Set up request flows

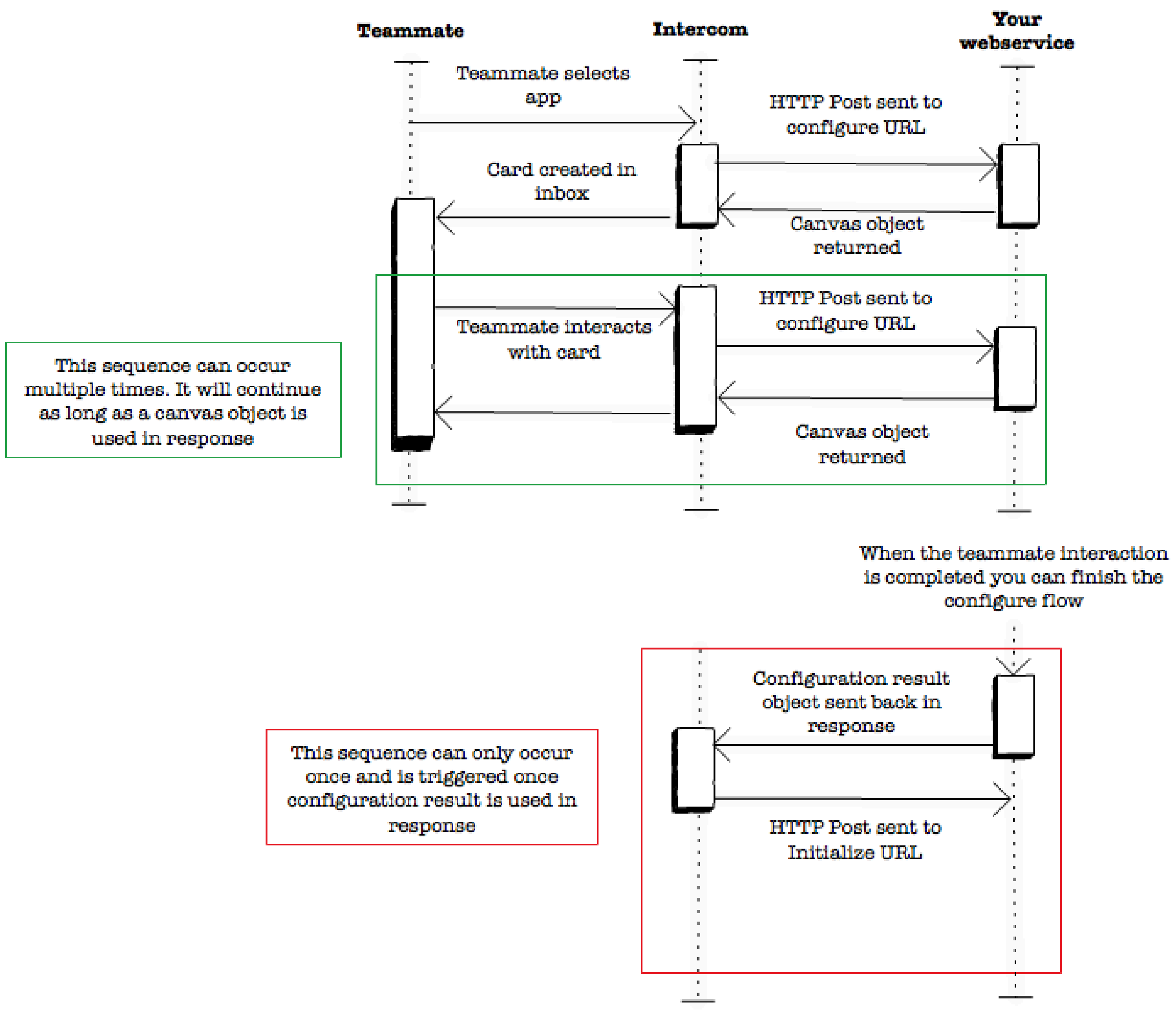
Configure Flow

This is an optional request flow that you can implement if your Messenger App requires configuration before displaying it to an end user. You can implement the `configure-flow` by specifying a `configure_url` when setting up your app within Developer Hub. Then, an HTTP `POST` will be made to that URL when a teammate selects your app.

This request allows you to respond with an updated canvas which the teammate can interact with to help construct the app. Or, you can return a result object containing card creation options which indicates the configuration is now complete.

Once a result object is sent, the collected information will be posted to the `initialize-url` URL.

Example Configure Flow


Teammate
Intercom
Your webservice
Teammate selects app
HTTP Post sent to configure URL
Card created in inbox
Canvas object returned
HTTP Post sent to configure URL
Teammate interacts with card
Canvas object returned
This sequence can occur multiple times. It will continue as long as a canvas object is used in response
When the teammate interaction is completed you can finish the configure flow
Configuration result object sent back in response
HTTP Post sent to Initialize URL
This sequence can only occur once and is triggered once configuration result is used in response

```
# This is the initial POST request which Intercom will send to your configure-card URL.

{
  "workspace_id": /* string id_code of the app using the card */,
  "admin": /* Admin object of admin who is performing configuration */,
  "context": /* Context object */
}

# You can then decide which of the two responses you would like to send; a canvas or res
```

```
# This object will be used to render a configuration UI inside the inserter or home scre

{
  "canvas": /* Canvas object */
}

# Taking an action in this UI will result in a further request to the configure URL with

{
  "workspace_id": /* string id_code of the app using the card */,
  "admin": /* Admin object of admin who is performing configuration */,
  "current_canvas": /* Canvas object */,
  "component_id": /* component_id, component which triggered action */,
  "input_values": {
    "<component_id>" : /* value entered in component */,
    ...
  },
  "context": /* Context object */
}

# As with the initial request, this may be responded to with either another canvas or a
```

```
# To finish configuration the developer responds with a results object indicating the co

{
  "results": {
    /* set of key-value pairs */
  }
}
```

```
# These results will then be posted to the initialize-url as card creation options.
```

Initialize Flow

Teammates can add a card instance of your Messenger App to conversations, to the homescreen, to a Message or to Workflows. When they do, the framework sends a HTTP `POST` to your webservice via the `initialize_url` you provided. This asks you what the initial canvas for your new card should look like.

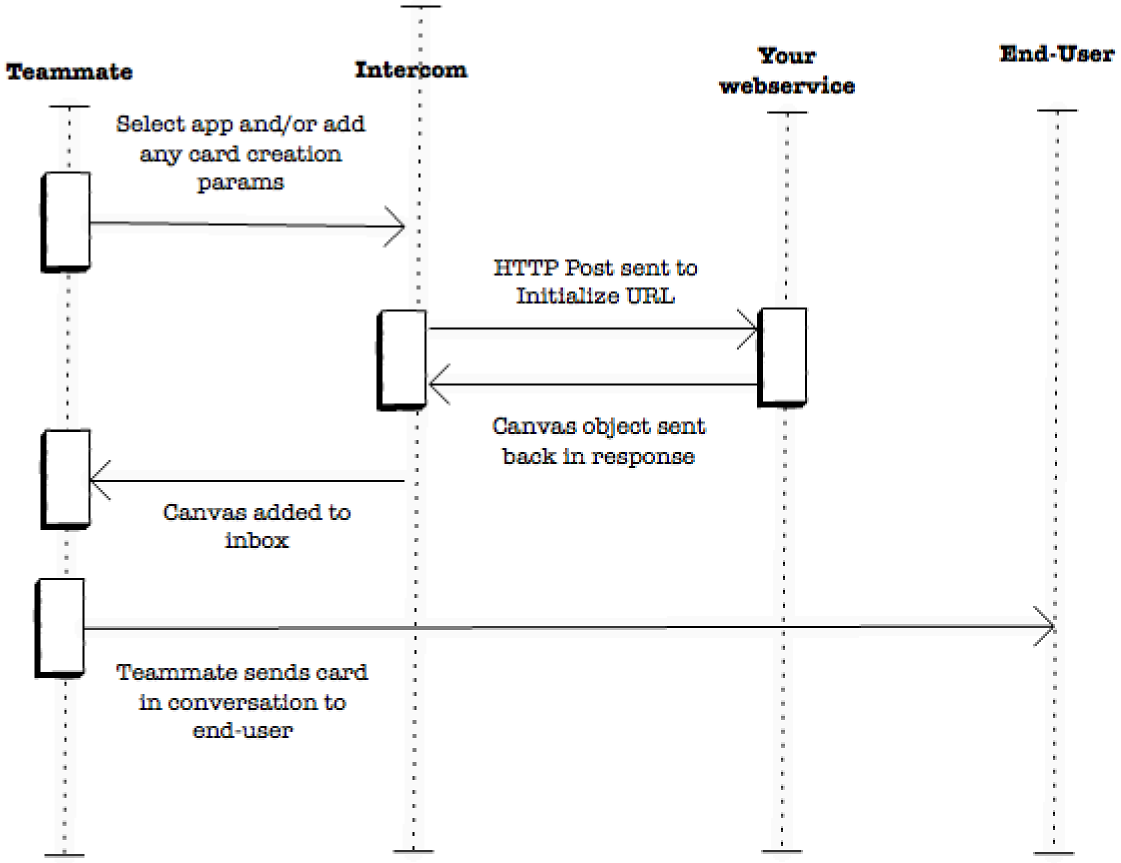
If you have implemented the configure flow then a teammate will have entered some configuration options. These are known as card creation options.

We gather those card creation options from the teammate, and send them to you as part of the payload of the **initialize-card** request. You return back a canvas containing the initial UI for the new card.

What if I have no configure flow?

Remember, the `configure flow` is optional, so if you haven't specified a `configure_url` then the framework will just start with the `initialize flow` when the teammate adds the app to the conversati.

Example Initialize Flow


Teammate
Intercom
Your webservice
End-User
Select app and/or add any card creation params
HTTP Post sent to Initialize URL
Canvas object sent back in response
Canvas added to inbox
Teammate sends card in conversation to end-user

```
# When a card is being added, Intercom POSTs a request to the Messenger app's initialize

{
  "card_creation_options": {
    /* set of key-value pairs */
  },
  "workspace_id": /* string id_code of the app using the card */,
  "context": /* Context object */
}
```

```
# The developer returns a response in the following format.

{
  "canvas": /* Canvas object */
}
```

Submit Flow

When an end user clicks a button or submits text to an input field on a card, the framework fires a HTTP POST to the `submit_url` you have provided. The purpose of this request is to tell you a user took an action on the card, and ask how it should be updated.

This request also gives you a chance to trigger any actions you want in your own code, like adding a user's email to your own database or scheduling a calendar event.

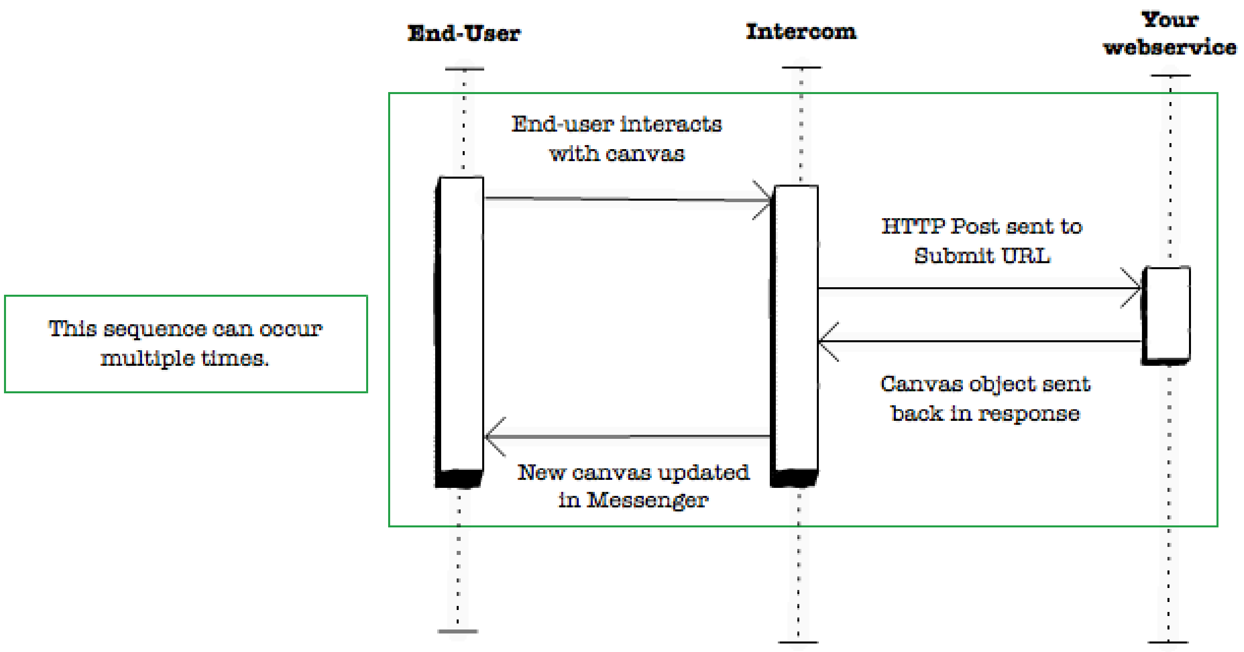
The payload contains information about the end user: the action they took, what values they had filled into each input field, and the card's current canvas. In the response, you return back a new canvas object, containing the updated UI for the card.

App Completion Tracking (optional)

App completion events will help you understand how your customers are using your app and will add more value to it, making it possible for bots to know when to continue with their flows.

You can optionally add an event object to your submit flow response to tell us if your Messenger app has completed its purpose. For example, an email collector app completes when the end-user submits their email address.

Example Submit Flow


End-User
Intercom
Your webservice
End-user interacts with canvas
HTTP Post sent to Submit URL
Canvas object sent back in response
New canvas updated in Messenger
This sequence can occur multiple times.

```
# When an end-user triggers a submit action on a card, Intercom POSTs a request to the m

{
  "workspace_id": /* string id_code of the app using the card */,
  "current_canvas": /* Canvas object */,
  "component_id": /* component_id */,
  "input_values": {
      "<component_id>" : /* value entered in component */,
    ...
  },
  "user": /* User object of end-user who triggered action */,
  "context": {
    "location": /* "conversation"/"home" */,
    "conversation_id": /* conversation_id */ //Only needed if location == conversation
  }
}
```

```
# The developer returns a response in the following format.

{
  "canvas": /* Canvas object */,
  "event": {
    "type": "completed"
  }, // App Completion Tracking (optional)
}

# The card which triggered the request is updated with the new canvas from the response.

# The app completion event is recorded by Intercom.
```

Live Canvas Flow

The `live canvas` is an optional extension of the canvas object which enables developers to provide up-to-date information in a `card`.

It can be implemented by simply returning a `content_url` in a `canvas` object.

Whenever the card containing that `canvas` object is rendered in the Messenger, a `POST` request will be sent to that content URL.

This enables a developer to provide a live (rather than the regular static) version of a `canvas`.

No interaction required

Note that no user interaction is required for the `POST` request to be triggered. It will occur whenever the card is viewed in a conversation. The live canvas can also be rendered in a teammate inbox.

Why would I need a live canvas?

You can usually realize similar functionality to a `live canvas` by requiring a user interaction.

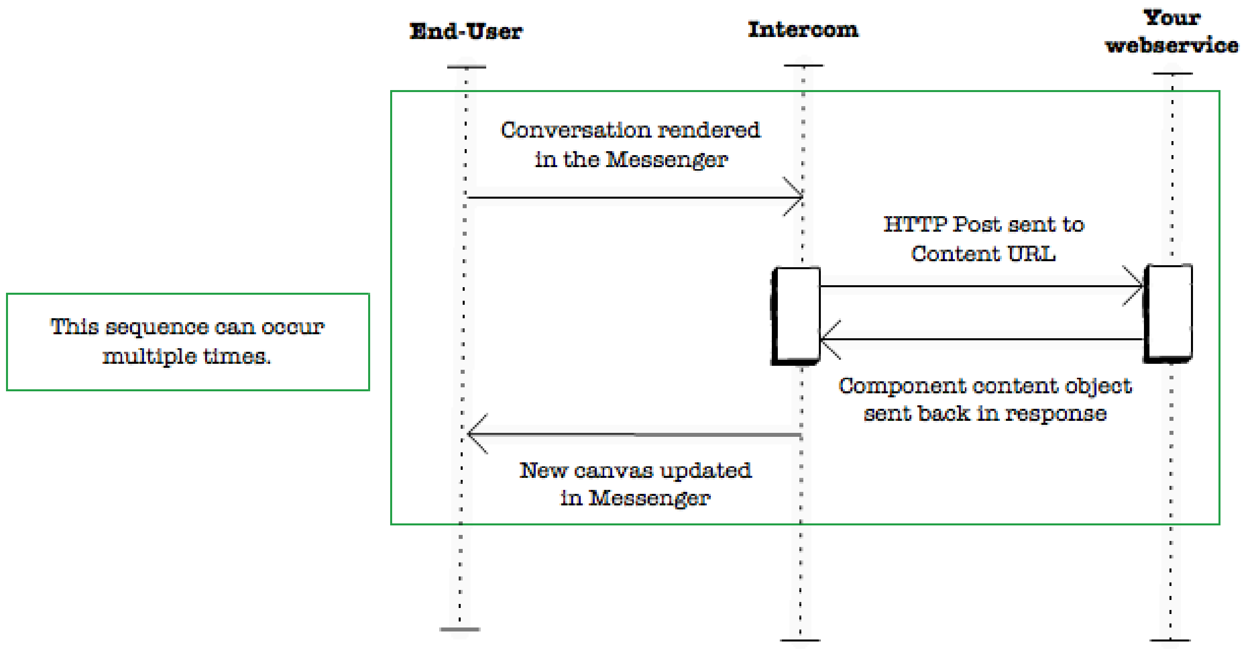
Think of a webinar booking app as an example scenario. You may have a button a user can click on to show them the availability and times for the next webinar.

However, in some cases this is not ideal. If you had an app that showed the current health status of your product or platform for example.

Ideally, you would like this to show the current state without requiring a user to manually trigger a status refresh. You would use a `live canvas` to do this.

As a result, whenever a user opens up their conversation with the health status card they will always get the latest state (which is hopefully green), and they will not need to instigate any action to do so.

Example Live Canvas Flow


End-User
Intercom
Your webservice
This sequence can occur multiple times.
Conversation rendered in the Messenger
HTTP Post sent to Content URL
Component content object sent back in response
New canvas updated in Messenger

Objects in the Live Canvas Flow

Please see the canvas object for more details on how to implement this flow and the expected response from your webservice.