# http-backend

The backend is a server that uses SSE Server Side Events

## Client subscribe to vaultstream/tokens

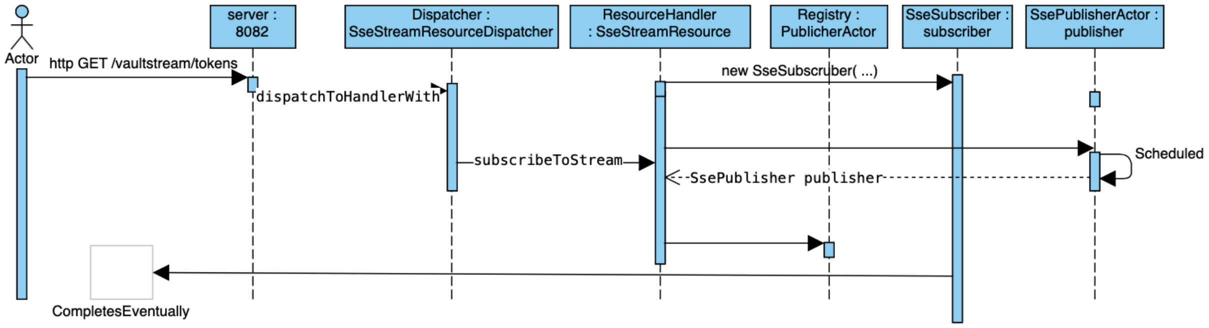

Actor
server : 8082
Dispatcher : SseStreamResourceDispatcher
ResourceHandler : SseStreamResource
Registry : PublicherActor
SseSubscriber : subscriber
SsePublisherActor : publisher
http GET /vaultstream/tokens
dispatchToHandlerWith
new SseSubscruber( ...)
subscribeToStream
Scheduled
SsePublisher publisher
CompletesEventually

Figure: Server side diagram for how backend builds up incommng listening to vaultstream/tokens. This takes place when http-frontend starts up.

streamName is mapped in ***vlingo-http.properties***
in key ***sse.stream.name.tokens*** – here text after last dot is the streamName

## GeneratePrivateToken

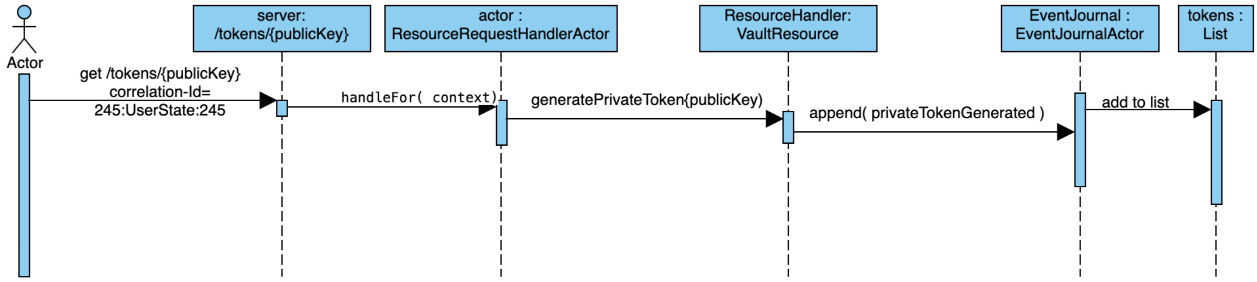

Actor
server: /tokens/{publicKey}
actor : ResourceRequestHandlerActor
ResourceHandler: VaultResource
EventJournal : EventJournalActor
tokens : List
get /tokens/{publicKey} correlation-Id= 245:UserState:245
handleFor( context)
generatePrivateToken{publicKey)
append( privateTokenGenerated )
add to list

Figure: a private token is generated from backend server by REST call to /tokens/{publicKey}

The correlation-Id must be set to something to match back on.

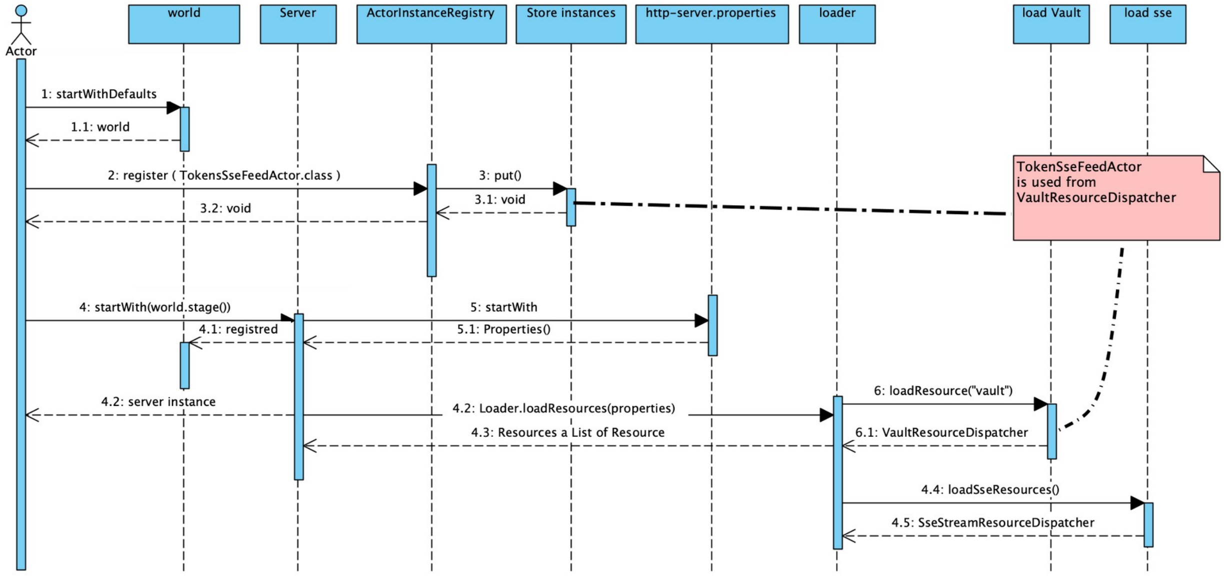
# Define server

To define the http web server (class Server) this sample uses two components from vlingo-http plugin. A vault component and a sse component.

This is basic what happens when server is declared.

1. A factory for TokenSseFeedActor is declated and stored in static factory
2. server reads property file called http-server.properties
3. Load web resources
    a) Load vault
    b) Load sse

A precondition to the server is that TokenSseFeedActor’s factory must be declared because that ActorInstanceRegistry works together with properties in http-server.properties.



Vlingo is designed with basics programming patterns

- Components are immutable
- Components are added with using a fluent configuration.
- World contains stages that contains actors.
- World is ***NOT*** immutable – actors are born and actors are retired and taken out.
- World can be terminated – and a new world can be started. This helps testing here each test in a test case can have it’s own world.
- Actors are born, updated and removed – actors are NOT immutable.

These patterns are in progress and some older code does not 100% live up to all this.

# Improvements

In general nothing should be ”hidden”

Fluent Building World