Overview of RFC 6570 URI Template

CSV on the web WG

Definitions

- **URI Template:** a compact sequence of characters for describing a range of Uniform Resource Identifiers through variable expansion; they are not URIs themselves
- **expression:** the text between ‘{‘ and ‘}’, including the enclosing braces
- **expansion:** the string result obtained from a template expression after processing it according to its expression type, list of variable names, and value modifiers
- **template processor:** a program of library that, given a URI Template and a set of variables with values, transforms the template string into a URI reference by parsing the template for expressions and substituting each one with with its corresponding expansion

Level 1 templates: *simple string expansion*

Variables

```
var        := “value”
hello      := “Hello World!”
```

Expression	Expansion
`{var}`	`value`
`{hello}`	`Hello%20World%21`

note percent-encoding of characters that are not in the set of unreserved URI characters: A-Z, a-z, 0-9, “-”, “.”, “_” and “~”

Level 2 templates: reserved string expansion

Variables

```
var          := "value"
hello        := "Hello World!"
path         := "/foo/bar"
```

Expression	Expansion
`{+var}`	`value`
`{+hello}`	`Hello%20World!`
`{+path}/here`	`/foo/bar/here`
`here?ref={+path}`	`here?ref=/foo/bar`

note that reserved URI characters are:
":", "/", "?", "#", "[", "]", "@", "!", "$", "&", "'", "(", ")", "", "+", ",", ";" and "="*

"+" operator: expansion is allowed to include reserved URI characters

Level 2 templates: *fragment expansion*

Variables

```
var        := “value”
hello      := “Hello World!”
path       := “/foo/bar”
```

Expression	Expansion
`X{#var}`	`X#value`
`X{#hello}`	`X#Hello%20World!`
`X{#undefined}`	`X`
	note omission of operator where variable is undefined

“#” operator: expansion of hash-prefixed fragment identifiers - including reserved URI characters

Level 3 templates: *simple expansion with multiple variables*

Variables

`var`	`:=`	`“value”`	`x`	`:=`	`“1024”`
`hello`	`:=`	`“Hello World!”`	`y`	`:=`	`“768”`
`path`	`:=`	`“/foo/bar”`	`empty`	`:=`	`“”`

Expression	Expansion
`map?{x,y}`	`map?1024,768`
`{x,hello,y}`	`1024,Hello%20World%21,768`

note use of comma “,” as default separator

multiple variables; comma separated list

Level 3 templates: *reserved expansion with multiple variables*

Variables

`var := "value"`		`x := "1024"`	
`hello := "Hello World!"`		`y := "768"`	
`path := "/foo/bar"`		`empty := ""`	

Expression	Expansion
`{+x,hello,y}`	`1024,Hello%20World!,768`
`{+path,x}/here`	`/foo/bar,1024/here`

Level 3 templates: *fragment expansion with multiple variables*

Variables

```
var       := “value”              x      := “1024”
hello     := “Hello World!”       y      := “768”
path      := “/foo/bar”           empty  := “”
```

Expression	Expansion
`{#x,hello,y}`	`#1024,Hello%20World!,768`
`{#path,x}/here`	`#/foo/bar,1024/here`

Level 3 templates: *label expansion*

Variables

```
var         := “value”                         x       := “1024”
hello       := “Hello World!”                  y       := “768”
path        := “/foo/bar”                      empty   := “”
```

Expression	Expansion
`X{.var}`	`X.value`
`X{.x,y}`	`X.1024.768`

“.” operator: expansion of dot-prefixed labels

Level 3 templates: *path segments*

Variables

`var := “value”`		`x := “1024”`	
`hello := “Hello World!”`		`y := “768”`	
`path := “/foo/bar”`		`empty := “”`	

Expression	Expansion
`{/var}`	`/value`
`{/var,x}/here`	`/value/1024/here`

“/” operator: expansion of slash-prefixed path segments

Level 3 templates: *path-style parameters*

Variables

```
var     := "value"               x     := "1024"
hello   := "Hello World!"        y     := "768"
path    := "/foo/bar"            empty := ""
```

Expression	Expansion
`{;x,y}`	`;x=1024;y=768`
`{;x,y,empty}`	`;x=1024;y=768;empty`

note omission of "=" token for 'empty'

";" operator: expansion of semi-colon prefixed path-style parameters

Level 3 templates: *form-style query*

Variables

`var := “value”`		`x := “1024”`	
`hello := “Hello World!”`		`y := “768”`	
`path := “/foo/bar”`		`empty := “”`	

Expression	Expansion
`{?x,y}`	`?x=1024&y=768`
`{?x,y,empty}`	`?x=1024&y=768&empty=`

“?” operator: expansion of ampersand separated form-style query

Level 3 templates: *form-style query continuation*

Variables

```
var      := "value"            x      := "1024"
hello    := "Hello World!"     y      := "768"
path     := "/foo/bar"         empty  := ""
```

Expression	Expansion
`?fixed=yes{&x}`	`?fixed=yes&x=1024`
`{&x,y,empty}`	`&x=1024&y=768&empty=`

note that only a single operator may be used in a given expression

"&" operator: expansion of form-style query continuation

Level 4 templates: *string expansion with substring value modifier*

Variables

```
var       := “value”
hello     := “Hello World!”
path      := “/foo/bar”
list      := (“red”, “green”, “blue”)
keys      := [(“semi”,”;”),(“dot”,”.”),(“comma”,”,”)]
```

Expression	Expansion
`{var:3}`	`val`
`{var:30}`	`value`

prefix modifier (“:”) indicates use of only a limited number of characters

Level 4 templates: *string expansion with list expansion value modifier*

Variables

```
var         := “value”
hello       := “Hello World!”
path        := “/foo/bar”
list        := (“red”, “green”, “blue”)
keys        := [(“semi”,”;”),(“dot”,”.”),(“comma”,”,”)]
```

Expression	Expansion
`{list}`	`red,green,blue`
`{list*}`	`red,green,blue`
`{keys}`	`semi,%3B,dot,.,comma,%2C`
`{keys*}`	`semi=%3B,dot=.,comma=%2C`

explode modifier (“”) indicates that the variable treated as a composite value - a list of names or associative array of (name,value) pairs, each of which is expanded as if it were a separate variable*

Level 4 templates: *reserved expansion with value modifiers*

Variables

```
var       := “value”
hello     := “Hello World!”
path      := “/foo/bar”
list      := (“red”, “green”, “blue”)
keys      := [(“semi”,”;”),(“dot”,”.”),(“comma”,”,”)]
```

Expression	Expansion
`{+path:6}/here`	`/foo/b/here`
`{+list}`	`red,green,blue`
`{+list*}`	`red,green,blue`
`{+keys}`	`semi,;,dot,.,comma,,`
`{+keys*}`	`semi=;,dot=.,comma=,`

Level 4 templates: *fragment expansion with value modifiers*

Variables

```
var       := “value”
hello     := “Hello World!”
path      := “/foo/bar”
list      := (“red”, “green”, “blue”)
keys      := [(“semi”,”;”),(“dot”,”.”),(“comma”,”,”)]
```

Expression	Expansion
`{#path:6}/here`	`#/foo/b/here`
`{#list}`	`#red,green,blue`
`{#list*}`	`#red,green,blue`
`{#keys}`	`#semi,;,dot,.,comma,,`
`{#keys*}`	`#semi=;,dot=.,comma=,`

Level 4 templates: *label expansion with value modifiers*

Variables

```
var        := “value”
hello      := “Hello World!”
path       := “/foo/bar”
list       := (“red”, “green”, “blue”)
keys       := [(“semi”,”;”),(“dot”,”.”),(“comma”,”,”)]
```

Expression	Expansion
`X{.var:3}`	`X.val`
`X{.list}`	`X.red,green,blue`
`X{.list*}`	`X.red.green.blue`
`X{.keys}`	`X.semi,%3B,dot,.,comma,%2C`
`X{.keys*}`	`X.semi=%3B.dot=..comma=%2C`

Level 4 templates: *path segments with value modifiers*

Variables

```
var          := “value”
hello        := “Hello World!”
path         := “/foo/bar”
list         := (“red”, “green”, “blue”)
keys         := [(“semi”,”;”),(“dot”,”.”),(“comma”,”,”)]
```

Expression	Expansion
`{/var:1,var}`	`/v/value`
`{/list}`	`/red,green,blue`
`{/list*}`	`/red/green/blue`
`{/list*,path:4}`	`/red/green/blue/%2Ffoo`
`{/keys}`	`/semi,%3B,dot,.,comma,%2C`
`{/keys*}`	`/semi=%3B/dot=./comma=%2C`

Level 4 templates: *path-style parameters with value modifiers*

Variables

```
var       := “value”
hello     := “Hello World!”
path      := “/foo/bar”
list      := (“red”, “green”, “blue”)
keys      := [(“semi”,”;”),(“dot”,”.”),(“comma”,”,”)]
```

Expression	Expansion
`{;hello:5}`	`;hello=Hello`
`{;list}`	`;list=red,green,blue`
`{;list*}`	`;list=red;list=green;list=blue`
`{;keys}`	`;keys=semi,%3B,dot,.,comma,%2C`
`{;keys*}`	`;semi=%3B;dot=.;comma=%2C`

Level 4 templates: *form-style query with value modifiers*

Variables

```
var       := “value”
hello     := “Hello World!”
path      := “/foo/bar”
list      := (“red”, “green”, “blue”)
keys      := [(“semi”,”;”),(“dot”,”.”),(“comma”,”,”)]
```

Expression	Expansion
`{?var:5}`	`?var=val`
`{?list}`	`?list=red,green,blue`
`{?list*}`	`?list=red&list=green&list=blue`
`{?keys}`	`?keys=semi,%3B,dot,.,comma,%2C`
`{?keys*}`	`?semi=%3B&dot=.&comma=%2C`

Level 4 templates: *form-style query continuation with value modifiers*

Variables

```
var        := “value”
hello      := “Hello World!”
path       := “/foo/bar”
list       := (“red”, “green”, “blue”)
keys       := [(“semi”,”;”),(“dot”,”.”),(“comma”,”,”)]
```

Expression	Expansion
`{&var:5}`	`&var=val`
`{&list}`	`&list=red,green,blue`
`{&list*}`	`&list=red&list=green&list=blue`
`{&keys}`	`&keys=semi,%3B,dot,.,comma,%2C`
`{&keys*}`	`&semi=%3B&dot=.&comma=%2C`

For more details, please refer to RFC 6570