
    <protocol name="ESTA.DMP">
      <propref_DMP
        size="2"
        loc="67"
        abs="false"
        read="true"
        write="true"
        event="false"
      />
    </protocol>
  </property>
</property>
```

property	no value			
	label	"pan axis"		
	(sub) property	value is on the network		
		label	"pan axis target position"	
		access	*protocol*	DMP
			address	67
			size	2 octets
			readable?	yes
			writeable?	yes
			events?	no

(**Note:** Most Boolean attributes are defined to default to "false" if they are not present. They are included here but may be omitted later.)

What about dynamic feedback?

The value we write to control pan (usually the only pan value in a DMX fixture) is its *target* position. Pan is not always at this position: it may take time to get there; it may be obstructed; or overridden; or out of range. In DDL the outermost element represents the actual physical position whilst the target value and any other properties which contribute to driving its position are properties of pan—with DMP the position can be accessible (but read-only), and in our example can generate events dynamically each time it changes so that any controller subscribing to these events can track the physical position in real time.

```
<property valuetype="network">
  <label>Pan axis</label>
  <protocol name="ESTA.DMP">
    <propref_DMP
      size="2"
      loc="154" abs="false"
      read="true" write="false"
      event="true"
    />
  </protocol>
  <property valuetype="network">
    <label>
      Pan axis target position
    </label>
    <protocol name="ESTA.DMP">
      <propref_DMP
        size="2"
        loc="67" abs="false"
        read="true" write="true"
        event="false"
      />
    </protocol>
  </property>
</property>
```

property	value is on the network			
	label	"pan axis"		
	access	*protocol*	DMP	
		address	154	
		size	2 octets	
		read	yes	
		write	no	
		events	yes	
	property	value is on the network		
		label	"pan axis target position"	
		access	*protocol*	DMP
			address	67
			size	2 octets
			readable?	yes
			writeable?	yes
			events?	no

Behaviors

Each property must carry one or more behavior declarations. Each behavior specifies a facet of how that property represents, describes, or controls a feature of the device:

```
<property valuetype="network">
  <label>Pan axis</label>
  <behavior name="pan"/>
  <behavior name="driven"/>
  <behavior name="type.sint"/>
  <protocol name="ESTA.DMP">
    <propref_DMP
      loc="154" size="2"
      read="true" event="true"
    />
  </protocol>
  <property valuetype="network">
    <label>
      Pan axis target position
    </label>
    <behavior name="target"/>
    <behavior name="type.sint"/>
    <protocol name="ESTA.DMP">
      <propref_DMP
        loc="67" size="2"
        read="true" write="true"
      />
    </protocol>
  </property>
</property>
```

property	value is on the network		
	label	"pan axis – actual position"	
	behaviors	pan, driven, ±integer	
	access	*protocol*	DMP
		address	154
		size	2 octets
		read	yes
		write	no
		events	yes
	property	value is on the network	
		label	"pan axis target position"
		behaviors	target, ±integer
		access	ESTA.DMP *address*=67 *size*=2 *read*=Yes *write*=Yes *events*=No

We now need to put our pan properties into context:

- All properties come within a device.
- A device is a DDL module, and all modules require an enclosing <DDL> element.
- Behaviors need to declare a *behaviorset*—another DDL module that specifies how those behaviors work.
- Our device needs a Universal Unique Identifier (UUID) and creator information.

```
<?xml version="1.0" encoding="UTF-16"?>
<DDL version="1.0">
  <device date="2008-10-26"
    UUID="fe748d76-9091-11dd-932d-0017316c497d"
    provider="http://www.esta.org/ddl/draft/">
    <UUIDname name="base.bset" UUID="71576eac-e94a-11dc-b664-0017316c497d"/>
    <UUIDname name="sl.bset" UUID="e5b07b68-9093-11dd-81b1-0017316c497d"/>
    <useprotocol name="ESTA.DMP"/>
    <property valuetype="network">
      <label>Pan axis</label>
      <behavior name="pan" set="sl.bset"/>
      <behavior name="driven" set="base.bset"/>
      <behavior name="type.sint" set="base.bset"/>
      <protocol name="ESTA.DMP">
        <propref_DMP loc="154" size="2" read="true" event="true"/>
      </protocol>
      <property valuetype="network">
        <label>Pan axis target position</label>
        <behavior name="target" set="base.bset"/>
        <behavior name="type.sint" set="base.bset"/>
        <protocol name="ESTA.DMP">
          <propref_DMP loc="67" size="2" read="true" write="true"/>
        </protocol>
      </property>
    </property>
  </device>
</DDL>
```

This is now a complete device description and validates against a DDL schema (see sidebar "Validating DDL" on page 25). However, it only describes one axis of the light and does not convey much detail.

Adding more detail

Adding detail does not disturb our existing structure, but means adding more of the same—more sub-properties, behaviors, and values—for example, we can declare our maximum speed. Properties whose values are fixed across all devices of the type, can be included in the DDL as *immediate* values so no network reference is needed:

```
<property valuetype="network">
  <label>Pan axis</label>
  <behavior name="pan"/>
  <behavior name="driven"/>
  <behavior name="type.sint"/>
  <protocol name="ESTA.DMP">
    ...
  </protocol>
  <property valuetype="network">
    <label>
      Pan axis target position
    </label>
    ...
  </property>
  <property valuetype="immediate">
    <label>Max rotation speed</label>
    <behavior name="rate1stLimit"/>
    <value type="uint">1200</value>
  </property>
</property>
```

property	value is on the network		
	label	"pan axis"	
	behaviors	pan, driven, ±integer	
	...		
	property	value is on the network	
		label	"pan axis target position"
		...	
	property	fixed value	
		label	"max rotation speed"
		behaviors	speed limit
		value	1200

How about a speed channel?

If instead of an immediate value, the maximum speed is network writeable variable, then instead of a simple declaration, we have a speed control channel—common on moving lights. In all other respects the DDL structure remains the same:

```
<property valuetype="network">
  <label>Pan axis</label>
  <behavior name="pan"/>
  <behavior name="driven"/>
  <behavior name="type.sint"/>
  <protocol name="ESTA.DMP">
    ...
  </protocol>
  <property valuetype="network">
    <label>
      Pan axis target position
    </label>
    ...
  </property>
  <property valuetype="network">
    <label>Max rotation speed</label>
    <behavior name="rate1stLimit"/>
    <behavior name="type.uint"/>
    <protocol name="ESTA.DMP">
      <propref_DMP loc="68" size="1"
        read="true" write="true"/>
    </protocol>
  </property>
</property>
```

property	value is on the network		
	label	"pan axis"	
	behaviors	pan, driven, ±integer	
	...		
	property	value is on the network	
		label	"pan axis target position"
		...	
	property	value is on the network	
		label	"max rotation speed"
		behaviors	speed limit
		access	ESTA.DMP *address*=68 *size*=1 *read*=Yes *write*=Yes *events*=No

Much information on the device can be coded in immediate properties which are available offline and require no network overhead—here we add range and scaling information. There are applications (e.g. the "flip" function on many consoles) that need this stuff.

```
<property valuetype="network">
  <label>Pan axis</label>
  <behavior name="pan"/>
  <behavior name="driven"/>
  <behavior name="type.sint"/>
  <behavior name="angle-deg"/>
  <protocol name="ESTA.DMP">
    <propref_DMP loc="154" size="2"
      read="true" event="true"/>
  </protocol>
  <property valuetype="network">
    <label>
      Pan axis target position
    </label>
    ...
  </property>
  <property valuetype="network">
    <label>Max rotation speed</label>
    ...
  </property>
  <property valuetype="immediate">
    <behavior name="unitscale"/>
    <value type="float">0.08</value>
  </property>
  <property valuetype="immediate">
    <behavior name="limitMinInc"/>
    <value type="sint">-4000</value>
  </property>
  <property valuetype="immediate">
    <behavior name="limitMaxInc"/>
    <value type="sint">4000</value>
  </property>
</property>
```

property	value is on the network		
	label	"pan axis"	
	behaviors	pan, driven, ±integer, units are degrees	
	access	ESTA.DMP address=154 size=2 read=yes write=no events=yes	
	property	value is on the network	
		label	"Pan axis target position"
		...	
	property	value is on the network	
		label	"Max rotation speed"
		...	
	property	fixed value	
		behaviors	one unit (in degrees)
		value	0.05
	property	fixed value	
		behaviors	minimum (inclusive)
		value	-200 deg
	property	fixed value	
		behaviors	maximum (inclusive)
		value	+200 deg

To complete the device module describing our pan axis, we again have to make the pan axis a property of a device with an enclosing DDL element:

```
<?xml version="1.0" encoding="UTF-16"?>
<DDL version="1.0">
  <device date="2008-10-26"
    UUID="fe748d76-9091-11dd-932d-0017316c497d"
    provider="http://www.esta.org/ddl/draft/">
    <UUIDname name="base.bset" UUID="71576eac-e94a-11dc-b664-0017316c497d"/>
    <UUIDname name="sl.bset" UUID="e5b07b68-9093-11dd-81b1-0017316c497d"/>
    <useprotocol name="ESTA.DMP"/>
    <property valuetype="network">
      <label>Pan axis</label>
      <behavior name="pan" set="sl.bset"/>
      <behavior name="driven" set="base.bset"/>
      <behavior name="type.sint" set="base.bset"/>
      <behavior name="angle-deg" set="base.bset"/>
      <protocol name="ESTA.DMP">
        <propref_DMP loc="154" size="2"
          read="true" event="true"/>
      </protocol>
      <property valuetype="network">
        <label>
          Pan axis target position
        </label>
        <behavior name="target" set="base.bset"/>
        <behavior name="type.sint" set="base.bset"/>
        <protocol name="ESTA.DMP">
          <propref_DMP loc="67" size="2"
            read="true" write="true"/>
        </protocol>
      </property>
      <property valuetype="network">
        <label>Max rotation speed</label>
        <behavior name="rate1stLimit" set="base.bset"/>
        <behavior name="type.uint" set="base.bset"/>
        <protocol name="ESTA.DMP">
          <propref_DMP loc="68" size="1"
            read="true" write="true"/>
        </protocol>
      </property>
      <property valuetype="immediate">
        <behavior name="unitscale" set="base.bset"/>
        <value type="float">0.08</value>
      </property>
      <property valuetype="immediate">
        <behavior name="limitMinInc" set="base.bset"/>
        <value type="sint">-4000</value>
      </property>
      <property valuetype="immediate">
        <behavior name="limitMaxInc" set="base.bset"/>
        <value type="sint">4000</value>
      </property>
    </property>
  </device>
</DDL>
```

Building the complete moving light description

To continue building up our moving light, we can either simply add other axes and features alongside our pan element within the same device, or we can take advantage of DDL's modularity, leave pan as a device on its own and include it as a sub-device of the moving light device:

```
...
<includedev UUID="fe748d76-9091-11dd-932d-0017316c497d">
  <label>Included pan axis subdevice</label>
</includedev>
...
```

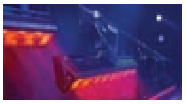
Conclusion

The concepts of DDL are mostly quite simple—a device is described as a hierarchy of properties that usually have a value and may have sub-properties (some properties are used just to group sub-properties and have no value of their own).

- The key to the way it works are the *behaviors* that must be attached to each property. Each behavior is a precise specification of a certain functionality. The specification forms a contract between the device and its controller.
- Whilst similar to the features found in many fixture libraries, behaviors are more precisely defined, are finer grained, and more flexible.
- There is a very rich set of behaviors already specified as part of the ACN standard. Currently the ACN Task Group in the Control Protocols Working Group is working on more. ■

Philip Nye is principal consultant and product design engineer at Engineering Arts. He has designed electronics and software, and consulted on all aspects of product design for many of the best known names in the entertainment technology industry. Philip is a member of the Control Protocols Working Group and has been a leading member of the ACN Task Group since very early in its existence. He is currently engaged in developing the industry funded open-source ACN implementation at www.openacn.org.

What does "Pan" mean?



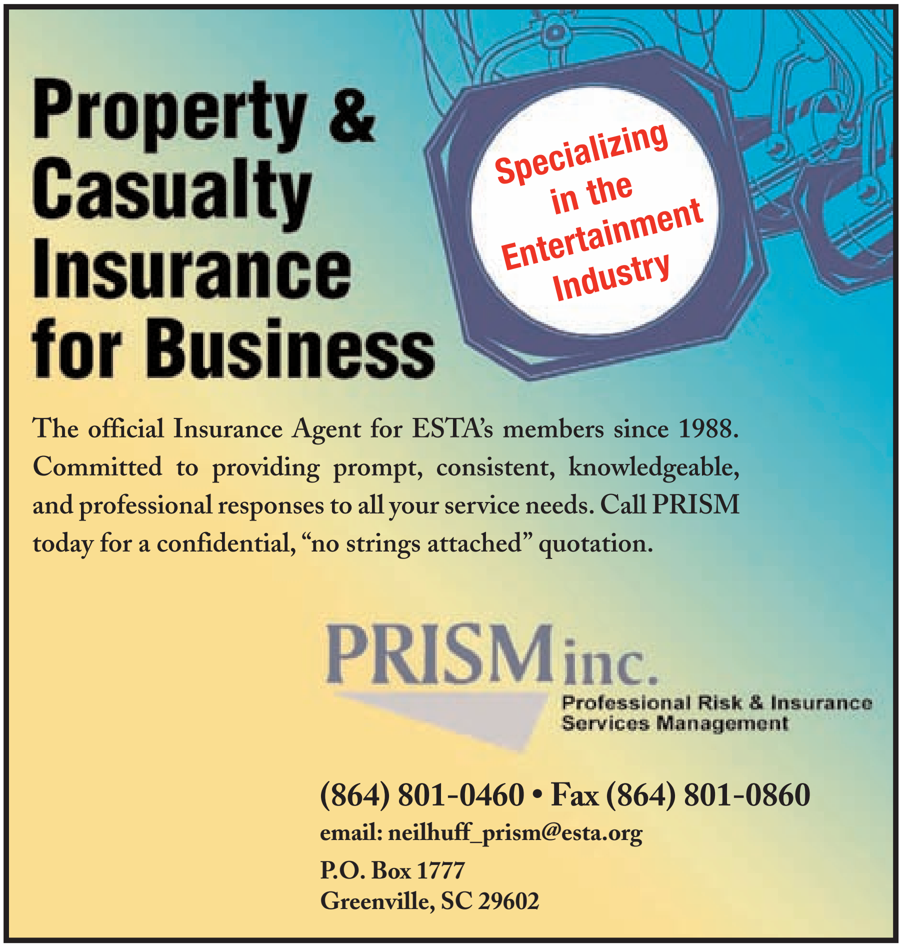
The descriptions use a behavior called "pan" to convey the idea of what the property is doing. This term is so instinctive to most lighting professionals that it can mask a lot of hidden assumptions. To illustrate this, think about what "pan" means in the context of a moving mirror versus a moving head light; then think of some of the less conventional moving lights, for example a Coemar "NAT"™ or Wynne-Wilson "Razorhead," an SGM® or Lighting Technology "Caterpillar," or a DHA "Pitching Light Curtain." You soon realize that simply calling a function "pan" or "tilt" does not fully describe what it does.

This may be okay for simple controllers, but more sophisticated ones need something more precise (remember that "flip" function). Fortunately, DDL has a full set of behaviors specifying precise geometrical descriptions of what is happening in three dimensions, and these can be used alongside the simpler behaviors, so the controller can use whichever best suits its functionality.

Validating DDL

Validation is an XML process which checks an XML document against a set of rules called a schema which specifies allowable element and attribute names, the order and context they can occur in, permissible values, and so on.

The DDL specification includes schemas in two languages: DTD and RelaxNG, others are possible. There are many validation tools available, including web tools, XML editors, command line utilities, and Java classes. Always validate DDL documents when you are writing them!



TLAs and more

It is hard to venture into the world of computer languages and protocols without encountering a vast number of TLAs (Three Letter Acronyms) and other jargon. Here are useful terms:

DDL—Device Description Language is part of the ACN standard, and is an XML-based language for describing controllable devices in a machine readable format.

ACN—Architecture for Control Networks, is the descriptive name of the ANSI E1.17 standard that contains many parts, including DDL.

DMP—Device Management Protocol is the protocol used by ACN for control.

CID—Component Identifier is a UUID that identifies an ACN component.

DCID—Device Class Identifier is a UUID that identifies a class of devices (a device type e.g., Acme Bendilite MkII).

XML—Extensible Markup Language is a framework based on tags and attributes that can be used to create customized languages for markup or data. Examples include XHTML (a more rigorous form of HTML the language of web pages) and of course DDL.

DTD—Document Type Declaration is a schema language defined in the XML specification. DTDs can be very useful, but are not required and there are also other schema languages.

IP—Internet Protocol

TFTP—Trivial File Transfer Protocol is an Internet standard protocol used for file transfer. It's limited but very easy to implement.

SLP—Service Location Protocol is an Internet Standard protocol and ACN's first-stage discovery protocol. It allows a controller to find devices that are on the network and their addresses.

UUID—Universal Unique Identifier is a 128-bit number that is guaranteed to be unique, and so is useful for unambiguously identifying almost anything that requires identification. UUIDs can be easily generated on demand.

DMX512—A protocol, originally for controlling dimmers, that has been around for years and is very successful.

Octet—A networking term for an 8-bit byte. Historically many computers had bytes of differing sizes, while an octet is always 8 bits.

A Lightning Speed Introduction to XML

DDL is written in XML (Extensible Markup Language) which is very widely used for structured data or text. There are many books and tutorials on XML available, so this is just a bare-bones guide.

XML adds structure to text data. It appears very simple to anyone familiar with HTML.	
It has opening and closing *tags* which define an *element*:	`<pets>...</pets>`
An element may contain text:	`<name>Fido></name>`

or *other elements*:
(or sometimes both – but never in DDL)
	`<pets>`
	`  <dog><name>Fido</name>...</dog>`
	`  <cat><name>Mog</name>...</cat>`
	`</pets>`

Empty elements can be abbreviated	`<pups></pups>` ↔ `<pups/>`
Elements may also have *attributes*:	`<dog breed="shitsu"/>`

Comments are delimited by “<!--” and “-->” and can go just about anywhere you would want them:	`<pets>`
	`  <!-- One record per pet -->`
	`  ...`
	`</pets>`