Chaincode Event Replay

Version 1.2 (git/design-doc/blockchain/commons/events/), 2024-06-03

Table of content

Introduction

The document describes how to replay the past chaincode events when needed and possible approaches to handle the double processing of events.

Summary

There can be scenarios where event subscribed callBack client have missed the event and processing of the particular event due to reasons like auth issue, service unavailability, processing error etc. Most of these cases should be handled by OBP reliable event delivery by setting the proper value to the attribute `maxCallbackRetry` during the subscription. Events will be retried starting with 1 second internal, following exponential backoff policy until it reaches 120 seconds (1, 2, 4, 8 ... 120, 120, 120) and continue to retry every 120 seconds until event is delivered successfully or `maxCallbackRetry` count is exhausted

In the cases, where `maxCallbackRetry` is exhausted, or HTTP response code returned from the client is non retriable, event replay methods can be considered with caution.

Event Replay

NOTE

There are different type of event subscriptions (transaction, block, filteredblock, chaincode). In this document, we focus only on `chaincode` type.

Event Replay Subscription

To replay the past events, we need to create a new subscription by passing 2 extra attributes.

NOTE

We cannot alter the existing subscription

```
{
    "type": "chaincode",
    "callbackURL": "https://callback.url",
    "chaincode": "chaincode_name",
    "expires": "1h",
    "event": ".*_TMAID",
    "seek":"from",
    "block":2730,
    "maxCallbackRetry": 15
}
```

1. `seek` - Param indicates which Blocks to be re-played. Valid options for this attributes are listed below
    a. oldest: delivers all blocks from the oldest block

b. newest: delivers the newest block

c. from: delivers from the block number specified for the block parameter.

2. `block` - This attribute is valid when the attribute value of`seek` is `from`

Most relevant attributes for us are

- `seek:"from"`
- `block: integer representing the starting block for event replay`
  In above sample subscription payload, events would be delivered starting from the block 2730, inclusive.

Similarly, for private events,

```
{
    "type": "privateChaincode",
    "callbackURL": "https://callback.url",
    "chaincode": "chaincode_name",
    "expires": "1h",
    "event": ".*",
    "seek":"from",
    "block":2730,
    "maxCallbackRetry": 15,
    "role":"09456f24-a42f-4d86-8984-81ac936ef2e8"
}
```

NOTE

Unlike public events, private events can only be replayed if the subscriber is actually the recipient of chaincode events to be replayed.

Few things to consider

1. It is recommended to keep one subscription per callBack client (no benefit of having multiple subscriptions for the same callBack client)
2. There is no restriction to re-use the same callBackURL for the replay subscription. When the same callBackURL is used, both the past events to be replayed and the new events generated would be delivered to the same callBack client
3. It is client's responsibility to implement the logic to avoid double processing of the same event twice if not intended
4. In case the new callBackURL is used, all the past events will be delivered to new callBack client
5. Multiple blockchain transactions can be part of the same Block which in turn indicates that multiple events may or may not be part of the same `BlockNumber` so to identify a unique event, we must consider `BlockNumber` and `TxnID` present in the event payload.

Handling of event double processing

1. Example scenario of events delivery when replay is in effect

Sequence	BlockNumber,TxnID	Scenario
1	100,0	Subscribed
2	101,1	Normal flow
3	103,1	Normal flow
4	106,1	Normal flow
5	106,2	Same block different TxnID to be processed
6	125,1	Issue with callBack client and events are dropped for some blocks [106,107,108,109]. Block 125 is processed after issue recovery
7	106,1	Existing subscription is deleted. A new replay subscription is created from BlockNumber=106. This is a scenario where we are handling the already processed event - double processing
8	106,2	Double processing, replay event
9	106,3	Normal flow, replay event
10	107,1	Normal flow, replay event
11	108,1	Normal flow, replay event
12	109,1	Normal flow, replay event
13	125,1	Double processing, replay event
14	127,1	Latest event block is caught up

1. From the above example scenario, it is evident that callBackClient should have the capability to handle double processing of same events when events are replayed

Approach 1

- Design the callBack client to persist `BlockNumber` and `TxnID` in the event
- Delete the old subscription which is not functioning anymore.
- Create the new subscription with `seek` and `from` attributes by passing the `BlockNumber` from where the events to be replayed

- For every new event received,

```
Fetch the `BlockNumber` and `TxnID` present in the event and compare the values
with persisted list of `BlockNumber` and `TxnID`
if Found
    Skip processing of the event
if Not Found
    Process the event
```

Approach 2

- Design the callBack client process the events in an idempotent manner. Meaning, re-processing of the same event doesn't have any implication on callBackClient/down systems.
- Delete the old subscription which is not functioning anymore.
- Create the new subscription with `seek` and `from` attributes by passing the `BlockNumber` from where the events to be replayed
- For every event received,

```
Process the event
```