Workgroup: CoRE Working Group
Internet-Draft:
draft-ietf-core-conditional-attributes-06
Published: 14 January 2023
Intended Status: Informational
Expires: 18 July 2023
A   M. Koster   A. Soloway
uDogtiger Labs   Qualcomm Technologies, Inc.
t
h
o
r
s
:
B. Silverajan, Ed.
Tampere University

Conditional Attributes for Constrained RESTful Environments

Abstract

This specification defines Conditional Notification and Control Attributes that work with CoAP Observe (RFC7641).

Editor note

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

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 18 July 2023.

Copyright Notice

Copyright (c) 2023 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this

document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction
2. Terminology
3. Conditional Attributes
  3.1. Conditional Notification Attributes
    3.1.1. Greater Than (c.gt)
    3.1.2. Less Than (c.lt)
    3.1.3. Change Step (c.st)
    3.1.4. Notification Band (c.band)
    3.1.5. Edge (c.edge)
  3.2. Conditional Control Attributes
    3.2.1. Minimum Period (c.pmin)
    3.2.2. Maximum Period (c.pmax)
    3.2.3. Minimum Evaluation Period (c.epmin)
    3.2.4. Maximum Evaluation Period (c.epmax)
    3.2.5. Confirmable Notification (c.con)
  3.3. Server processing of Conditional Attributes
4. Implementation Considerations
5. Security Considerations
6. IANA Considerations
7. Acknowledgements
8. Contributors
9. Changelog
10. Normative References
Appendix A. Pseudocode: Processing Conditional Attributes
Appendix B. Examples
  B.1. Minimum Period (c.pmin) example
  B.2. Maximum Period (c.pmax) example
  B.3. Greater Than (c.gt) example
  B.4. Greater Than (c.gt) and Period Max (c.pmax) example
Authors' Addresses

1. Introduction

IETF Standards for machine-to-machine communication in constrained environments describe 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 Attributes 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.

3. Conditional Attributes

This specification defines conditional attributes for use with CoRE Observe [RFC7641]. Conditional attributes provide fine-grained control of notification and synchronization of resource states. When observing a resource, a CoAP client conveys conditional attributes as metadata using the query component of a CoAP URI. A conditional attribute can be represented as a "name=value" query parameter or simply a "name" without a value. Multiple conditional attributes in a query component are separated with an ampersand "&". A resource marked as Observable in its link description SHOULD support these conditional attributes.

Note: In this draft, we assume 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 attributes 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. Conditional Notification Attributes

Conditional Notification Attributes define the conditions that trigger a notification. Conditional Notification Attributes 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 Attributes 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 Attributes MAY be included in an Observe request.

Conditional Notification Attributes are defined below:

Attribute	Name	Value
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 Attributes

3.1.1. 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 can only be supported on resources with a scalar numeric value.

3.1.2. 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 can only be supported on resources with a scalar numeric value.

3.1.3. 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" attribute, 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 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" (or equivalent).

The Change Step parameter can only 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".

3.1.4. Notification Band (c.band)

The Notification Band attribute 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" attribute works as a modifier to the behaviour of "c.gt" and "c.lt". Its use is determined only by its presence, and not its 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 in which 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 in which 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 not between the "c.gt" and "c.lt" values, excluding equal to "c.gt" and "c.lt".

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

3.1.5. Edge (c.edge)

When present, the Edge attribute 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" attribute 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" attribute is 1 (True), the server notifies the client each time a resource state changes from False to True.

The "c.edge" attribute can only be supported on resources with a boolean value.

3.2. Conditional Control Attributes

Conditional Control Attributes define the time intervals between consecutive notifications as well as the cadence of the evaluation

of the conditions that trigger a notification. Conditional Control Attributes 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 Attributes MAY be included in an Observe request.

Conditional Control Attributes are defined below:

Attribute	Name	Value
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 Attributes

3.2.1. 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). 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" (or equivalent).

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.

3.2.2. 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). 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" (or equivalent).

3.2.3. 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. 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" (or equivalent).

3.2.4. 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. 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" (or equivalent).

3.2.5. Confirmable Notification (c.con)

When present with a value of 1 (True), Confirmable Notification indicates 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.

3.3. Server processing of Conditional Attributes

Conditional Notification Attributes and Conditional Control Attributes 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. 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

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.

This specification defines conditional attributes 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".

5. Security Considerations

The security considerations in Section 11 of [RFC7252] apply. Additionally, the security considerations in Section 7 of [RFC7641] also apply.

6. IANA Considerations

This specification requests a new Conditional Attributes registry to ensure attributes map uniquely to parameter names.

Note to IANA: Please replace "RFC XXXX" with the assigned RFC number in the table below.

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

Table 3

7. 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. Marco Tiloca provided an extensive review.

8. Contributors

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

Zach Shelby
ARM
Vuokatti
FINLAND
phone: +358 40 7796297
email: zach.shelby@arm.com

Matthieu Vial
Schneider-Electric
Grenoble
France
phone: +33 (0)47657 6522
eMail: matthieu.vial@schneider-electric.com

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

9. Changelog

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

10. 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/info/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/info/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/info/rfc7641>.

[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/info/rfc8174>.

Appendix A. Pseudocode: Processing Conditional Attributes

This appendix is informative. It describes the possible logic of how a server processes conditional attributes 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 attributes described in this document.

```
// 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 <=
    #define OUTOFBAND_TRUE ( lt < gt && (gt < curr_state || curr_state <

    #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_EXIS
             return true;
         }
        }
    }

    return false;

}
```

Figure 1: Pseudocode showing the logic for processing conditional attributes

Appendix B. Examples

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

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

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

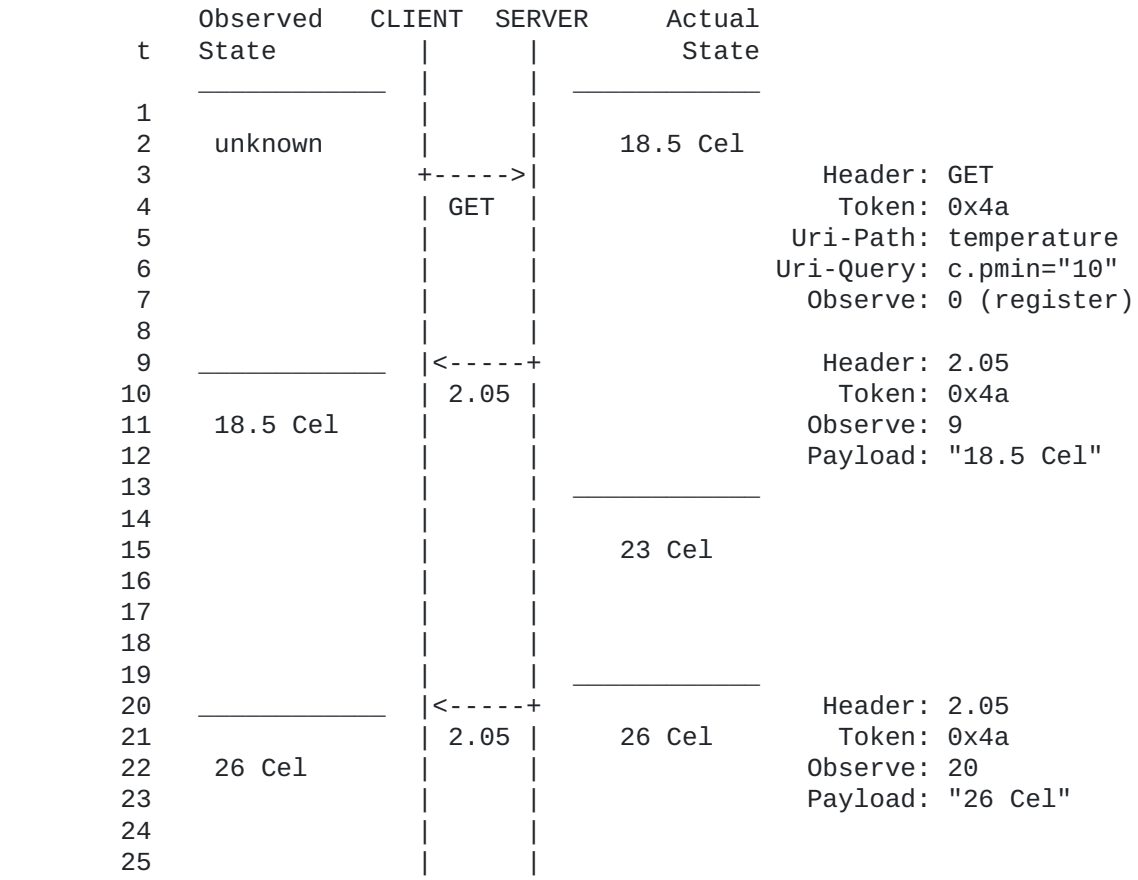

Figure 2: Client registers and receives one notification of the current state and one of a new state state when c.pmin time expires.

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

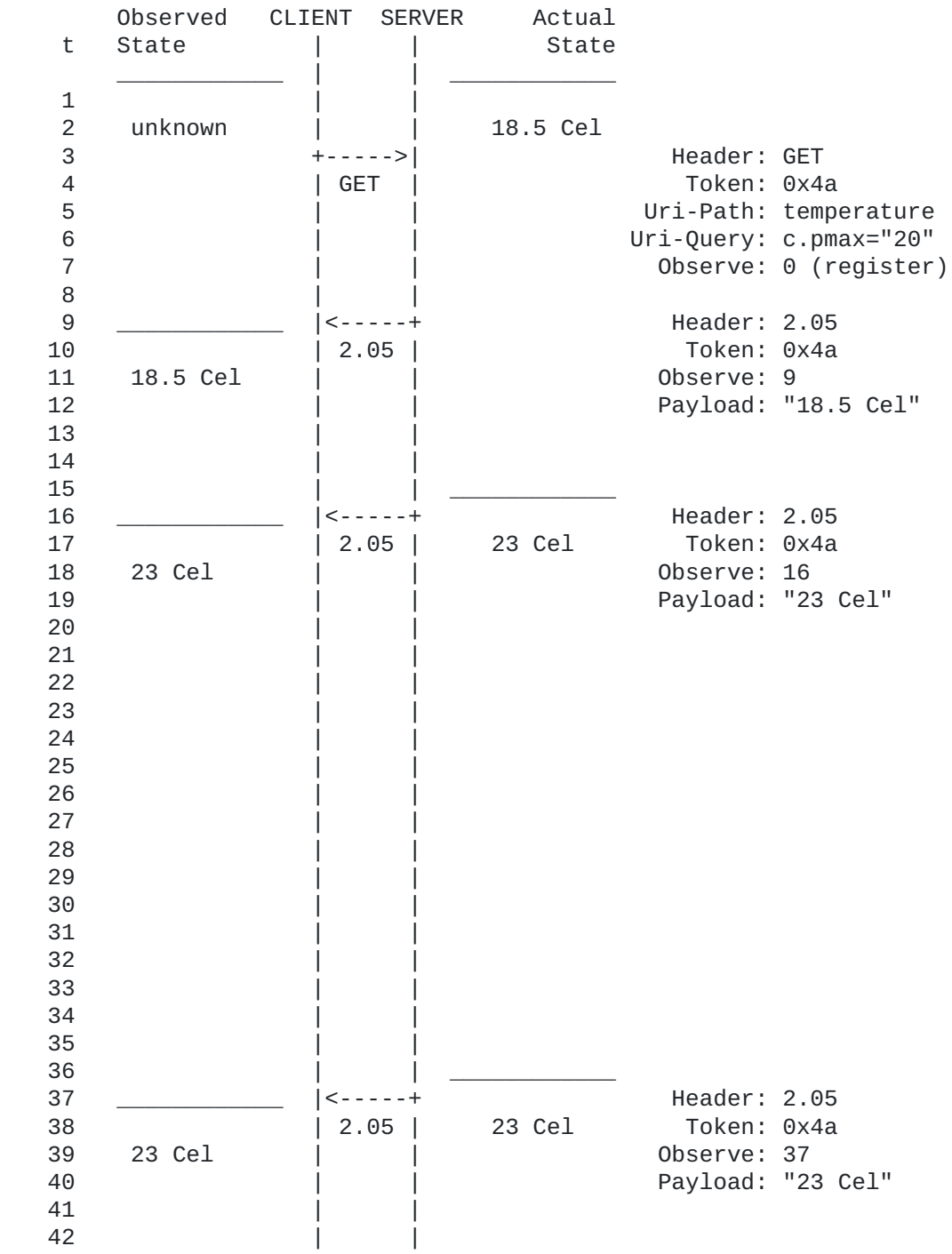

Figure 3: Client registers and receives one notification of the current state, one of a new state and one of an unchanged state when c.pmax time expires.

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

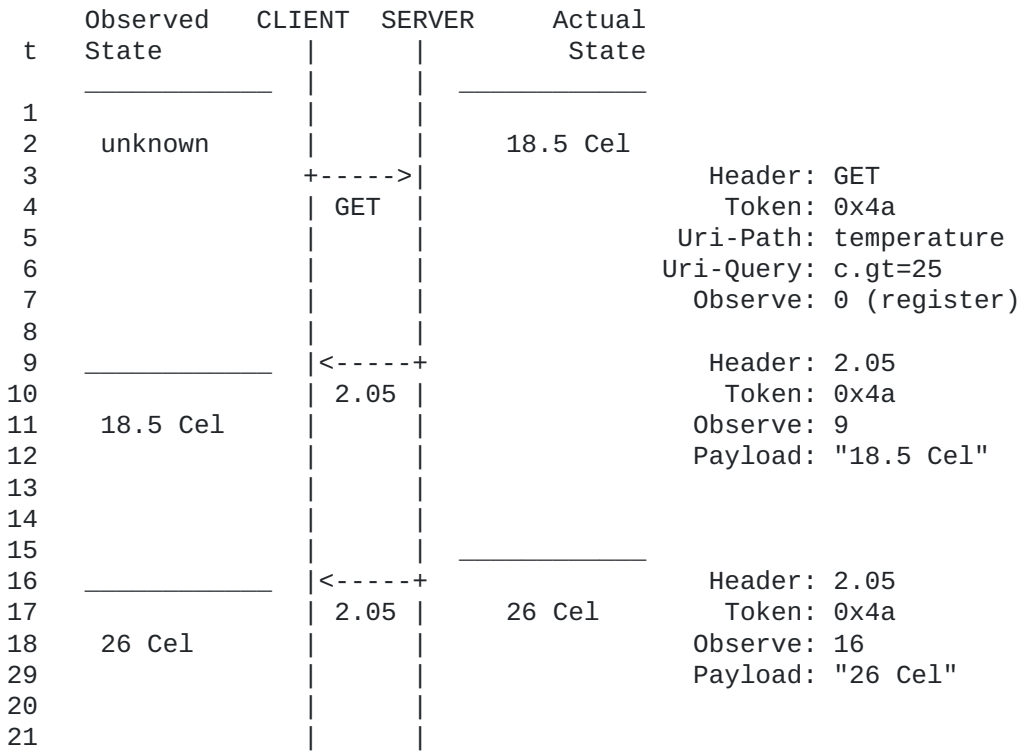

Figure 4: Client registers and receives one notification of the current state and one of a new state when it passes through the greater than threshold of 25.

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

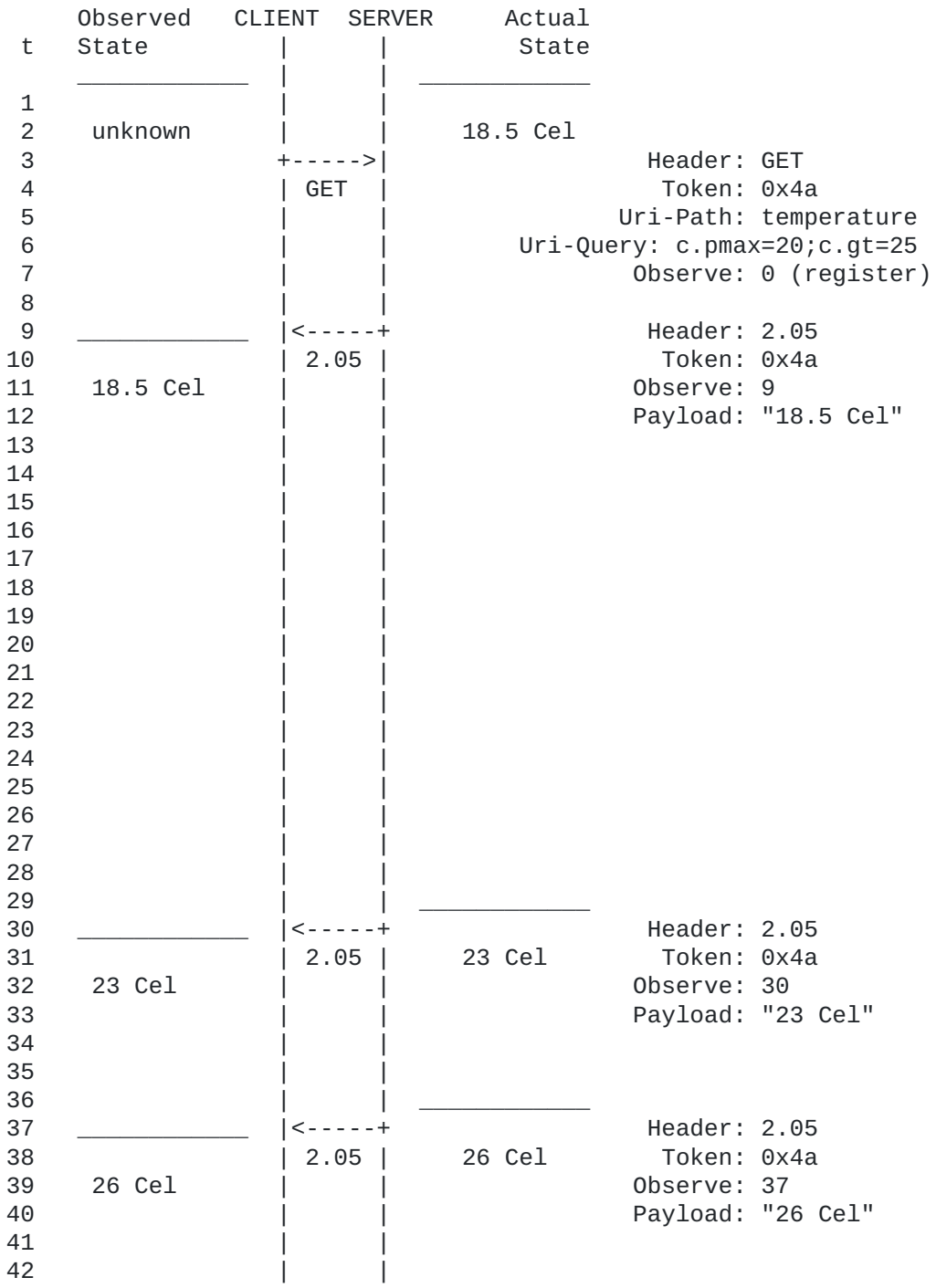

Figure 5: Client registers and receives one notification of the current state, one when c.pmax time expires and one of a new state when it passes through the greater than threshold of 25.

Authors' Addresses

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

Bilhanan Silverajan (editor)
Tampere University
Kalevantie 4
FI-33100 Tampere
Finland

Email: bilhanan.silverajan@tuni.fi