Dynamic Resource Linking for Constrained RESTful Environments
draft-ietf-core-dynlink-01

Abstract

   For CoAP [RFC7252] Dynamic linking of state updates between
   resources, either on an endpoint or between endpoints, is defined
   with the concept of Link Bindings. This specification defines
   conditional observation attributes that work with Link Bindings or
   with CoAP Observe [RFC7641].

   Editor's note:

   o The git repository for the draft is found at https://github.com/
   core-wg/dynlink

   o Examples need to be added.

Copyright Notice

Table of Contents

1. Introduction

   IETF Standards for machine to machine communication in constrained
   environments describe a REST protocol and a set of related
   information standards that may be used to represent machine data and
   machine metadata in REST interfaces. CoRE Link-format is a standard
   for doing Web Linking [RFC5988] in constrained environments.

   This specification introduces the concept of a Link Binding, which
   defines a new link relation type to create a dynamic link between
   resources over which to exchange state updates. Specifically, a Link
   Binding is a link for binding the state of 2 resources together such
   that updates to one are sent over the link to the other. CoRE Link
   Format representations are used to configure, inspect, and maintain
   Link Bindings. This specification additionally defines a set of
   conditional Observe Attributes for use with Link Bindings and with
   the standalone CoRE Observe [RFC7641] method.

2. Terminology

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

   This specification requires readers to be familiar with all the terms
   and concepts that are discussed in [RFC5988] and [RFC6690]. This
   specification makes use of the following additional terminology:

   Link Binding: A unidirectional logical link between a source
      resource and a destination resource, over which state information
      is synchronized.

   State Synchronization: Depending on the binding method (Polling,
      Observe, Push) different REST methods may be used to synchronize
      the resource values between a source and a destination. The
      process of using a REST method to achieve this is defined as
      "State Synchronization". The endpoint triggering the state
      synchronization is the synchronization initiator.

3. Link Bindings

   In a M2M RESTful environment, endpoints may directly exchange the
   content of their resources to operate the distributed system. For
   example, a light switch may supply on-off control information that
   may be sent directly to a light resource for on-off control.
   Beforehand, a configuration phase is necessary to determine how the
   resources of the different endpoints are related to each other. This

can be done either automatically using discovery mechanisms or by
means of human intervention and a so-called commissioning tool. In
this specification the abstract relationship between two resources is
called a link Binding. The configuration phase necessitates the
exchange of binding information so a format recognized by all CoRE
endpoints is essential. This specification defines a format based on
the CoRE Link-Format to represent binding information along with the
rules to define a binding method which is a specialized relationship
between two resources. The purpose of a binding is to synchronize
the content between a source resource and a destination resource.
The destination resource MAY be a group resource if the authority
component of the destination URI contains a group address (either a
multicast address or a name that resolves to a multicast address).
Since a binding is unidirectional, the binding entry defining a
relationship is present only on one endpoint. The binding entry may
be located either on the source or the destination endpoint depending
on the binding method.

3.1. Binding Methods

A binding method defines the rules to generate the web-transfer
exchanges that synchronize state between source and destination
resources. By using REST methods content is sent from the source
resource to the destination resource.

The following table gives a summary of the binding methods defined in
this specification.

```
+---------+------------+-------------+---------------+
| Name    | Identifier | Location    | Method        |
+---------+------------+-------------+---------------+
| Polling | poll       | Destination | GET           |
|         |            |             |               |
| Observe | obs        | Destination | GET + Observe |
|         |            |             |               |
| Push    | push       | Source      | PUT           |
+---------+------------+-------------+---------------+
```

Table 1: Binding Method Summary

The description of a binding method must define the following
aspects:

Identifier: This is the value of the "bind" attribute used to
   identify the method.

Location: This information indicates whether the binding entry is
   stored on the source or on the destination endpoint.

REST Method: This is the REST method used in the Request/Response
   exchanges.

Conditions: A binding method definition must state how the condition
   attributes of the abstract binding definition are actually used in
   this specialized binding.

The binding methods are described in more detail below.

3.1.1. Polling

The Polling method consists of sending periodic GET requests from the
destination endpoint to the source resource and copying the content
to the destination resource. The binding entry for this method MUST
be stored on the destination endpoint. The destination endpoint MUST
ensure that the polling frequency does not exceed the limits defined
by the pmin and pmax attributes of the binding entry. The copying
process MAY filter out content from the GET requests using value-
based conditions (e.g based on the Change Step, Less Than, Greater
Than attributes).

3.1.2. Observe

The Observe method creates an observation relationship between the
destination endpoint and the source resource. On each notification
the content from the source resource is copied to the destination
resource. The creation of the observation relationship requires the
CoAP Observation mechanism [RFC7641] hence this method is only
permitted when the resources are made available over CoAP. The
binding entry for this method MUST be stored on the destination
endpoint. The binding conditions are mapped as query string
parameters (see Section 4.2).

3.1.3. Push

When the Push method is assigned to a binding, the source endpoint
sends PUT requests to the destination resource when the binding
condition attributes are satisfied for the source resource. The
source endpoint MUST only send a notification request if the binding
conditions are met. The binding entry for this method MUST be stored
on the source endpoint.

3.2. Link Relation

Since Binding involves the creation of a link between two resources,
Web Linking and the CoRE Link-Format are a natural way to represent
binding information. This involves the creation of a new relation
type, named "boundto". In a Web link with this relation type, the

target URI contains the location of the source resource and the
context URI points to the destination resource.

3.3. Binding Attributes

Web link attributes allow a fine-grained control of the type of state
synchronization along with the conditions that trigger an update.
This specification defines the attributes below:

```
+--------------------+-----------+------------------+
| Attribute          | Parameter | Value            |
+--------------------+-----------+------------------+
| Binding method     | bind      | xsd:string       |
|                    |           |                  |
| Minimum Period (s) | pmin      | xsd:integer (>0) |
|                    |           |                  |
| Maximum Period (s) | pmax      | xsd:integer (>0) |
|                    |           |                  |
| Change Step        | st        | xsd:decimal (>0) |
|                    |           |                  |
| Greater Than       | gt        | xsd:decimal      |
|                    |           |                  |
| Less Than          | lt        | xsd:decimal      |
+--------------------+-----------+------------------+
```

Table 2: Binding Attributes Summary

3.3.1. Bind Method (bind)

This is the identifier of a binding method which defines the rules to
synchronize the destination resource. This attribute is mandatory.

3.3.2. Minimum Period (pmin)

When present, the minimum period indicates the minimum time to wait
(in seconds) before triggering a new state synchronization (even if
it has changed). In the absence of this parameter, the minimum
period is up to the synchronization initiator. The minimum period
MUST be greater than zero otherwise the receiver MUST return a CoAP
error code 4.00 "Bad Request" (or equivalent).

3.3.3. Maximum Period (pmax)

When present, the maximum period indicates the maximum time in
seconds between two consecutive state synchronizations (regardless if
it has changed). In the absence of this parameter, the maximum
period is up to the synchronization initiator. The maximum period
MUST be greater than zero and MUST be greater than the minimum period

   parameter (if present) otherwise the receiver MUST return a CoAP
   error code 4.00 "Bad Request" (or equivalent).

[3.3.4](). Change Step (st)

   When present, the change step indicates how much the value of a
   resource SHOULD change before triggering a new state synchronization
   (compared to the value of the previous synchronization). Upon
   reception of a query including the st attribute the current value
   (CurrVal) of the resource is set as the initial value (STinit). Once
   the resource value differs from the STinit value (i.e. CurrVal >=
   STinit + ST or CurrVal <= STint - ST) then a new state
   synchronization occurs. STinit is then set to the state
   synchronization value and new state synchronizations are based on a
   change step against this value. The change step MUST be greater than
   zero otherwise the receiver MUST return a CoAP error code 4.00 "Bad
   Request" (or equivalent).

   Note: Due to the state synchronization based update of STint it may
   result in that resource value received in two sequential state
   synchronizations differs by more than st.

[3.3.5](). Greater Than (gt)

   When present, Greater Than indicates the upper limit value the
   resource value SHOULD cross before triggering a new state
   synchronization. State synchronization only occurs when the resource
   value exceeds the specified upper limit value. The actual resource
   value is used for the synchronization rather than the gt value. If
   the value continues to rise, no new state synchronizations are
   generated as a result of gt. If the value drops below the upper
   limit value and then exceeds the upper limit then a new state
   synchronization is generated.

[3.3.6](). Less Than (lt)

   When present, Less Than indicates the lower limit value the resource
   value SHOULD cross before triggering a new state synchronization.
   State synchronization only occurs when the resource value is less
   than the specified lower limit value. The actual resource value is
   used for the synchronization rather than the lt value. If the value
   continues to fall no new state synchronizations are generated as a
   result of lt. If the value rises above the lower limit value and
   then drops below the lower limit then a new state synchronization is
   generated.

3.3.7. **Attribute Interactions**

Pmin, pmax, st, gt and lt may be present in the same query.

If pmin and pmax are present in a query then they take precedence
over the other parameters. Thus even if st, gt or lt are met, if
pmin has not been exceeded then no state synchronization occurs.
Likewise if st, gt or lt have not been met and pmax time has expired
then state synchronization occurs. The current value of the resource
is used for the synchronization. If pmin time is exceeded and st, gt
or lt are met then the current value of the resource is synchronized.
If st is also included, a state synchronization resulting from pmin
or pmax updates STinit with the synchronized value.

If gt and lt are included gt MUST be greater than lt otherwise an
error CoAP error code 4.00 "Bad Request" (or equivalent) MUST be
returned.

If st is included in a query with a gt or lt attribute then state
synchronizations occur only when the conditions described by st AND
gt or st AND gl are met.

4. **Binding Table**

The binding table is a special resource that gives access to the
bindings on a endpoint. A binding table resource MUST support the
Binding interface defined below. A profile SHOULD allow only one
resource table per endpoint.

4.1. **Binding Interface Description**

This section defines a REST interface for Binding table resources.
The interface supports the link-format type.

The if= column defines the Interface Description (if=) attribute
value to be used in the CoRE Link Format for a resource conforming to
that interface. When this value appears in the if= attribute of a
link, the resource MUST support the corresponding REST interface
described in this section. The resource MAY support additional
functionality, which is out of scope for this specification.
Although this interface description is intended to be used with the
CoRE Link Format, it is applicable for use in any REST interface
definition.

The Methods column defines the REST methods supported by the
interface, which are described in more detail below.

```
+-----------+----------+------------------+----------------+
| Interface | if=      | Methods          | Content-Formats |
+-----------+----------+------------------+----------------+
| Binding   | core.bnd | GET, POST, DELETE | link-format    |
+-----------+----------+------------------+----------------+
```

Table 3: Binding Interface Description

The Binding interface is used to manipulate a binding table. A
request with a POST method and a content format of application/link-
format simply appends new bindings to the table. All links in the
payload MUST have a relation type "boundTo". A GET request simply
returns the current state of a binding table whereas a DELETE request
empties the table. Individual entries may be dreeleted from the
table by specifying the resource path in a DELETE request.

The following example shows requests for adding, retrieving and
deleting bindings in a binding table.

```
Req: POST /bnd/ (Content-Format: application/link-format)
<coap://sensor.example.com/s/light>;
  rel="boundto";anchor="/a/light";bind="obs";pmin="10";pmax="60"
Res: 2.04 Changed

Req: GET /bnd/
Res: 2.05 Content (application/link-format)
<coap://sensor.example.com/s/light>;
  rel="boundto";anchor="/a/light";bind="obs";pmin="10";pmax="60"

Req: DELETE /bnd/a/light
Res: 2.04 Changed

Req: DELETE /bnd/
Res: 2.04 Changed
```

Figure 1: Binding Interface Example

4.2. Resource Observation Attributes

When resource interfaces following this specification are made
available over CoAP, the CoAP Observation mechanism [RFC7641] MAY be
used to observe any changes in a resource, and receive asynchronous
notifications as a result. In addition, a set of query string
parameters are defined here to allow a client to control how often a
client is interested in receiving notifications and how much a
resource value should change for the new representation to be
interesting. These query parameters are described in the following
table. A resource using an interface description defined in this

specification and marked as Observable in its link description SHOULD
support these observation parameters. The Change Step parameter can
only be supported on resources with an atomic numeric value.

These query parameters MUST be treated as resources that are read
using GET and updated using PUT, and MUST NOT be included in the
Observe request. Multiple parameters MAY be updated at the same time
by including the values in the query string of a PUT. Before being
updated, these parameters have no default value.

```
+----------------+------------------+------------------+
| Resource       | Parameter        | Data Format      |
+----------------+------------------+------------------+
| Minimum Period | /{resource}?pmin | xsd:integer (>0) |
|                |                  |                  |
| Maximum Period | /{resource}?pmax | xsd:integer (>0) |
|                |                  |                  |
| Change Step    | /{resource}?st   | xsd:decimal (>0) |
|                |                  |                  |
| Less Than      | /{resource}?lt   | xsd:decimal      |
|                |                  |                  |
| Greater Than   | /{resource}?gt   | xsd:decimal      |
+----------------+------------------+------------------+
```

Table 4: Resource Observation Attribute Summary

Minimum Period: As per Section 3.3.2

Maximum Period: As per Section 3.3.3

Change Step: As per Section 3.3.4

Greater Than: As per Section 3.3.5

Less Than: As per Section 3.3.6

5. Security Considerations

An implementation of a client needs to be prepared to deal with
responses to a request that differ from what is specified in this
specification. A server implementing what the client thinks is a
resource with one of these interface descriptions could return
malformed representations and response codes either by accident or
maliciously. A server sending maliciously malformed responses could
attempt to take advantage of a poorly implemented client for example
to crash the node or perform denial of service.

6. IANA Considerations

6.1. Interface Description

The specification registers the "binding" CoRE interface description
link target attribute value as per [RFC6690].

Attribute Value: binding

Editor's note: RFC6690 actually indicates the use of core. for CoRE
WG documents. Therefore it probably is more correct to register
core.binding . However this may cause a problem for existing
implementations. One approach may be to register two attributes
"binding" and "core.binding."

Description: The binding interface is used to manipulate a binding
table which describes the link bindings between source and
destination resources for the purposes of synchronizing their
content.

Reference: This specification. Note to RFC editor: please insert the
RFC of this specification.

Notes: None

6.2. Link Relations Type

This specification registers the new "bounto" link relation type as
per [RFC5988].

Relation Name: boundto

Description: The purpose of a boundto relation type is to indicate
   that there is a binding between a source resource and a
   destination resource for the purposes of synchronizing their
   content.

Reference: This specification. Note to RFC editor: please insert
   the RFC of this specification.

Notes: None

Application Data: None

7. Acknowledgements

   Acknowledgement is given to colleagues from the SENSEI project who
   were critical in the initial development of the well-known REST
   interface concept, to members of the IPSO Alliance where further
   requirements for interface types have been discussed, and to Szymon
   Sasin, Cedric Chauvenet, Daniel Gavelle and Carsten Bormann who have
   provided useful discussion and input to the concepts in this
   specification.

8. Changelog

   draft-ietf-core-dynlink-01

   o General: The term state synchronization has been introduced to
      describe the process of synchronization between destination and
      source resources.

   o General: The document has been restructured the make the
      information flow better.

   o Clause 3.1: The descriptions of the binding attributes have been
      updated to clarify their usage.

   o Clause 3.1: A new clause has been added to discuss the
      interactions between the resources.

   o Clause 3.4: Has been simplified to refer to the descriptions in
      3.1. As the text was largely duplicated.

   o Clause 4.1: Added a clarification that individual resources may be
      removed from the binding table.

   o Clause 6: Formailised the IANA considerations.

   draft-ietf-core-dynlink Initial Version 00:

   o This is a copy of draft-groves-core-dynlink-00

   draft-groves-core-dynlink Draft Initial Version 00:

   o This initial version is based on the text regarding the dynamic
      linking functionality in I.D.ietf-core-interfaces-05.

   o The WADL description has been dropped in favour of a thorough
      textual description of the REST API.

9. References

9.1. Normative References

[RFC2119]   Bradner, S., "Key words for use in RFCs to Indicate
            Requirement Levels", BCP 14, RFC 2119,
            DOI 10.17487/RFC2119, March 1997,
            <http://www.rfc-editor.org/info/rfc2119>.

[RFC5988]   Nottingham, M., "Web Linking", RFC 5988,
            DOI 10.17487/RFC5988, October 2010,
            <http://www.rfc-editor.org/info/rfc5988>.

[RFC6690]   Shelby, Z., "Constrained RESTful Environments (CoRE) Link
            Format", RFC 6690, DOI 10.17487/RFC6690, August 2012,
            <http://www.rfc-editor.org/info/rfc6690>.

9.2. Informative References

[RFC7252]   Shelby, Z., Hartke, K., and C. Bormann, "The Constrained
            Application Protocol (CoAP)", RFC 7252,
            DOI 10.17487/RFC7252, June 2014,
            <http://www.rfc-editor.org/info/rfc7252>.

[RFC7641]   Hartke, K., "Observing Resources in the Constrained
            Application Protocol (CoAP)", RFC 7641,
            DOI 10.17487/RFC7641, September 2015,
            <http://www.rfc-editor.org/info/rfc7641>.

Authors' Addresses

   Zach Shelby
   ARM
   150 Rose Orchard
   San Jose 95134
   FINLAND

   Phone: +1-408-203-9434
   Email: zach.shelby@arm.com

   Matthieu Vial
   Schneider-Electric
   Grenoble
   FRANCE

   Phone: +33 (0)47657 6522
   Email: matthieu.vial@schneider-electric.com

Michael Koster
SmartThings
665 Clyde Avenue
Mountain View 94043
USA

Email: michael.koster@smartthings.com

Christian Groves
Huawei
Australia

Email: Christian.Groves@nteczone.com