PHILIPS

hue PERSONAL WIRELESS LIGHTING

How to **develop for Hue?**

Datatypes and Time Patterns

All data types have a short hand used throughout the API. All uses of the shorthand in the main API documentation should link through to the data types page, with an appropriate anchor, and/or display the data type description on hover.

Datatypes

Below is a list of data types used by the API and their short hand used in the API documentation.

Short Hand	Description
string n..m	A string in UTF8 encoding where n..m specifies the minimum number of characters, n, and maximum number, m. If n and m are not specified then only certain values are accepted as specified in the description. Although the character % is a valid UTF8 character it is not allowed in strings, if it occurs it will be removed.
ASCII string n..m	A string in UTF8 encoding only containing characters in the set [a-z], [A-Z], [0-9] and [-] n..m specifies the minimum number of characters, n, and maximum number, m, for free format strings. If n and m are not specified then only certain values are accepted specified in the description. Undefined is “none” for attributes <1.x.0 will be changed to null attributes as of 1.x.0 use null for undefined

On this page:

uint8	8 bit, unsigned, non-wrapping integer. i.e. an integer in the range of 0 to 255 where integer values outside this range are invalid.
uint16	16 bit, unsigned, non-wrapping integer. i.e. an integer in the range of 0 to 65535 where integer values outside this range are invalid.
hex n..m	Hexadecimal string. E.g. “AB4314” n..m specifies the minimum number of characters, n, and maximum number, m.
list n..m of x	A list of items of type x. Where x is another entry in this table. A list is formatted as comma separated values totally enclosed in square brackets e.g. `[1,2,3].` n..m specifies the minimum number of entries in list, n, and maximum number, m.
bool	A Boolean value which can take the values true or false only.
object	An object value is a JSON compliant object. This is of the format of zero or more key value pairs encapsulated in curly braces {}.
time	string (19..19) ISO8601:2004 `[YYYY]-[MM]-[DD]T[hh]:[mm]:[ss]` Example: 2013-12-31T14:12:45
timePattern	string (2..64) Undefined is “none” for attributes <1.x.0 will be changed to null attributes as of 1.x.0 use null for undefined
resource	A resource can have sub-resources. A resource differs from an object that it is not included in a GET. Though it is possible to make a GET on the sub-resources or its childs. A resource can have a freeformat id or a name based on uint or ASCII

Time Patterns

Based on ISO8601:2004

Absolute time	`[YYYY]-[MM]-[DD]T[hh]:[mm]:[ss]`([date]T[time])
Randomized time	`[YYYY]:[MM]:[DD]T[hh]:[mm]:[ss]A[hh]:[mm]:[ss]` ([date]T[time]A[time])

Recurring times	`W[bbb]/T[hh]:[mm]:[ss]` Every day of the week given by bbb at given time
Recurring randomized times	`W[bbb]/T[hh]:[mm]:[ss]A[hh]:[mm]:[ss]` Every weekday given by bbb at given left side time, randomized by right side time. Right side time has to be smaller than 12 hours
Time intervals	`T[hh]:[mm]:[ss]/T[hh]:[mm]:[ss]` Every day from left time to right time (maximal interval length is 23 hours) `W[bbb]/T[hh]:[mm]:[ss]/T[hh]:[mm]:[ss]` Every weekday given by bbb from left side time to right side time
Timers	

`PT[hh]:[mm]:[ss]`	Timer, expiring after given time
`PT[hh]:[mm]:[ss]`	Timer, expiring after given time
`PT[hh]:[mm]:[ss]A[hh]:[mm]:[ss]`	Timer with random element
`R[nn]/PT[hh]:[mm]:[ss]`	Recurring timer
`R/PT[hh]:[mm]:[ss]`	Recurring timer
`R[nn]/PT[hh]:[mm]:[ss]A[hh]:[mm]:[ss]`	Recurring timer with random element