Handling Errors in Asynchronous Processes

Introduction

FOLIO is increasingly adopting asynchronous processes. This has an impact of both the organisation process and the technical process. This is particularly apparent when identifying and compensating for failures.

What are asynchronous processes?

These are processes where the overall outcome is (usually) known after the initiator has already been informed of a (partial) outcome.

I think there are broadly two kinds of asynchronous processes in FOLIO at the moment.

Triggered by an external client

These are processes where the first activity occurs synchronously when a client triggers it (usually by making a HTTP request to the back end gateway) and follow up steps are performed asynchronously after the client has been informed of the initial outcome.

These follow up steps might be performed within the same module or they might be done by other modules.

Importing a MARC source record is an example of this process, as are many of the circulation processes that schedules / sends patron notices after the client has received a response. I believe orders also has processes like this.

These current implementations use a variety of approaches e.g. pub-sub, vert.x event bus, asynchronous futures.

There are a number of processes in development that could / will also fit into this pattern, see links below for design documents for those.

Triggered by the system

These are background processes, usually triggered at a configured interval.

The current implementations typically use the Okapi timer interface where Okapi makes a periodic HTTP request to a configured endpoint. As I understand it, any response is effectively ignored by Okapi.

Loan anonymization or request expiration are examples of this kind of process.

Communication Errors

These are errors that occur during the communication between two modules. These are part of a more general class of technical errors, this document could be expanded to describe as well.

Synchronous Requests

These are requests where a response is sent after the activity has completed, e.g. update an item in storage. Most of FOLIO's HTTP APIs are synchronous requests.

Whilst this is mostly outside of the scope of this document, I believe it is relevant context because (as far as I am aware) FOLIO does not have a standard way for compensating for these either.

There might be some situations which are a mixture, e.g. scheduling patron notices occurs after the main circulation process has finished, yet involved synchronous requests.

Asynchronous Requests

These are requests where the response is sent (usually) before the step or process is completed. This means that errors that occur during further communication (e.g. forwarding) or processing of the request may not be noticed and sufficiently reacted to.

This kind of process is more akin to messaging approaches. FOLIO's current architectural pattern for this is a HTTP based pub-sub that is implemented by mod-pubsub.

Questions

- What should happen if a message cannot be published?
- What should happen if a message cannot be delivered to a subscriber?
- What should happen if a message is lost after it is published and before delivery?
- What should happen if a subscriber cannot interpret a message?
Process Errors

These are errors that could occur in the organisational process. For example, when a source record is imported, what happens if there is a mistake in the mapping that means that an instance cannot be created?

Questions

- What should happen if an asynchronous process cannot be performed due to business logic?
- What should happen if an asynchronous process cannot be completed due to a technical error?

Systemic Questions

How to identify that an error has occurred?

How does the system know that an error has occurred? In many cases, this might be obvious because the client receives a response stating the error. What happens if it occurs after that initial response, should other parts of the system know about the error?

How to compensate for an error?

Can the system automatically compensate for the error e.g. by retrying or rolling back?

What should happen if the system cannot automatically compensate?

Which errors should a user be notified about?

How might a user perform manual compensation (some actions are not allowed in the reference UI)?

Appendices

Links

- Automated patron blocks design document
- Closing loans upon closure of fees and fines design document
- Withdrawing items design document
- Silent failures of loan anonymisation
- Silent failures of sending patron notices

Worked Example