Independent Submission
Request for Comments: 7967
Category: Informational
ISSN: 2070-1721

A. Bhattacharyya
S. Bandyopadhyay
A. Pal
T. Bose
Tata Consultancy Services Ltd.

August 2016

Constrained Application Protocol (CoAP) Option for No Server Response

Abstract

There can be machine-to-machine (M2M) scenarios where server responses to client requests are redundant. This kind of open-loop exchange (with no response path from the server to the client) may be desired to minimize resource consumption in constrained systems while updating many resources simultaneously or performing high-frequency updates. CoAP already provides Non-confirmable (NON) messages that are not acknowledged by the recipient. However, the request/response semantics still require the server to respond with a status code indicating "the result of the attempt to understand and satisfy the request", per RFC 7252.

This specification introduces a CoAP option called 'No-Response'. Using this option, the client can explicitly express to the server its disinterest in all responses against the particular request. This option also provides granular control to enable expression of disinterest to a particular response class or a combination of response classes. The server MAY decide to suppress the response by not transmitting it back to the client according to the value of the No-Response option in the request. This option may be effective for both unicast and multicast requests. This document also discusses a few examples of applications that benefit from this option.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This is a contribution to the RFC Series, independently of any other RFC stream. The RFC Editor has chosen to publish this document at its discretion and makes no statement about its value for implementation or deployment. Documents approved for publication by the RFC Editor are not a candidate for any level of Internet Standard; see Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7967.

Copyright Notice

Table of Contents

1. Introduction

This specification defines a new option for the Constrained Application Protocol (CoAP) [RFC7252] called 'No-Response'. This option enables clients to explicitly express their disinterest in receiving responses back from the server. The disinterest can be expressed at the granularity of response classes (e.g., 2.xx) or a combination of classes (e.g., 2.xx and 5.xx). By default, this option indicates interest in all response classes. The server MAY decide to suppress the response by not transmitting it back to the client according to the value of the No-Response option in the request.

Along with the technical details, this document presents some practical application scenarios that highlight the usefulness of this option. [ITS-LIGHT] and [CoAP-ADAPT] contain the background research for this document.

In this document, when it is mentioned that a request from a client is with No-Response, the intended meaning is that the client expresses its disinterest for all or some selected classes of responses.

1.1. Potential Benefits

The use of the No-Response option should be driven by typical application requirements and, particularly, characteristics of the information to be updated. If this option is opportunistically used in a fitting M2M application, then the concerned system may benefit in the following aspects. (However, note that this option is elective, and servers can simply ignore the preference expressed by the client.)

* Reduction in network congestion due to effective reduction of the overall traffic.

* Reduction in server-side load by relieving the server from responding to requests for which responses are not necessary.

* Reduction in battery consumption at the constrained endpoint(s).

* Reduction in overall communication cost.

1.2. Terminology

The terms used in this document are in conformance with those defined in [RFC7252].

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

2. Option Definition

The properties of the No-Response option are given in Table 1. In this table, the C, U, N, and R columns indicate the properties Critical, Unsafe, NoCacheKey, and Repeatable, respectively.

Number	C	U	N	R	Name	Format	Length	Default
258		X	-		No-Response	uint	0-1	0

Table 1: Option Properties

This option is a request option. It is elective and not repeatable. This option is Unsafe-to-Forward, as the intermediary MUST know how to interpret this option. Otherwise, the intermediary (without knowledge about the special unidirectional nature of the request) would wait for responses.

Note: Since CoAP maintains a clear separation between the request/response and the message sub-layer, this option does not have any dependency on the type of message (Confirmable/Non-confirmable). So, even the absence of a message sub-layer (e.g., CoAP over TCP [CoAP-TCP-TLS]) should have no effect on the interpretation of this option. However, considering the CoAP-over-UDP scenario [RFC7252], NON messages are best suited to this option because of the expected benefits. Using No-Response with NON messages gets rid of any kind of reverse traffic, and the interaction becomes completely open loop.

Using this option with CON requests may not serve the desired purpose if piggybacked responses are triggered. But, if the server responds with a separate response (which, perhaps, the client does not care about), then this option can be useful. Suppressing the separate response reduces traffic by one additional CoAP message in this case.

This option contains values to indicate disinterest in all or a particular class or combination of classes of responses as described in Section 2.1.

2.1. Granular Control over Response Suppression

This option enables granular control over response suppression by allowing the client to express its disinterest in a typical class or combination of classes of responses. For example, a client may explicitly tell the receiver that no response is required unless

something 'bad' happens and a response of class 4.xx or 5.xx is to be fed back to the client. No response of the class 2.xx is required in such case.

Note: Section 2.7 of [RFC7390] describes a scheme where a server in the multicast group may decide on its own to suppress responses for group communication with granular control. The client does not have any knowledge about that. However, on the other hand, the No-Response option enables the client to explicitly inform the servers about its disinterest in responses. Such explicit control on the client side may be helpful for debugging network resources. An example scenario is described in Section 4.2.1.

The server MUST send back responses of the classes for which the client has not expressed any disinterest. There may be instances where a server, on its own, decides to suppress responses. An example is suppression of responses by multicast servers as described in Section 2.7 of [RFC7390]. If such a server receives a request with a No-Response option showing 'interest' in specific response classes (i.e., not expressing disinterest for these options), then any default behavior of suppressing responses, if present, MUST be overridden to deliver those responses that are of interest to the client.

So, for example, suppose a multicast server suppresses all responses by default and receives a request with a No-Response option expressing disinterest in 2.xx (success) responses only. Note that the option value naturally expresses interest in error responses 4.xx and 5.xx in this case. Thus, the server must send back a response if the concerned request caused an error.

The option value is defined as a bit map (Table 2) to achieve granular suppression. Its length can be 0 bytes (empty) or 1 byte.

Value	Binary Representation	Description
0	<empty>	Interested in all responses.
2	00000010	Not interested in 2.xx responses.
8	00001000	Not interested in 4.xx responses.
16	00010000	Not interested in 5.xx responses.

Table 2: Option Values

The conventions used in deciding the option values are:

1. To suppress an individual class: Set bit number (n-1) starting from the least significant bit (bit number 0) to suppress all responses belonging to class n.xx. So,

   option value to suppress n.xx class = 2**(n-1)

2. To suppress a combination of classes: Set each corresponding bit according to point 1 above. Example: The option value will be 18 (binary: 00010010) to suppress both 2.xx and 5.xx responses. This is essentially bitwise OR of the corresponding individual values for suppressing 2.xx and 5.xx. The "CoAP Response Codes" registry (see Section 12.1.2 of [RFC7252]) defines 2.xx, 4.xx, and 5.xx responses. So, an option value of 26 (binary: 00011010) will request to suppress all response codes defined in [RFC7252].

Note: When No-Response is used with value 26 in a request, the client endpoint SHOULD cease listening to response(s) to the particular request. On the other hand, showing interest in at least one class of response means that the client endpoint can no longer completely cease listening activity and must be configured to listen during some application specific time-out period for the particular request. The client endpoint never knows whether the present request will be a success or a failure. Thus, for example, if the client decides to open up the response for errors (4.xx and 5.xx), then it has to wait for the entire time-out period -- even for the instances where the request is successful (and the server is not supposed to send back a response). Note that in this context there may be situations when the response to errors might get lost. In such a situation, the client would wait during the time-out period but would not receive any response. However, this should not give the client the impression that the request was necessarily successful. In other words, in this case, the client cannot distinguish between response suppression and message loss. The application designer needs to tackle this situation. For example, while performing frequent updates, the client may strategically interweave requests without No-Response option into a series of requests with No-Response to check periodically that things are fine at the server end and the server is actively responding.

2.2. Method-Specific Applicability Considerations

The following table provides a ready reference on the possible applicability of this option with four REST methods. This table is for the type of possible interactions foreseen at the time of preparing this specification. The key words from RFC 2119 such as "SHOULD NOT", etc., deliberately have not been used in this table because it only contains suggestions.

Method Name	Remarks on Applicability
GET	This should not be used with a conventional GET request when the client requests the contents of a resource. However, this option may be useful for exceptional cases where GET requests have side effects. For instance, the proactive cancellation procedure for observing a request [RFC7641] requires a client to issue a GET request with the Observe option set to 1 (’deregister’). If it is more efficient to use this deregistration instead of reactive cancellation (rejecting the next notification with RST), the client MAY express its disinterest in the response to such a GET request.
PUT	Suitable for frequent updates (particularly in NON messages) on existing resources. Might not be useful when PUT is used to create a new resource, as it may be important for the client to know that the resource creation was actually successful in order to carry out future actions. Also, it may be important to ensure that a resource was actually created rather than updating an existing resource.
POST	If POST is used to update a target resource, then No-Response can be used in the same manner as in PUT. This option may also be useful while updating through query strings rather than updating a fixed target resource (see Section 4.1.2.2 for an example).
DELETE	Deletion is usually a permanent action. If the client wants to ensure that the deletion request was properly executed, then this option should not be used with the request.

Table 3: Suggested Applicability of No-Response with REST Methods

3. Miscellaneous Aspects

This section further describes important implementation aspects worth considering while using the No-Response option. The following discussion contains guidelines and requirements (derived by combining [RFC7252], [RFC7390], and [RFC5405]) for the application developer.

3.1. Reusing Tokens

Tokens provide a matching criteria between a request and the corresponding response. The life of a Token starts when it is assigned to a request and ends when the final matching response is received. Then, the Token can again be reused. However, a request with No-Response typically does not have any guaranteed response path. So, the client has to decide on its own about when it can retire a Token that has been used in an earlier request so that the Token can be reused in a future request. Since the No-Response option is 'elective', a server that has not implemented this option will respond back. This leads to the following two scenarios:

The first scenario is when the client is never going to care about any response coming back or about relating the response to the original request. In that case, it MAY reuse the Token value at liberty.

However, as a second scenario, let us consider that the client sends two requests where the first request is with No-Response and the second request (with the same Token) is without No-Response. In this case, a delayed response to the first one can be interpreted as a response to the second request (client needs a response in the second case) if the time interval between using the same Token is not long enough. This creates a problem in the request-response semantics.

The most ideal solution would be to always use a unique Token for requests with No-Response. But if a client wants to reuse a Token, then in most practical cases the client implementation SHOULD implement an application-specific reuse time after which it can reuse the Token. A minimum reuse time for Tokens with a similar expression as in Section 2.5 of [RFC7390] SHOULD be used:

```
TOKEN_REUSE_TIME = NON_LIFETIME + MAX_SERVER_RESPONSE_DELAY +
                   MAX_LATENCY
```

NON_LIFETIME and MAX_LATENCY are defined in Section 4.8.2 of [RFC7252]. MAX_SERVER_RESPONSE_DELAY has the same interpretation as in Section 2.5 of [RFC7390] for a multicast request. For a unicast request, since the message is sent to only one server, MAX_SERVER_RESPONSE_DELAY means the expected maximum response delay

from the particular server to that client that sent the request. For multicast requests, MAX_SERVER_RESPONSE_DELAY has the same interpretation as in Section 2.5 of [RFC7390]. So, for multicast it is the expected maximum server response delay "over all servers that the client can send a multicast request to", per [RFC7390]. This response delay for a given server includes its specific Leisure period; where Leisure is defined in Section 8.2 of [RFC7252]. In general, the Leisure for a server may not be known to the client. A lower bound for Leisure, lb_Leisure, is defined in [RFC7252], but not an upper bound as is needed in this case. Therefore, the upper bound can be estimated by taking N (N>>1) times the lower bound lb_Leisure:

lb_Leisure = S * G / R

where
S = estimated response size
G = group size estimate
R = data transfer rate

Any estimate of MAX_SERVER_RESPONSE_DELAY MUST be larger than DEFAULT_LEISURE, as defined in [RFC7252].

Note: If it is not possible for the client to get a reasonable estimate of the MAX_SERVER_RESPONSE_DELAY, then the client, to be safe, SHOULD use a unique Token for each stream of messages.

3.2. Taking Care of Congestion Control and Server-Side Flow Control

This section provides guidelines for basic congestion control. Better congestion control mechanisms can be designed as future work.

If this option is used with NON messages, then the interaction becomes completely open loop. The absence of any feedback from the server-end affects congestion-control mechanisms. In this case, the communication pattern maps to the scenario where the application cannot maintain an RTT estimate as described in Section 3.1.2 of [RFC5405]. Hence, per [RFC5405], a 3-second interval is suggested as the minimum interval between successive updates, and it is suggested to use an even less aggressive rate when possible. However, in case of a higher rate of updates, the application MUST have some knowledge about the channel, and an application developer MUST interweave occasional closed-loop exchanges (e.g., NON messages without No-Response, or CON messages) to get an RTT estimate between the endpoints.

Interweaving requests without No-Response is a MUST in case of an aggressive request rate for applications where server-side flow control is necessary. For example, as proposed in [CoAP-PUBSUB], a

broker MAY return 4.29 (Too Many Requests) in order to request a client to slow down the request rate. Interweaving requests without No-Response allows the client to listen to such a response.

3.3. Considerations regarding Caching of Responses

The cacheability of CoAP responses does not depend on the request method, but it depends on the Response Code. The No-Response option does not lead to any impact on cacheability of responses. If a request containing No-Response triggers a cacheable response, then the response MUST be cached. However, the response MAY not be transmitted considering the value of the No-Response option in the request.

For example, if a request with No-Response triggers a cacheable response of 4.xx class with Max-Age not equal to 0, then the response must be cached. The cache will return the response to subsequent similar requests without No-Response as long as the Max-Age has not elapsed.

3.4. Handling the No-Response Option for a HTTP-to-CoAP Reverse Proxy

A HTTP-to-CoAP reverse proxy MAY translate an incoming HTTP request to a corresponding CoAP request indicating that no response is required (showing disinterest in all classes of responses) based on some application-specific requirement. In this case, it is RECOMMENDED that the reverse proxy generate an HTTP response with status code 204 (No Content) when such response is allowed. The generated response is sent after the proxy has successfully sent out the CoAP request.

If the reverse proxy applies No-Response for one or more classes of responses, it will wait for responses up to an application-specific maximum time (T_max) before responding to the HTTP side. If a response of a desired class is received within T_max, then the response gets translated to HTTP as defined in [HTTP-to-CoAP]. However, if the proxy does not receive any response within T_max, it is RECOMMENDED that the reverse Proxy send an HTTP response with status code 204 (No Content) when allowed for the specific HTTP request method.

4. Application Scenarios

This section describes some examples of application scenarios that may potentially benefit from the use of the No-Response option.

4.1. Frequent Update of Geolocation from Vehicles to Backend Server

Let us consider an intelligent traffic system (ITS) consisting of vehicles equipped with a sensor gateway comprising sensors like GPS and accelerometer sensors. The sensor gateway acts as a CoAP client. It connects to the Internet using a low-bandwidth cellular connection (e.g., General Packet Radio Service (GPRS)). The GPS coordinates of the vehicle are periodically updated to the backend server.

While performing frequent location updates, retransmitting (through the CoAP CON mechanism) a location coordinate that the vehicle has already left is not efficient as it adds redundant traffic to the network. Therefore, the updates are done using NON messages. However, given the huge number of vehicles updating frequently, the NON exchange will also trigger a huge number of responses from the backend. Thus, the cumulative load on the network will be quite significant. Also, the client in this case may not be interested in getting responses to location update requests for a location it has already passed and when the next location update is imminent.

On the contrary, if the client endpoints on the vehicles explicitly declare that they do not need any status response back from the server, then load will be reduced significantly. The assumption is that the high rate of updates will compensate for the stray losses in geolocation reports.

Note: It may be argued that the above example application can also be implemented using the Observe option [RFC7641] with NON notifications. But, in practice, implementing with Observe would require lot of bookkeeping at the data collection endpoint at the backend (observer). The observer needs to maintain all the observe relationships with each vehicle. The data collection endpoint may be unable to know all its data sources beforehand. The client endpoints at vehicles may go offline or come back online randomly. In the case of Observe, the onus is always on the data collection endpoint to establish an observe relationship with each data source. On the other hand, implementation will be much simpler if initiating is left to the data source to carry out updates using the No-Response option. Another way of looking at it is that the implementation choice depends on where there is interest to initiate the update. In an Observe scenario, the interest is expressed by the data consumer. In contrast, the

classic update case applies when the interest is from the data producer. The No-Response option makes classic updates consume even less resources.

The following subsections illustrate some sample exchanges based on the application described above.

4.1.1. Using No-Response with PUT

Each vehicle is assigned a dedicated resource "vehicle-stat-<n>", where <n> can be any string uniquely identifying the vehicle. The update requests are sent using NON messages. The No-Response option causes the server not to respond back.

Figure 1: Example of Unreliable Update with No-Response Option Using PUT

4.1.2. Using No-Response with POST

4.1.2.1. POST Updating a Fixed Target Resource

In this case, POST acts the same way as PUT. The exchanges are the same as above. The updated values are carried as payload of POST as shown in Figure 2.

Figure 2: Example of Unreliable Update with No-Response Option Using POST as the Update Method

4.1.2.2. POST Updating through Query String

It may be possible that the backend infrastructure deploys a dedicated database to store the location updates. In such a case, the client can update through a POST by sending a query string in the URI. The query string contains the name/value pairs for each update. No-Response can be used in the same manner as for updating fixed resources. The scenario is depicted in Figure 3.

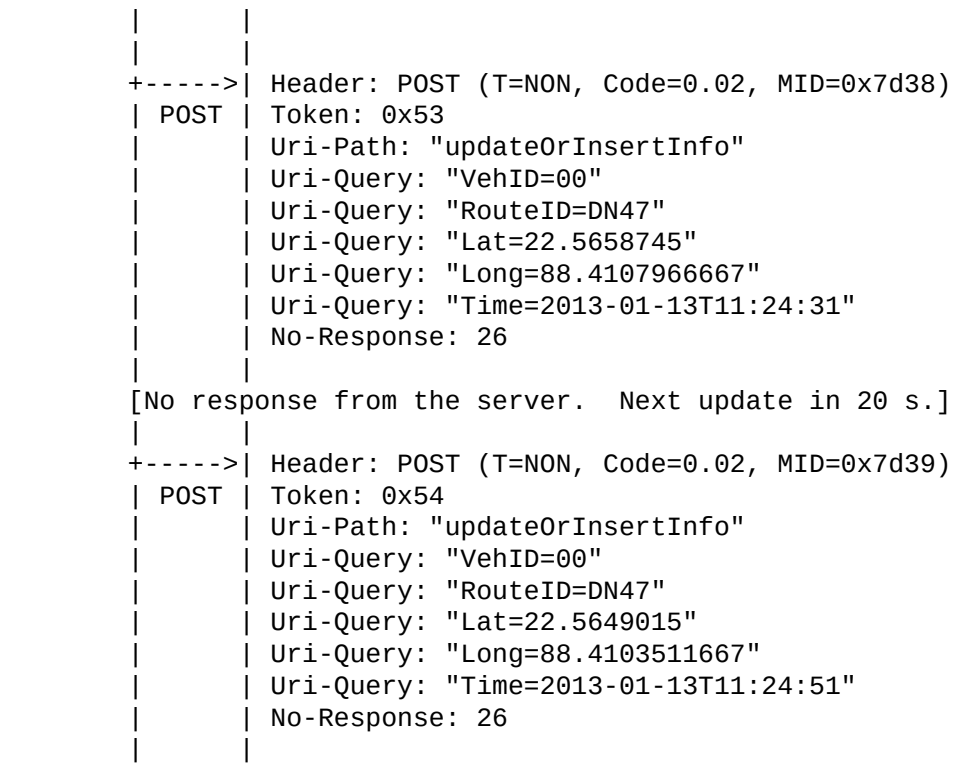

Figure 3: Example of Unreliable Update with No-Response Option Using POST with a Query String to Insert Update Information into the Backend Database

4.2. Multicasting Actuation Command from a Handheld Device to a Group of Appliances

A handheld device (e.g., a smart phone) may be programmed to act as an IP-enabled switch to remotely operate on one or more IP-enabled appliances. For example, a multicast request to switch on/off all the lights of a building can be sent. In this case, the IP switch

application can use the No-Response option in a NON request message to reduce the traffic generated due to simultaneous CoAP responses from all the lights.

Thus, No-Response helps in reducing overall communication cost and the probability of network congestion in this case.

4.2.1. Using Granular Response Suppression

The IP switch application may optionally use granular response suppression such that the error responses are not suppressed. In that case, the lights that could not execute the request would respond back and be readily identified. Thus, explicit suppression of option classes by the multicast client may be useful to debug the network and the application.

5. IANA Considerations

The IANA had previously assigned number 284 to this option in the "CoAP Option Numbers" registry. IANA has updated this as shown below:

Number	Name	Reference
258	No-Response	RFC 7967

6. Security Considerations

The No-Response option defined in this document presents no security considerations beyond those in Section 11 of the base CoAP specification [RFC7252].

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://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, <http://www.rfc-editor.org/info/rfc7252>.

7.2. Informative References

[CoAP-ADAPT]
Bandyopadhyay, S., Bhattacharyya, A., and A. Pal, "Adapting protocol characteristics of CoAP using sensed indication for vehicular analytics", 11th ACM Conference on Embedded Networked Sensor Systems (SenSys '13), DOI 10.1145/2517351.2517422, November 2013.

[CoAP-PUBSUB]
Koster, M., Keranen, A., and J. Jimenez, "Publish-Subscribe Broker for the Constrained Application Protocol (CoAP)", Work in Progress, draft-koster-core-coap-pubsub-05, July 2016.

[CoAP-TCP-TLS]
Bormann, C., Lemay, S., Tschofenig, H., Hartke, K., Silverajan, B., and B. Raymor, Ed., "CoAP (Constrained Application Protocol) over TCP, TLS, and WebSockets", Work in Progress, draft-ietf-core-coap-tcp-tls-04, August 2016.

[HTTP-to-CoAP]
Castellani, A., Loreto, S., Rahman, A., Fossati, T., and E. Dijk, "Guidelines for HTTP-to-CoAP Mapping Implementations", Work in Progress, draft-ietf-core-http-mapping-13, July 2016.

[ITS-LIGHT]
Bhattacharyya, A., Bandyopadhyay, S., and A. Pal, "ITS-light: Adaptive lightweight scheme to resource optimize intelligent transportation tracking system (ITS) - Customizing CoAP for opportunistic optimization", 10th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous 2013), DOI 10.1007/978-3-319-11569-6_58, December 2013.

[RFC5405] Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers", BCP 145, RFC 5405, DOI 10.17487/RFC5405, November 2008, <http://www.rfc-editor.org/info/rfc5405>.

[RFC7390] Rahman, A., Ed., and E. Dijk, Ed., "Group Communication for the Constrained Application Protocol (CoAP)", RFC 7390, DOI 10.17487/RFC7390, October 2014, <http://www.rfc-editor.org/info/rfc7390>.

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

Acknowledgments

Thanks to Carsten Bormann, Matthias Kovatsch, Esko Dijk, Bert Greevenbosch, Akbar Rahman, and Klaus Hartke for their valuable input.

Authors' Addresses

Abhijan Bhattacharyya
Tata Consultancy Services Ltd.
Kolkata, India

Email: abhijan.bhattacharyya@tcs.com

Soma Bandyopadhyay
Tata Consultancy Services Ltd.
Kolkata, India

Email: soma.bandyopadhyay@tcs.com

Arpan Pal
Tata Consultancy Services Ltd.
Kolkata, India

Email: arpan.pal@tcs.com

Tulika Bose
Tata Consultancy Services Ltd.
Kolkata, India

Email: tulika.bose@tcs.com