Eiffel-Collector

The CI Systems best friend for collecting Eiffel Event and trigger CI activities

Tobias Åkervik
Ericsson

Overview

- What is Eiffel-Collector?
- Eiffel-Collector idea/project presented last Eiffel-Summit 2023
- What happened after Eiffel-Summit 2023
- Some short diffs between Eiffel-Intelligence (EI) and Eiffel-Collector (EC)
- How does Eiffel-Collector work?
- Demo
- Q&A

What is Eiffel-Collector?

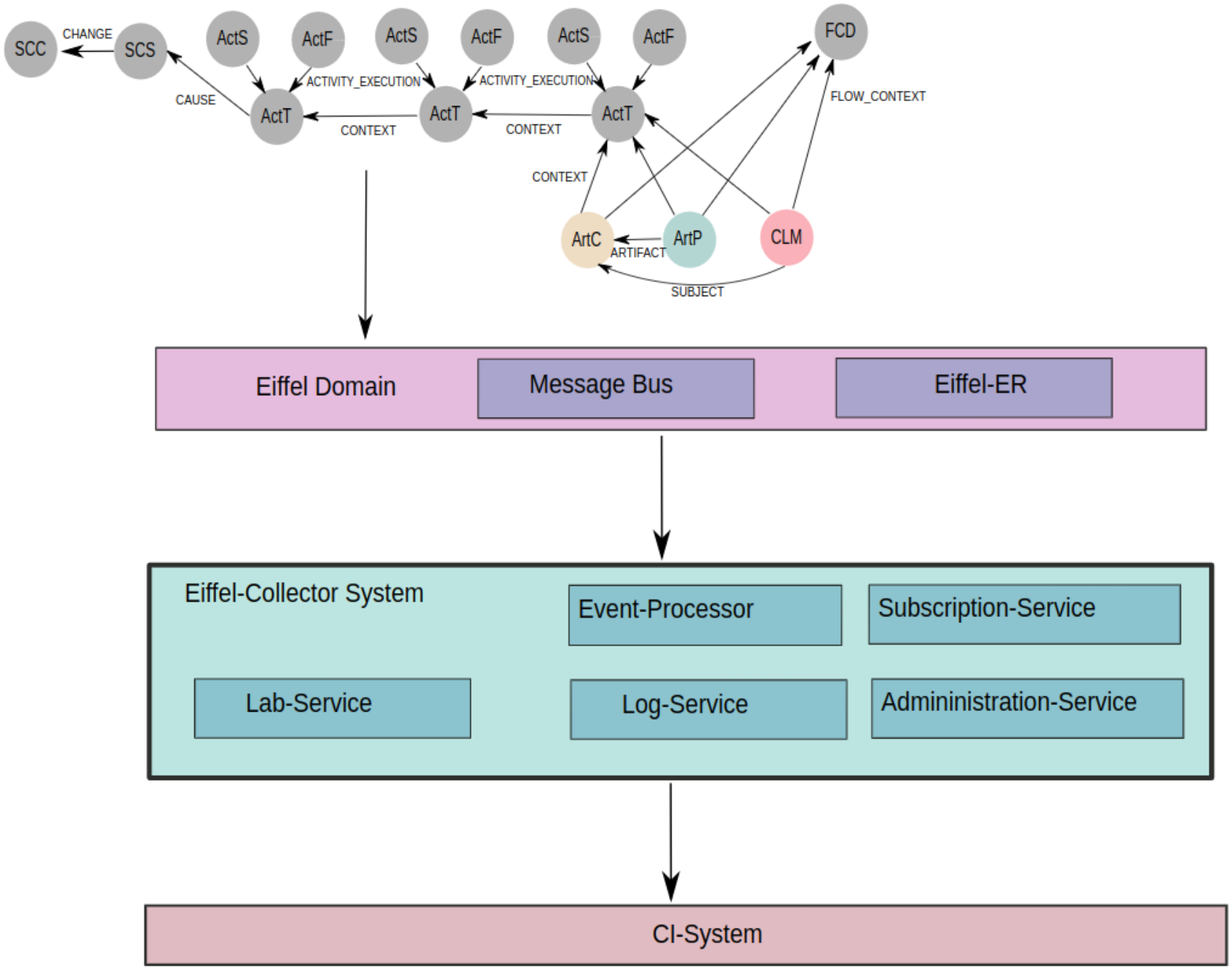
- Eiffel Collector (EC) system is a collection of microservices that consumes Eiffel events and supports subscribing to specific data in Eiffel event(s).
- When an Eiffel event is consumed by EC and is matched against one or more subscriptions based on event conditions, a notification is made to the configured receiver in the subscription.
- The receiver is often a CI System that can trigger an activity, e.g a pipeline or a dependency update (a use-case explained by Emil Bäckmark in an earlier presentation)

Eiffel-Collector project

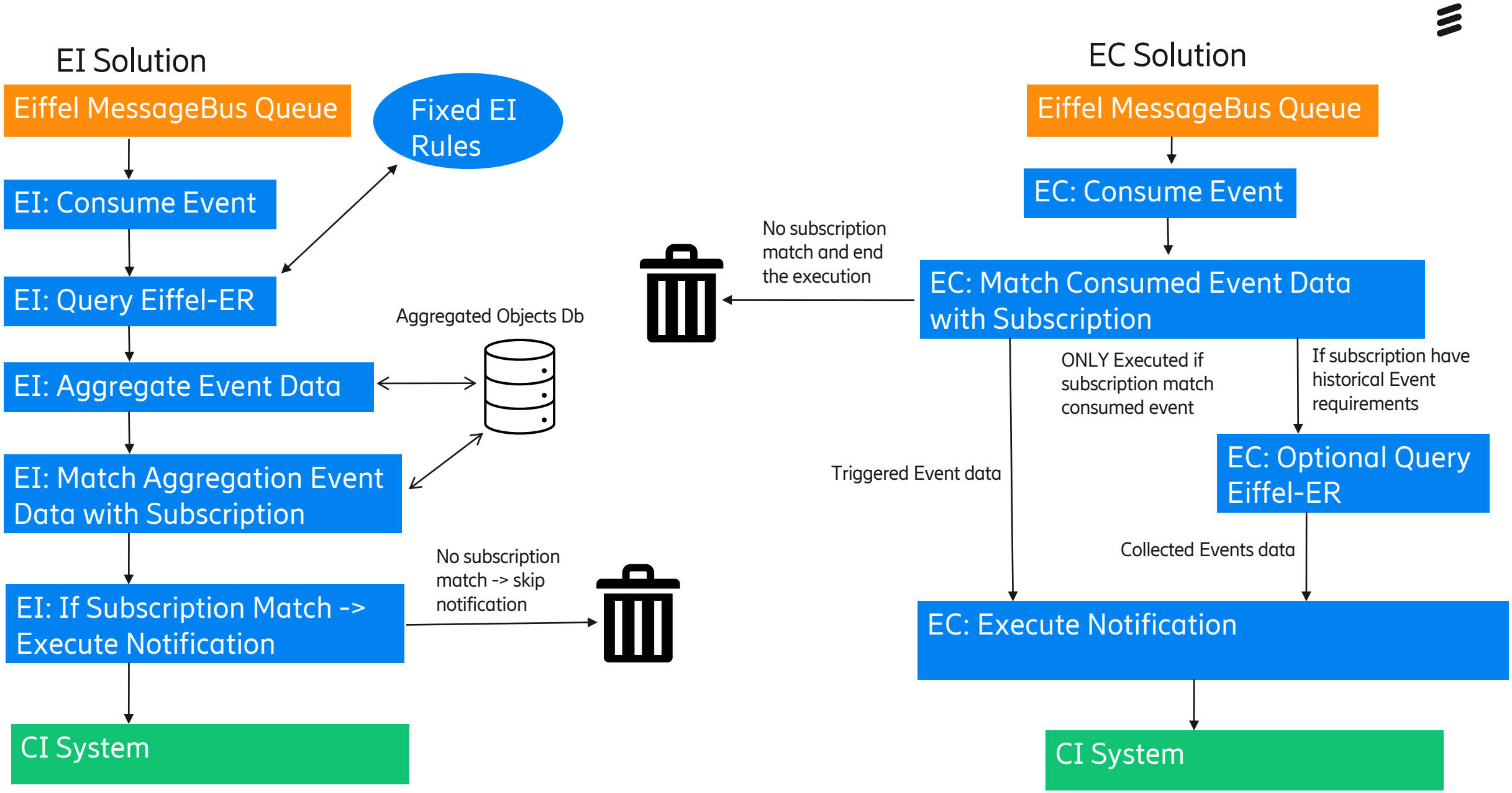
- Simple to create subscriptions and event conditions
  - No JMEPaths expressions (Eiffel-Intelligence "EI" solution) -> More commonly known JsonPaths and regular expressions in event conditions.
- No fixed rules (EI solution) of which events that are collected -> define which events to collect in each subscription, a more dynamic solution for different Eiffel event flows.
- Make the Event processing more transparent for the user -> Easy way to check why a Notification never was executed.

What happened after Eiffel-Summit 2023

- Development continue internal at Ericsson.
- Eiffel-Collector PoC state around August/September 2023.
  (Deployment in Docker and Kubernetes)
- September -> November 2023: Presentation in different architecture boards.
- Continue development and productization of Eiffel-Collector work started January/February 2024.
- September 2024 released and deployed in some of development unit CI systems in Ericsson.


SCC
CHANGE
SCS
ActS
ActF
ActS
ActF
ActS
ActF
FCD
CAUSE
ACTIVITY_EXECUTION
ACTIVITY_EXECUTION
FLOW_CONTEXT
ActT
ActT
ActT
CONTEXT
CONTEXT
CONTEXT
ArtC
ARTIFACT
ArtP
CLM
SUBJECT
Eiffel Domain
Message Bus
Eiffel-ER
Eiffel-Collector System
Event-Processor
Subscription-Service
Lab-Service
Log-Service
Admininistration-Service
CI-System

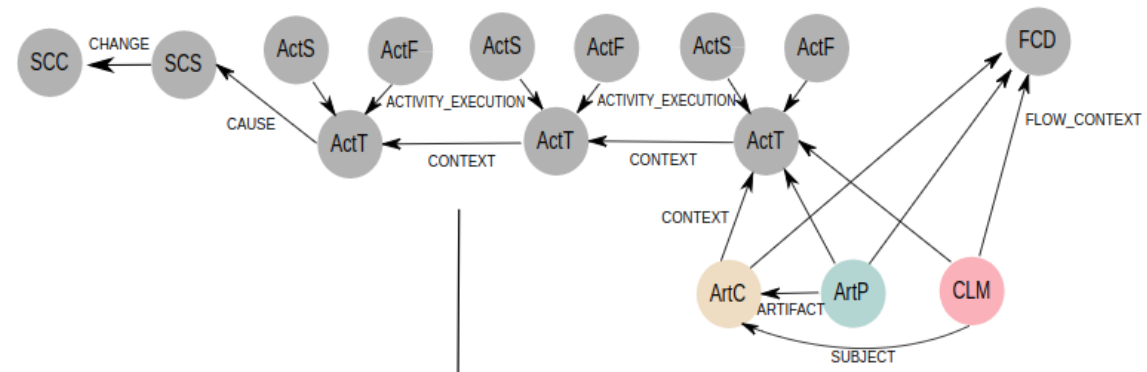
Some short diffs between EI and EC

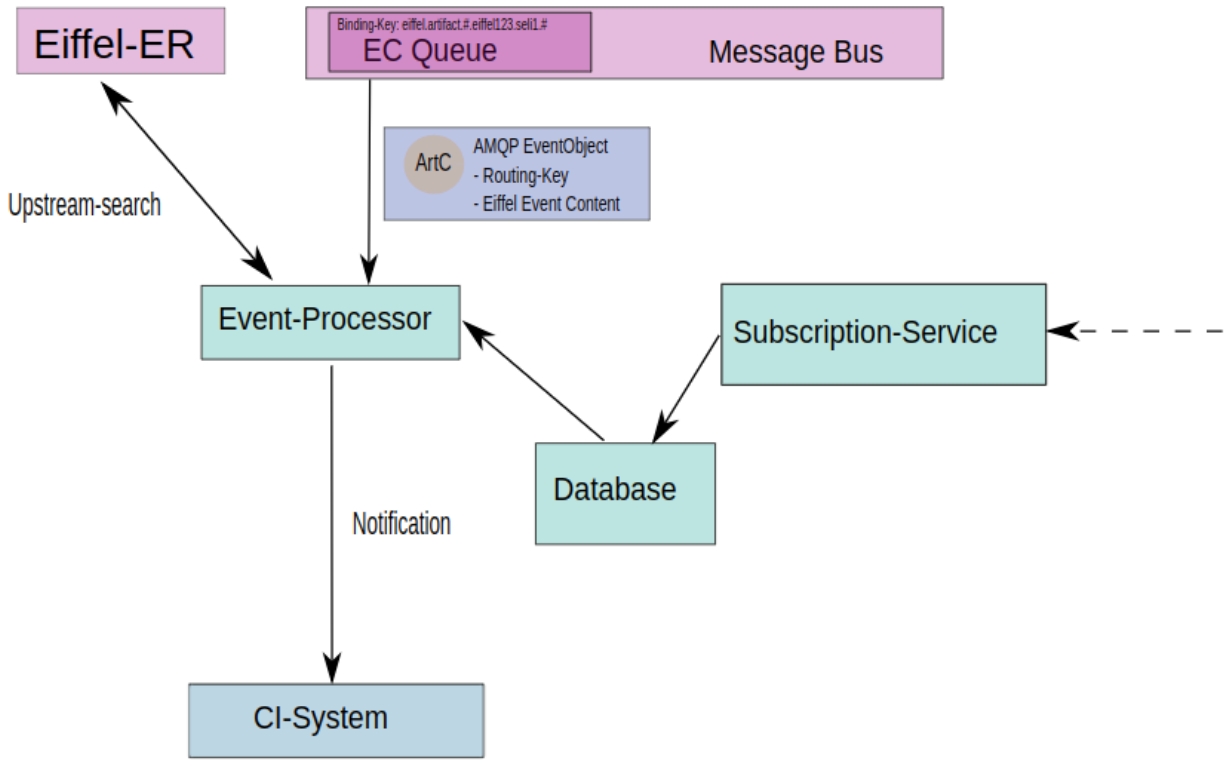


Cont. Some short diffs between EI and EC

- EI is Open Source <-> EC is Ericsson internal product
- EI Aggregate Event Data to Database and wait for one level of downstream events <-> EC collect event data upstream (only) and execute notification without storing data in Database.
- EI have fixed rules (per instance) for which Eiffel events data to collect and cannot be changed during run-time <-> EC have no fixed rules for which Events to collect. Which events data to collect are defined in each subscription. One EC instance can collect any Eiffel events data.
- EC has a log-service where any user can query the event processing logs <-> EI don't have support for this.

How does Eiffel-Collector work?





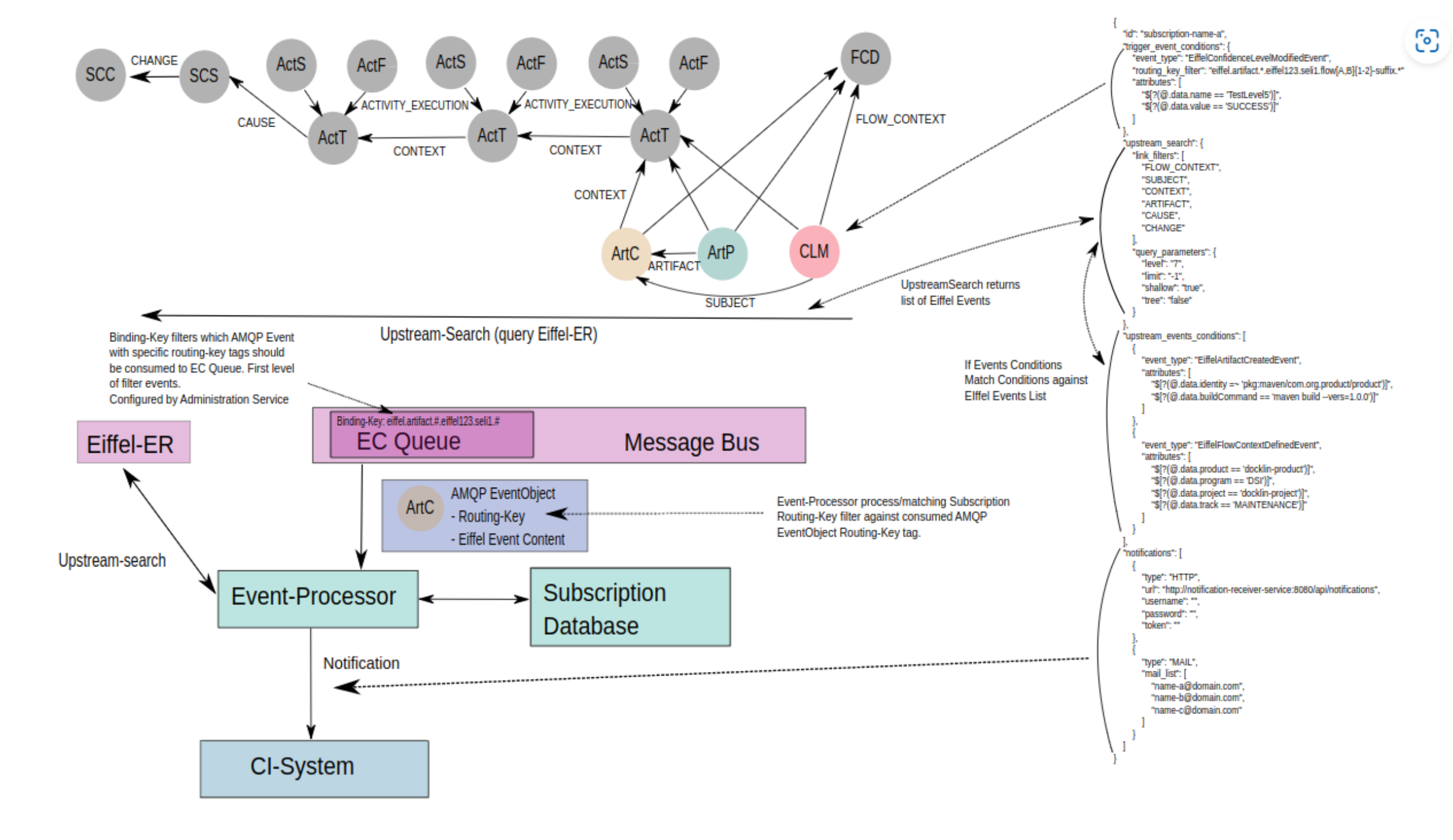
Subscription

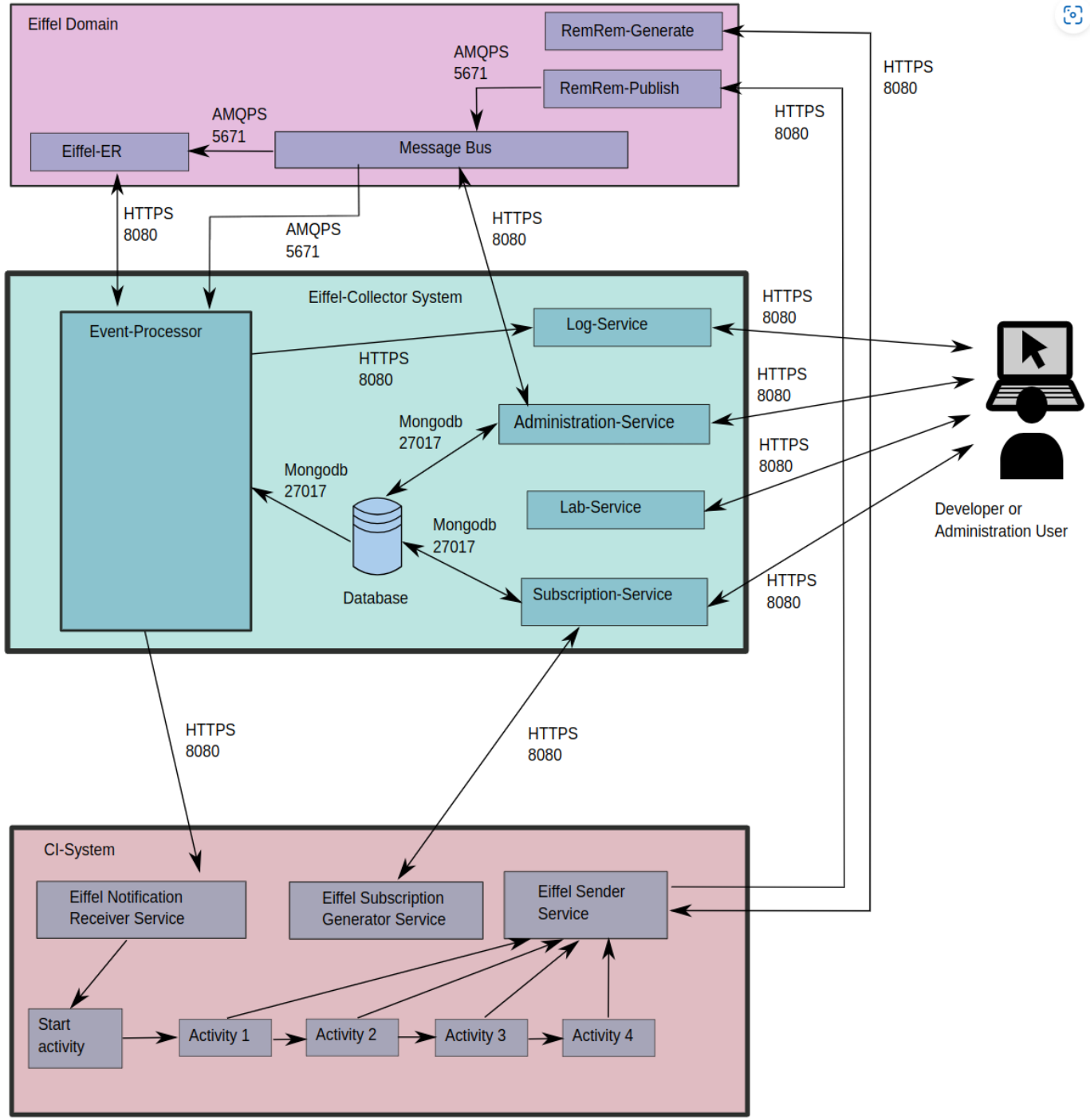

```
{
  "id": "subscription-name-a",
  "trigger_event_conditions": {
    "event_type": "EiffelConfidenceLevelModifiedEvent",
    "routing_key_filter": "eiffel.artifact.*.eiffel123.seli1.flow[A,B]{1-2}-suffix.*"
    "attributes": [
      "$[?(@.data.name == 'TestLevel5')]",
      "$[?(@.data.value == 'SUCCESS')]"
    ]
  },
  "upstream_search": {
    "link_filters": [
      "FLOW_CONTEXT",
      "SUBJECT",
      "CONTEXT",
      "ARTIFACT",
      "CAUSE",
      "CHANGE"
    ],
    "query_parameters": {
      "level": "7",
      "limit": "-1",
      "shallow": "true",
      "tree": "false"
    }
  },
  "upstream_events_conditions": [
    {
      "event_type": "EiffelArtifactCreatedEvent",
      "attributes": [
        "$[?(@.data.identity =~ 'pkg:maven/com.org.product/product')]",
        "$[?(@.data.buildCommand == 'maven build --vers=1.0.0')]"
      ]
    },
    {
      "event_type": "EiffelFlowContextDefinedEvent",
      "attributes": [
        "$[?(@.data.product == 'docklin-product')]",
        "$[?(@.data.program == 'DSI')]",
        "$[?(@.data.project == 'docklin-project')]",
        "$[?(@.data.track == 'MAINTENANCE')]"
      ]
    }
  ],
  "notifications": [
    {
      "type": "HTTP",
      "url": "http://notification-receiver-service:8080/api/notifications",
      "username": "",
      "password": "",
      "token": ""
    },
    {
      "type": "MAIL",
      "mail_list": [
        "name-a@domain.com",
        "name-b@domain.com",
        "name-c@domain.com"
      ]
    }
  ]
}
```

Input and Output

- Subscription – Is the input to the EC system
  - Defines which event type and Event conditions to trigger and start the event processing on.
  - Define binding-key filters
  - Define Event Upstream Search and Upstream Event conditions
  - Define Notifications, supported types: HTTP and Mail.
- Notification
  - Contains trigger/cause Event-id
  - Collected Eiffel-Events

Some System Overview pictures


SCC
CHANGE
SCS
ActS
ActF
ActS
ActF
ActS
ActF
FCD
CAUSE
ACTIVITY_EXECUTION
ACTIVITY_EXECUTION
ActT
ActT
ActT
CONTEXT
CONTEXT
FLOW_CONTEXT
CONTEXT
ArtC
ARTIFACT
ArtP
CLM
SUBJECT
{
"id": "subscription-name-a",
"trigger_event_conditions": {
"event_type": "EiffelConfidenceLevelModifiedEvent",
"routing_key_filter": "eiffel.artifact.*.eiffel123.seli1.flow[A,B]{1-2}-suffix.*"
"attributes": [
"$[?(@.data.name == 'TestLevel5')]",
"$[?(@.data.value == 'SUCCESS')]"
]
},
"upstream_search": {
"link_filters": [
"FLOW_CONTEXT",
"SUBJECT",
"CONTEXT",
"ARTIFACT",
"CAUSE",
"CHANGE"
],
"query_parameters": {
"level": "7",
"limit": "-1",
"shallow": "true",
"tree": "false"
}
},
"upstream_events_conditions": [
{
"event_type": "EiffelArtifactCreatedEvent",
"attributes": [
"$[?(@.data.identity =~ 'pkg:maven/com.org.product/product')]",
"$[?(@.data.buildCommand == 'maven build --vers=1.0.0')]"
]
},
{
"event_type": "EiffelFlowContextDefinedEvent",
"attributes": [
"$[?(@.data.product == 'docklin-product')]",
"$[?(@.data.program == 'DSI')]",
"$[?(@.data.project == 'docklin-project')]",
"$[?(@.data.track == 'MAINTENANCE')]"
]
}
],
"notifications": [
{
"type": "HTTP",
"url": "http://notification-receiver-service:8080/api/notifications",
"username": "",
"password": "",
"token": ""
},
{
"type": "MAIL",
"mail_list": [
"name-a@domain.com",
"name-b@domain.com",
"name-c@domain.com"
]
}
]
}
UpstreamSearch returns
list of Eiffel Events
If Events Conditions
Match Conditions against
EIffel Events List
Upstream-Search (query Eiffel-ER)
Binding-Key filters which AMQP Event
with specific routing-key tags should
be consumed to EC Queue. First level
of filter events.
Configured by Administration Service
Eiffel-ER
Binding-Key: eiffel.artifact.#.eiffel123.seli1.#
EC Queue
Message Bus
ArtC
AMQP EventObject
- Routing-Key
- Eiffel Event Content
Event-Processor process/matching Subscription
Routing-Key filter against consumed AMQP
EventObject Routing-Key tag.
Upstream-search
Event-Processor
Subscription
Database
Notification
CI-System


Eiffel Domain
RemRem-Generate
RemRem-Publish
AMQPS
5671
AMQPS
5671
Eiffel-ER
Message Bus
HTTPS
8080
HTTPS
8080
HTTPS
8080
AMQPS
5671
HTTPS
8080
HTTPS
8080
Eiffel-Collector System
Event-Processor
Log-Service
HTTPS
8080
HTTPS
8080
Mongodb
27017
Administration-Service
HTTPS
8080
Mongodb
27017
Lab-Service
Mongodb
27017
Database
Subscription-Service
HTTPS
8080
Developer or
Administration User
HTTPS
8080
HTTPS
8080
CI-System
Eiffel Notification
Receiver Service
Eiffel Subscription
Generator Service
Eiffel Sender
Service
Start
activity
Activity 1
Activity 2
Activity 3
Activity 4

Demo

Time for Eiffel-Collector Demo

Q&A