TOTSCO

TOTSCo Bulletin No 67
Date: 1 August 2024
Subject: Revised Guidance on receiving HTTP Messages from the TOTSCo Hub

This bulletin has been written by the IPG to provide additional guidance on handling receipt of messages from the TOTSCO hub, following issues experienced in testing & trials.

This guidance was originally issued as Bulletin 66. Feedback was received from industry that the advice in that bulletin could be interpreted as a change to the API specification. This had never been the intent of the authors, so the IPG agreed to review the advice and completely re-write and re-issue the bulletin. **For the avoidance of any doubt, this revised bulletin replaces Bulletin 66.**

Disclaimer: *This bulletin has been produced in good faith by industry participants, facilitated by TOTSCo, to assist other industry participants. It is not legal or regulatory advice nor approved and/or endorsed by Ofcom, ICO and/or TOTSCo. Communications providers may not rely on its contents and are responsible for their own regulatory compliance.*

TOTSCo
August 2024

Issue 1: Slow response to messages by RCPs is causing TOTSCo Hub to retry, sometimes leading to duplicate messages

As part of the implementation specifications for the TOTSCo Hub, the letterbox API specification lays out a set of expectations for processing and acceptance of a message when delivered to a CP by the TOTSCo Hub. This is a mirror specification, as the TOTSCo Hub is also required to follow the same standards.

Alongside the API specification, the One Touch Switch Message Delivery Policies v1.0 document specifies the SLAs for processing messages, and this is the area that is causing issues. The document states a connection timeout after 1 second, and a response timeout after 3 seconds. When OAuth2 is used for authentication, this has been implemented as a "GLOBAL_TIMEOUT" of 2 seconds for receipt of http response to the POST of the message (after getting an OAuth2 token). Other authentication mechanisms have a slightly longer timeout.

What does this mean? If the TOTSCo Hub does not see a valid http response to a message sent to an RCP after 2 seconds, it will regard the message as undelivered, and will retry delivery. However, in some cases, the RCP has in fact received and is processing the message.

What does the issue look like for RCPs? As an example:

- RCP1 sends a match request via the TOTSCo Hub to RCP2.
- RCP2 is slow at generating a http 202 response back to the Hub, but has received the message and generates a match response.
- However the Hub sees a timeout, and retries delivery of the message to RCP2. RCP2 sees a second match request, and generates a second response.

- RCP1 wonders why one match request got two match responses, perhaps with different Switch Order Reference (SOR) values, and wonders which to use.

This behaviour has been exhibited by multiple CPs during testing, for a variety of reasons including system start up time, intermittent connectivity etc.

Background to message delivery design

The delivery policies detail the response timeout period that the Hub will wait for a CP to respond before retrying delivery, with applicable retry policies depending on the type of message sent.

The API specification also describes how the POST process is independent of the message content, and only the envelope is relevant to the onward deliver of that message.

Some background here is relevant to the design process. The JSON envelope message architecture is intended to be an open message standard for inter CP communications regardless of the industry process or message content. The Hub design, and associated letterbox API specifications, were intended to be lightweight data processors, allowing messages to be moved from one CP to another rapidly without "over processing". This is further emphasized with the asynchronous message protocols where generation of the message response can be performed out of band and subject to SLAs appropriate to the individual process, and not encumber the API itself with the data processing in real time.

In practice, it is understood that some CPs have gone beyond a basic level of validation on receipt of a message from the TOTSCo Hub, to the extent that the whole message content is being validated, and perhaps even processing the transaction contained within the message.

This over-processing of the message on receipt has resulted in situations where the response to the message arrives after the GLOBAL_TIMEOUT of 2 seconds, with consequences as described above.

Guidance on letterbox API implementation to ensure fast response

For any message sent to a CP using the letterbox API interface, here are a list of suggested do's and don'ts that can be used to optimise processing time, and to ensure the solution is fit for further message applications in future.

<u>Do's</u>

1. Schema validation: It is reasonable that the schema of the JSON document can be validated on receipt of a message on the letterbox API. TOTSCo will only validate the schema up to the envelope, and not beyond, although they will validate that the message as a whole is a validly constructed JSON document. CPs may include extended schema validation, aligned with the swagger definitions published in support of the OTS process. This may be extended in the future with support for other processes as well.
2. Envelope validation: The intent of the letterbox API was that only the envelope should be validated on receipt of the message to ensure it can be responded to. TOTSCo performs this on behalf of any messages it receives and rejects the message at the letterbox API if it knows the message cannot be onward delivered, or if the sender is not authorised to send the message (e.g. misuse of source RCPID).
3. Target a processing time of under 200ms for a response. This is well below the response timeout SLA, and provides contingencies for lags or delays in internal processes, such as writing onward to queues or databases where real processing is performed.
4. Each CP should understand their own systems and ensure they are started up and available for processing transactions as soon as the socket listener interfaces is opened and is listening

for incoming messages. For example, ensuring all database connections are established before receiving messages. This will help to ensure messages are processed within the response timeout described above.

Don'ts

1. Whole message validation: If the envelope is well formed, and the content valid, then any errors resulting from the body of the message can be dealt with either in asynchronous response messages if a request message contained invalid content (e.g. surname is missing), or by discarding the message if the message did not expect a reply, for example an acknowledgement contained an unrecognised SOR.
2. Don't generate a synchronous rejection for any data validation for which there is a documented asynchronous fault code (e.g. 1101 Missing or incomplete address).

Although it is recognised that asynchronous messaging does have weaknesses in some situations (for example being unable to respond to a message where no reply is expected), it is expected that extensive testing between CPs will identify faults, and that such real time validations should not be necessary at run-time.

Handling Duplicate Message

Even with a lightweight Letterbox API, there can be situations beyond your control where you may not be able to process the message sent to you within the response timeout SLA and respond to the hub before it closes its connection.

RCPs will need to consider their own methods of handling retried messages which may appear to them to be duplications of the original message.

Recommendations

Consider your existing letterbox API designs and try to ensure they are as lightweight as possible. Long response times have knock on impacts to the TOTSCo Hub as well, so it is in every body's interest to keep the end-to-end delivery mechanisms as light as possible.

Issue 2: Uniqueness of correlation id

The TOTSCo API Specification has the following definition of a source correlation id:

> In a source element, the correlationID must always be provided, the format can be anything the originator chooses to support their messaging process but should be sufficiently unique to allow correlation of response with request over a reasonable period.

The IPG understands that across the base of RCPs, both of the following extremes have been implemented:

1. As LRCP, strict duplicate detection of the combination of correlationID and RCPID.
2. As GRCP, re-using the same correlationID for all messages related to a switch for a single customer.

What does this mean? If a sending RCP re-uses the same correlationID to a destination RCP which has implemented strict duplication detection, subsequent messages will be discarded silently by the receiving RCP (since they consider them to be duplicate messages, e.g. caused by the delivery timeouts explained in issue 1 above), with no rejection messages sent back to the sender.

What does the issue look like for RCPs? As an example:

- RCP1 generates an internal correlation id for each sales order journey (fresh value for each customer handled by a sales agent) and planned to use the same value in the match request to the LRCP.
- Match request to RCP2 fails to match; sales agent adds account number and tries again.
- RCP2 does not reply as the correlation id is repeated from the first match attempt.
- RCP1 sees a failure by RCP2 to reply to their 2nd match attempt.

Recommendations
The IPG recommendations are two-fold:

Do not perform duplicate message detection based only on correlationID
Performing duplicate message detection based only on correlationID (even along with RCPID) could result in erroneous discarding of messages that the sending RCP regards as valid, based on the information in the Industry Process[1] (which is referenced from Ofcom's GC C7.18) and TOTSCo API Specification[2].

Generate a unique correlationID for each message
If as originator of a message, you generate a unique correlationID for each message, you are guaranteed to not fall foul of any duplicate message detection.

There are several ways to achieve this, including:

- Generate a UUID to act as correlationID just before you generate the overall message.
- If you wish to use a single value across multiple messages for a single customer, append a hyphen or colon and sequence number to act as your correlationID. The latest Swagger spec defines a max length of 256 characters.

Issue 3: Slow response to messages by RCPs will affect your throughput.

As well as a slow response causing multiple delivery responses it also impacts a CP's individual performance. The One Touch Switch Message Delivery Policies v1.0 document explains that the TOTSCo Hub operates by delivering messages in sequential order, and each CP has two queues: one for match requests, and one for other message types. The sequential ordering (first-in first-out) was established to ensure that race conditions would not form within business processes in the event of an outage.

What does this mean? During a peak period where you may receive a number of match requests from other CPs, a slower response may mean that a queue of messages could form at the Hub waiting to be sent to you and therefore you may not accept match requests inside the 30 second match request delivery window.

What does the issue look like? As an example:

- RCP1 receives 15 Switch Match Requests from RCP2 following a recommendation of RCP2 during a TV programme.
- RCP3 also sends a Switch Match Request shortly after.
- RCP1 responds individually to the Switch Match Requests with an average response rate of 2 seconds – a total of 30 seconds to process the 15 requests.

[1] OTS Industry Process v4.3. §5.10 last paragraph permits re-use of correlation identifier. Note that §22 Appendix 4 suggests duplicate correlationID detections at the Hub (which has not been implemented).
[2] TOTSCO API Specification v1.1. §2.1.5, definition of JSON element correlationID.

- RCP3's message times out at the TOTSCO Hub (as it could not be delivered inside 30 seconds) and a Message Delivery Failure message is received by RCP3.

Guidance for throughput considerations:
The table below should be used as a guide for CPs to understand the total number of match request messages that can queue based on their average response time and assure themselves that they can meet demand.

Average response time (milliseconds)	**Total messages that can be processed in 30 seconds**	**Queued match requests that are at risk of Hub reporting message delivery failure**
100	300	301+
200	150	151+
It is recommended that RCPs aim to respond in under 200 ms. The rows below indicate the potential impacts of slower response times.		
500	60	61+
1000	30	31+
2000	15	16+
3000	10	11+