

### **Comment 1**

<b>Commenter</b>	Philip Nye
<b>Section Number</b>	General
<b>Section Title</b>	N/A
<b>Description</b>	There is a lot in UDR and I have not had time to do more than scratch the surface in these comments. Some comments may be invalid because I haven't read as far as the place they are resolved yet.)
<b>Resolution</b>	No Action
<b>Resolution Description</b>	No action required.

### **Comment 2**

<b>Commenter</b>	Philip Nye
<b>Section Number</b>	General
<b>Section Title</b>	N/A
<b>Description</b>	Nearly 20 years after the standardization of Device Description Language (E1.17 DDL) it is good to see the industry catching up with the necessity for something like this. The similarities between UDR (E1.73) and DDL are startling including - a broad ambition to "...describe device capabilities and information in such a way they can be interpreted reliably and consistently by machines" - the intent that Readers may interpret the description at any level appropriate to their function, from the basic to the advanced - the separation of the control model from any particular data protocol (serialization) - inclusion of a detailed specific serialization for DMX512 - a huge amount of the detail of expressing parameter relationships using structures and constraints - the definitions of classes and instances and the mechanisms for building libraries of classes from which more complex models are constructed There is not much in UDR that was not covered using remarkably similar concepts in DDL and its standardized behaviors. However, the definitions of UDR follow more modern trends in text representation (JSON vs XML) and improve on DDL by using much better defined mechanisms for classes and inheritance, versioning and localization. As one of the original architects of DDL I cannot fault much of the design of E1-73 and wish the standard success.)
<b>Resolution</b>	No Action
<b>Resolution Description</b>	No action required.

### Comment 3

<b>Commenter</b>	Philip Nye
<b>Section Number</b>	General
<b>Section Title</b>	N/A
<b>Description</b>	<p>The structure and layout of the standard documentation must anticipate that, particularly in its early stages, there will be frequent additions or changes. Where details are embedded in a monolithic standard, making agile changes is very unwieldy in the context of the ESTA standards process.</p> <p>In the current documentation there are many tables which contain a list of allowable values along with a directive which specifies the action of a Reader encountering a value outside those defined. e.g for Table 6-6 'Readers of a Device Library [JSON] object shall ignore any key not defined...' or for Table 8-4 'Readers shall interpret any Parameter using a unit not defined as "unitless"'. Many of these tables clearly have scope for additions in future and in some cases such additions are very probable in quite a short time. This presents a problem for standardization when all such lists and tables are part of a monolithic standard document because the procedural overhead for updating a major standard document is high and much needed small changes can get bogged down in a major review process.</p> <p>The standard should separate such tables out and for tables which are likely to grow quickly, provide a process for updating them which is separate from the core standard so that additions do not require that the main body of the standard be revised (and so opened for full review) in order to update them. ACN used an EPI mechanism (which could still be used) and DDL additionally pushed many such features into behaviors within behaviorsets to avoid this issue (UDR proposes that structures should act in a similar way). For example, if the list of units (table 8-4) is a separate mini-standard then only that standard needs to go through the review process when it is extended, as is highly likely. If not creating additional documents, consider at least putting many of these lists and tables into appendices.)</p>
<b>Resolution</b>	Accept in Part
<b>Resolution Description</b>	The task group agrees with some portions of the comment and is aiming to design the document to make it easy to update wherever possible. However, the task group does not believe lots of mini-standards is the appropriate solution to this problem. The separation of the Library documents is one mechanism currently being used to aid easier, more incremental updates.

### Comment 4

<b>Commenter</b>	Philip Nye
<b>Section Number</b>	Schemas
<b>Section Title</b>	N/A
<b>Description</b>	<p>JSON schemas provide a rigorous way to define the markup that is integral to UDR and are a normative part of the standard. However, they are barely referenced in the text. The documents themselves contain schema fragments, some using JSON-Schema and others using a graphical format which is not referenced, but these fragments do not define the entire markup anyway.</p> <p>The schemas on gitlab.com are a normative part of the standard. Include them as an appendix if necessary and provide the full references. Then make clear where schemas appear in the text that they are non-normative and that if they differ from the official schema the latter takes priority.)</p>
<b>Resolution</b>	Accept in Part
<b>Resolution Description</b>	JSON schemas are non-normative in these standards. We have made some modifications to the document to make this clearer and explain how JSON schemas are intended to be used. We have also removed schemas from the documents as they are accompanying material.

## Comment 5

<b>Commenter</b>	Philip Nye
<b>Section Number</b>	4.1
<b>Section Title</b>	Basics
<b>Description</b>	<p>While I grasped the concepts easily, I struggled to understand how the parts go together and the relationship between the object model, the markup and the standards documents.</p> <ol style="list-style-type: none"><li>1. Give the object model a name for easy reference.</li><li>2. Make it clear that the basic unit of a UDR description is a Document (ref. §15) which is read by a Reader (where is this defined?) which takes the JSON document and uses as little or as much of the information therein as its application needs, to build its own internal representation. Give some examples of Readers to show the variety of needs to be met: a test tool might just build an annotated tree view of network accessible parameters and some generic slider controls to exercise a device. Some manufacturer specific configurators will be tuned to filter for very specific context and parameter types but ignore much else. Meanwhile a visualizer has to simulate each device in minute detail so it need to know, not just the networked parameters but the physical geometry and characteristics and a lot of detail on how the device responds to DMX or other control protocols; not only this but it has to understand how all the devices relate to each other to make a System - information which cannot be provided by a device description but can be built within UDR (and once built could be available for multiple purposes e.g. a theatre could provide machine-readable models of its space and all the standard equipment in it).</li><li>3. Add brief text to clarify that a Document may be retrievable from multiple places, including from some components themselves, can be packaged with others into an Archive (ref. §16) or may be dynamically generated e.g. where Systems or Devices are auto-configuring or software defined.</li><li>4. Mention schemas as formal definitions of allowable JSON Structures which can be used to check for compliant markup. Reference the schema for this draft. Ensure JSON-Schema and the actual schemas for E1.73 are in the Normative references.</li><li>5. Be more explicit and rigorous about which parts of UDR are machine readable (as JSON) and which aren't. The statement "UDR is composed of a series of Libraries, Device Classes, and Systems" and much of the rest of the text is suggesting that Libraries, Device Classes, and Systems are fundamental top level types machine readable as JSON Markup. However, the libraries and classes defined by ESTA (e.g. E1.73-3 Intensity/Color) are PDF documents, not JSON, and include a lot of fundamental normative semantic material which does not go into the markup. Thus a library object in JSON markup is a different thing from an ESTA UDR library and the standard is often ambiguous. Essentially, any ESTA UDR standard consists of:<ul style="list-style-type: none"><li>■ an object model for Readers together with its JSON markup</li><li>■ a JSON Schema which rigorously defines the format of that markup</li><li>■ normative text providing semantics for anything that is new (i.e. not part of existing standards) in the object model</li><li>■ informative text which provides examples, clarification, suggested usage etc.</li></ul>The semantics of an object which are normative but not included in markup will multiply as UDR develops, particularly for structures and structure classes since these express a semantic relationship that goes beyond mere hierarchy and connections. A place this is very significant is in versioning where throughout the document repeated text notes which version changes are required by different types of markup change, yet none of these suggest that there may be changes to the external semantic definitions that require a version change.</li><li>6. Classes vs Instances: "UDR is composed of a series of Libraries, Device Classes, and Systems...". I think (haven't had time to investigate fully) that in general the JSON for a Class is defined by a JSON-Schema while the JSON for an Instance is defined by JSON conforming to that schema? If this is true (maybe only for certain types of class?), it helps clarify the distinction and should be mentioned.)</li></ol>
<b>Resolution</b>	Accept in Part

<b>Resolution Description</b>	<p>5.1 Accept: We have given the model a name.</p> <p>5.2 Reject: The task group does not agree that the basic unit is a Document. Reader is defined in the definitions section of the document.</p> <p>5.3 Reject: A document is not a special thing, it is just another way of organizing objects. This standard aims to be transport agnostic, so it is not intended to describe how data is accessed.</p> <p>5.4 Accept in Principle: We have added a new JSON schema section and clarified use of JSON schemas within the document.</p> <p>5.5 Accept: We have made a pass through the document to clarify this.</p> <p>5.6 No Action: This is an incorrect assumption.</p>
-------------------------------	--

### **Comment 6**

<b>Commenter</b>	Philip Nye
<b>Section Number</b>	5.3.1
<b>Section Title</b>	End User Identifier
<b>Description</b>	Apart from being a way for anyone to freely generate their own Entities without reference to any coordinating body or the need for a long-lived registered domain, it is not clear what the intent or expectation is. The text mentions a possible mechanism to 'allow the identifier to be linked to the end user's personal identity in an online Device Class repository.' but this seems very vague and unclear in its expectation - this in turn may lead to widely differing and incompatible interpretations. Are there plans to offer such a repository? or a framework for organizations or individuals wishing to do so? Alternatively should there be a requirement or defined mechanism for the End User to provide some associated metadata in their class or system?)
<b>Resolution</b>	Accept in Part
<b>Resolution Description</b>	The intent of the standard is to enable the use cases described, but not to specify them explicitly. We have removed the sentence related to an online repository, as this can be deduced.

### **Comment 7**

<b>Commenter</b>	Philip Nye
<b>Section Number</b>	5.4
<b>Section Title</b>	Qualified Identifier
<b>Description</b>	"a dot" in 4 places. Add clarification that a 'dot' is a delimiter as defined in the ABNF. It may not be obvious, particularly to less fluent users of English.)
<b>Resolution</b>	Accept
<b>Resolution Description</b>	Delimiter is now included in definitions and use of the word dot has been replaced throughout the document.

### **Comment 8**

<b>Commenter</b>	Philip Nye
<b>Section Number</b>	5.4.1
<b>Section Title</b>	Versioned Qualified Identifier
<b>Description</b>	Typo: para 2 "followed by a dot..." should be "followed by a version-delimiter...")
<b>Resolution</b>	Accept in Principle
<b>Resolution Description</b>	We have changed the way objects are identified, so this comment is no longer applicable.

### **Comment 9**

<b>Commenter</b>	Philip Nye
<b>Section Number</b>	5.5
<b>Section Title</b>	Referencing
<b>Description</b>	"When referencing an Atomic Data Object..." this is the first time that atomic data objects are encountered in the text (aside from the definitions section). It would help to add a paragraph in section 4 explaining the next level down from Libraries, Device Classes, and Systems.)
<b>Resolution</b>	Reject
<b>Resolution Description</b>	The group believes the definition in Section 3 is sufficient to explain this layer, however, this particular reference is no longer present.

### **Comment 10**

<b>Commenter</b>	Philip Nye
<b>Section Number</b>	6.1.3.3
<b>Section Title</b>	Author
<b>Description</b>	<p>This property is required but the standard makes no attempt to define or constrain it in a way to make it useful (e.g. by ensuring it is unique within the domain of the organisation).</p> <p>Given the library has a fully qualified identifier with an unambiguous organisation ID, the author should be an optional property (easy) or should carry additional requirements (harder.)</p>
<b>Resolution</b>	Accept in Part
<b>Resolution Description</b>	Added clarification that this is human-readable, and that the value should remain the same for libraries from the same individual or organization.

### Comment 11

<b>Commenter</b>	Philip Nye
<b>Section Number</b>	6.1.4
<b>Section Title</b>	Markup
<b>Description</b>	<p>This section (and others defining markup) makes no sense until the concepts of Documents and schemas have been described and is far easier to read in the context of the appropriate part of the JSON-Schema.</p> <p>Examples should show how markup combines and aggregates (if at all). This section gives 2 examples:</p> <pre>{   "org.esta.lib.intensity-color#1.0.2": {    } }</pre> <p>and</p> <pre>{   "@description": "esta_intensity-color_library",   "publishDate": "2021-11-10T09:00:00Z",   "author": "ESTA",   "parameterClasses": {   },   ... etc. }</pre> <p>Do these nest like this?</p> <pre>{   "org.esta.lib.intensity-color#1.0.2": {     "@description": "esta_intensity-color_library",     "publishDate": "2021-11-10T09:00:00Z",     "author": "ESTA",     "parameterClasses": {     },     ... etc.   } }</pre> <p>or concatenate? or are they retrieved as separate entities? )</p>
<b>Resolution</b>	Accept in Principle
<b>Resolution Description</b>	Section 4.5 was added explaining how examples work in the document. All examples were updated to include comments as described in Section 4.5.

### Comment 12

<b>Commenter</b>	Philip Nye
<b>Section Number</b>	7
<b>Section Title</b>	Atomic Data Objects
<b>Description</b>	<p>The use of 'par', 'str', 'ser' &amp; 'res' appears to an attempt to force these concepts into a symmetrical three character abbreviation which just sows confusion and misunderstanding. The common abbreviation 'struct' is immediately recognisable while 'str' is elsewhere nearly always an abbreviation for 'string'. Similarly 'res' and 'par' while not evoking contrary meanings quite so strongly do not convey immediate understanding. Consider slightly longer abbreviations. e.g. 'param', 'resrc' or 'rsrc'.)</p>
<b>Resolution</b>	Accept in Principle
<b>Resolution Description</b>	We have changed the way objects are identified, so this comment is no longer applicable.

### Comment 13

<b>Commenter</b>	Philip Nye
<b>Section Number</b>	8.1.1.1
<b>Section Title</b>	Category
<b>Description</b>	<p>The libraries defined in E1.73 are reasonably consistent in how they define categories but there is nothing to enforce this. Category identifiers are defined in the separate schema for device-class and it is not clear how a library can overload this.</p> <p>1. "Categories...shall not be defined by individual manufacturers." This may have unintended consequences.</p> <ul style="list-style-type: none"><li>■ it may put a brake on expansion of UDR into new applications</li><li>■ to get around this, manufacturers will co-opt existing categories for purposes they are not intended for, thus corrupting and devaluing their usefulness.</li></ul> <p>Consider a mechanism like Media types where names beginning "x-" are unapproved and considered unreliable.</p> <p>2. "Categories are defined in library documents..."</p> <p>This means that a library is more than can be expressed in its JSON markup. Section 6 defines libraries differently and implies that the entire thing can be expressed in JSON. See comments on §4.1</p> <p>3. Consider removing category definitions from the standard altogether and aggregating those from all ESTA libraries into either a separate standard or an ESTA registry where they will share a common format and it is easy to check for inconsistencies and conflicts of meaning or, when considering a new category, to check how similar things have been handled before.</p> <p>4. Categories as defined are a strict tree hierarchy, each name only has meaning in the context of its parents and ancestors. This has problems:</p> <ul style="list-style-type: none"><li>■ the same name may appear in different contexts with very different meanings. This is not a problem for a Reader but very confusing for a human.</li><li>■ there is nothing to link the same concept when it occurs in different contexts, even closely related ones. I suspect that humans will not be fully aware of this and assume links where none exist.</li></ul> <p>)</p>
<b>Resolution</b>	Accept in Principle
<b>Resolution Description</b>	The task group agrees this is confusing. We have clarified the text around categories. We are also looking to add a catch-all category and are to move the definitions of categories outside of the documents. This is planned for a future review cycle.

## Comment 14

<b>Commenter</b>	Philip Nye
<b>Section Number</b>	8.1.3.4
<b>Section Title</b>	Unit
<b>Description</b>	<p>The list is woefully short, restrictive and random in its choice. It mostly uses common metric, mainly SI, units which is sensible but does not attempt any rigour or to reference SI or provide guidance to maintain consistency in introducing new units which will almost instantly be needed (see Anticipating Extensions and Updates to the Standard). The list conflates units with prefixes expressing powers of ten ("milli", "kilo" etc.) and scaling. More consistent dimensional analysis in UDR will allow a much richer set of cases to be built from just a small set of primitives.</p> <p>2. Separate prefixes from the base units as a different property and allow them in any combination. (i.e. just one unit metre can be combined with micro-, milli-, kilo- etc.).</p> <p>3. Degrees Fahrenheit is an aberration! Delete it unless you can provide special justification which would not also oblige inclusion of inches, pounds, quarts and other arbitrary units (which would open a can of worms!)</p> <p>4. percent is a misnomer and confusing if the variable runs from 0 to 1 as described. Change the name to 'proportion' or 'fraction' or something (or have it run from 0 to 100). Furthermore, it is not truly a unit and while it is intended to address a common use-case, it provides no clues about what it represents. In many cases, including very common ones such as conventional dimmers, this sort of property is modulating some other function. In a moving light, a dimmer is known to be modulating intensity. However for a discrete dimmer, what is modulated may not be known or even discoverable - what is plugged in to it? a 500W or a 1200W fixture? or a heater or fan? or nothing at all? I am not convinced that tagging a parameter as having units of 'fraction' or 'proportion' answers any of these questions but it does express what a value from 0...1 is doing better than percentage. See also comment on logarithmic units.</p> <p>5. Logarithmic units are widely used in audio and wireless but also elsewhere. dB is a measure of ratio very much like percent/proportion but with an attached reference value become an absolute measure, some of which are very common (dBV, dBA, dBm, dB SPL). pH (acidity) is another common logarithmic value. Either provide a logarithmic fraction mechanism (best, see below) or include common logarithmic values directly.</p> <p>6. Logarithmic quantities, proportions/percentages and other modulation functions or non-linearities may be best captured by addressing them head on using structures where one parameter is generated by applying a modulation function to another (one or other of which may be virtual or hidden parameters). As well as logarithms and linear modulation, this technique can address arbitrary non-linear relationships such as dimmer curves or trigonometric relationships arising from mechanical linkages (e.g. a linear actuator operating via a linkage to rotate a part). These concepts may fit into the constraints mechanism in UDR.</p> <p>7. Here are some other units to consider including from the start - a few minutes thinking would come up with more:</p> <ul style="list-style-type: none"> <li>■ litre (and millilitre etc)</li> <li>■ amp-hours (watt-hours which are also used for battery charge are easily converted to/from joules)</li> <li>■ Newton-metres (torque, not Joules)</li> </ul> <p>8. Bytes – it should be clearly stated in the standard that Byte means 8-bits throughout since historically this was not always the case. Alternatively use the term "octet" as most data protocol standards do.</p> <p>9. RPM is an arcane non-SI unit and not necessary as Hz is included (a console can easily perform conversions if a UI requires RPM). If having both, it would be sensible to include radians and radians-per-second too.</p> <p>In approaching units, the task group should think about policy on whether the standard aims to provide the most minimal set of units possible and rely on controllers to provide conversions to something familiar in context, or to allow a pragmatic larger set of common units which are familiar but overlap with each other and may cause endless issues for implementers. There are some )</p>
<b>Resolution</b>	Accept in Part
<b>Resolution Description</b>	The units have been totally overhauled and now follows SI standards. We have added a definition of byte. The group believes RPM is accepted and widely used within the industry.

**Comment 15**

<b>Commenter</b>	Philip Nye
<b>Section Number</b>	8.2.3.3>5
<b>Section Title</b>	Dynamic Count Min/Max, Count
<b>Description</b>	<p>Remove Dynamic Min/Max Count properties and replace with a single dynamicCount boolean.</p> <p>Justification This scheme declares an array of similar parameters which is important. When count is statically declared, this works fine and no further properties or constraints/structures are necessary. Where count is dynamic, there must be other parameters or mechanisms to access the value for count. §17.1 also makes clear that instantiation must or may be as a sparse array. Given that this is run-time instantiation, the mechanism for discovering these details from the component should be declared in other parts of the UDR and that is the place to define the min and max values. In the simplest case, there is a dynamic 'count' parameter declared elsewhere and that has its own min and max constraints. In other cases min and max may not be known until run-time (e.g. they may depend on the amount of RAM installed in the individual device or on how much resource is currently taken up by other optional functions).</p> <p>)</p>
<b>Resolution</b>	Reject
<b>Resolution Description</b>	These fields are intended to communicate the possible range of the count of objects so Readers can understand how best to display them to a user. The actual count is outside the scope of the data model and is intended to be handled by another layer.

## Comment 16

<b>Commenter</b>	Philip Nye
<b>Section Number</b>	8.2.3.6
<b>Section Title</b>	Access and Lifetime
<b>Description</b>	<p>readwrite is ambiguous and inaccurate. After some thought, I have concluded that for unambiguous expression there should be two types of parameter "read" (read-only) and "write" (write-only). Any parameter shall be declared one or the other*. Where a read and a write parameter relate closely to the same functionality this should then be expressed as a structure detailing the exact way they relate. There are several possible ways to express this syntactically with a boolean access value probably being the simplest.</p> <p>Justification</p> <ul style="list-style-type: none"><li>• In the main protocols UDR is initially aimed at DMX and sACN, dynamically accessible parameters are write-only by definition as the protocol provides no mechanism to read them. Defining these as readwrite is meaningless.</li><li>• Where a dynamically accessible parameter is truly accessible via a bidirectional protocol there is ambiguity over what the relationship is between the value written and the value read and what the read value represents. Consider this example: A moving light has a pan function driven by parameter 'Pan' with limits of -200° and +200°. The manufacturer has chosen to implement the device so that writing a value outside the limits will be accepted but the pan movement will stop at the limit. Initially the device is at rest with pan at -180°. Now parameter Pan is set to +270°. What does it mean to read the value of the parameter after a short interval? There are several possibilities:<ul style="list-style-type: none"><li>■ the value read is +270°, exactly what was written.</li><li>■ the value read is +200°, the value written subjected to the max constraint.</li><li>■ the value read reflects in real time where the pan function currently is as it moves towards the limit.</li></ul>All the ambiguities in this example go away if all the cases are declared as separate parameters. The fact they relate to the Pan function and its write-only parameter, and the exact nature of that relationship must be expressed in one or more structure relationships. The device designer is free to implement the three possibilities separately in any combination from none to all. The fact that one may be at the same protocol address as the write value is expressed in the relevant serializer. * There may also be a need for parameters that are virtual or hidden for use as intermediate nodes in structures. )</li></ul>
<b>Resolution</b>	Accept in Part
<b>Resolution Description</b>	We have separated read into target and actual. The group believes this is sufficient complexity for this document. There is no need for an additional structure to define this information.

### Comment 17

<b>Commenter</b>	Philip Nye
<b>Section Number</b>	8.2.3.6
<b>Section Title</b>	Access and Lifetime
<b>Description</b>	<p>For writable parameters, what happens when an out of range value is set? There are at least three common approaches, each of which may be encountered in real-life equipment and the most appropriate choice of behavior varies from case to case. These need to be describable.</p> <ol style="list-style-type: none"><li>1. reject the request and leave the set point unchanged.</li><li>2. set the parameter to the nearest acceptable value (e.g. set it to max value, or truncate a string to max length).</li><li>3. behave the same as case 2. but store the requested value so that if the limit changes (e.g. by an external configuration change) the set point does too.</li></ol>
<b>Resolution</b>	Reject
<b>Resolution Description</b>	This standard describes how data is modelled, it does not attempt to describe behavior.

### Comment 18

<b>Commenter</b>	Philip Nye
<b>Section Number</b>	8.2.3.6
<b>Section Title</b>	Access and Lifetime
<b>Description</b>	<p>Add volatility to the lifetime property.</p> <p>Lifetime gets the bare bones of its concept clearly but it does not express whether a parameter may change independently of direct control via the protocol. Sometimes this is implicit and obvious to a human but may not be obvious to a Reader - a read-only parameter for ambient temperature or incident light will vary frequently through the runtime of the device but consider a DMX address or some other configuration parameter, settable by RDM, in some cases these may also be set by user action (via a front panel, configuration web-page or other protocols) - in other cases, the only way to set a value is via the access protocol and the Reader can be sure that having set that value it will not change.</p>
<b>Resolution</b>	Reject
<b>Resolution Description</b>	There is no concept of a Parameter that is non-volatile. It is expected that any Parameter which can be read and which has a lifetime of persistent or runtime may change at any time, which is implicit in the fact it is readable.

### Comment 19

<b>Commenter</b>	Philip Nye
<b>Section Number</b>	8.2.3.7
<b>Section Title</b>	Atomic Identifier
<b>Description</b>	<p>A very important concept - be more explicit about what this means. The second paragraph is a cop-out. I would suggest "Methods for ensuring Parameters are set atomically, and behavior on errors should be defined or described within serialization definitions for individual access protocols.")</p>
<b>Resolution</b>	Reject
<b>Resolution Description</b>	This standard is a data model, it provides mechanisms to associated atomic Parameters. The method of setting and reporting on this is up to another layer, e.g. transport.

### Comment 20

<b>Commenter</b>	Philip Nye
<b>Section Number</b>	8.2.3.8>11
<b>Section Title</b>	Min, Max, Min Modifier, Max Modifier
<b>Description</b>	These properties should be removed. In describing max and min modifiers, the document already acknowledges that in many cases these are not simply properties of a parameter but are separate parameters with a particular relationship. It defines a special case way to reference those parameters and has to define priorities in case of conflict between this mechanism and a structure. The place to express all of this is in a structure and to define and use constraints structures for very common relationships like max and min.)
<b>Resolution</b>	Reject
<b>Resolution Description</b>	Min and Max have significant value as they will be used in the majority of cases today. The additional capabilities provide for more complex setups.

### Comment 21

<b>Commenter</b>	Philip Nye
<b>Section Number</b>	8.2.3.8>9
<b>Section Title</b>	Min, Max
<b>Description</b>	Note this comment applies even if accepting previous comment max and min and their modifiers Allow both inclusive and exclusive values ( $p \leq \max$ vs. $p < \max$ ). Inclusive values are generally preferable or at least acceptable for integer parameters but exclusive values are sometimes much more appropriate and especially for floating point values and wrapping (see Wrapping). For example, for a value which must be positive and non-zero the exact inclusive minimum will be highly dependent on the precise floating point representation and the case is much better expressed by an exclusive minimum of zero. )
<b>Resolution</b>	Reject
<b>Resolution Description</b>	The task group's intuition based on the set of parameters modelled so far is that we do not need exclusive limits. If we come across examples where we do then we will revisit this.

### Comment 22

<b>Commenter</b>	Philip Nye
<b>Section Number</b>	8.2.3.12
<b>Section Title</b>	Default
<b>Description</b>	<p>It is not clear what default means. Is this:</p> <ul style="list-style-type: none"><li>• the value the parameter takes after power up? or soft reset?</li><li>• the value the parameter takes after any calibration procedure - which may only occur after an explicit command?</li><li>• a 'safe' value which the parameter can be set to which will minimize interference with other equipment?</li><li>• the value a persistent parameter will take after factory reset?</li></ul> <p>Of course it could be any of these things depending on context but you are likely to find that otherwise very similar devices from different manufacturers have wildly different interpretations; without more detail it is hard to see how this value can be useful to a Reader.</p> <p>In contrast, a default value in a parameter class could be a useful way to provide a value which would not then have to be declared separately in each instance unless overriding the default.</p> <p>)</p>
<b>Resolution</b>	Accept in Principle
<b>Resolution Description</b>	This value is intended for controllers to use when a "home" or "default" option is available. Clarifying text has been added.

### Comment 23

<b>Commenter</b>	Philip Nye
<b>Section Number</b>	8.2.3.13
<b>Section Title</b>	Wrapping
<b>Description</b>	<p>The text for Min and Max dictates that both are inclusive values. This creates trouble with values that wrap. In the gobo wheel example given, when wrapping is true, the minimum and maximum values must be co-incident so that setting the parameter to max (= 6) is identical to setting it to min (= 0); however this is not spelled out clearly and also makes for ambiguity. In most wrapping cases it is preferable to make max an exclusive value while min is inclusive. In the case of a wrapping parameter with a limit supplied by a related parameter via the access protocol, using an inclusive max may not even be possible - consider the case where a 1-byte parameter wraps from 255 to 0. To express this using the inclusive max requires that max is 256 which cannot be expressed in 1 byte.)</p>
<b>Resolution</b>	Accept in Principle
<b>Resolution Description</b>	The text has been updated to clarify how wrapping is intended to work (now called Looping). Max is effectively exclusive, but can still be addressed.

### Comment 24

<b>Commenter</b>	Philip Nye
<b>Section Number</b>	15
<b>Section Title</b>	Document
<b>Description</b>	This section should come first. It provides a structural context which makes the following sections much easier to read. It also introduces the schema (which is a normative part of the standard and should be included in references). For someone wanting to implement, the schema is crucial and the rest of the document should be read in that context.)
<b>Resolution</b>	Reject
<b>Resolution Description</b>	Documents are only one way that object conforming to the model can be provided to Readers. It is expected that in the future other methods will become more popular than a document. Moving Document to the beginning would also require many forward references.

### Comment 25

<b>Commenter</b>	Philip Nye
<b>Section Number</b>	17.2
<b>Section Title</b>	Entity Information
<b>Description</b>	"When requesting an Entity from a Component via whatever method"  change to  "When requesting an Entity for a Component via whatever method and from whatever source"  This recognizes that the component itself may not have the capability to provide the entity.  In practice a useful generic Reader would probably be structured to make requests to a resolver which determines where and how to retrieve the actual JSON in the context of the access protocols in use. )
<b>Resolution</b>	Accept in Principle
<b>Resolution Description</b>	Section 17.2 was removed from the document due to other changes to how Entities are identified, resulting in it becoming irrelevant.

**Comment 26**

<b>Commenter</b>	Philip Nye
<b>Section Number</b>	2: General
<b>Section Title</b>	Core definitions and overall concepts
<b>Description</b>	<p>There is a gap in UDR between the architecture and structure defined in Part 1 and the core structures defined here. What is missing is a set of low level semantic concepts that apply right across the device representation. These should be expressed in this document.</p> <p>Below are some of the missing concepts. Whether they are put into the first release of UDR or not, care must be taken that the first release does not operate so as to exclude them from all future updates. None of these examples are currently addressed by any semantic definitions in UDR and this means that different assumptions will be made by those implementing it and Readers will struggle to discern any commonality. It also means that differing models, or simply differing terminology, are likely to develop to describe the same thing in different contexts meaning that the prospects for creating a generic controller become remote.</p> <p>)</p>
<b>Resolution</b>	No Action
<b>Resolution Description</b>	No action required.

## Comment 27

<b>Commenter</b>	Philip Nye
<b>Section Number</b>	2: General
<b>Section Title</b>	Reading and writing parameters
<b>Description</b>	<p>There is an assumption that control, configuration and monitoring of a device is achieved by writing or reading values to parameters which are the primitives from which the model is built. This is broadly how DMX, RDM, ACN-DMP and a lot of other protocols work (e.g. MIDI, SNMP) and more message and command oriented protocols can usually be translated into the same concept. A controller, monitor or other Reader builds its own model of a device from the structures and parameters of UDR to suit its application. The serializers then tell it how to access and manipulate those parameters to achieve its purpose.</p> <p>It is important that "Parameters" represent any model specific data value which may be embedded in the JSON itself or accessible from the component. If the latter, a parameter may still be constant across all instances, constant but specific to each instance, read-only but varying across time or writable. For an access protocol like DMX or sACN, no parameter can be readable so any parameter must either be write-only or be a literal value embedded in the JSON</p> <p>This assumption is not spelled out and a whole lot goes missing. Repeating the example from my comment on part 1, Access and Lifetime:</p> <p>A moving light has a pan function driven by parameter Pan with limits of <math>-200^{\circ}</math> and <math>+200^{\circ}</math>. The value written to Pan represents a target. The manufacturer has chosen to implement the device so that writing a value outside the limits will be accepted but the pan movement will stop at the limit. The access protocol(s) allow both writing and reading.</p> <p>Initially the device is at rest with pan at <math>-180^{\circ}</math>. Now Pan is written to <math>+270^{\circ}</math>. What does it mean to read the value of Pan after a short interval? There are several possibilities: - the value read is <math>+270^{\circ}</math>, exactly what was written. - the value read is <math>+200^{\circ}</math>, the value written subjected to the max constraint. - the value read reflects in real time where the pan function currently is as it moves towards the limit. - in a multi-controller environment, other inputs may override or supersede the written value. The value read is the current target which is the result of any HTP, LTP or any other priority processing to resolve multiple conflicting values.</p> <p>These alternatives may all be useful to a controller and the manufacturer may choose to implement more than one of them as separate parameters - how does a controller distinguish between them and know how they relate? Even if only one is implemented, a Reader may need to know which it is.</p> <p>)</p>
<b>Resolution</b>	Accept in Principle
<b>Resolution Description</b>	This comment has been addressed through changes to provide two methods for reading values along with descriptions explaining their functionality and purpose.

### Comment 28

<b>Commenter</b>	Philip Nye
<b>Section Number</b>	2: General
<b>Section Title</b>	Behavior of Device in Response to our of range Values
<b>Description</b>	<p>As hinted in the previous point, if an attempt is made to set a parameter to a value outside its allowable range, there two main responses possible:</p> <ul style="list-style-type: none"><li>• use the closest value which is in range (best-efforts policy)</li><li>• reject the request (reject policy)</li></ul> <p>Which is used may depend on the protocol but will often be a choice of the implementer in a specific context. DMX for example, provides no mechanism to feed back that a request has been rejected and users of a dimmer who had set an output limit would not want it to ignore a snap to maximum because it exceeded the limit. However, a DMX controlled pyrotechnic device could legitimately be much fussier. In contrast RDM provides a NAK message and reason code NR_DATA_OUT_OF_RANGE but this does not preclude an implementer using a best-efforts policy for any specific PID.</p> <p>Some Readers are likely to want to know which policy is implemented for any particular parameter (consider a visualizer which is trying to second-guess the effect of protocol changes on a fixture based on its UDR description. The proper place to define this may be in the Serializer or could be in he form of a structure definition.</p> <p>)</p>
<b>Resolution</b>	Reject
<b>Resolution Description</b>	This is mostly a transport domain problem. We would expect any transport to detail what happens if a value out of range is set, or alternatively this may not be possible at all such as when using DMX.

### Comment 29

<b>Commenter</b>	Philip Nye
<b>Section Number</b>	2: General
<b>Section Title</b>	Constraints on parameters
<b>Description</b>	<p>In the current draft spec Max and Min are privileged special cases, being defined in the structure of UDR (Part-1 §8.2.3.8-11). However, that document then gets into difficulties when these are not simple static values which can be embedded as literals within the JSON but must be described as parameters - which may be constant or variable. See max and min and their modifiers. Once Max or Min becomes a parameter, that parameter itself may be subject to constraints - an output limit on a dimmer level is a variable Max constraint which itself has limits.</p> <p>There should be no special case for Max and Min. They are examples of constraints which are (or should be) independent of the context or category in which they operate (Min for a Red Intensity level should be no different from Min for a Winch Speed) and these should be described using a consistent structure format: this parameter behaves in that way or this paramater represents a constraint on that one. Other examples might be more complex ranges (e.g. ones with gaps), locks or mutexes, rate of change limits (max speed for a movement, dimmer rise times) etc.</p> <p>A basic set of these kinds of constraint should be included in the core definitions for UDR. If these are defined clearly and consistently, further constraints have a model to follow and should fit the same outline.</p> <p>)</p>
<b>Resolution</b>	Reject
<b>Resolution Description</b>	Although the comment has merit, as described in the answer to comment 20, the task group believes Min and Max have value for the most common use cases.

**Comment 30**

<b>Commenter</b>	Philip Nye
<b>Section Number</b>	2: General
<b>Section Title</b>	Define Parameter Groups
<b>Description</b>	<p>The concepts above and many others, are obvious candidates for using group structures* In one way to do this, almost anywhere a simple parameter is referenced in the markup, it should be replaceable with a structure which references one or more parameters which operate together to achieve the action or retrieve the value required and defines the semantics of how they work together. Thus, the Pan example above (and using the Stick model of part-4) could have a single "parameter": /par/pan which in a DXM only device is all there is. But with a bidirectional protocol, where there are multiple read parameters representing different facets of Pan it could be replaced with something like:</p> <pre>"parameter": {   "readWriteParam" : {     "writeTarget" : "par/pan/setTarget",     "readBack" : "par/pan/readback",     "readTarget" : "par/pan/target",     "readCurrentValue" : "par/pan/currentPosition"   } }</pre> <p>This can optionally replace almost any plain writable parameter. readBack (the same value that was written), readTarget (the target the device is actually using after constraints are applied and multiple controller priorities resolved) and readCurrentValue (the real-time position) are optional parameter references which associate both the parameter identifiers if those parameters are provided and their exact meaning (from the group definition) together. Alternatively this could be replaced with multiple simpler structures:</p> <pre>"control": {   ...   "parameter": "par/pan/setTarget" }, ... "constrain": {   "parameter": "par/pan/setTarget",   "min": "par/pan/loLimit"   "max": "par/pan/hiLimit" }, "constrain": {   "parameter": "par/pan/loLimit",   "min": -200,   "maxExclusive": "par/pan/hiLimit" }, "constrain": {   "parameter": "par/pan/hiLimit",   "minExclusive": "par/pan/loLimit",   "max": 200 }, ... etc.</pre> <p>* I thought this was what Control Groups (§7.2) were for. However the statement "for a particular Device Instance, only one Control Groups Structure can be active at any point" doesn't fit.</p>
<b>Resolution</b>	Accept in Part
<b>Resolution Description</b>	A method like this has been implemented for Parameter access. However, the task group does not believe this is a general principle that applies to other concepts.

### Comment 31

<b>Commenter</b>	Philip Nye
<b>Section Number</b>	2: General
<b>Section Title</b>	Semantics vs Structure
<b>Description</b>	<p>Categories and Structures are the two principle means to convey semantic information that goes with an otherwise generic set of parameters. Part-1 (§8.1.1.1) suggests Categories are used primarily for filtering and grouping using a hierarchy of keywords intended for human readers. However, at some level, they are the only way that a Reader can recognize certain special semantics.</p> <p>Structures on the other hand, are used for grouping parameters which work together to a particular end but in the standard defining the structure class they are also overloaded with semantics that go way beyond the mere hierarchy of grouping.</p> <p>This means there is an overlap of possible ways to describe semantics and relationships and potential for conflict or confusion. UDR should be clear on where the boundary lies. In the Pan example in the comment above, a parameter representing the instantaneous value of pan as the light moves could be identified using a category "motion/rotate/current-position". However, this is not a good idea for several reasons. Despite this, the strict tree of categories means that semantic information may be lost and commonality certainly will.</p> <p>Part-4 gives examples of Motion Definitions and defines a Control Object which is defined in fairly rigorous mathematical terms with the axis property being defined as "X", "Y" or "Z". However, it also provides a "commonAxis" property giving a 'common name' for that axis. How does this relate to the Category of the corresponding parameter? What if commonAxis="pan" and category=".../.../tilt"?)</p>
<b>Resolution</b>	Reject
<b>Resolution Description</b>	<p>Structures are the method of associating Parameters together in multiple ways with UDR. This is a foundational principle as there is no "one way" to associate Parameters, or to provide added value information. While Categories may seem like another candidate for a Structure, the task group believes they have some foundational importance to provide a cleaner namespacing and help associate "like named" Parameters with each other. The list of Categories is governed by ESTA, so this limits conflicts.</p> <p>In the example of motion, a commonAxis of "pan" would be applied to a motion/rotate/offset parameter, so there is no conflict.</p>

### Comment 32

<b>Commenter</b>	Philip Nye
<b>Section Number</b>	2: General
<b>Section Title</b>	Terminology masks commonality
<b>Description</b>	<p>The Control Object of Part-4 provides a good generic method for describing basic motion control and relating the various parameters. However, it introduces the commonAxis name and restricts that to "pan", "tilt", "barrel", "x", "y" or "z". This is very lighting-centric and excludes or devalues use in applications where the concept is the same but terminology different - e.g. "azimuth" and "altitude" or "bearing" and "elevation" for "pan" and "tilt". The same factors are likely to occur (only worse) for categories where there is nothing at all to make child terminology consistent between one parent category and another.</p> <p>I suggest the standard defines key terms for these concepts based on well defined and accepted mathematical notations, e.g. Angle-Z, and then replaces the words used in categories and properties like commonAxis with a system of aliases which define more application dependent terminology. e.g. alias "lighting/pan" = "rotate-Z"</p> <p>)</p>
<b>Resolution</b>	Accept in Part
<b>Resolution Description</b>	This standard has a target application market and is not intended to describe everything. Lighting and cameras are two common applications for this standard, and use of pan/tilt are common within entertainment devices. The task group recognizes that the x, y, z common axes are confusing so these have been removed. If explicit axes are required these are already provided within the Control object using the "axis" key/value.

### Comment 33

<b>Commenter</b>	Philip Nye
<b>Section Number</b>	2: General
<b>Section Title</b>	Control Groups
<b>Description</b>	It is not clear (to me) how these are intended to work. The requirement "for a particular Device Instance, only one Control Groups Structure can be active at any point" suggests they correspond to a personality setting or perhaps to specific variants on a device class. See note above)
<b>Resolution</b>	Reject
<b>Resolution Description</b>	The document explains what active means, and provides a reference to E1.37-1 with further information on this. If this is still considered unclear, the task group would value further information on how to best clarify this for a reader.

### Comment 34

<b>Commenter</b>	Philip Nye
<b>Section Number</b>	3: General
<b>Section Title</b>	Intensity/Color Definitions
<b>Description</b>	I have not read this part in enough depth to comment.)
<b>Resolution</b>	No Action
<b>Resolution Description</b>	No action required.

**Comment 35**

<b>Commenter</b>	Philip Nye
<b>Section Number</b>	4: 7.1.1.4.2
<b>Section Title</b>	Control
<b>Description</b>	<p>The example defines "pan" as angle-Z and "tilt" as angle-X which is an intuitive interpretation of the coordinate space defined in §7.1.1.2. (the light is pointing away from me so tilt is around the X axis). However, this conflicts with the Z-Y-X order convention of [ODT] which would put the light pointing along axis-X with "tilt" as angle-Y and gobo rotate as angle-X.</p> <p>A consequence of the Z-Y-X convention for a most luminaires is that the 2D positional coordinates on a gobo, slide or projected image are not given by (X,Y) but by (-Y,Z). it is worth clarifying this. )</p>
<b>Resolution</b>	Reject
<b>Resolution Description</b>	The task group assumes the reference to [ODT] means [OTP]. However, there is no conflict between an orientation of a device and the order of rotation. Order of rotation can be the same for a device oriented in any direction.