URI Templates for Marlin

Version 1.0
Final

Source	Marlin Developer Community
Date	Sept 10, 2007

Notice

Intellectual Property

A commercial implementation of this specification requires a license from the Marlin Trust Management Organization.

Contact Information

Feedback on this specification should be addressed to: editor@marlin-community.com

Contact information for the Marlin Trust Management Organization can be found at: http://www.marlin-trust.com/

Contents

1 Introduction

The purpose of URI templates is to be able to represent partially-formatted URIs that a Marlin Device can transform into a final URI, such as a URL, by substituting parts of the URI template with the value of variables for which the value may not be known at the time the URI template was created or transmitted.

1.1 Syntax Notation

This specification uses the Augmented Backus-Naur Form (ABNF) notation of [RFC2234], including the following core ABNF syntax rules defined by that specification: ALPHA (letters), DIGIT (decimal digits) and CHAR (character).

1.2 Conformance Conventions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this specification are to be interpreted as described in IETF RFC 2119 [RFC2119].

These capitalized key words are used to unambiguously specify requirements and behavior that affect the interoperability and security of implementations. When these key words are not capitalized they are meant in their natural-language sense.

All Elements of this specification are considered **Normative** unless specifically marked **Informative**. All Normative Elements are **Mandatory** to implement, except where such an element is specifically marked **OPTIONAL**. Finally, where **Normative** elements are described as **OPTIONAL**, they MAY be omitted from an implementation, but when implemented, they MUST be implemented as described.

1.3 References

1.3.1 Normative References

[MRL-BB]	Marlin Engineering Work Group Marlin Broadband Delivery Specification Version 1.2
[RFC2234]	Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997
[CORE]	Marlin Engineering Work Group Marlin Core System Specification Version 1.3
[RFC2119]	Key words for use in RFCs to Indicate Requirement Levels. Internet Engineering Task Force, 1997
[RFC3986]	IETF RFC 3986 Uniform Resource Identifiers (URI): Generic Syntax
[RFC3629]	F. Yergeau, “UTF-8, a transformation format of ISO 10646”, November 2003

1.3.2 Informative References

[Marlin]	Marlin http://www.marlin-community.com/

1.4 Definitions

Percent-Encoding, percent-encoded	Escaping of certain characters as specified in section 2.1 of [RFC3986]

1.5 Acronyms

URI	Uniform Resource Identifier
UTF	Unicode Transformation Format

2 URI Template Syntax

A URI template is a UTF-8 encoded string [RFC3629] that contains zero or more placeholder substrings. The syntax for a URI template is:

```
uri-template = 1*VCHAR
```

The URI template MAY contain one or more variable placeholders substrings. A placeholder substring represents a single template variable. A placeholder substring has the following syntax:

```
placeholder = "{" var-name "}"
var-name = [var-modifier] var-namespace ":" var-local-name
var-modifier = "~"
var-namespace = ALPHA
var-local-name = 1*( ALPHA / DIGIT / "%" / ":" / "@" / "-" / "." /
"_" / "~" / "!" / "$" / "&" / "*" / "+" / "=" )

```

The template string MUST NOT contain any “{“ or “}” characters outside the placeholder substrings except in they percent-encoded form.

3 Template Transform

The template transform takes a template string as input and produces a URI string as output. The output is obtained by replacing all placeholder substrings with the string value of that placeholder's variable (var-name). Variable names are case sensitive. The template MUST be constructed in such a way that the URI string produced is a valid URI as defined in [RFC3986]. This implies that all reserved characters in the values of the variable MUST be percent-encoded as specified in section 2.1 of [RFC3986].

The value of a variable that is not understood by the entity that is performing the transform depends on the presence of the optional variable modifier: if the modifier "~" is present, the value SHALL be the fixed string "~"; if it is not present the value SHALL be an empty string.

4 Template Variables

4.1 Marlin Variables

The variable namespace for Marlin variables is “m”.
The general syntax for a Marlin variable is:

```
marlin-var = match * ( "&" match )
match = attribute-name "=" attribute-value
attribute-name = "type" / URI
attribute-value = 1*VCHAR
```

attribute-name: either the fixed string “type” or the name of an Octopus Node attribute as defined in section 3.2 of Marlin Core [CORE].

attribute-value: a string.

Since the “=” and “&” characters are used as delimiters, any occurrence of those characters in attribute-name or attribute-value MUST be percent-encoded as specified in section 2.1 of [RFC3986].

The value of the variable is the unique identifier (uid) of an Octopus node selected by the client application.
The node selected by the application MUST satisfy all the match filters indicated in the variable name using the ‘match’ syntax. Each match filter indicates an attribute name (attribute-name) and an attribute value (attribute-value). A node satisfies the match filter if and only if it has an attribute whose name is the filter’s attribute-name with a value equal to the filter’s attribute-value. When attribute-name is ‘type’, the node attribute to match is the node attribute named ‘urn:marlin:core:node:attribute:type’.
Since the attribute names and attribute values to match may contain percent-encoded characters, the attribute-name and attribute-value subcomponent strings MUST be percent-decoded before matching.

If the application finds more than one node that can be selected, it MAY use other criteria, such as user preferences, to select one of them.
The application MAY use the empty string for the value of this variable if it is not able to select a suitable node.

5 Examples

Input	`http://www.bok.net/music/get-token?user-node={m:type=user}&cid=8967F56D`
Output	`http://www.bok.net/music/get-token?user-node=urn:marlin:organization:boknet:user:006789F5&cid=8967F56D`

Input	`http://www.bok.net/music/get-token?user-node={m:type=user&urn:marlin:organization:test:MyAttribute=urn:marlin:acme:myServiceId}&cid=8967F56D`
Output	`http://www.bok.net/music/get-token?user-node=urn:marlin:organization:acme:myNode:006789F5&cid=8967F56D`

Input	`http://www.bok.net/test?z={z:known_var}&foo=3`
Output	`http://www.bok.net/test?z=some_value&foo=3`

Input	`http://www.bok.net/test?z={z:unknown_var}&foo=3`
Output	`http://www.bok.net/test?z=&foo=3`

Input	`http://www.bok.net/test?z={~z:unknown_var}&foo=3`
Output	`http://www.bok.net/test?z=~&foo=3`