The Pub/Sub subscriber data APIs, such as pull, provide limited access to message data. Normally, acknowledged messages are inaccessible to subscribers of a given subscription. In addition, subscriber clients must process every message in a subscription even if only a subset is needed.

The **Seek** feature extends subscriber functionality by allowing you to alter the acknowledgement state of messages in bulk. For example, you can replay previously acknowledged messages or purge messages in bulk. In addition, you can copy the state of one subscription to another by using seek in combination with a **Snapshot**, introduced as part of the seek feature. Recovering acknowledged messages generally requires the source subscription to be configured in advance and results in additional storage fees.

These features are described below. However, you can look at the quickstart (/pubsub/docs/replay-qs) for a working example.

Seeking to a time marks every message received by Pub/Sub before the time as acknowledged, and all messages received after the time as unacknowledged. You can seek to a time in the future to purge messages. To replay and reprocess previously acknowledged messages, seek to a prior time. The message publication time is generated by the Pub/Sub servers (see `publishTime` (/pubsub/docs/reference/rest/v1/PubsubMessage) in the API reference). This approach is imprecise due to:

- Possible clock skew among Pub/Sub servers.
- The fact that Pub/Sub has to work with the arrival time of the publish request rather than when an event occurred in the source system.

To seek to a prior time, you must first configure your subscription to retain acknowledged messages:

- An acknowledged message is retained in a subscription only if the subscription's `retain_acked_messages` (/pubsub/docs/reference/rest/v1/projects.subscriptions/create#body.request_body.FIELDS.retain_acked_messages) property is set to true (the default is false), for up to `message_retention_duration`

(/pubsub/docs/reference/rest/v1/projects.subscriptions/create#body.request_body.FIELDS.message_retention_duration)
after it is published (the default is 7 days). Acknowledged messages are retained only if they are acknowledged after the subscription's `retain_acked_messages` (/pubsub/docs/reference/rest/v1/projects.subscriptions/create#body.request_body.FIELDS.retain_acked_messages)
is set to `true`.

- An unacknowledged message is retained in a subscription for up to `message_retention_duration` (/pubsub/docs/reference/rest/v1/projects.subscriptions/create#body.request_body.FIELDS.message_retention_duration)
after it is published (the default is 7 days).

- Both the `retain_acked_messages` (/pubsub/docs/reference/rest/v1/projects.subscriptions/create#body.request_body.FIELDS.retain_acked_messages)
and the `message_retention_duration` (/pubsub/docs/reference/rest/v1/projects.subscriptions/create#body.request_body.FIELDS.message_retention_duration)
properties of a subscription can be specified at subscription creation, or updated for an existing subscription.

When you modify either the message retention duration or subscription expiration policy, the expiration duration mus
a value greater than the message retention duration. The defaults are 7 and 31 days, respectively.

The snapshot feature allows you to capture the message acknowledgment state of a subscription. Once a snapshot is created, it retains:

- All messages that were unacknowledged in the source subscription at the time of the snapshot's creation.
- Any messages published to the topic thereafter.

You can replay these unacknowledged messages by using a snapshot to seek to any of the topic's subscriptions.

Unlike with seeking to a time, you don't need to perform any special subscription configuration to seek to a snapshot. You just need to create the snapshot ahead of time. For example, you might create a snapshot when deploying new subscriber code, in case you need to recover from unexpected or erroneous acknowledgements.

Snapshots expire and are deleted in the following cases (whichever comes first):

- The snapshot reaches a lifespan of seven days.
- The oldest unacknowledged message in the snapshot exceeds the `message retention duration` (/pubsub/docs/reference/rest/v1/projects.subscriptions/create#body.request_body.FIELDS.message_retention_duration) .

For example, consider a snapshot of a subscription with a backlog where the oldest unacknowledged message is a day old. The snapshot expires after six days, rather than seven. This timeline is necessary for snapshots to offer strong at-least-once delivery guarantees.

Seek operations are strictly consistent in regard to message delivery guarantees. This means that any message that is to become unacknowledged based on the seek condition is guaranteed to be eventually delivered at least once after the seek operation succeeds. However, delivered messages do not instantly become consistent with the seek operation. So a message that was published before the seek timestamp or that is acknowledged in a snapshot might be delivered after the seek operation. In a sense, message delivery operates as an eventually consistent system with respect to the seek operation: it might take as long as a minute for the operation to take full effect.

- **Update subscriber code safely.** A concern with deploying new subscriber code is that the new executable may erroneously acknowledge messages, leading to message loss. Incorporating snapshots into your deployment process gives you a way to recover from bugs in new subscriber code.
- **Recover from unexpected subscriber problems.** In cases where subscriber problems are not associated with a specific deployment event, you might not have a relevant snapshot. In this case, if you have enabled *acknowledged message retention* for a subscription, seeking to a past time gives you a way to recover from the error.
- **Save processing time and cost.** Perform a bulk acknowledgement on a large backlog of messages that are no longer relevant.

- **Test subscriber code on known data.** When testing subscriber code for performance and consistency, it is useful to use the same data in every run. Snapshots enable consistent data with strong semantics. In addition, snapshots can be applied to any subscription on a given topic, including a newly created one.

You can use Pub/Sub with Dataflow (/dataflow/docs/). However, we do not recommend direct access to Pub/Sub Seek from within a running Dataflow pipeline. For the recommended workflow, see Using Pub/Sub with Dataflow (/dataflow/docs/guides/using-cloud-pubsub-seek).