Workgroup: ACE Working Group
Internet-Draft:
draft-ietf-ace-revoked-token-notification-03
Published: 24 October 2022
Intended Status: Standards Track
Expires: 27 April 2023
Authors: M. Tiloca RISE AB
L. Seitz Combitech
F. Palombini Ericsson AB
S. Echeverria CMU SEI
G. Lewis CMU SEI

# Notification of Revoked Access Tokens in the Authentication and Authorization for Constrained Environments (ACE) Framework

## Abstract

This document specifies a method of the Authentication and Authorization for Constrained Environments (ACE) framework, which allows an Authorization Server to notify Clients and Resource Servers (i.e., registered devices) about revoked Access Tokens. The method allows Clients and Resource Servers to access a Token Revocation List on the Authorization Server, with the possible additional use of resource observation for the Constrained Application Protocol (CoAP). Resulting (unsolicited) notifications of revoked Access Tokens complement alternative approaches such as token introspection, while not requiring additional endpoints on Clients and Resource Servers.

## Discussion Venues

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

Discussion of this document takes place on the Authentication and Authorization for Constrained Environments Working Group mailing list (ace@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/ace/.

Source for this draft and an issue tracker can be found at https://github.com/ace-wg/ace-revoked-token-notification.

## 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 27 April 2023.

## Copyright Notice



## Table of Contents

## 1. Introduction

Authentication and Authorization for Constrained Environments (ACE) [RFC9200] is a framework that enforces access control on IoT devices acting as Resource Servers. In order to use ACE, both Clients and Resource Servers have to register with an Authorization Server and become a registered device. Once registered, a Client can send a request to the Authorization Server, to obtain an Access Token for a Resource Server. For a Client to access the Resource Server, the Client must present the issued Access Token at the Resource Server, which then validates it before storing it (see Section 5.10.1.1 of [RFC9200]).

Even though Access Tokens have expiration times, there are circumstances by which an Access Token may need to be revoked before its expiration time, such as: (1) a registered device has been compromised, or is suspected of being compromised; (2) a registered device is decommissioned; (3) there has been a change in the ACE profile for a registered device; (4) there has been a change in access policies for a registered device; and (5) there has been a change in the outcome of policy evaluation for a registered device (e.g., if policy assessment depends on dynamic conditions in the execution environment, the user context, or the resource utilization).

As discussed in Section 6.1 of [RFC9200], only client-initiated revocation is currently specified [RFC7009] for OAuth 2.0 [RFC6749], based on the assumption that Access Tokens in OAuth are issued with a relatively short lifetime. However, this is not expected to be the

case for constrained, intermittently connected devices, that need Access Tokens with relatively long lifetimes.

This document specifies a method for allowing registered devices to access and possibly subscribe to a Token Revocation List (TRL) resource on the Authorization Server, in order to obtain an updated list of revoked, but yet not expired, pertaining Access Tokens. In particular, registered devices can subscribe to the TRL at the Authorization Server by using resource observation [RFC7641] for the Constrained Application Protocol (CoAP) [RFC7252].

Unlike in the case of token introspection (see Section 5.9 of [RFC9200]), a registered device does not provide an owned Access Token to the Authorization Server for inquiring about its current state. Instead, registered devices simply obtain an updated list of revoked, but yet not expired, pertaining Access Tokens, as efficiently identified by corresponding hash values.

The benefits of this method are that it complements token introspection, and it does not require any additional endpoints on the registered devices. The only additional requirements for registered devices are a request/response interaction with the Authorization Server to access and possibly subscribe to the TRL (see Section 2), and the lightweight computation of hash values to use as Token identifiers (see Section 3).

### 1.1. 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.

Readers are expected to be familiar with the terms and concepts described in the ACE framework for Authentication and Authorization [RFC9200], as well as with terms and concepts related to CBOR Web Tokens (CWTs) [RFC8392], and JSON Web Tokens (JWTs) [RFC7519]. The terminology for entities in the considered architecture is defined in OAuth 2.0 [RFC6749]. In particular, this includes Client, Resource Server, and Authorization Server.

Readers are also expected to be familiar with the terms and concepts related to CBOR [RFC8949], JSON [RFC8259], the CoAP protocol [RFC7252], CoAP Observe [RFC7641], and the use of hash functions to name objects as defined in [RFC6920].

Note that, unless otherwise indicated, the term "endpoint" is used here following its OAuth definition, aimed at denoting resources such as /token and /introspect at the Authorization Server, and /

authz-info at the Resource Server. This document does not use the CoAP definition of "endpoint", which is "An entity participating in the CoAP protocol."

This specification also refers to the following terminology.

- Token hash: identifier of an Access Token, in binary format encoding. The token hash has no relation to other possibly used token identifiers, such as the "cti" (CWT ID) claim of CBOR Web Tokens (CWTs) [RFC8392].

- Token Revocation List (TRL): a collection of token hashes such that the corresponding Access Tokens have been revoked but are not expired yet.

- TRL resource: a resource on the Authorization Server, with a TRL as its representation.

- TRL endpoint: an endpoint at the Authorization Server associated with the TRL resource. The default name of the TRL endpoint in a url-path is '/revoke/trl'. Implementations are not required to use this name, and can define their own instead.

- Registered device: a device registered at the Authorization Server, i.e., as a Client, or a Resource Server, or both. A registered device acts as a caller of the TRL endpoint.

- Administrator: entity authorized to get full access to the TRL at the Authorization Server, and acting as a caller of the TRL endpoint. An administrator is not necessarily a registered device as defined above, i.e., a Client requesting Access Tokens or a Resource Server consuming Access Tokens. How the administrator authorization is established and verified is out of the scope of this specification.

- Pertaining Access Token:

  - With reference to an administrator, an Access Token issued by the Authorization Server.

  - With reference to a registered device, an Access Token intended to be owned by that device. An Access Token pertains to a Client if the Authorization Server has issued the Access Token and provided it to that Client. An Access Token pertains to a Resource Server if the Authorization Server has issued the Access Token to be consumed by that Resource Server.

Examples throughout this document are expressed in CBOR diagnostic notation without the tag and value abbreviations.

## 2. Protocol Overview

This protocol defines how a CoAP-based Authorization Server informs Clients and Resource Servers, i.e., registered devices, about pertaining revoked Access Tokens. How the relationship between a registered device and the Authorization Server is established is out of the scope of this specification.

At a high level, the steps of this protocol are as follows.

* Upon startup, the Authorization Server creates a single TRL resource. At any point in time, the TRL resource represents the list of all revoked Access Tokens issued by the Authorization Server that are not expired yet.

* When a device registers at the Authorization Server, it also receives the url-path to the TRL resource.

  After the registration procedure is finished, the registered device can send an Observation Request to the TRL resource as described in [RFC7641], i.e., a GET request including the CoAP Observe Option set to 0 (register). By doing so, the registered device effectively subscribes to the TRL resource, as interested to receive notifications about its update. Upon receiving the request, the Authorization Server adds the registered device to the list of observers of the TRL resource.

  At any time, the registered device can send a GET request to the TRL endpoint. When doing so, it can request for: the current list of pertaining revoked Access Tokens (see Section 6); or the most recent TRL updates occurred over the list of pertaining revoked Access Tokens (see Section 7). In either case, the registered device may also rely on an Observation Request for subscribing to the TRL resource as discussed above.

* When an Access Token is revoked, the Authorization Server adds the corresponding token hash to the TRL. Also, when a revoked Access Token eventually expires, the Authorization Server removes the corresponding token hash from the TRL.

  In either case, after updating the TRL, the Authorization Server sends Observe Notifications as per [RFC7641]. That is, an Observe Notification is sent to each registered device subscribed to the TRL resource and to which the Access Token pertains.

  Depending on the specific subscription established through the observation request, the notification provides the current updated list of revoked Access Tokens in the portion of the TRL pertaining to that device (see Section 6), or rather the most

recent TRL updates occurred over that list of pertaining revoked Access Tokens (see Section 7).

Further Observe Notifications may be sent, consistently with ongoing additional observations of the TRL resource.

*An administrator can access and subscribe to the TRL like a registered device, while getting the full updated representation of the TRL.

Figure 1 shows a high-level overview of the service provided by this protocol. In particular, it shows the Observe Notifications sent by the Authorization Server to one administrator and four registered devices, upon revocation of the issued Access Tokens t1, t2 and t3, with token hash th1, th2 and th3, respectively. Each dotted line associated with a pair of registered devices indicates the Access Token that they both own.

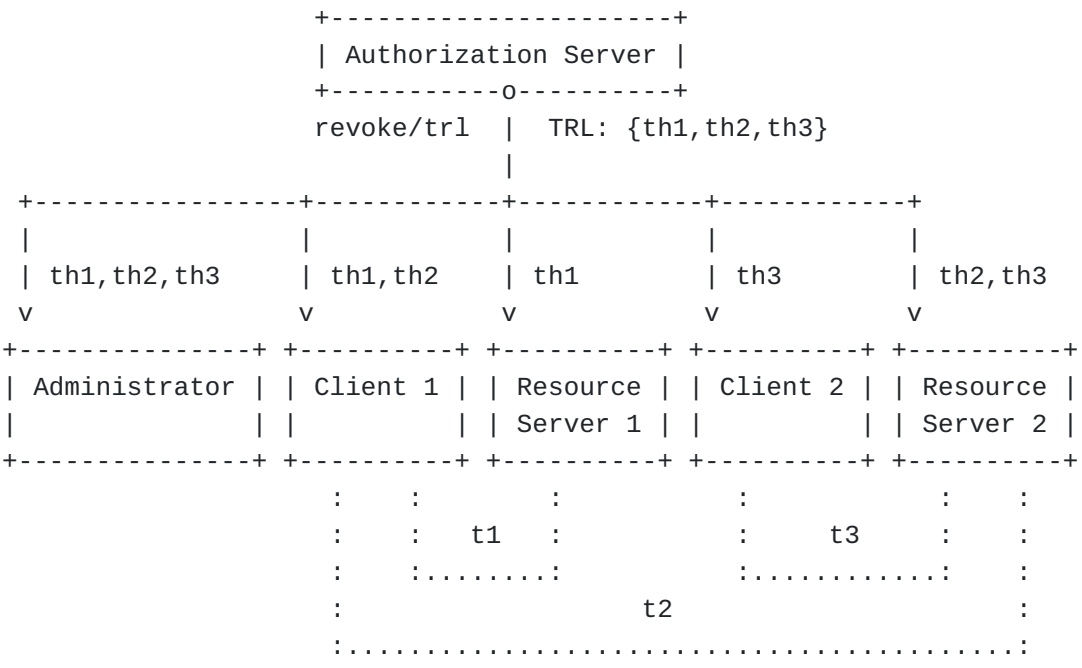



Figure 1: Protocol Overview

Section 11 provides examples of the protocol flow and message exchange between the Authorization Server and a registered device.

## 3. Token Hash

The token hash of an Access Token is computed as follows.

1. The Authorization Server defines ENCODED_TOKEN, as the content of the 'access_token' parameter in the Authorization Server

response (see Section 5.8.2 of [RFC9200]), where the Access Token was included and provided to the requesting Client.

Note that the content of the 'access_token' parameter is either:

- A CBOR byte string, if the Access Token was transported using CBOR. With reference to the example in Figure 2, and assuming the string's length in bytes to be 119 (i.e., 0x77 in hexadecimal), then ENCODED_TOKEN takes the bytes {0x58 0x77 0xd0 0x83 0x44 0xa1 ...}, i.e., the raw content of the parameter 'access_token'.

- A text string, if the Access Token was transported using JSON. With reference to the example in Figure 3, ENCODED_TOKEN takes "2YotnFZFEjr1zCsicMWpAA", i.e., the raw content of the parameter 'access_token'.

2. The Authorization Server defines HASH_INPUT as follows.

   - If CBOR was used to transport the Access Token (as a CWT or JWT), HASH_INPUT takes the same value of ENCODED_TOKEN.

   - If JSON was used to transport the Access Token (as a CWT or JWT), HASH_INPUT takes the serialization of ENCODED_TOKEN.

     In either case, HASH_INPUT results in the binary representation of the content of the 'access_token' parameter from the Authorization Server response.

3. The Authorization Server generates a hash value of HASH_INPUT as per Section 6 of [RFC6920]. The resulting output in binary format is used as the token hash. Note that the used binary format embeds the identifier of the used hash function, in the first byte of the computed token hash.

   The specifically used hash function MUST be collision-resistant on byte-strings, and MUST be selected from the "Named Information Hash Algorithm" Registry [Named.Information.Hash.Algorithm].

   The Authorization Server specifies the used hash function to registered devices during their registration procedure (see Section 9).

```
2.01 Created
Content-Format: application/ace+cbor
Max-Age: 85800
Payload:
{
   "access_token" : h'd08344a1 ...
    (remainder of the Access Token omitted for brevity) ...',
   "token_type" : pop,
   "expires_in" : 86400,
   "profile" : coap_dtls,
   (remainder of the response omitted for brevity)
}
```

Figure 2: Example of Authorization Server response using CBOR

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache
Payload:
{
   "access_token" : "2YotnFZFEjr1zCsicMWpAA ...
    (remainder of the Access Token omitted for brevity) ...",
   "token_type" : "pop",
   "expires_in" : 86400,
   "profile" : "coap_dtls",
   (remainder of the response omitted for brevity)
}
```

Figure 3: Example of Authorization Server response using JSON

## 4. The TRL Resource

Upon startup, the Authorization Server creates a single TRL resource, encoded as a CBOR array.

Each element of the array is a CBOR byte string, with value the token hash of an Access Token. The order of the token hashes in the CBOR array is irrelevant, and the CBOR array MUST be treated as a set in which the order of elements has no significant meaning.

The TRL is initialized as empty, i.e., the initial content of the TRL resource representation MUST be an empty CBOR array.

### 4.1. Update of the TRL Resource

The Authorization Server updates the TRL in the following two cases.

* When a non-expired Access Token is revoked, the token hash of the Access Token is added to the TRL resource representation. That

is, a CBOR byte string with the token hash as its value is added to the CBOR array used as TRL resource representation.

* When a revoked Access Token expires, the token hash of the Access Token is removed from the TRL resource representation. That is, the CBOR byte string with the token hash as its value is removed from the CBOR array used as TRL resource representation.

## 5. The TRL Endpoint

Consistent with Section 6.5 of [RFC9200], all communications between a caller of the TRL endpoint and the Authorization Server MUST be encrypted, as well as integrity and replay protected. Furthermore, responses from the Authorization Server to the caller MUST be bound to the caller's request.

Following a request to the TRL endpoint, the messages defined in this document that the Authorization Server sends as response use Content-Format "application/ace-trl+cbor". Their payload is formatted as a CBOR map, and the CBOR values for the parameters included therein are defined in Section 12.

The Authorization Server MUST implement measures to prevent access to the TRL endpoint by entities other than registered devices and authorized administrators.

The TRL endpoint supports only the GET method, and allows two types of query of the TRL.

* Full query: the Authorization Server returns the token hashes of the revoked Access Tokens currently in the TRL and pertaining to the requester.

  The Authorization Server MUST support this type of query. The processing of a full query and the related response format are defined in Section 6.

* Diff query: the Authorization Server returns a list of diff entries. Each diff entry is related to one of the most recent updates, in the portion of the TRL pertaining to the requester.

  The entry associated with one of such updates contains a list of token hashes, such that: i) the corresponding revoked Access Tokens pertain to the requester; and ii) they were added to or removed from the TRL at that update.

  The Authorization Server MAY support this type of query. In such a case, the Authorization Server maintains the history of updates to the TRL resource as defined in Section 5.1. The processing of

a diff query and the related response format are defined in Section 7.

If it supports diff queries, the Authorization Server MAY additionally support its "Cursor" extension, which has two benefits. First, the Authorization Server can avoid excessively big latencies when several diff entries have to be transferred, by delivering one adjacent subset at the time, in different diff query responses. Second, a requester can retrieve diff entries associated with TRL updates that, even if not the most recent ones, occurred after a TRL update indicated as reference point.

If it supports the "Cursor" extension, the Authorization Server stores additional information when maintaining the history of updates to the TRL resource, as defined in Section 5.1.1. Also, the processing of full query requests and diff query requests, as well as the related response format, are further extended as defined in Section 8.

## 5.1. Supporting Diff Queries

If the Authorization Server supports diff queries, it is able to transfer a list of diff entries, as a series of TRL updates. That is, when replying to a diff query performed by a requester, the Authorization Server specifies the most recent updates to the portion of the TRL pertaining to that requester.

The following defines how the Authorization Server builds and maintains consistent histories of TRL updates for each registered device and administrator, hereafter referred to as requesters.

For each requester, the Authorization Server maintains an update collection of maximum N_MAX series items, where N_MAX is a pre-defined, constant positive integer. The Authorization Server MUST keep track of the N_MAX most recent updates to the portion of the TRL that pertains to each requester. The Authorization Server SHOULD provide requesters with the value of N_MAX, upon their registration (see Section 9).

The series items in the update collection MUST be strictly ordered in a chronological fashion. That is, at any point in time, the current first series item is the one least recently added to the update collection and still retained by the Authorization Server, while the current last series item is the one most recently added to the update collection. The particular method used to achieve this is implementation-specific.

Each time the TRL changes, the Authorization Server performs the following operations for each requester.

1. The Authorization Server considers the portion of the TRL pertaining to that requester. If the TRL portion is not affected by this TRL update, the Authorization Server stops the processing for that requester.

2. Otherwise, the Authorization Server creates two sets "trl_patch" of token hashes, i.e., one "removed" set and one "added" set, as related to this TRL update.

3. The Authorization Server fills the two sets with the token hashes of the removed and added Access Tokens, respectively, from/to the TRL portion considered at step 1.

4. The Authorization Server creates a new series item, which includes the two sets from step 3.

5. If the update collection associated with the requester currently includes N_MAX series items, the Authorization Server MUST delete the oldest series item in the update collection.

   This occurs when the number of TRL updates pertaining to the requester and currently stored at the Authorization Server is equal to N_MAX.

6. The Authorization Server adds the series item to the update collection associated with the requester, as the most recent one.

#### 5.1.1. Supporting the "Cursor" Extension

If it supports the "Cursor" extension for diff queries, the Authorization Server performs also the following actions.

The Authorization Server defines the constant, unsigned integer MAX_INDEX <= ((2 ** 64) - 1), where "**" is the exponentiation operator. In particular, the value of MAX_INDEX is REQUIRED to be at least (N_MAX - 1), and is RECOMMENDED to be at least ((2 ** 32) - 1). Note that MAX_INDEX is practically expected to be order of magnitudes greater than N_MAX.

When maintaining the history of updates to the TRL resource, the following applies separately for each update collection.

*Each series item X in the update collection is also associated with an unsigned integer 'index', whose minimum value is 0 and whose maximum value is MAX_INDEX. The first series item ever added to the update collection MUST have 'index' with value 0.

If i_X is the value of 'index' associated with a series item X, then the following series item Y will take 'index' with value i_Y = (i_X + 1) % (MAX_INDEX + 1). That is, after having added a series item whose associated 'index' has value MAX_INDEX, the next added series item will result in a wrap-around of the 'index' value, and will thus take 'index' with value 0.

For example, assuming N_MAX = 3, the values of 'index' in the update collection chronologically evolve as follows, as new series items are added and old series items are deleted.

-...

-( i_A = MAX_INDEX - 2, i_B = MAX_INDEX - 1, i_C = MAX_INDEX )

-( i_B = MAX_INDEX - 1, i_C = MAX_INDEX, i_D = 0 )

-( i_C = MAX_INDEX, i_D = 0, i_E = 1 )

-( i_D = 0, i_E = 1, i_F = 2 )

-...

*The unsigned integer 'last_index' is also defined, with minimum value 0 and maximum value MAX_INDEX.

If the update collection is empty (i.e., no series items have been added yet), the value of 'last_index' is not defined. If the update collection is not empty, 'last_index' has the value of 'index' currently associated with the latest added series item in the update collection.

That is, after having added V series items to the update collection, the last and most recently added series item has 'index' with value 'last_index' = (V - 1) % (MAX_INDEX + 1).

As long as a wrap-around of the 'index' value has not occurred, the value of 'last_index' is the absolute counter of series items added to that update collection until and including V, minus 1.

When processing a diff query using the "Cursor" extension, the values of 'index' are used as cursor information, as defined in Section 8.2.

For each update collection, the Authorization Server also defines a constant, positive integer MAX_DIFF_BATCH <= N_MAX, whose value specifies the maximum number of diff entries to be included in a single diff query response. The specific value depends on the specific registered device or administrator associated with the update collection in question. If supporting the "Cursor" extension,

the Authorization Server SHOULD provide registered devices and administrators with the value of MAX_DIFF_BATCH, upon their registration (see Section 9).

### 5.2. Query Parameters

The TRL endpoint allows the following query parameters to be present in a GET request. The Authorization Server MUST silently ignore unknown query parameters.

* 'pmax': if included, it follows the semantics defined in Section 3.2.2 of [I-D.ietf-core-conditional-attributes]. This query parameter is relevant only in case the GET request is specifically an Observation Request, i.e., if it includes the CoAP Observe Option set to 0 (register). In such a case, this parameter indicates the maximum time, in seconds, between two consecutive notifications for the observation in question, regardless whether the TRL resource has changed or not.

  If the Observation Request does not include the 'pmax' parameter, the maximum time to consider is up to the Authorization Server. If the Observation Request includes the 'pmax' parameter, its value MUST be greater than zero, otherwise the Authorization Server MUST return a 4.00 (Bad Request) response.

  If the GET request is not an Observation Request, the Authorization Server MUST ignore the 'pmax' parameter, in case this is included.

* 'diff': if included, it indicates to perform a diff query of the TRL (see Section 7). Its value MUST be either:

  - the integer 0, indicating that a (notification) response should include as many diff entries as the Authorization Server can provide in the response; or

  - a positive integer strictly greater than 0, indicating the maximum number of diff entries that a (notification) response should include.

  If the Authorization Server does not support diff queries, it ignores the query parameter 'diff' when present in the GET request and proceeds like when processing a full query of the TRL (see Section 6).

  Otherwise, the Authorization Server MUST return a 4.00 (Bad Request) response in case the query parameter 'diff' of the GET request specifies a value other than 0 or than a positive integer. The response MUST have Content-Format "application/ace-trl+cbor". The payload of the response is a CBOR map, which MUST

include the 'error' field with value 0 ("Invalid parameter value") and MAY include the 'error_description' field to provide additional context.

* 'cursor': if included, it indicates to perform a diff query of the TRL together with the "Cursor" extension, as defined in Section 8.2. Its value MUST be either 0 or a positive integer.

  If included, the query parameter 'cursor' specifies an unsigned integer value that was provided by the Authorization Server in a previous response from the TRL endpoint (see Section 8.1, Section 8.2.2 and Section 8.2.3).

  If the Authorization Server does not support the "Cursor" extension, it ignores the query parameter 'cursor' when present in the GET request. In such a case, the Authorization Server proceeds: i) like when processing a diff query of the TRL (see Section 7), if it supports diff queries and the query parameter 'diff' is present in the GET request, or ii) like when processing a full query of the TRL (see Section 6) otherwise.

  If the Authorization Server supports both diff queries and the "Cursor" extension, and the GET request specifies the query parameter 'cursor', then the Authorization Server MUST return a 4.00 (Bad Request) response in case any of the following conditions holds.

  - The GET request does not specify the query parameter 'diff'.

    The 'error' parameter within the CBOR map carried in the response payload MUST have value 1 ("Invalid set of parameters").

  - The query parameter 'cursor' has a value other than 0 or than a positive integer.

    The 'error' parameter within the CBOR map carried in the response payload MUST have value 0 ("Invalid parameter value").

  - The query parameter 'cursor' has a value strictly greater than MAX_INDEX (see Section 5.1.1).

    The 'error' parameter within the CBOR map carried in the response payload MUST have value 0 ("Invalid parameter value"). The CBOR map MUST also include the 'cursor' parameter, which MUST specify either: the CBOR simple value "null" (0xf6), if the update collection associated with the requester is empty; or the corresponding current value of 'last_index' otherwise.

- All of the following hold: the update collection associated with the requester is not empty; no wrap-around of its 'index' value has occurred; and the query parameter 'cursor' has a value strictly greater than the current 'last_index' on the update collection (see Section 5.1.1).

  The 'error' parameter within the CBOR map carried in the response payload MUST have value 2 ("Out of bound cursor value"). The CBOR map MUST also include the 'cursor' parameter, which MUST specify the current value of 'last_index' for the update collection associated with the requester.

The 4.00 (Bad Request) response MUST have Content-Format "application/ace-trl+cbor". The payload of the response MUST be a CBOR map, which MUST include the 'error' parameter and MAY include the 'error_description' parameter to provide additional context.

## 6. Full Query of the TRL

In order to produce a (notification) response to a GET request asking for a full query of the TRL, the Authorization Server performs the following actions.

1. From the current TRL resource representation, the Authorization Server builds a set HASHES, such that:

   * If the requester is a registered device, HASHES specifies the token hashes of the Access Tokens pertaining to that registered device. The Authorization Server can use the authenticated identity of the registered device to perform the necessary filtering on the TRL resource representation.

   * If the requester is an administrator, HASHES specifies all the token hashes in the current TRL resource representation.

2. The Authorization Server sends a 2.05 (Content) response to the requester. The response MUST have Content-Format "application/ace-trl+cbor". The payload of the response is a CBOR map, which MUST be formatted as follows.

   * The 'full_set' parameter MUST be included and specifies a CBOR array 'full_set_value'. Each element of 'full_set_value' specifies one of the token hashes from the set HASHES, encoded as a CBOR byte string. If the set HASHES is empty, the 'full_set' parameter specifies the empty CBOR array.

The order of the token hashes in the CBOR array is irrelevant, i.e., the CBOR array MUST be treated as a set in which the order of elements has no significant meaning.

*The 'cursor' parameter MUST be included if the Authorization Server supports both the diff queries and the related "Cursor" extension (see Section 5.1 and Section 5.1.1). Its value is specified according to what is defined in Section 8.1, and provides the requester with information for performing a follow-up diff query using the "Cursor" extension (see Section 8.2).

If the Authorization Server does not support both diff queries and the "Cursor" extension, this parameter MUST NOT be included. In case the requester does not support both diff queries and the "Cursor" extension, it MUST silently ignore the 'cursor' parameter if present.

Figure 4 provides the CDDL definition [RFC8610] of the CBOR array 'full_set_value' specified in the response from the Authorization Server, as value of the 'full_set' parameter.

```
token_hash = bytes
full_set_value = [* token_hash]
```

Figure 4: CDDL definition of 'full_set_value'

Figure 5 shows an example of response from the Authorization Server, following a full query request to the TRL endpoint. In this example, the Authorization Server does not support the "Cursor" extension (if it supports diff queries at all), hence the 'cursor' parameter is not included in the payload of the response. Also, full token hashes are omitted for brevity.

```
2.05 Content
Content-Format: application/ace-trl+cbor
Payload:
{
   "full_set" : [
     h'01fa51cc ... ', h'01748190 ... '
   ]
}
```

Figure 5: Example of response following a Full Query request to the TRL endpoint

## 7. Diff Query of the TRL

In order to produce a (notification) response to a GET request asking for a diff query of the TRL, the Authorization Server performs the following actions.

1. The Authorization Server defines the positive integer NUM as follows. If the value N specified in the query parameter 'diff' in the GET request is equal to 0 or greater than the pre-defined positive integer N_MAX (see Section 5.1), then NUM takes the value of N_MAX. Otherwise, NUM takes N.

2. The Authorization Server determines U = min(NUM, SIZE), where SIZE <= N_MAX is the number of TRL updates pertaining to the requester and currently stored at the Authorization Server.

3. The Authorization Server prepares U diff entries. If U is equal to 0 (e.g., because SIZE is equal to 0 at step 2), then no diff entries are prepared.

   The prepared diff entries are related to the U most recent TRL updates pertaining to the requester, as maintained in the update collection for that requester (see Section 5.1). In particular, the first diff entry refers to the most recent of such updates, the second diff entry refers to the second from last of such updates, and so on.

   Each diff entry is a CBOR array 'diff_entry', which includes the following two elements.

   * The first element is a CBOR array 'removed'. Each element of the array is a CBOR byte string, with value the token hash of an Access Token such that: it pertained to the requester; and it was removed from the TRL during the update associated with the diff entry.

   * The second element is a CBOR array 'added'. Each element of the array is a CBOR byte string, with value the token hash of an Access Token such that: it pertains to the requester; and it was added to the TRL during the update associated with the diff entry.

   The order of the token hashes in the CBOR arrays 'removed' and 'added' is irrelevant. That is, the CBOR arrays 'removed' and 'added' MUST be treated as a set in which the order of elements has no significant meaning.

4. The Authorization Server prepares a 2.05 (Content) response for the requester. The response MUST have Content-Format

"application/ace-trl+cbor". The payload of the response is a CBOR map, which MUST be formatted as follows.

* The 'diff_set' parameter MUST be present and specifies a CBOR array 'diff_set_value' of U elements. Each element of 'diff_set_value' specifies one of the CBOR arrays 'diff_entry' prepared above as diff entry. Note that U might have value 0, in which case 'diff_set_value' is the empty CBOR array.

  Within 'diff_set_value', the CBOR arrays 'diff_entry' MUST be sorted to reflect the corresponding updates to the TRL in reverse chronological order. That is, the first 'diff_entry' element of 'diff_set_value' relates to the most recent update to the portion of the TRL pertaining to the requester. The second 'diff_entry' element relates to the second from last most recent update to that portion, and so on.

* The 'cursor' parameter and the 'more' parameter MUST be included if the Authorization Server supports both the diff queries and the related "Cursor" extension (see Section 5.1.1). Their values are specified according to what is defined in Section 8.2, and provide the requester with information for performing a follow-up query to the TRL endpoint (see Section 8.2).

  If the Authorization Server does not support both diff queries and the "Cursor" extension, these parameters MUST NOT be included. In case the requester does not support both diff queries and the "Cursor" extension, it MUST silently ignore the 'cursor' parameter and the 'more' parameter if present.

Figure 6 provides the CDDL definition [RFC8610] of the CBOR array 'diff_set_value' specified in the response from the Authorization Server, as value of the 'diff_set' parameter.

```
token_hash = bytes
trl_patch = [* token_hash]
diff_entry = [removed: trl_patch, added: trl_patch]
diff_set_value = [* diff_entry]
```

Figure 6: CDDL definition of 'diff_set_value'

Figure 7 shows an example of response from the Authorization Server, following a Diff Query request to the TRL endpoint, where U = 3 diff entries are specified. In this example, the Authorization Server does not support the "Cursor" extension, hence the 'cursor'

parameter and the 'more' parameter are not included in the payload of the response. Also, full token hashes are omitted for brevity.

```
2.05 Content
Content-Format: application/ace-trl+cbor
Payload:
{
   "diff_set" : [
     [
       [ h'01fa51cc ... ', h'01748190 ... '],
       [ h'01cdf1ca ... ', h'01be41a6 ... ']
     ],
     [
       [ h'0144dd12 ... ', h'01231fff ... '],
       []
     ],
     [
       [],
       [ h'01ca986f ... ', h'01fe1a2b ... ']
     ]
   ]
}
```

Figure 7: Example of response following a Diff Query request to the TRL endpoint

Appendix A discusses how performing a diff query of the TRL is in fact a usage example of the Series Transfer Pattern defined in [I-D.bormann-t2trg-stp].

## 8. Response Messages when Using the "Cursor" Extension

If it supports both diff queries and the "Cursor" extension, the Authorization Server composes a response to a full query request or diff query request as defined in Section 8.1 and Section 8.2, respectively.

The exact format of the response depends on the request being a full query or diff query request, on the presence of the query parameter 'cursor' in the diff query request, and on the current status of the update collection associated with the requester.

Error handling and the possible resulting error responses are as defined in Section 5.2.

### 8.1. Response to Full Query

When processing a full query request to the TRL endpoint, the Authorization Server composes a response as defined in Section 6.

In particular, the 'cursor' parameter included in the CBOR map carried in the response payload specifies either the CBOR simple value "null" (0xf6) or a CBOR unsigned integer.

The 'cursor' parameter MUST specify the CBOR simple value "null" in case there are currently no TRL updates pertinent to the requester, i.e., the update collection for that requester is empty. This is the case from when the requester registers at the Authorization Server until a first update pertaining to that requester occurs to the TRL.

Otherwise, the 'cursor' parameter MUST specify a CBOR unsigned integer. This MUST take the 'index' value of the last series item in the update collection associated with the requester (see Section 5.1.1), as corresponding to the most recent update pertaining to the requester occurred to the TRL.

### 8.2. Response to Diff Query

When processing a diff query request to the TRL endpoint, the Authorization Server composes a response as defined in the following.

#### 8.2.1. Empty Collection

If the update collection associated with the requester has no elements, the Authorization Server returns a 2.05 (Content) response. The response MUST have Content-Format "application/ace-trl+cbor" and its payload MUST be a CBOR map formatted as follows.

* The 'diff_set' parameter MUST be included and specifies the empty CBOR array.

* The 'cursor' parameter MUST be included and specifies the CBOR simple value "null" (0xf6).

* The 'more' parameter MUST be included and specifies the CBOR simple value "false" (0xf4).

Note that the above applies when the update collection associated with the requester has no elements, regardless whether the query parameter 'cursor' is included or not in the diff query request, and irrespective of the specified unsigned integer value if present.

#### 8.2.2. Cursor Not Specified in the Diff Query Request

If the update collection associated with the requester is not empty and the diff query request does not include the query parameter

'cursor', the Authorization Server performs the same actions defined in Section 7, with the following differences.

* At step 3, the Authorization Server considers the value MAX_DIFF_BATCH (see Section 5.1.1), and prepares L = min(U, MAX_DIFF_BATCH) diff entries. If L is equal to 0 (e.g., because U is equal to 0), then no diff entries are prepared.

  If U <= MAX_DIFF_BATCH, the prepared diff entries are the last series items in the update collection associated with the requester, corresponding to the L most recent TRL updates pertaining to the requester.

  If U > MAX_DIFF_BATCH, the prepared diff entries are the eldest of the last U series items in the update collection associated with the requester, as corresponding to the first L of the U most recent TRL updates pertaining to the requester.

* At step 4, the CBOR map to carry in the payload of the 2.05 (Content) response MUST be formatted as follows.

  - The 'diff_set' parameter MUST be present and specifies a CBOR array 'diff_set_value' of L elements. Each element of 'diff_set_value' specifies one of the CBOR arrays 'diff_entry' prepared as diff entry. Note that L might have value 0, in which case 'diff_set_value' is the empty CBOR array.

  - The 'cursor' parameter MUST be present and specifies a CBOR unsigned integer. This MUST take the 'index' value of the series item of the update collection included as first diff entry in the 'diff_set_value' CBOR array, which is specified by the 'diff_set' parameter. That is, the 'cursor' parameter takes the 'index' value of the series item in the update collection corresponding to the most recent update pertaining to the requester and returned in this diff query response.

    Note that the 'cursor' parameter takes the same 'index' value of the last series item in the update collection when U <= MAX_DIFF_BATCH.

  - The 'more' parameter MUST be present and MUST specify the CBOR simple value "false" (0xf4) if U <= MAX_DIFF_BATCH, or the CBOR simple value "true" (0xf5) otherwise.

    If the 'more' parameter has value "true", the requester can send a follow-up diff query request including the query parameter 'cursor', with the same value of the 'cursor' parameter specified in this diff query response. As defined in Section 8.2.3, this would result in the Authorization Server transferring the following subset of series items as diff

entries, thus resuming from where interrupted in the previous transfer.

#### 8.2.3. Cursor Specified in the Diff Query Request

If the update collection associated with the requester is not empty and the diff query request includes the query parameter 'cursor' with value P, the Authorization Server proceeds as follows, depending on which of the following two cases hold.

* Case A - The series item X with 'index' having value P and the series item Y with 'index' having value (P + 1) % (MAX_INDEX + 1) are both not found in the update collection associated with the requester. This occurs when the item Y (and possibly further ones after it) has been previously removed from the history of updates for that requester (see step 5 at Section 5.1).

  In this case, the Authorization Server returns a 2.05 (Content) response. The response MUST have Content-Format "application/ace-trl+cbor" and its payload MUST be a CBOR map formatted as follows.

  - The 'diff_set' parameter MUST be included and specifies the empty CBOR array.

  - The 'cursor' parameter MUST be included and specifies the CBOR simple value "null" (0xf6).

  - The 'more' parameter MUST be included and specifies the CBOR simple value "true" (0xf5).

  With the combination ('cursor', 'more') = ("null", "true"), the Authorization Server is signaling that the update collection is in fact not empty, but that one or more series items have been lost due to their removal. These include the item with 'index' value (P + 1) % (MAX_INDEX + 1), that the requester wished to obtain as the first one following the specified reference point with 'index' value P.

  When receiving this diff query response, the requester should send a new full query request to the Authorization Server. A successful response provides the requester with the full, current pertaining portion of the TRL, as well as with a valid value of the 'cursor' parameter (see Section 8.1) to be possibly used as query parameter in a following diff query request.

* Case B - The series item X with 'index' having value P is found in the update collection associated with the requester; or the series item X is not found and the series item Y with 'index'

having value (P + 1) % (MAX_INDEX + 1) is found in the update collection associated with the requester.

In this case, the Authorization Server performs the same actions defined in Section 7, with the following differences.

- At step 3, the Authorization Server considers the value MAX_DIFF_BATCH (see Section 5.1.1), and prepares L = min(SUB_U, MAX_DIFF_BATCH) diff entries, where SUB_U = min(NUM, SUB_SIZE), and SUB_SIZE is the number of series items in the update collection starting from and including the series item added immediately after X. If L is equal to 0 (e.g., because SUB_U is equal to 0), then no diff entries are prepared.

  If SUB_U <= MAX_DIFF_BATCH, the prepared diff entries are the last series items in the update collection associated with the requester, corresponding to the L most recent TRL updates pertaining to the requester.

  If SUB_U > MAX_DIFF_BATCH, the prepared diff entries are the eldest of the last SUB_U series items in the update collection associated with the requester, corresponding to the first L of the SUB_U most recent TRL updates pertaining to the requester.

- At step 4, the CBOR map to carry in the payload of the 2.05 (Content) response MUST be formatted as follows.

  - The 'diff_set' parameter MUST be present and specifies a CBOR array 'diff_set_value' of L elements. Each element of 'diff_set_value' specifies one of the CBOR arrays 'diff_entry' prepared as diff entry. Note that L might have value 0, in which case 'diff_set_value' is the empty CBOR array.

  - The 'cursor' parameter MUST be present and MUST specify a CBOR unsigned integer. In particular:

    - If L is equal to 0, i.e., the series item X is the last one in the update collection, then the 'cursor' parameter MUST take the same 'index' value of the last series item in the update collection.

    - If L is different than 0, then the 'cursor' parameter MUST take the 'index' value of the series element of the update collection included as first diff entry in the 'diff_set' CBOR array. That is, the 'cursor' parameter takes the 'index' value of the series item in the update collection corresponding to the most recent update

pertaining to the requester and returned in this diff query response.

Note that the 'cursor' parameter takes the same 'index' value of the last series item in the update collection when SUB_U <= MAX_DIFF_BATCH.

oThe 'more' parameter MUST be present and MUST specify the CBOR simple value "false" (0xf4) if SUB_U <= MAX_DIFF_BATCH, or the CBOR simple value "true" (0xf5) otherwise.

If 'more' has value "true", the requester can send a follow-up diff query request including the query parameter 'cursor', with the same value of the 'cursor' parameter specified in this diff query response. This would result in the Authorization Server transferring the following subset of series items as diff entries, thus resuming from where interrupted in the previous transfer.

## 9. Upon Registration

During the registration process at the Authorization Server, an administrator or a registered device receives the following information as part of the registration response.

*The url-path to the TRL endpoint at the Authorization Server.

*The hash function used to compute token hashes. This is specified as an integer or a text string, taking value from the "ID" or "Hash Name String" column of the "Named Information Hash Algorithm" Registry [Named.Information.Hash.Algorithm], respectively.

*Optionally, a positive integer N_MAX, if the Authorization Server supports diff queries of the TRL resource (see Section 5.1 and Section 7).

*Optionally, a positive integer MAX_DIFF_BATCH, if the Authorization Server supports diff queries of the TRL resource as well as the related "Cursor" extension (see Section 5.1.1 and Section 8).

After the registration procedure is finished, the administrator or registered device can send a GET request to the TRL resource, including the CoAP Observe Option set to 0 (register), in order to start an observation of the TRL resource at the Authorization Server as per Section 3.1 of [RFC7641]. The GET request can express the wish for a full query (see Section 6) or a diff query (see Section 7) of the TRL.

In case the request is successfully processed, the Authorization Server replies with a response specifying the CoAP response code 2.05 (Content) and including the CoAP Observe Option. The payload of the response is formatted as defined in Section 6 or in Section 7, in case the GET request yielded the execution of a full query or a diff query of the TRL, respectively.

Further details about the registration process at the Authorization Server are out of scope for this specification. Note that the registration process is also out of the scope of the ACE framework for Authentication and Authorization (see Section 5.5 of [RFC9200]).

## 10. Notification of Revoked Tokens

When the TRL is updated (see Section 4.1), the Authorization Server sends Observe Notifications to the observers of the TRL resource. Observe Notifications are sent as per Section 4.2 of [RFC7641].

If the 'pmax' query parameter was specified in the Observation Request starting an observation (see Section 5.2), the Authorization Server might accordingly send additional Observe Notifications to the associated observer. That is, the Authorization Server ensures that no more than pmax seconds elapse between two consecutive notifications sent to that observer, regardless whether the TRL resource has changed or not. If the 'pmax' query parameter was not specified in the Observation Request, a possible maximum time to consider is up to the Authorization Server.

The payload of each Observe Notification is formatted as defined in Section 6 or in Section 7, in case the original Observation Request yielded the execution of a full query or a diff query of the TRL, respectively.

Furthermore, an administrator or a registered device can send additional GET requests to the TRL endpoint at any time, in order to retrieve the token hashes of the pertaining revoked Access Tokens. When doing so, the caller of the TRL endpoint can perform a full query (see Section 6) or a diff query (see Section 7) of the TRL.

When receiving a response from the TRL endpoint, a registered device MUST expunge every stored Access Token associated with a token hash specified in the response.

When a Resource Server RS receives a response from the TRL endpoint specifying the token hash th1 associated with a revoked Access Token t1, the RS might not have received and stored that Access Token yet. This occurs if the Access Token is revoked before it is successfully posted to the Authorization Information Endpoint at the RS (see Section 5.10.1 of [RFC9200]). Such a delay can be due, for example, to messages that get lost in transmission, or rather to the Client

experiencing failures in sending the Access Token to the RS, or deliberately holding the Access Token back.

Thus, in order to ensure that no revoked Access Tokens are accepted and stored, the RS performs the following actions.

- The RS MUST store the token hash th1, until gaining knowledge that the associated revoked Access Token t1 is also expired.

  This can happen when receiving a subsequent response from the TRL endpoint (i.e., indicating that the token hash th1 is not in the TRL portion pertaining to the RS anymore), or when the Access Token t1 is posted to the Authorization Information Endpoint and is found to be expired based on its 'exp' claim [RFC7519], if included.

- The RS MUST NOT accept as valid and store an Access Token t1 posted to the Authorization Information Endpoint, if the corresponding token hash th1 is among the stored ones.

## 11. Interaction Examples

This section provides examples of interactions between a Resource Server RS as a registered device and an Authorization Server AS. The Authorization Server supports both full query and diff query of the TRL, as defined in Section 6 and Section 7, respectively.

The details of the registration process are omitted, but it is assumed that the Resource Server sends an unspecified payload to the Authorization Server, which replies with a 2.01 (Created) response.

The payload of the registration response is a CBOR map, which includes the following entries:

- a "trl_path" parameter, specifying the path of the TRL resource;

- a "trl_hash" parameter, specifying the hash function used to computed token hashes as defined in Section 3;

- an "n_max" parameter, specifying the value of N_MAX, i.e., the maximum number of TRL updates pertaining to each registered device that the Authorization Server retains for that device (see Section 7);

- possible further parameters related to the registration process.

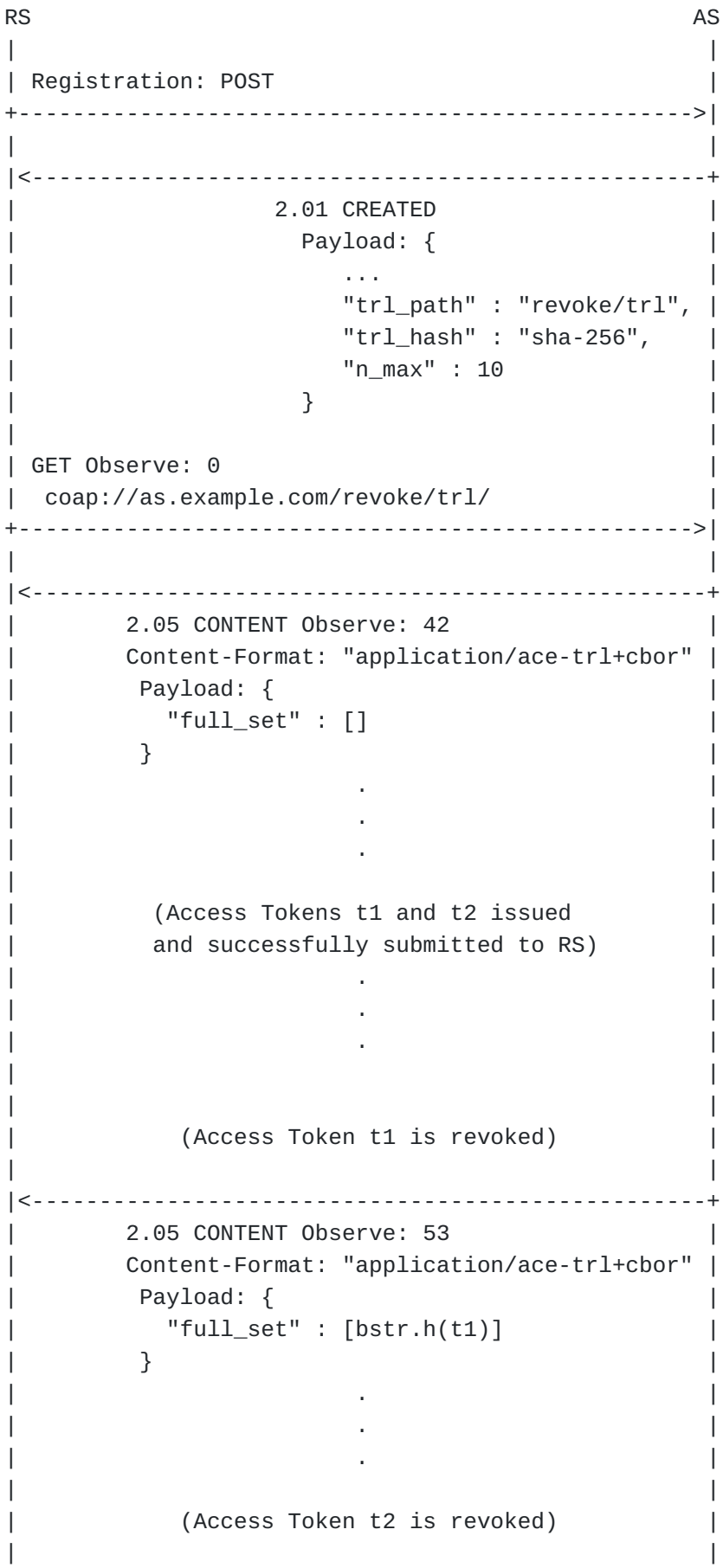
Furthermore, 'h(x)' refers to the hash function used to compute the token hashes, as defined in Section 3 of this specification and according to [RFC6920]. Assuming the usage of CWTs transported in CBOR, 'bstr.h(t1)' and 'bstr.h(t2)' denote the byte-string

representations of the token hashes for the Access Tokens t1 and t2, respectively.

### 11.1. Full Query with Observation

Figure 8 shows an interaction example considering a CoAP observation and a full query of the TRL.

In this example, the Authorization Server does not support the "Cursor" extension (if it supports diff queries at all). Hence the 'cursor' parameter is not included in the payload of the responses to a full query request.


RS
AS
Registration: POST
2.01 CREATED
Payload: {
...
"trl_path" : "revoke/trl",
"trl_hash" : "sha-256",
"n_max" : 10
}
GET Observe: 0
coap://as.example.com/revoke/trl/
2.05 CONTENT Observe: 42
Content-Format: "application/ace-trl+cbor"
Payload: {
"full_set" : []
}
(Access Tokens t1 and t2 issued
and successfully submitted to RS)
(Access Token t1 is revoked)
2.05 CONTENT Observe: 53
Content-Format: "application/ace-trl+cbor"
Payload: {
"full_set" : [bstr.h(t1)]
}
(Access Token t2 is revoked)

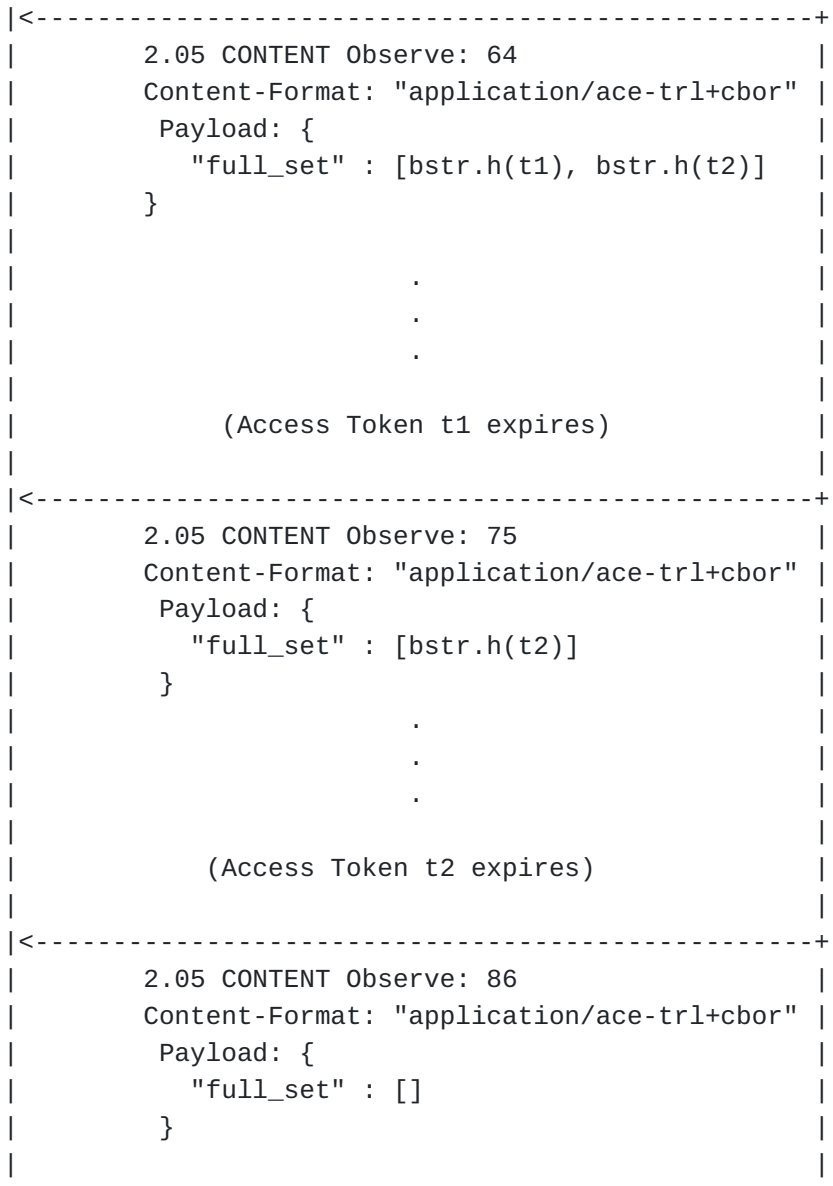

2.05 CONTENT Observe: 64
Content-Format: "application/ace-trl+cbor"
Payload: {
"full_set" : [bstr.h(t1), bstr.h(t2)]
}
(Access Token t1 expires)
2.05 CONTENT Observe: 75
Content-Format: "application/ace-trl+cbor"
Payload: {
"full_set" : [bstr.h(t2)]
}
(Access Token t2 expires)
2.05 CONTENT Observe: 86
Content-Format: "application/ace-trl+cbor"
Payload: {
"full_set" : []
}

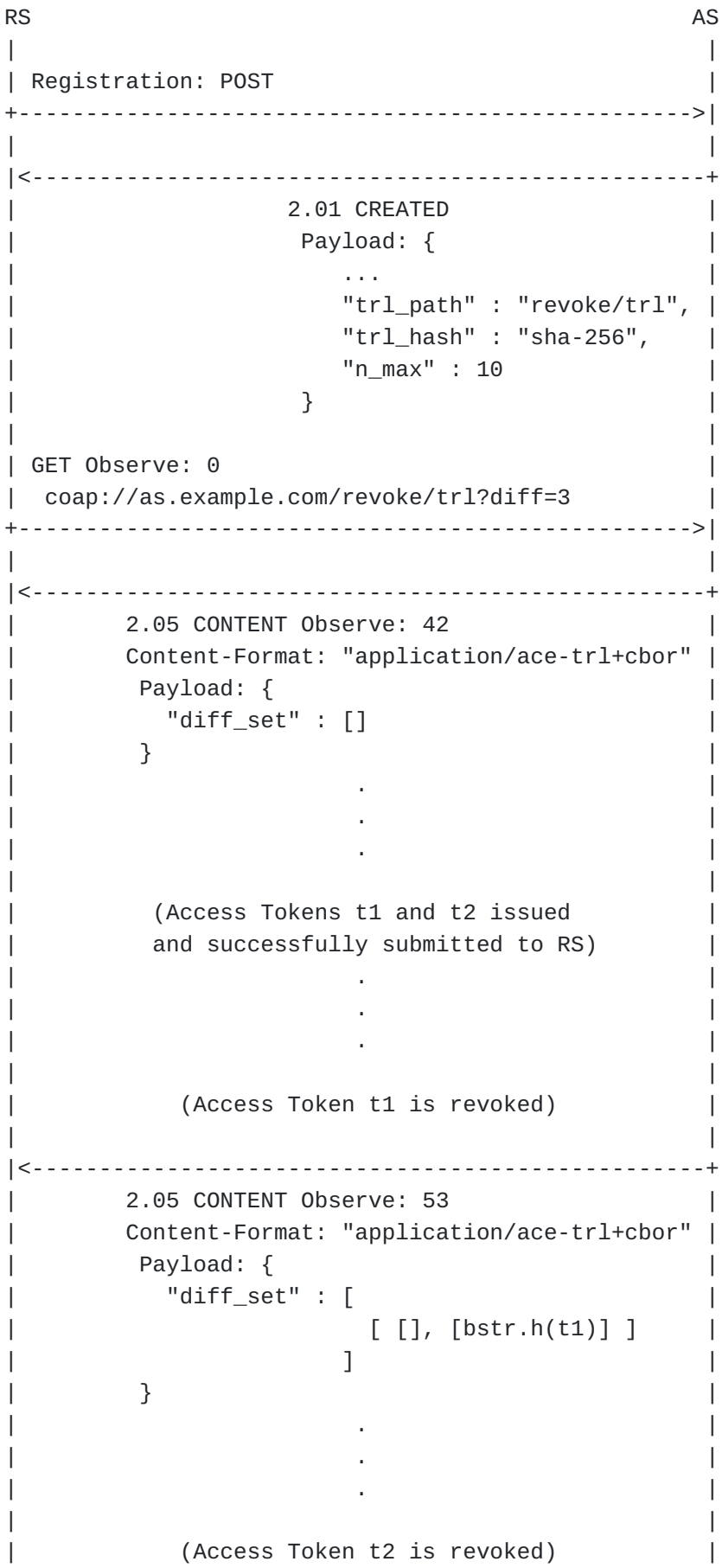
Figure 8: Interaction for Full Query with Observation

### 11.2. Diff Query with Observation

Figure 9 shows an interaction example considering a CoAP observation and a diff query of the TRL.

The Resource Server indicates N=3 as value of the query parameter 'diff', i.e., as the maximum number of diff entries to be specified in a response from the Authorization Server.

In this example, the Authorization Server does not support the "Cursor" extension. Hence the 'cursor' parameter and the 'more' parameter are not included in the payload of the responses to a diff query request.


RS
AS
Registration: POST
2.01 CREATED
Payload: {
...
"trl_path" : "revoke/trl",
"trl_hash" : "sha-256",
"n_max" : 10
}
GET Observe: 0
coap://as.example.com/revoke/trl?diff=3
2.05 CONTENT Observe: 42
Content-Format: "application/ace-trl+cbor"
Payload: {
"diff_set" : []
}
(Access Tokens t1 and t2 issued
and successfully submitted to RS)
(Access Token t1 is revoked)
2.05 CONTENT Observe: 53
Content-Format: "application/ace-trl+cbor"
Payload: {
"diff_set" : [
[ [], [bstr.h(t1)] ]
]
}
(Access Token t2 is revoked)

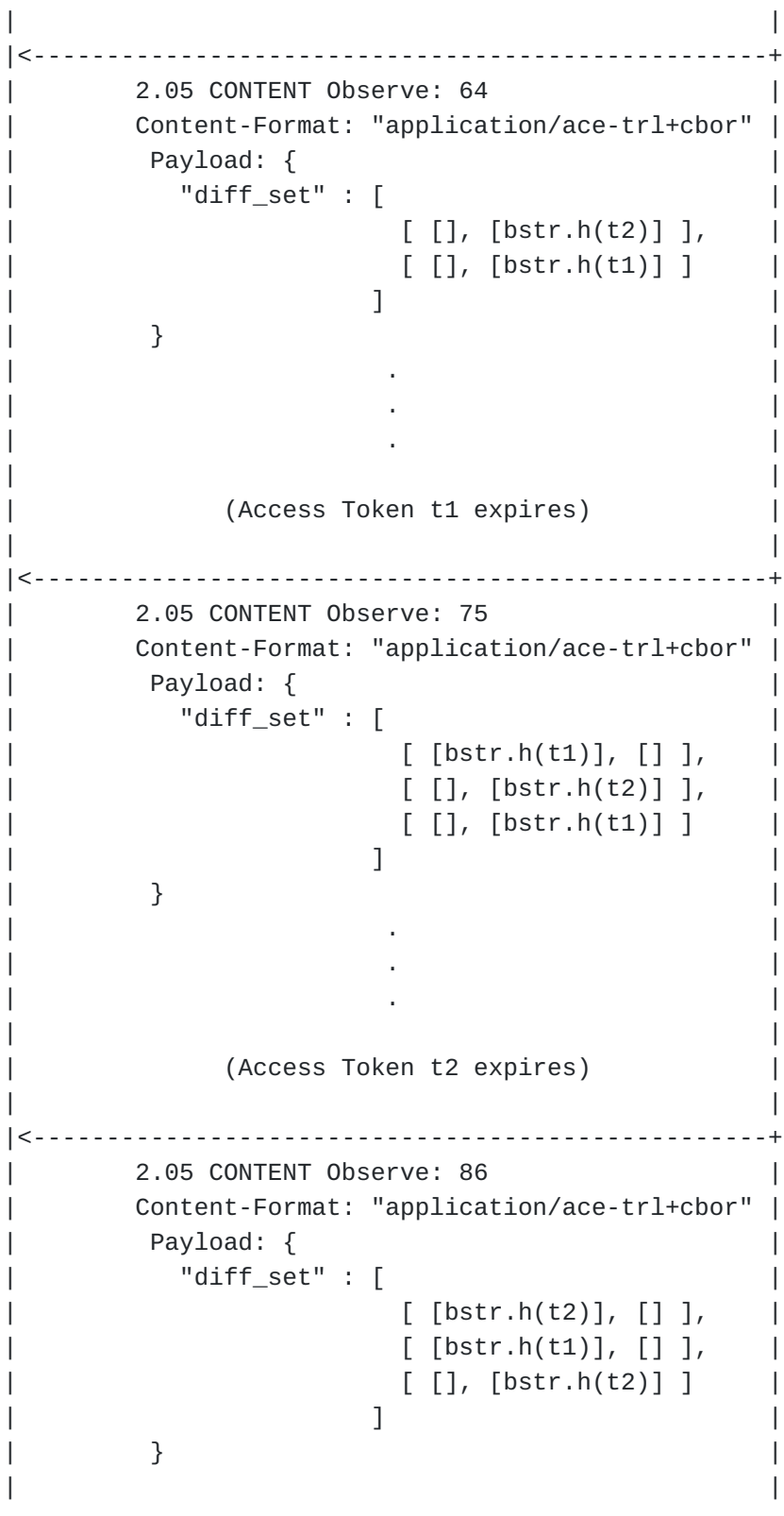

2.05 CONTENT Observe: 64
Content-Format: "application/ace-trl+cbor"
Payload: {
"diff_set" : [
[ [], [bstr.h(t2)] ],
[ [], [bstr.h(t1)] ]
]
}
(Access Token t1 expires)
2.05 CONTENT Observe: 75
Content-Format: "application/ace-trl+cbor"
Payload: {
"diff_set" : [
[ [bstr.h(t1)], [] ],
[ [], [bstr.h(t2)] ],
[ [], [bstr.h(t1)] ]
]
}
(Access Token t2 expires)
2.05 CONTENT Observe: 86
Content-Format: "application/ace-trl+cbor"
Payload: {
"diff_set" : [
[ [bstr.h(t2)], [] ],
[ [bstr.h(t1)], [] ],
[ [], [bstr.h(t2)] ]
]
}

Figure 9: Interaction for Diff Query with Observation

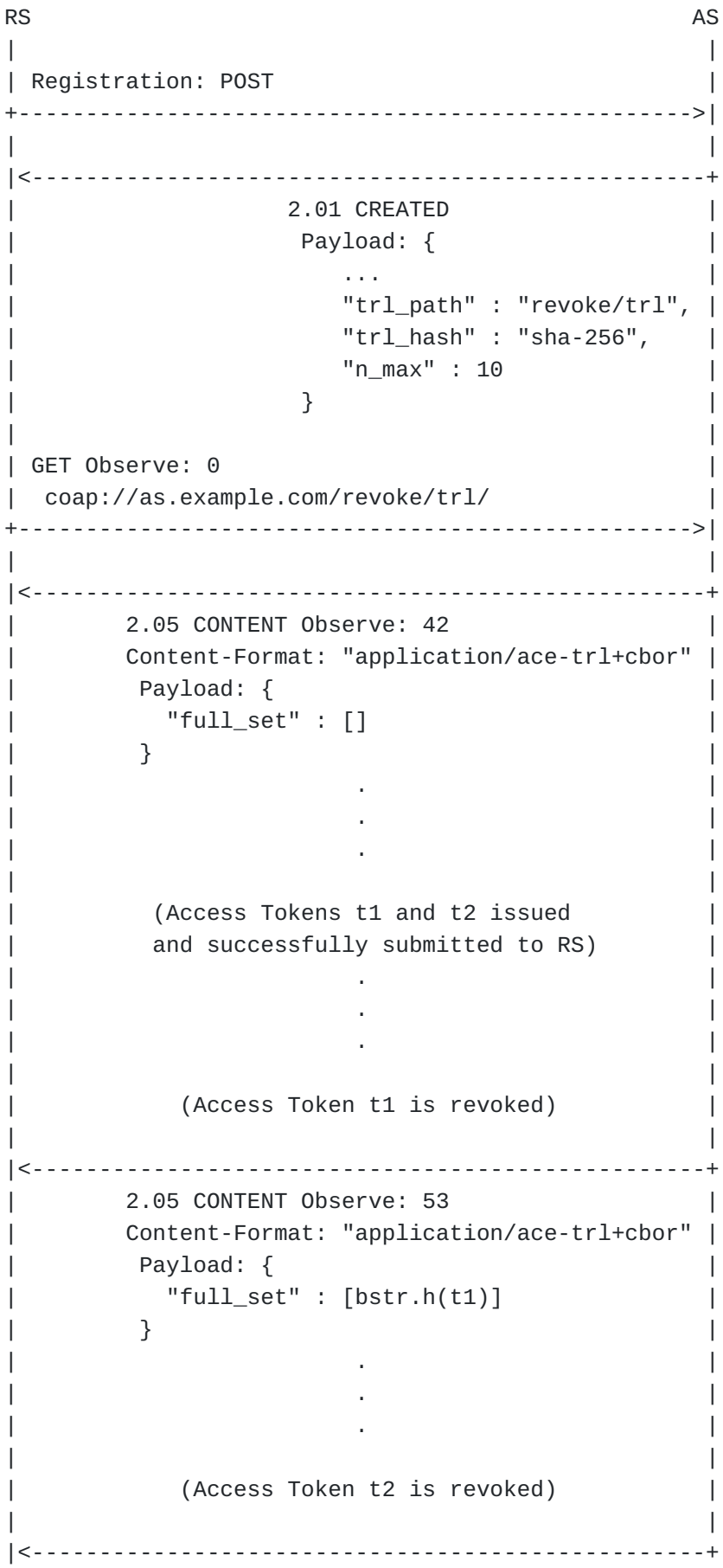
### 11.3. Full Query with Observation and Additional Diff Query

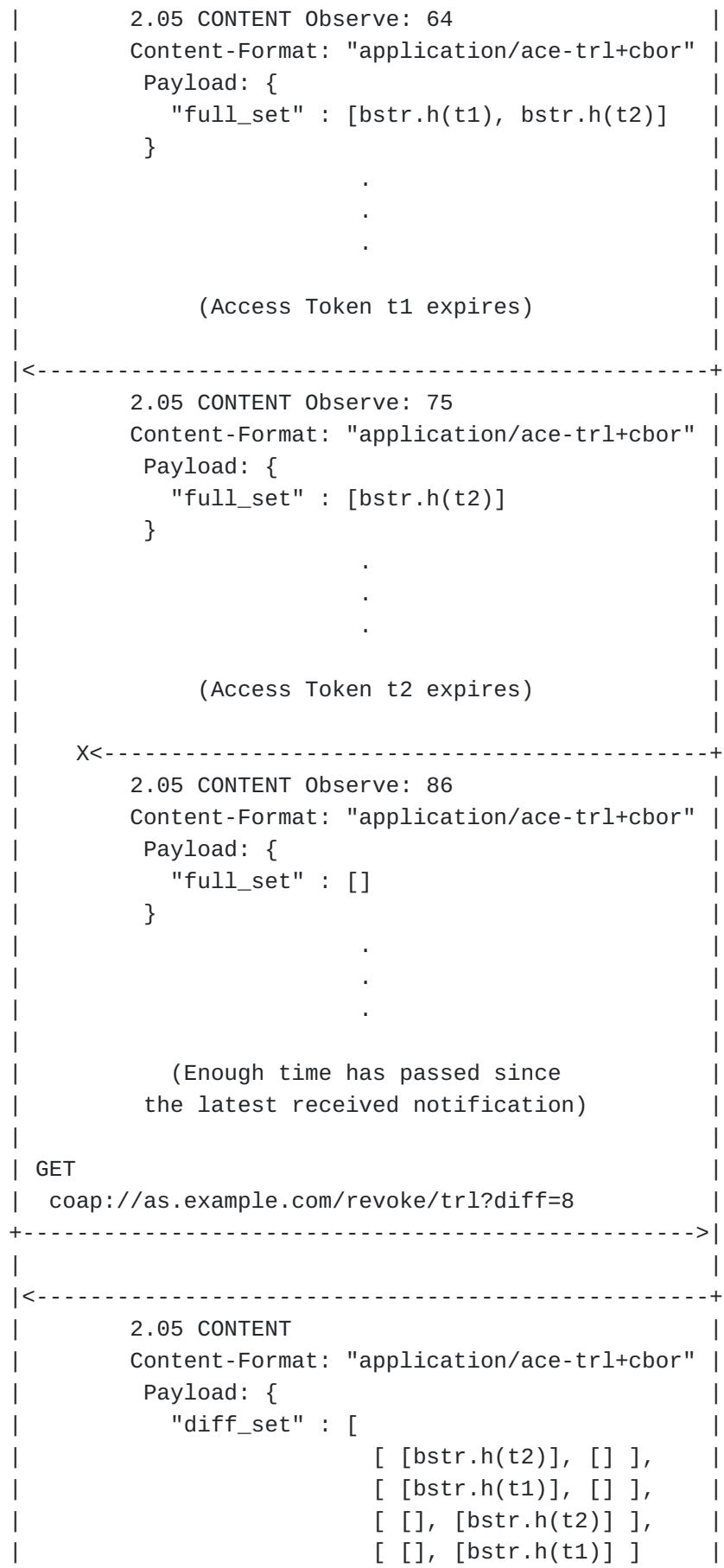
Figure 10 shows an interaction example considering a CoAP observation and a full query of the TRL.

The example also considers one of the notifications from the Authorization Server to get lost in transmission, and thus not reaching the Resource Server.

When this happens, and after a waiting time defined by the application has elapsed, the Resource Server sends a GET request with no Observe Option to the Authorization Server, to perform a diff query of the TRL. The Resource Server indicates N=8 as value of the query parameter 'diff', i.e., as the maximum number of diff entries to be specified in a response from the Authorization Server.

In this example, the Authorization Server does not support the "Cursor" extension. Hence, the 'cursor' parameter is not included in the payload of the responses to a full query request. Also, the 'cursor' parameter and the 'more' parameter are not included in the payload of the responses to a diff query request.


RS
AS
Registration: POST
2.01 CREATED
Payload: {
...
"trl_path" : "revoke/trl",
"trl_hash" : "sha-256",
"n_max" : 10
}
GET Observe: 0
coap://as.example.com/revoke/trl/
2.05 CONTENT Observe: 42
Content-Format: "application/ace-trl+cbor"
Payload: {
"full_set" : []
}
(Access Tokens t1 and t2 issued
and successfully submitted to RS)
(Access Token t1 is revoked)
2.05 CONTENT Observe: 53
Content-Format: "application/ace-trl+cbor"
Payload: {
"full_set" : [bstr.h(t1)]
}
(Access Token t2 is revoked)


2.05 CONTENT Observe: 64
Content-Format: "application/ace-trl+cbor"
Payload: {
"full_set" : [bstr.h(t1), bstr.h(t2)]
}
(Access Token t1 expires)
2.05 CONTENT Observe: 75
Content-Format: "application/ace-trl+cbor"
Payload: {
"full_set" : [bstr.h(t2)]
}
(Access Token t2 expires)
2.05 CONTENT Observe: 86
Content-Format: "application/ace-trl+cbor"
Payload: {
"full_set" : []
}
(Enough time has passed since
the latest received notification)
GET
coap://as.example.com/revoke/trl?diff=8
2.05 CONTENT
Content-Format: "application/ace-trl+cbor"
Payload: {
"diff_set" : [
[ [bstr.h(t2)], [] ],
[ [bstr.h(t1)], [] ],
[ [], [bstr.h(t2)] ],
[ [], [bstr.h(t1)] ]

```
|                          ]                          |
|         }                                           |
|                                                     |
```

Figure 10: Interaction for Full Query with Observation and Diff Query

## 12. ACE Token Revocation List Parameters

This specification defines a number of parameters that can be transported in the response from the TRL endpoint, when the response payload is a CBOR map. Note that such a response MUST use the Content-Format "application/ace-trl+cbor" defined in Section 15.2 of this specification.

The table below summarizes them, and specifies the CBOR value to use as abbreviation instead of the full descriptive name.

| Name | CBOR Value | CBOR Type |
|---|---|---|
| full_set | 0 | array |
| diff_set | 1 | array |
| cursor | 2 | unsigned integer / simple value "null" |
| more | 3 | simple value "false" / simple value "true" |
| error | -1 | int |
| error_description | -2 | tstr |

Figure 11: CBOR abbreviations for the ACE Token Revocation List parameters

## 13. ACE Token Revocation List Error Identifiers

This specification defines a number of values that the Authorization Server can include as error identifiers, in the 'error' field of an error response from the TRL endpoint. This applies to error responses whose payload is a CBOR map and whose Content-Format is "application/ace-trl+cbor".

| Value | Description |
|---|---|
| 0 | Invalid parameter value |
| 1 | Invalid set of parameters |
| 2 | Out of bound cursor value |

Figure 12: ACE Token Revocation List Error Identifiers

## 14. Security Considerations

Security considerations are inherited from the ACE framework for Authentication and Authorization [RFC9200], from [RFC8392] as to the usage of CWTs, from [RFC7519] as to the usage of JWTs, from [RFC7641] as to the usage of CoAP Observe, and from [RFC6920] with regard to resource naming through hashes. The following considerations also apply.

The Authorization Server MUST ensure that each registered device can access and retrieve only its pertaining portion of the TRL. To this end, the Authorization Server can perform the required filtering based on the authenticated identity of the registered device, i.e., a (non-public) identifier that the Authorization Server can securely relate to the registered device and the secure association that they use to communicate.

Disclosing any information about revoked Access Tokens to entities other than the intended registered devices may result in privacy concerns. Therefore, the Authorization Server MUST ensure that, other than registered devices accessing their own pertaining portion of the TRL, only authorized and authenticated administrators can retrieve the full TRL. To this end, the Authorization Server may rely on an access control list or similar.

If a registered device has many non-expired Access Tokens associated with itself that are revoked, the pertaining portion of the TRL could grow to a size bigger than what the registered device is prepared to handle upon reception, especially if relying on a full query of the TRL resource (see Section 6). This could be exploited by attackers to negatively affect the behavior of a registered device. Issuing Access Tokens with not too long expiration time could help reduce the size of a TRL, but an Authorization Server SHOULD take measures to limit this size.

Most of the communication about revoked Access Tokens presented in this specification relies on CoAP Observe Notifications sent from

the Authorization Server to a registered device. The suppression of those notifications by an external attacker that has access to the network would prevent registered devices from ever knowing that their pertaining Access Tokens have been revoked. In order to avoid this, a registered device SHOULD NOT rely solely on the CoAP Observe notifications. In particular, a registered device SHOULD also regularly poll the Authorization Server for the most current information about revoked Access Tokens, by sending GET requests to the TRL endpoint according to a related application policy.

## 15. 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.

### 15.1. Media Type Registrations

IANA is asked to register the media type "application/ace-trl+cbor" for messages of the protocols defined in this document encoded in CBOR. This registration follows the procedures specified in [RFC6838].

Type name: application

Subtype name: ace-trl+cbor

Required parameters: N/A

Optional parameters: N/A

Encoding considerations: Must be encoded as CBOR map containing the protocol parameters defined in [RFC-XXXX].

Security considerations: See Section 14 of this document.

Interoperability considerations: N/A

Published specification: [RFC-XXXX]

Applications that use this media type: The type is used by Authorization Servers, Clients and Resource Servers that support the notification of revoked Access Tokens, according to a Token Revocation List maintained by the Authorization Server as specified in [RFC-XXXX].

Fragment identifier considerations: N/A

Additional information: N/A

Person & email address to contact for further information:
<iesg@ietf.org>

Intended usage: COMMON

Restrictions on usage: None

Author: Marco Tiloca <marco.tiloca@ri.se>

Change controller: IESG

### 15.2. CoAP Content-Formats Registry

IANA is asked to add the following entry to the "CoAP Content-Formats" registry within the "CoRE Parameters" registry group.

Media Type: application/ace-trl+cbor

Encoding: -

ID: TBD

Reference: [RFC-XXXX]

### 15.3. ACE Token Revocation List Parameters Registry

This specification establishes the "ACE Token Revocation List Parameters" IANA registry. The registry has been created to use the "Expert Review" registration procedure [RFC8126]. Expert Review guidelines are provided in Section 15.5. It should be noted that, in addition to the Expert Review, some portions of the registry require a specification, potentially a Standards Track RFC, to be supplied as well.

The columns of this registry are:

* Name: This is a descriptive name that enables easier reference to the item. The name MUST be unique. It is not used in the encoding.

* CBOR Value: This is the value used as CBOR abbreviation of the item. These values MUST be unique. The value can be a positive integer or a negative integer. Different ranges of values use different registration policies [RFC8126]. Integer values from -256 to 255 are designated as Standards Action. Integer values from -65536 to -257 and from 256 to 65535 are designated as Specification Required. Integer values greater than 65535 are designated as Expert Review. Integer values less than -65536 are marked as Private Use.

*CBOR Type: This contains the CBOR type of the item, or a pointer to the registry that defines its type, when that depends on another item.

*Reference: This contains a pointer to the public specification for the item.

This registry has been initially populated by the values in Section 12. The "Reference" column for all of these entries refers to this document.

### 15.4. ACE Token Revocation List Errors

This specification establishes the "ACE Token Revocation List Errors" IANA registry. The registry has been created to use the "Expert Review" registration procedure [RFC8126]. Expert Review guidelines are provided in Section 15.5. It should be noted that, in addition to the Expert Review, some portions of the registry require a specification, potentially a Standards Track RFC, to be supplied as well.

The columns of this registry are:

*Value: The value to be used to identify the error. The value MUST be unique. The value can be a positive integer or a negative integer. Integer values between 0 and 255 are designated as Standards Track Document required. Integer values from 256 to 65535 are designated as Specification Required. Integer values greater than 65535 are designated as Expert Review. Integer values less than -65536 are marked as private use.

*Description: This field contains a brief description of the error.

*Reference: This field contains a pointer to the public specification defining the error, if one exists.

This registry has been initially populated by the values in Section 13. The "Reference" column for all of these entries refers to this document.

### 15.5. Expert Review Instructions

The IANA registries established in this document is defined as Expert Review. This section gives some general guidelines for what the experts should be looking for, but they are being designated as experts for a reason so they should be given substantial latitude.

Expert reviewers should take into consideration the following points:

* Point squatting should be discouraged. Reviewers are encouraged to get sufficient information for registration requests to ensure that the usage is not going to duplicate one that is already registered and that the point is likely to be used in deployments. The zones tagged as private use are intended for testing purposes and closed environments. Code points in other ranges should not be assigned for testing.

* Specifications are required for the Standards Track range of point assignment. Specifications should exist for Specification Required ranges, but early assignment before a specification is available is considered to be permissible. Specifications are needed for the Expert Review range if they are expected to be used outside of closed environments in an interoperable way. When specifications are not provided, the description provided needs to have sufficient information to identify what the point is being used for.

* Experts should take into account the expected usage of fields when approving point assignment. The fact that there is a range for Standards Track documents does not mean that a Standards Track document cannot have points assigned outside of that range. The length of the encoded value should be weighed against how many code points of that length are left, the size of device it will be used on, and the number of code points left that encode to that size.

# 16. References

## 16.1. Normative References


**[I-D.ietf-core-conditional-attributes]** Koster, M., Soloway, A., and B. Silverajan, "Conditional Attributes for Constrained RESTful Environments", Work in Progress, Internet-Draft, draft-ietf-core-conditional-attributes-04, 10 May 2022, <https://www.ietf.org/archive/id/draft-ietf-core-conditional-attributes-04.txt>.

**[Named.Information.Hash.Algorithm]** IANA, "Named Information Hash Algorithm", <https://www.iana.org/assignments/named-information/named-information.xhtml>.

**[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>.

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/info/rfc6749>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <https://www.rfc-editor.org/info/rfc6838>.

[RFC6920] Farrell, S., Kutscher, D., Dannewitz, C., Ohlman, B., Keranen, A., and P. Hallam-Baker, "Naming Things with Hashes", RFC 6920, DOI 10.17487/RFC6920, April 2013, <https://www.rfc-editor.org/info/rfc6920>.

[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>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/info/rfc7519>.

[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>.

[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/info/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/info/rfc8174>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

[RFC8392] Jones, M., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "CBOR Web Token (CWT)", RFC 8392, DOI 10.17487/RFC8392, May 2018, <https://www.rfc-editor.org/info/rfc8392>.

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to

Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/info/rfc8610>.

**[RFC8949]** Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/info/rfc8949>.

**[RFC9200]** Seitz, L., Selander, G., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "Authentication and Authorization for Constrained Environments Using the OAuth 2.0 Framework (ACE-OAuth)", RFC 9200, DOI 10.17487/RFC9200, August 2022, <https://www.rfc-editor.org/info/rfc9200>.

### 16.2. Informative References

**[I-D.bormann-t2trg-stp]** Bormann, C. and K. Hartke, "The Series Transfer Pattern (STP)", Work in Progress, Internet-Draft, draft-bormann-t2trg-stp-03, 7 April 2020, <https://www.ietf.org/archive/id/draft-bormann-t2trg-stp-03.txt>.

**[RFC7009]** Lodderstedt, T., Ed., Dronia, S., and M. Scurtescu, "OAuth 2.0 Token Revocation", RFC 7009, DOI 10.17487/RFC7009, August 2013, <https://www.rfc-editor.org/info/rfc7009>.

## Appendix A. On using the Series Transfer Pattern

Performing a diff query of the TRL as specified in Section 7 is in fact a usage example of the Series Transfer Pattern defined in [I-D.bormann-t2trg-stp].

That is, a diff query enables the transfer of a series of TRL updates, with the Authorization Server specifying U <= N_MAX diff entries as the U most recent updates to the portion of the TRL pertaining to a requester, i.e., a registered device or an administrator.

When responding to a diff query request from a requester (see Section 7), 'diff_set' is a subset of the update collection associated with the requester, where each 'diff_entry' record is a series item from that update collection. Note that 'diff_set' specifies the whole current update collection when the value of U is equal to SIZE, i.e., the current number of series items in the update collection.

The value N of the query parameter 'diff' in the GET request allows the requester and the Authorization Server to trade the amount of provided information with the latency of the information transfer.

Since the update collection associated with each requester includes up to N_MAX series item, the Authorization Server deletes the oldest series item when a new one is generated and added to the end of the update collection, due to a new TRL update pertaining to that requester (see Section 5.1). This addresses the question "When can the server decide to no longer retain older items?" raised in Section 3.2 of [I-D.bormann-t2trg-stp].

Furthermore, performing a diff query of the TRL together with the "Cursor" extension as specified in Section 8 in fact relies on the "Cursor" pattern of the Series Transfer Pattern (see Section 3.3 of [I-D.bormann-t2trg-stp]).

## Appendix B. Document Updates

RFC EDITOR: Please remove this section.

### B.1. Version -02 to -03

* Definition of MAX_INDEX for the "Cursor" extension.

* Handling wrap-around of 'index' when using the "Cursor" extension.

* Error handling for the case where 'cursor' > MAX_INDEX.

* Improved error handling in case 'index' is out-of-bound.

* Clarified parameter semantics, message content and examples.

* Editorial improvements.

### B.2. Version -01 to -02

* Earlier mentioning of error cases.

* Clearer distinction between maintaining the history of TRL updates and preparing the response to a diff query.

* Defined the use of "cursor" in the document body, as an extension of diff queries.

* Both success and error responses have a CBOR map as payload.

* Corner cases of message processing explained more explcitly.

*Clarifications and editorial improvements.

### B.3. Version -00 to -01

*Added actions to perform upon receiving responses from the TRL endpoint.

*Fixed off-by-one error when using the "Cursor" pattern.

*Improved error handling, with registered error codes.

*Section restructuring (full- and diff-query as self-standing sections).

*Renamed identifiers and CBOR parameters.

*Clarifications and editorial improvements.

## Acknowledgments

The authors sincerely thank Christian Amsüss, Carsten Bormann, Benjamin Kaduk, David Navarro, Marco Rasori, Michael Richardson, Jim Schaad, Göran Selander and Travis Spencer for their comments and feedback.

The work on this document has been partly supported by VINNOVA and the Celtic-Next project CRITISEC; and by the H2020 project SIFIS-Home (Grant agreement 952652).

## Authors' Addresses

Marco Tiloca
RISE AB
Isafjordsgatan 22
SE-16440 Kista
Sweden

Email: marco.tiloca@ri.se

Ludwig Seitz
Combitech
Djaeknegatan 31
SE-21135 Malmoe
Sweden

Email: ludwig.seitz@combitech.com

Francesca Palombini
Ericsson AB
Torshamnsgatan 23

SE-16440 Kista
Sweden

Email: francesca.palombini@ericsson.com

Sebastian Echeverria
CMU SEI
4500 Fifth Avenue
Pittsburgh, PA, 15213-2612
United States of America

Email: secheverria@sei.cmu.edu

Grace Lewis
CMU SEI
4500 Fifth Avenue
Pittsburgh, PA, 15213-2612
United States of America

Email: glewis@sei.cmu.edu