CoRE Working Group
Internet-Draft
Intended status: Standards Track
Expires: 17 September 2026

B. Silverajan
Tampere University
M. Koster
Dogtiger Labs
A. Soloway
Qualcomm Technologies, Inc.

16 March 2026

Conditional Query Parameters for CoAP Observe
draft-ietf-core-conditional-attributes-12

Abstract

This specification defines Conditional Notification and Control Query Parameters compatible with CoAP Observe (RFC7641).

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-core-conditional-attributes/.

Discussion of this document takes place on the core Working Group mailing list (mailto:core@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/core/. Subscribe at https://www.ietf.org/mailman/listinfo/core/.

Source for this draft and an issue tracker can be found at https://github.com/core-wg/conditional-attributes.

Status of This Memo

This Internet-Draft will expire on 17 September 2026.

Table of Contents

1. Introduction

IETF Standards for Internet of Things (IoT) communication in constrained environments define the Constrained Application Protocol (CoAP) [RFC7252], a RESTful application protocol, as well as a set of related information standards that may be used to represent machine data and machine metadata in REST interfaces.

This specification defines Conditional Notification and Control Parameters for use with CoAP Observe [RFC7641].

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This specification requires readers to be familiar with all the terms and concepts that are discussed in [RFC7252] and [RFC7641]. This specification makes use of the following additional terminology:

Notification Band: A resource value range that may be bounded by a minimum and maximum value or may be unbounded having either a minimum or maximum value.

"xs:boolean" and "xs:decimal" types: Data types from XML Schema [W3C.REC-xmlschema-2-20041028], used in this document to describe boolean and scalar values in CoAP resources. The "xs:" prefix notation is used solely for type indication and does not imply the use of XML or XML Schema in protocol encoding.

3. Conditional Query Parameters

This specification defines conditional query parameters (or more simply, "conditional parameters" in this document) for use with CoRE Observe [RFC7641]. Conditional parameters provide fine-grained control of notification and synchronization of resource states. A CoAP client conveys conditional parameters as metadata using the query component of a CoAP URI. A conditional parameter can be

represented as a "name=value" query parameter or simply a "name" without a value. A conditional parameter can be one of two kinds: A conditional notification parameter, or a conditional control parameter. Multiple conditional parameters in a query component are separated with an ampersand "&". A resource marked as Observable in its link description SHOULD support these conditional parameters.

This specification also defines conditional query parameters as parameters that apply to scalar and boolean values in CoAP resources. While complex data structures (e.g., SenML, CBOR arrays, or other structured formats) are commonly used in IoT systems, this document does not provide explicit guidance on how conditional parameters should interact with these formats.

This specification assumes that there are finite quantization effects in the internal or external updates to the value representing the state of a resource; specifically, that a resource state may be updated at any time with any valid value. We therefore avoid any continuous-time assumptions in the description of the conditional parameters and instead use the phrase "sampled value" to refer to a member of a sequence of values that may be internally observed from the resource state over time.

3.1. Overview

If a CoAP client is interested in obtaining all the state representations of a resource from a CoAP server as they change, the client is able to do so by using CoAP Observe. If a CoAP client is instead interested in receiving only state representations fulfilling certain constraints (such as a minimum/maximum value), it can do so by indicating conditional parameters as query parameters in its request to a CoAP server, when registering its interest in observing a resource.

The usage of conditional attributes employs the notion of resource state projection. This is an idea that aligns with established practices employed by RESTful API designs that allow clients to retrieve specific representations or subsets of a resource’s data, enhancing efficiency and flexibility.

In constrained environments, CoAP clients can employ resource state projections as a technique to reduce unnecessary data transfer in constrained environments. By using Observe with query parameters, the client requests the server to project a new state from the current resource representation and deliver only a subset of updates, based on the received requests. When a server receives a request containing conditional query parameters from a client, the server maintains a projected resource state separate from a resource state requested without conditional query parameters.

The mechanism can be explained in the following subsections in terms of registration, operation and cancellation.

3.2. Registration

In this example, three CoAP endpoints are shown: Clients A and B are interested in obtaining updates to state representations describing the current CO2 level, provided by a CoAP Server.

In Figure 1, Client A uses CoAP Observe to register its interest in receiving all updates to the CO2 resource state from the Server.

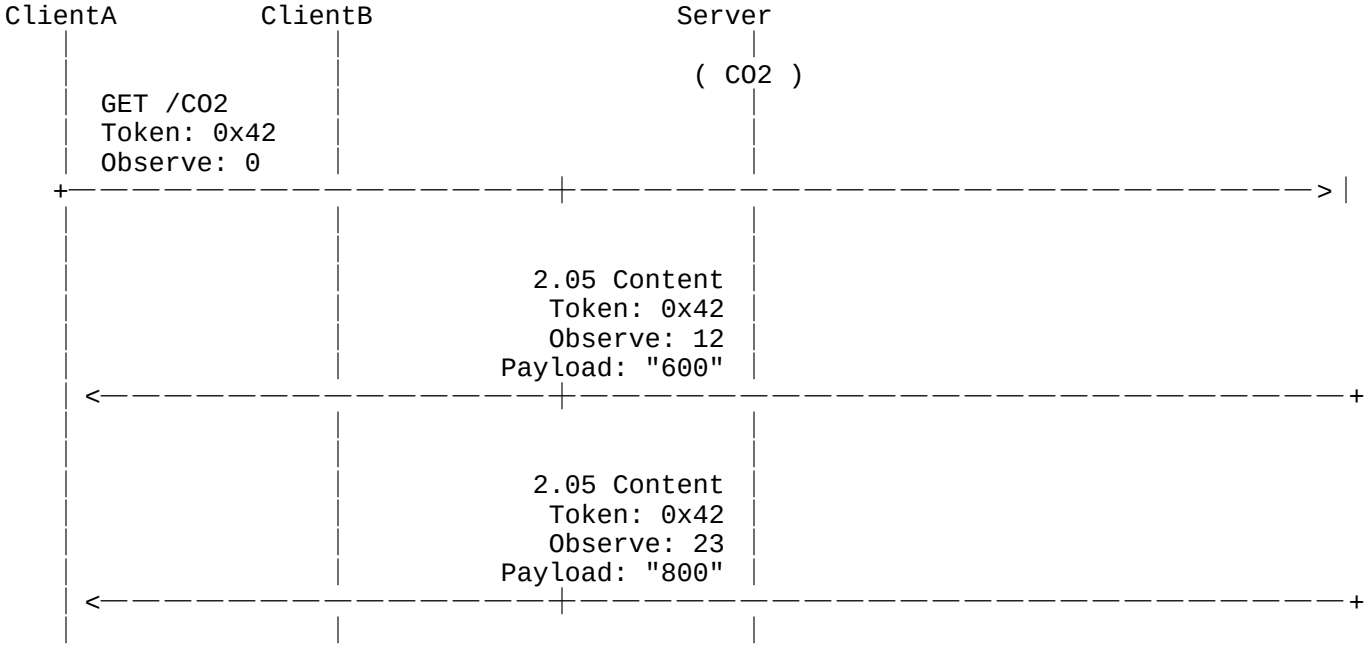

Figure 1: Client A registers and receives one notification of the current state and one state update.

Client B, on the other hand is interested in receiving only a subset of updates from the Server. In Figure 2, Client B is depicted using CoAP Observe with a conditional parameter to register its interest in receiving specific updates to the C02 resource state from the Server. The Server provides a representation of the current state and creates and creates a new state projection registering Client B's interest.

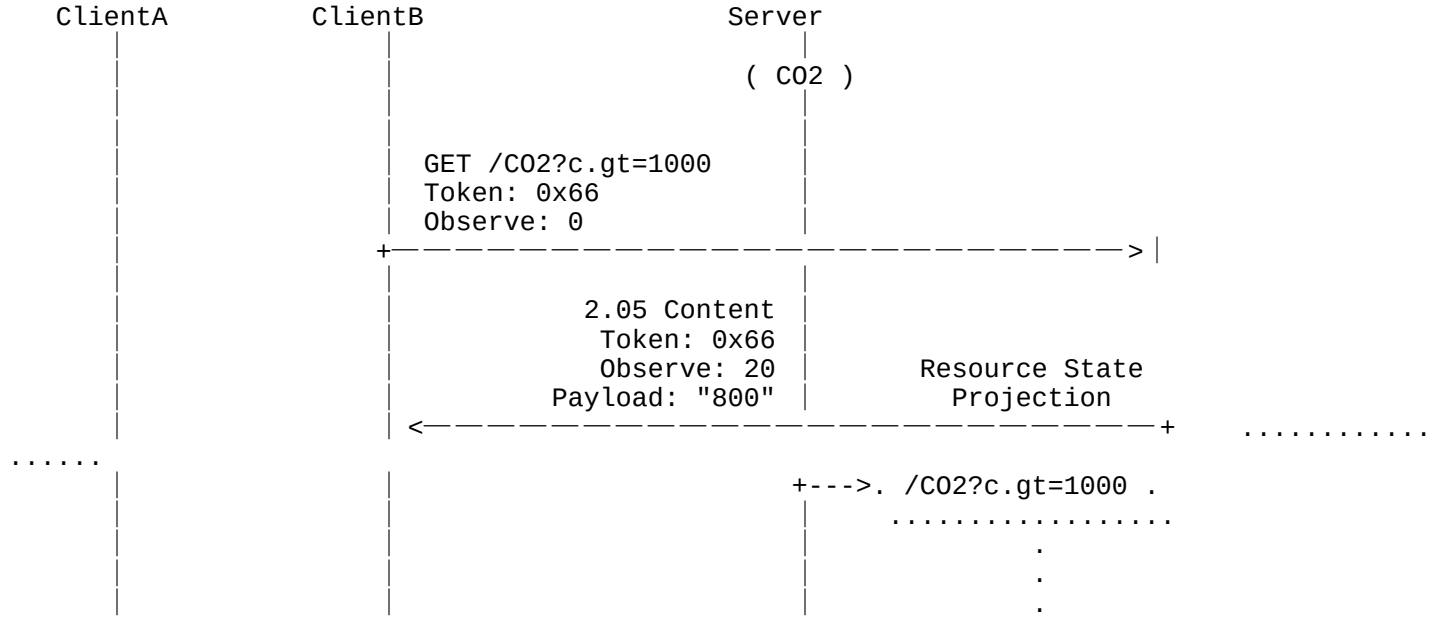

Figure 2: Client B registers with conditional parameters, and receives one notification of the current state and a state projection is created.

3.3. Operation

In subsequent interactions for providing state updates, the Server will continue to provide all state updates to Client A, while Client B receives state updates fulfilling the conditions specified by the conditional parameter.

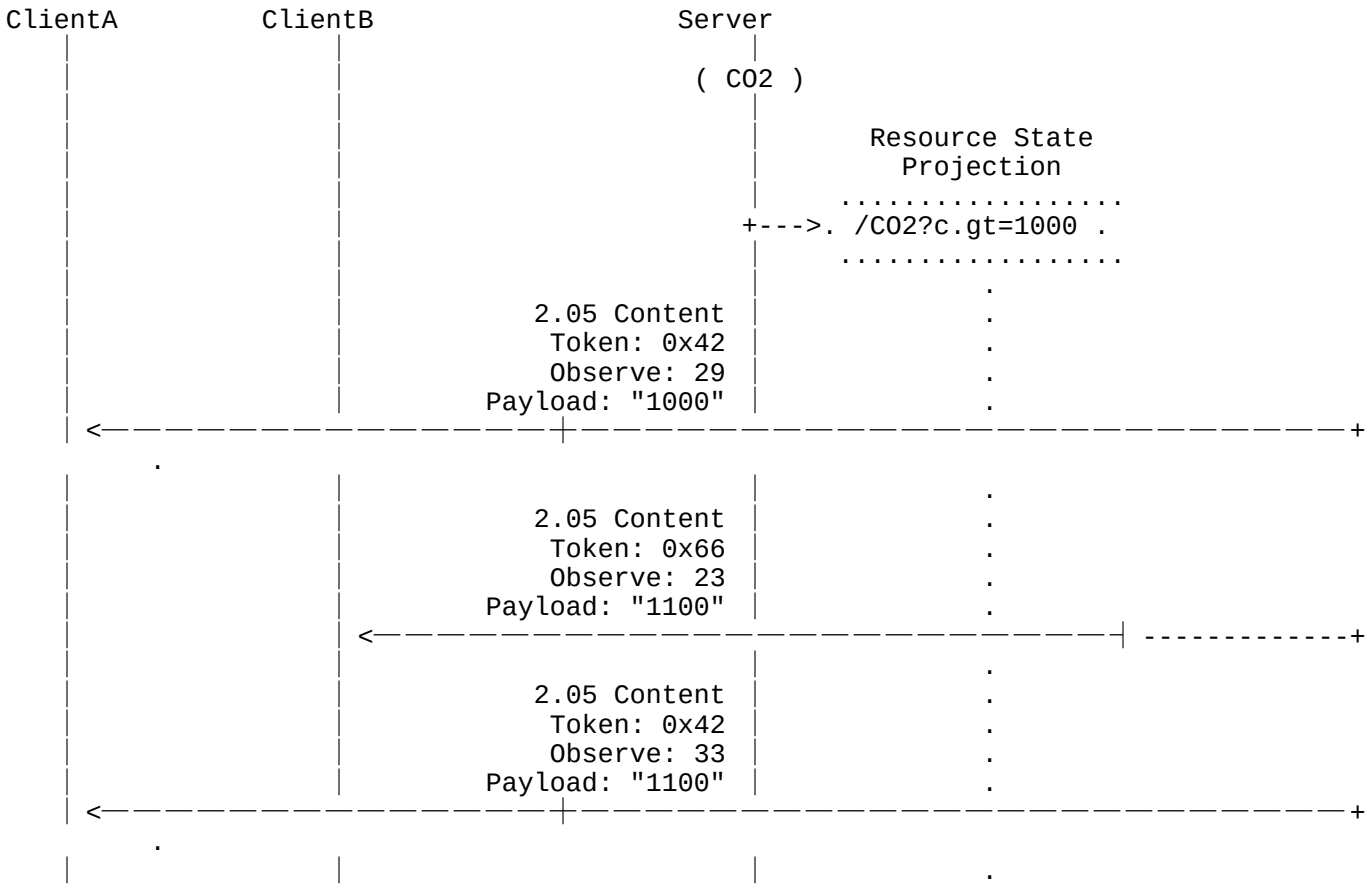

Figure 3: Clients A and B receiving C02 state updates from the Server, without and with conditional parameters, respectively.

3.4. Cancellation

A client that wishes to cancel an existing registration can do so in accordance with Section 3.6 of [RFC7641]. If a client wishes to explicitly cancel an existing registration by issuing a GET request, it MUST also additionally supply the original URI containing the conditional parameters that was conveyed to the server during the registration. This is depicted in Figure 4 for Client B.

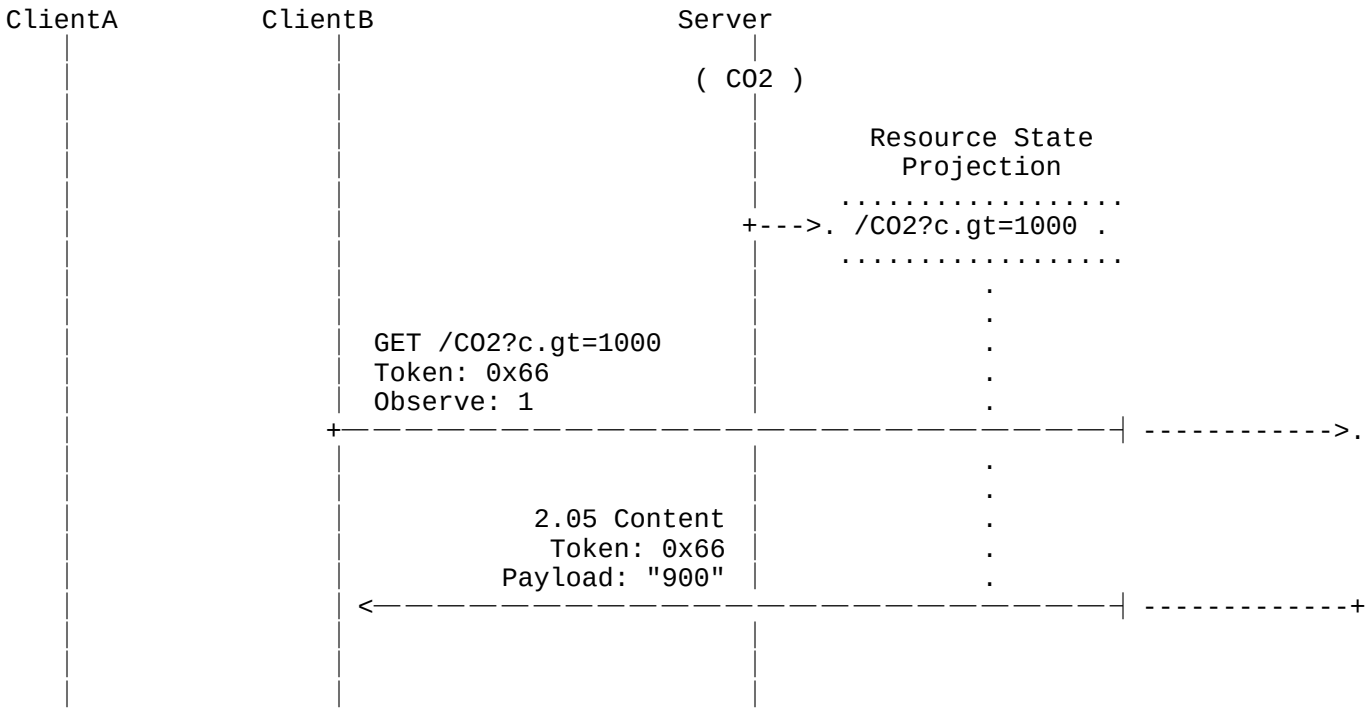

Figure 4: Client B explicitly cancelling an existing registration.

3.5. Conditional Notification Parameters

Conditional Notification Parameters define the conditions that trigger a notification. Conditional Notification Parameters SHOULD be evaluated on all potential notifications from a resource, whether resulting from an internal server-driven sampling process or from external update requests to the server.

The set of Conditional Notification Parameters defined here allows a client to control how often a notification is received and how much a representation state should change in order to trigger a notification. One or more Conditional Notification Parameters MAY be included in an Observe request.

Conditional Notification Parameters are defined below:

Parameter	Name	Value Type
Greater Than	c.gt	xs:decimal
Less Than	c.lt	xs:decimal
Change Step	c.st	xs:decimal (>0)
Notification Band	c.band	(none)
Edge	c.edge	xs:boolean

Table 1: Conditional Notification Parameters

3.5.1. Greater Than (c.gt)

When present, Greater Than indicates the upper limit value the sampled value SHOULD cross before triggering a notification. A notification is sent whenever the sampled value crosses the specified upper limit value, relative to the last reported value, and the time for "c.pmin" has elapsed since the last notification. The sampled value is sent in the notification. If the value continues to rise, no notifications are generated as a result of "c.gt". If the value drops below the upper limit value then a notification is sent, subject again to the "c.pmin" time.

The Greater Than parameter MUST be supported on resources with a scalar numeric value.

3.5.2. Less Than (c.lt)

When present, Less Than indicates the lower limit value the resource value SHOULD cross before triggering a notification. A notification is sent whenever the sampled value crosses the specified lower limit value, relative to the last reported value, and the time for "c.pmin" has elapsed since the last notification. The sampled value is sent in the notification. If the value continues to fall no notifications are generated as a result of "c.lt". If the value rises above the lower limit value then a new notification is sent, subject to the "c.pmin" time.

The Less Than parameter MUST be supported on resources with a scalar numeric value.

3.5.3. Change Step (c.st)

When present, Change step indicates how much the value representing a resource state SHOULD change before triggering a notification, compared to the previous resource state. Upon reception of a query including the "c.st" parameter, the current resource state representing the most recently sampled value is reported, and then set as the last reported value (last_rep_v). When a subsequent sampled value or update of the resource state differs from the last reported state by an amount, positive or negative, greater than or equal to "c.st", and the time for "c.pmin" has elapsed since the last notification, a notification is sent and the last reported value is updated to the new resource state sent in the notification. The change step MUST be greater than zero, otherwise the receiver MUST return a CoAP error code 4.00 "Bad Request".

The Change Step parameter MUST be supported on resources with a scalar numeric value.

Note: due to sampling and other constraints, e.g., "c.pmin", the change in resource states received in two sequential notifications may differ by more than "c.st".

3.5.4. Notification Band (c.band)

The Notification Band parameter allows a bounded or unbounded (based on a minimum or maximum) value range that may trigger multiple notifications. This enables use cases where different ranges result in differing behaviour. For example, in monitoring the temperature of machinery, whilst the temperature is in the normal operating range, only periodic updates are needed. However as the temperature moves to more abnormal ranges, more frequent state updates may be sent to clients.

Without a notification band, a transition across a Less Than (c.lt), or Greater Than (c.gt) limit only generates one notification. This means that it is not possible to describe a case where multiple notifications are sent so long as the limit is exceeded.

The "c.band" parameter works as a modifier to the behaviour of "c.gt" and "c.lt". Its use is determined only by its presence, as this parameter takes no value. Therefore, if "c.band" is present in a query, "c.gt", "c.lt", or both, MUST be included.

When "c.band" is present with "c.lt" but without "c.gt", the lower bound for the notification band (notification band minimum) is defined. Notifications occur when the resource value is equal to or above the notification band minimum. No maximum values exist for the band.

When "c.band" is present with "c.gt" but without "c.lt", the upper bound for the notification band (notification band maximum) is defined. Notifications occur when the resource value is equal to or below the notification band maximum. No minimum values exist for the band.

If "c.band" is specified and the value of "c.gt" is less than that of "c.lt", in-band notification occurs. That is, notification occurs whenever the resource value is between the "c.gt" and "c.lt" values, including equal to "c.gt" or "c.lt".

If "c.band" is specified and the value of "c.gt" is greater than that of "c.lt", out-of-band notification occurs. That is, notification occurs when the resource value is not between the "c.gt" and "c.lt" values, excluding equal to "c.gt" and "c.lt".

The Notification Band parameter MUST be supported on resources with a scalar numeric value.

3.5.5. Edge (c.edge)

When present, the Edge parameter indicates interest for receiving notifications of either the falling edge or the rising edge transition of a boolean resource state. When the value of the "c.edge" parameter is 0 (False), the server notifies the client each time a resource state changes from True to False. When the value of the "c.edge" parameter is 1 (True), the server notifies the client each time a resource state changes from False to True.

The "c.edge" parameter MUST be supported on resources with a boolean value.

3.6. Conditional Control Parameters

Conditional Control Parameters define the time intervals between consecutive notifications as well as the cadence of the evaluation of the conditions that trigger a notification. Conditional Control Parameters can be used to configure the internal server-driven sampling process for performing evaluations of the conditions of a resource. One or more Conditional Control Parameters MAY be included in an Observe request.

Conditional Control Parameters are defined below:

Parameter	Name	Value Type
Minimum Period (s)	c.pmin	xs:decimal (>0)
Maximum Period (s)	c.pmax	xs:decimal (>0)
Minimum Evaluation Period (s)	c.epmin	xs:decimal (>0)
Maximum Evaluation Period (s)	c.epmax	xs:decimal (>0)
Confirmable Notification	c.con	xs:boolean

Table 2: Conditional Control Parameters

3.6.1. Minimum Period (c.pmin)

When present, Minimum Period indicates the minimum time, in seconds, between two consecutive notifications (whether or not the resource state has changed). The value is a floating-point number, allowing for sub-second precision (e.g., 0.5 for half a second or 0.01 for 10 milliseconds). If a server does not support sub-second intervals, it MAY round the value up to the nearest supported resolution. In the absence of this parameter, the minimum period is up to the server. Minimum Period MUST be greater than zero, otherwise the receiver MUST return a CoAP error code 4.00 "Bad Request".

A server MAY update the resource state with the last sampled value that occurred during the "c.pmin" interval, after the "c.pmin" interval expires.

Note: due to finite quantization effects, the time between notifications may be greater than "c.pmin" even when the sampled value changes within the "c.pmin" interval. "c.pmin" may or may not be used to drive the internal sampling process.

3.6.2. Maximum Period (c.pmax)

When present, Maximum Period indicates the maximum time, in seconds, between two consecutive notifications (regardless of whether or not the resource state has changed). The value is a floating-point number, allowing for sub-second precision (e.g., 0.5 for half a second or 0.01 for 10 milliseconds). If a server does not support sub-second intervals, it MAY round the value up to the nearest supported resolution. In the absence of this parameter, the maximum

period is up to the server. Maximum Period MUST be greater than zero and MUST be greater than or equal to Minimum Period (if present), otherwise the receiver MUST return a CoAP error code 4.00 "Bad Request".

3.6.3. Minimum Evaluation Period (c.epmin)

When present, Minimum Evaluation Period indicates the minimum time, in seconds, the client recommends to the server to wait between two consecutive evaluations of the conditions of a resource, since the client has no interest in the server doing more frequent evaluations. The value is a floating-point number, allowing for sub-second precision (e.g., 0.5 for half a second or 0.01 for 10 milliseconds). If a server does not support sub-second intervals, it MAY round the value up to the nearest supported resolution. When the value of Minimum Evaluation Period expires after the previous evaluation, the server MAY immediately perform a new evaluation. In the absence of this parameter, the minimum evaluation period is not defined and thus not used by the server. The server MAY use "c.pmin", if defined, as a guidance on the desired evaluation cadence. Minimum Evaluation Period MUST be greater than zero, otherwise the receiver MUST return a CoAP error code 4.00 "Bad Request".

3.6.4. Maximum Evaluation Period (c.epmax)

When present, Maximum Evaluation Period indicates the maximum time, in seconds, the server MAY wait between two consecutive evaluations of the conditions of a resource. The value is a floating-point number, allowing for sub-second precision (e.g., 0.5 for half a second or 0.01 for 10 milliseconds). If a server does not support sub-second intervals, it MAY round the value up to the nearest supported resolution. When the value of Maximum Evaluation Period expires after the previous evaluation, the server MUST immediately perform a new evaluation. In the absence of this parameter, the maximum evaluation period is not defined and thus not used by the server. Maximum Evaluation Period MUST be greater than zero and MUST be greater than Minimum Evaluation Period (if present), otherwise the receiver MUST return a CoAP error code 4.00 "Bad Request".

3.6.5. Confirmable Notification (c.con)

When present with a value of 1 (True), Confirmable Notification indicates that a notification MUST be confirmable, i.e., the server MUST send the notification in a confirmable CoAP message, to request an acknowledgement from the client. When present with a value of 0 (False), Confirmable Notification indicates a notification can be confirmable or non-confirmable, i.e., it can be sent in a confirmable or a non-confirmable CoAP message.

3.7. Server processing of Conditional Parameters

Conditional Notification Parameters and Conditional Control Parameters may be present in the same query. However, they are not defined at multiple prioritization levels. The server sends a notification whenever any of the parameter conditions are met, upon which it updates its last notification value and time to prepare for the next notification. When Conditional Notification Parameters and Conditional Control Parameters are present in the same query, notifications may be subjected to the presence of a Conditional Control Parameter such as "c.pmin" or "c.pmax". Only one notification occurs when there are multiple conditions being met at the same time. As a general example, the pseudocode illustrated in Appendix A shows one way to determine when a notification is to be sent.

4. Implementation Considerations

If a conditional parameter is provided with an inappropriate data type (e.g., "c.edge=10" where "c.edge" is expected to be boolean), the server MUST reject the request with 4.00 Bad Request.

When "c.pmax" and "c.pmin" are equal, the expected behaviour is that notifications will be sent every (c.pmin == c.pmax) seconds. However, these notifications can only be fulfilled by the server on a best effort basis. Because "c.pmin" and "c.pmax" are designed as acceptable tolerance bounds for sending state updates, a query from an interested client containing equal "c.pmin" and "c.pmax" values must not be seen as a hard real-time scheduling contract between the client and the server.

The use of the notification band minimum and maximum allows for a synchronization whenever a change in the resource value occurs. Theoretically, this could occur in-line with the server internal sample period or as defined by the "c.epmin" and "c.epmax" values for determining the resource value. Implementors SHOULD consider the resolution needed before updating the resource, e.g., updating the resource when a temperature sensor value changes by 0.001 degree versus 1 degree.

When a server has multiple observations with different measurement cadences as defined by the "c.epmin" and "c.epmax" values, the server MAY evaluate all observations when performing the measurement of any one observation.

An implementation might choose to apply conditions like c.gt or c.lt to the v (value) field in SenML-based resources. However, this behavior is not defined in this document. Implementers are

encouraged to consider how such formats may be adapted in their specific deployments. Future extensions or additional mechanisms may provide explicit guidance on supporting conditional parameters for complex data structures as well as data structures having multiple records.

This specification defines conditional parameters that can be used with CoAP Observe relationships between CoAP clients and CoAP servers. However, it is recognised that the presence of one or more proxies between a client and a server can interfere with clients receiving resource updates, if a proxy does not supply resource representations when the value remains unchanged (e.g., if "c.pmax" is set, and the server sends multiple updates when the resource state contains the same value). A server SHOULD use the Max-Age option to mitigate this, by setting Max-Age to be less than or equal to "c.pmax".

This document defines conditional query parameters that refine the behavior of a resource when used in conjunction with the Observe mechanism. As such, this specification does not require resources to advertise explicit support for conditional parameters through resource discovery. More specifically, it does not define a new CoRE Link Format (if=) interface type for advertising support of these conditional parameters. This specification intentionally avoids defining such an interface type at this stage, in order to preserve flexibility and to avoid introducing unnecessary coupling between resource interface semantics and request-time projection behavior.

Future specifications MAY define a Link Format interface type or other discovery mechanisms to explicitly advertise support for conditional parameters, should deployment experience indicate that proactive capability discovery is necessary. Such mechanisms would need to clearly specify the behavioral guarantees associated with advertising that interface.

5. Security Considerations

The security considerations in Section 11 of [RFC7252] apply.

Additionally, the security considerations in Section 7 of [RFC7641] also apply, particularly towards mitigating amplification attacks.

As noted in Section 2.2 of [I-D.irtf-t2trg-amplification-attacks], an attacker could craft GET requests combining observations with conditional parameters such as c.pmax or c.epmax with values that are below a minimum implementation-specific threshold. If a server receives such a request and is unwilling to register the observer client, the server MAY silently ignore the registration request and

process the GET request as usual. The resulting response MUST NOT include an Observe Option, the absence of which signals to the client that it will not be added to the list of observers by the server.

6. IANA Considerations

This document has the following actions for IANA:

Note to RFC Editor: Please replace all occurrences of "[RFC-XXXX]" with the RFC number of this specification and delete this paragraph.

This document establishes the "Conditional parameters" registry within the "Constrained RESTful Environments (CoRE) Parameters" registry group.

Each entry in the registry must include:

* Name: This is the human-readable name and description of the conditional parameter,

* Parameter: This is the short name, as used in query parameters,

* Value Type: The value type of the parameter (if any),

* Reference: The link to reference documentation, which must give details describing the conditional notification or control parameter and how it is to be processed.

Initial entries in this subregistry are as follows:

Name	Parameter	Value Type	Change Controller	Reference
Minimum Period (s)	c.pmin	xs:decimal (>0)	IETF	RFC XXXX
Maximum Period (s)	c.pmax	xs:decimal (>0)	IETF	RFC XXXX
Minimum Evaluation Period (s)	c.epmin	xs:decimal (>0)	IETF	RFC XXXX
Maximum Evaluation Period (s)	c.epmax	xs:decimal (>0)	IETF	RFC XXXX
Confirmable Notification	c.con	xs:boolean	IETF	RFC XXXX
Greater Than	c.gt	xs:decimal	IETF	RFC XXXX
Less Than	c.lt	xs:decimal	IETF	RFC XXXX
Change Step	c.st	xs:decimal (>0)	IETF	RFC XXXX
Notification Band	c.band	(none)	IETF	RFC XXXX
Edge	c.edge	xs:boolean	IETF	RFC XXXX

Table 3: New Conditional Parameters registry

The IANA policy for future additions to the subregistry is Expert Review, as described in [RFC8126]. The evaluation of a registration request should consider the following points:

* Clarity and correctness of registrations. Experts are expected to check the clarity of purpose and use of the new conditional parameters and associated query parameters, which have to be clearly defined in the corresponding reference documentation. Conditional parameters that do not meet these objectives of clarity and completeness MUST NOT be registered.

* Point squatting is to be discouraged. Reviewers are encouraged to get sufficient information for registration requests to ensure that a new conditional parameter is likely to be used in deployments and is not going to duplicate one that is already registered. To reduce the potential for conflict with commonly used query parameter names, it is strongly recommended that new entry names be prepended with "c." (such as entries described in Table 3).

7. References

7.1. 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, <https://www.rfc-editor.org/rfc/rfc2119>.

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

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

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/rfc/rfc8126>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[W3C.REC-xmlschema-2-20041028] Malhotra, A., Ed. and P. V. Biron, Ed., "XML Schema Part 2: Datatypes Second Edition", W3C REC REC-xmlschema-2-20041028, W3C REC-xmlschema-2-20041028, 28 October 2004, <https://www.w3.org/TR/2004/REC-xmlschema-2-20041028/>.

7.2. Informative References

[I-D.irtf-t2trg-amplification-attacks] Mattsson, J. P., Selander, G., and C. Amsss, "Amplification Attacks Using the Constrained Application

Protocol (CoAP)", Work in Progress, Internet-Draft, draft-irtf-t2trg-amplification-attacks-05, 18 June 2025, <https://datatracker.ietf.org/doc/html/draft-irtf-t2trg-amplification-attacks-05>.

Appendix A. Pseudocode: Processing Conditional Parameters

This appendix is informative. It describes the possible logic of how a server processes conditional parameters to determine when to send a notification to a client.

Note: The pseudocode is not exhaustive nor should it be treated as reference code. It depicts a subset of the conditional parameters described in this specification.

```
// struct Resource {
//
//  bool band;
//  int pmin;
//  int pmax;
//  int epmin;
//  int epmax;
//  int st;
//  int gt;
//  int lt;
//
//  time_t last_sampled_time;
//  time_t last_rep_time;

//  int curr_state;
//  int prev_state;
//
//  ...
//
// };


boolean is_notifiable( Resource * r ) {

  time_t curr_time = get_current_time();

  #define BAND_EXISTS ( r->band )

  #define LT_EXISTS ( r->lt )
  #define GT_EXISTS ( r->gt )

  #define EPMIN_TRUE ( curr_time - r->last_sampled_time >= r->epmin )
  #define EPMAX_TRUE ( curr_time - r->last_sampled_time > r->epmax )
```

```
     #define PMIN_TRUE ( curr_time - r->last_reported_time >= r->pmin )
     #define PMAX_TRUE ( curr_time - r->last_reported_time > r->pmax )

     #define LT_TRUE ( r->curr_state < r->lt ^ r->prev_state < r->lt )
     #define GT_TRUE ( r->curr_state > r->gt ^ r->prev_state > r->gt )

     #define ST_TRUE ( abs( r->curr_state - r->prev_state ) >= r->st )

     #define INBAND_TRUE ( gt < lt && \\
                          (gt <= curr_state && curr_state <= lt ))
     #define OUTOFBAND_TRUE ( lt < gt && \\
                             (gt < curr_state || curr_state < lt ))

     #define BANDMIN_TRUE ( r->lt <= r->curr_state)
     #define BANDMAX_TRUE (r->curr_state <= r->gt)


     if PMAX_TRUE {
           return true;
     }

     if PMIN_TRUE {
         if !BAND_EXISTS {
             if LT_TRUE || GT_TRUE || ST_TRUE {
                 return true;
             }
         }
         else {
          if ( (BANDMIN_TRUE && !GT_EXISTS) || \
               (BANDMAX_TRUE && !LT_EXISTS) || \
                INBAND_TRUE || \
                OUTOFBAND_TRUE ) {
              return true;
          }
         }
     }

     return false;

   }
```

Figure 5: Pseudocode showing the logic for processing conditional parameters

Appendix B. Examples

This appendix is informative. It provides some examples of the use of Conditional Parameters.

Note: For brevity, only the method or response code is shown in the header field.

B.1. Minimum Period (c.pmin) example

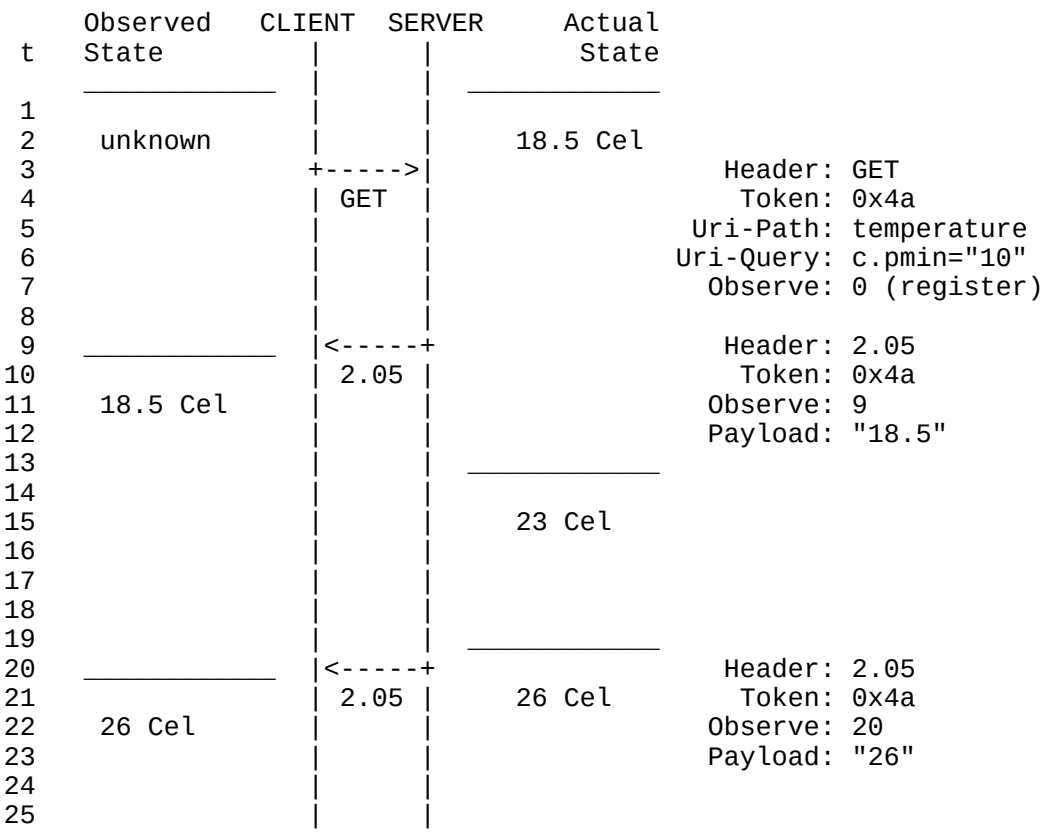

Figure 6: Client registers and receives one notification of the current state and one of a new state state when c.pmin time expires.

B.2. Maximum Period (c.pmax) example

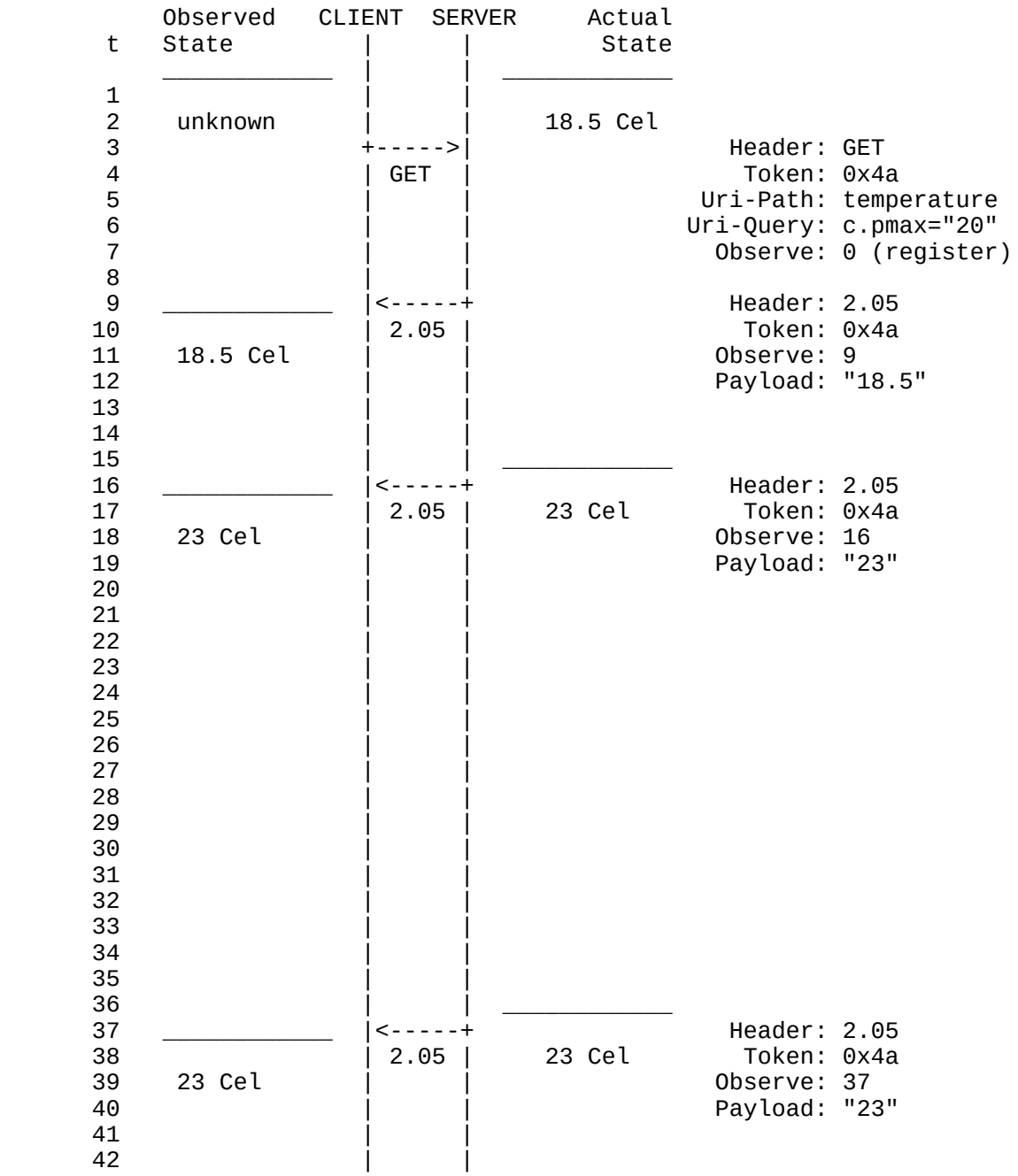
Observed State
CLIENT
SERVER
Actual State
t
unknown
18.5 Cel
GET
Header: GET
Token: 0x4a
Uri-Path: temperature
Uri-Query: c.pmax="20"
Observe: 0 (register)
2.05
Header: 2.05
Token: 0x4a
Observe: 9
Payload: "18.5"
18.5 Cel
2.05
Header: 2.05
Token: 0x4a
Observe: 16
Payload: "23"
23 Cel
23 Cel
2.05
Header: 2.05
Token: 0x4a
Observe: 37
Payload: "23"
23 Cel

Figure 7: Client registers and receives one notification of the current state, one of a new state and one of an unchanged state when c.pmax time expires.

B.3. Greater Than (c.gt) example

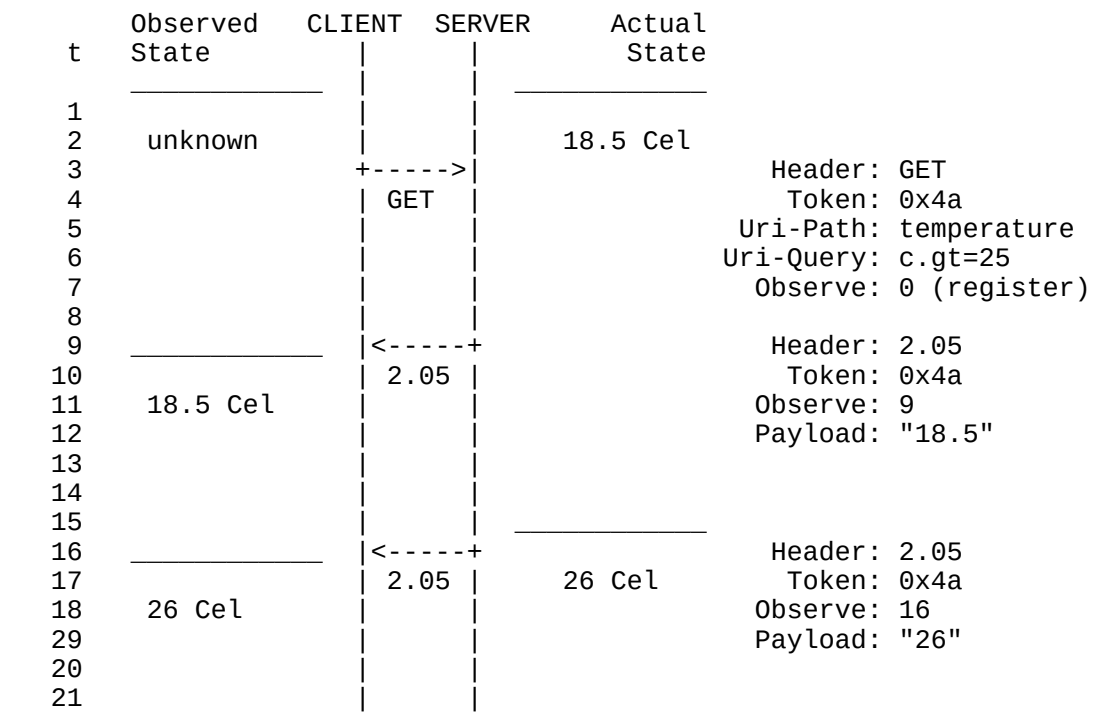

Figure 8: Client registers and receives one notification of the current state and one of a new state when it passes through the greater than threshold of 25.

B.4. Greater Than (c.gt) and Period Max (c.pmax) example

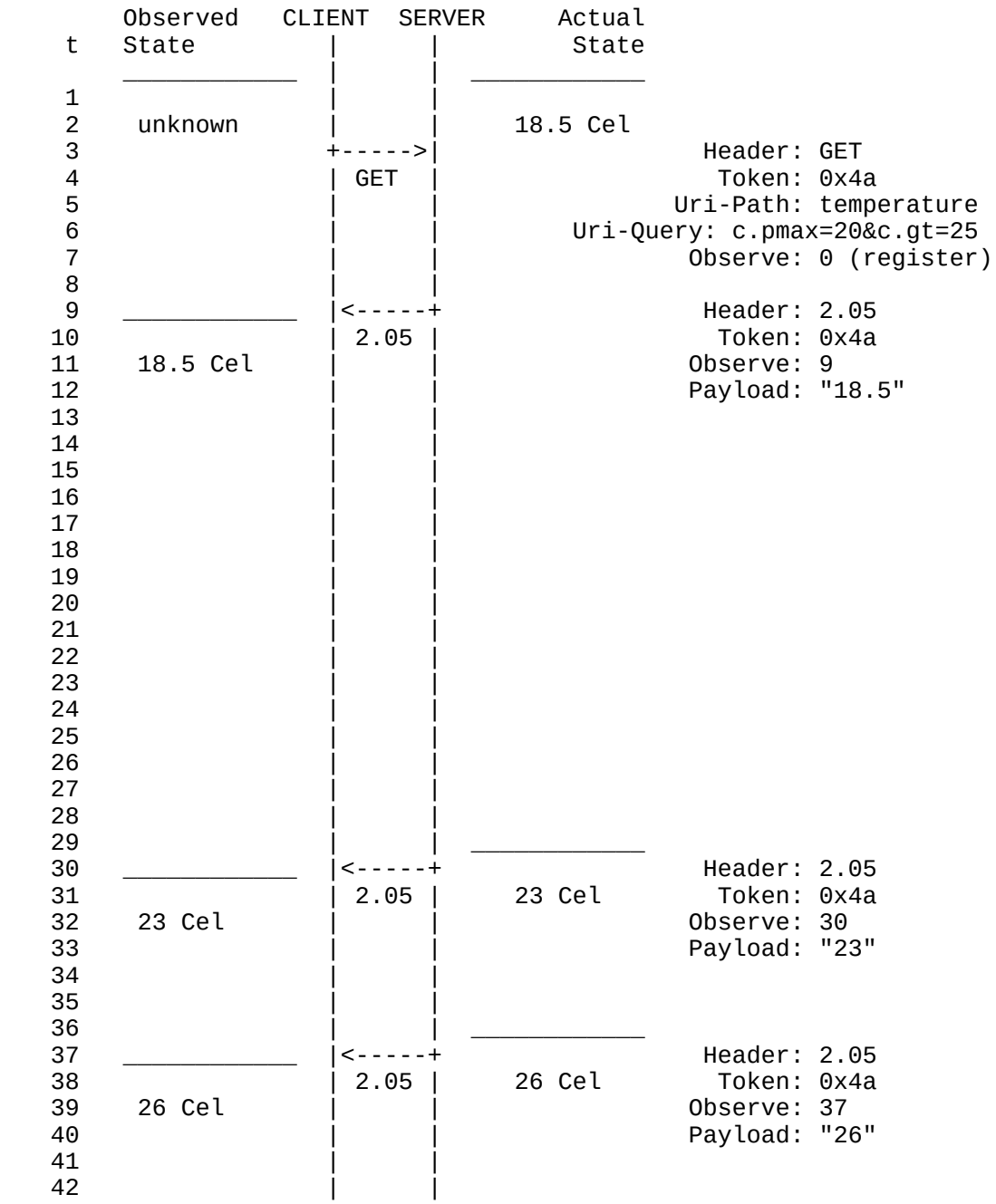
Observed State
CLIENT
SERVER
Actual State
t
unknown
18.5 Cel
GET
Header: GET
Token: 0x4a
Uri-Path: temperature
Uri-Query: c.pmax=20&c.gt=25
Observe: 0 (register)
2.05
Header: 2.05
Token: 0x4a
Observe: 9
Payload: "18.5"
18.5 Cel
23 Cel
Header: 2.05
Token: 0x4a
Observe: 30
Payload: "23"
23 Cel
26 Cel
Header: 2.05
Token: 0x4a
Observe: 37
Payload: "26"
26 Cel

Figure 9: Client registers and receives one notification of the current state, one when c.pmax time expires, and one of a new state when it passes through the greater than threshold of 25.

Acknowledgements

Hannes Tschofenig and Mert Ocak highlighted syntactical corrections in the usage of pmax and pmin in a query. David Navarro proposed allowing for pmax to be equal to pmin. Jaime Jimnez, Marco Tiloca and Ines Robles provided extensive reviews. Suggestions from Klaus Hartke aided greatly in clarifying how conditional parameters work with CoAP Observe. Security considerations were improved based on authors' observations in Section 2.2 of [I-D.irtf-t2trg-amplification-attacks].

Changelog

This section is to be removed before publishing as an RFC.

draft-ietf-core-conditional-attributes-12

* Added "xs:boolean" and "xs:decimal" types in Terminology
* Added avoidance of complex data structures in Section 3
* Extra text regarding resource state projection in Section 3.1
* Implementation Considerations now discusses SenML as well as if= interface descriptions.

draft-ietf-core-conditional-attributes-11

* Title of the document changed, and conditional attributes are now called conditional query parameters for accuracy.
* Clarifying decimal support for conditional control parameters.
* Text for error handling of type mismatches added
* Editorial fixes

draft-ietf-core-conditional-attributes-10

* Rectifying text and a table column in IANA Considerations, that version -09 erroneously omitted.

draft-ietf-core-conditional-attributes-09

* IANA Considerations section updated

* Editorial and formatting fixes

draft-ietf-core-conditional-attributes-08

* Various editorial fixes and corrections based on review comments on mailing list from Marco Tiloca.

draft-ietf-core-conditional-attributes-07

* Expanded how conditional attributes work with Observe in sections 3.1 to 3.4

* Addressed early review from IoT Directorate

* Security Considerations section expanded

draft-ietf-core-conditional-attributes-06

* Removed code block from Section 3.5

* Added an appendix containing pseudocode for server processing.

draft-ietf-core-conditional-attributes-05

* Multiple (mostly editorial) clarifications and updates based on review comments on mailing list from Marco Tiloca.

draft-ietf-core-conditional-attributes-04

* Reference code updated to include behaviour for edge attribute.

draft-ietf-core-conditional-attributes-03

* Attribute names updated to create uniqueness for use as conditional observe attributes.

draft-ietf-core-conditional-attributes-02

* Clarifications on usage and value of the band parameter

* Implementation considerations for proxies added

* Security considerations added

* IANA considerations added

draft-ietf-core-conditional-attributes-01

* Clarifications on True and False values for Edge and Con Attributes

* Alan Soloway added as author

draft-ietf-core-conditional-attributes-00

* Conditional Atttributes section from draft-ietf-core-dynlink-13 separated into own WG draft

Contributors

Christian Groves
Australia
Email: cngroves.std@gmail.com

Zach Shelby
ARM
FI- Vuokatti
Finland
Email: zach.shelby@arm.com

Matthieu Vial
Schneider-Electric
Grenoble
France
Email: matthieu.vial@schneider-electric.com

Jintao Zhu
Huawei
Xi’ an, Shaanxi Province
China
Email: jintao.zhu@huawei.com

Authors’ Addresses

Bilhanan Silverajan
Tampere University
Kalevantie 4
FI-33100 Tampere
Finland
Email: bilhanan.silverajan@tuni.fi

Michael Koster
Dogtiger Labs
524 H Street
Antioch, CA, 94509
United States of America
Email: michaeljohnkoster@gmail.com

Alan Soloway
Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, 92121
United States of America
Email: asoloway@qti.qualcomm.com