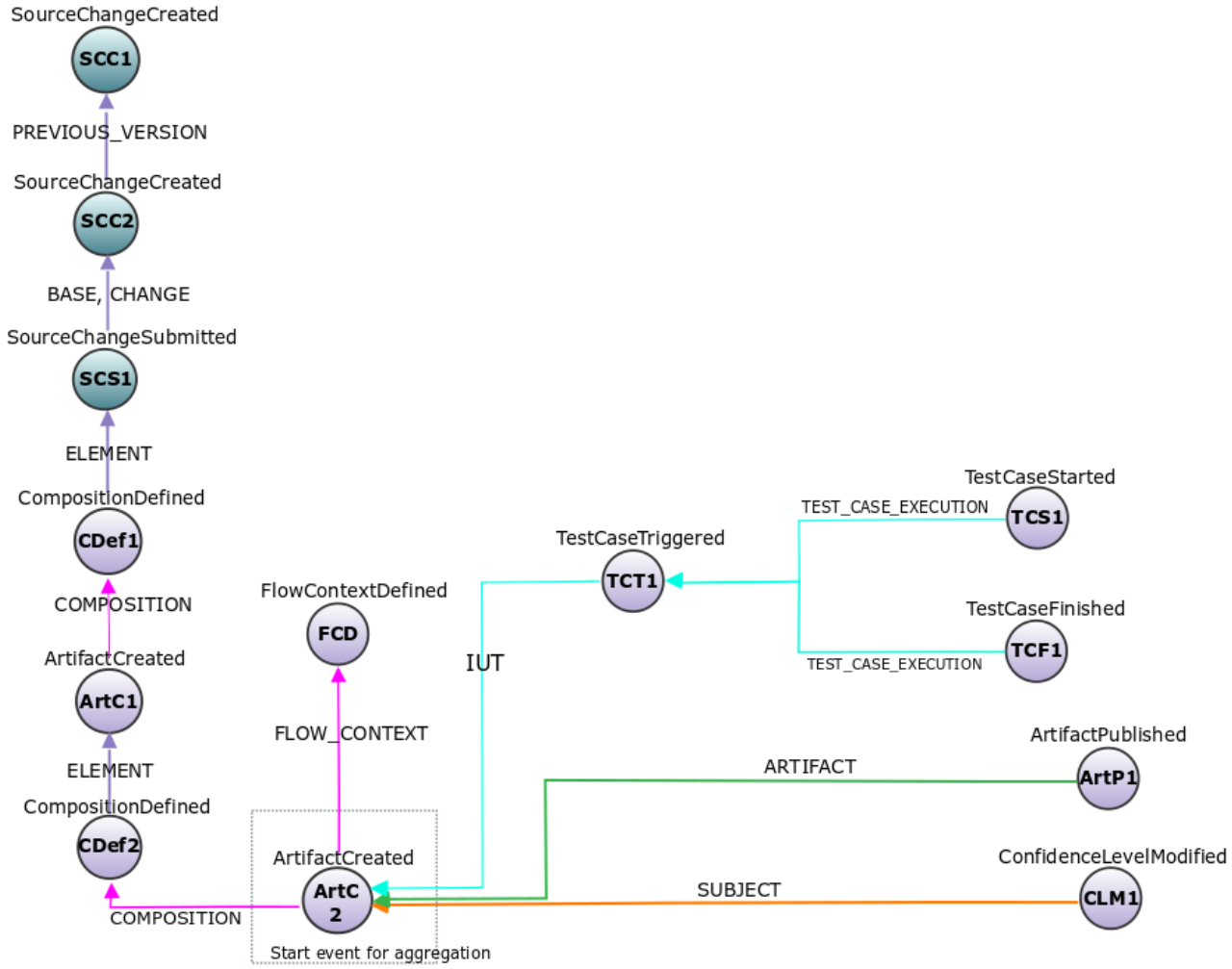
Eiffel-Collector Services

Tobias Åkervik

Background

- Eiffel-Intelligence, "EI", the Event Aggregator application that exist today



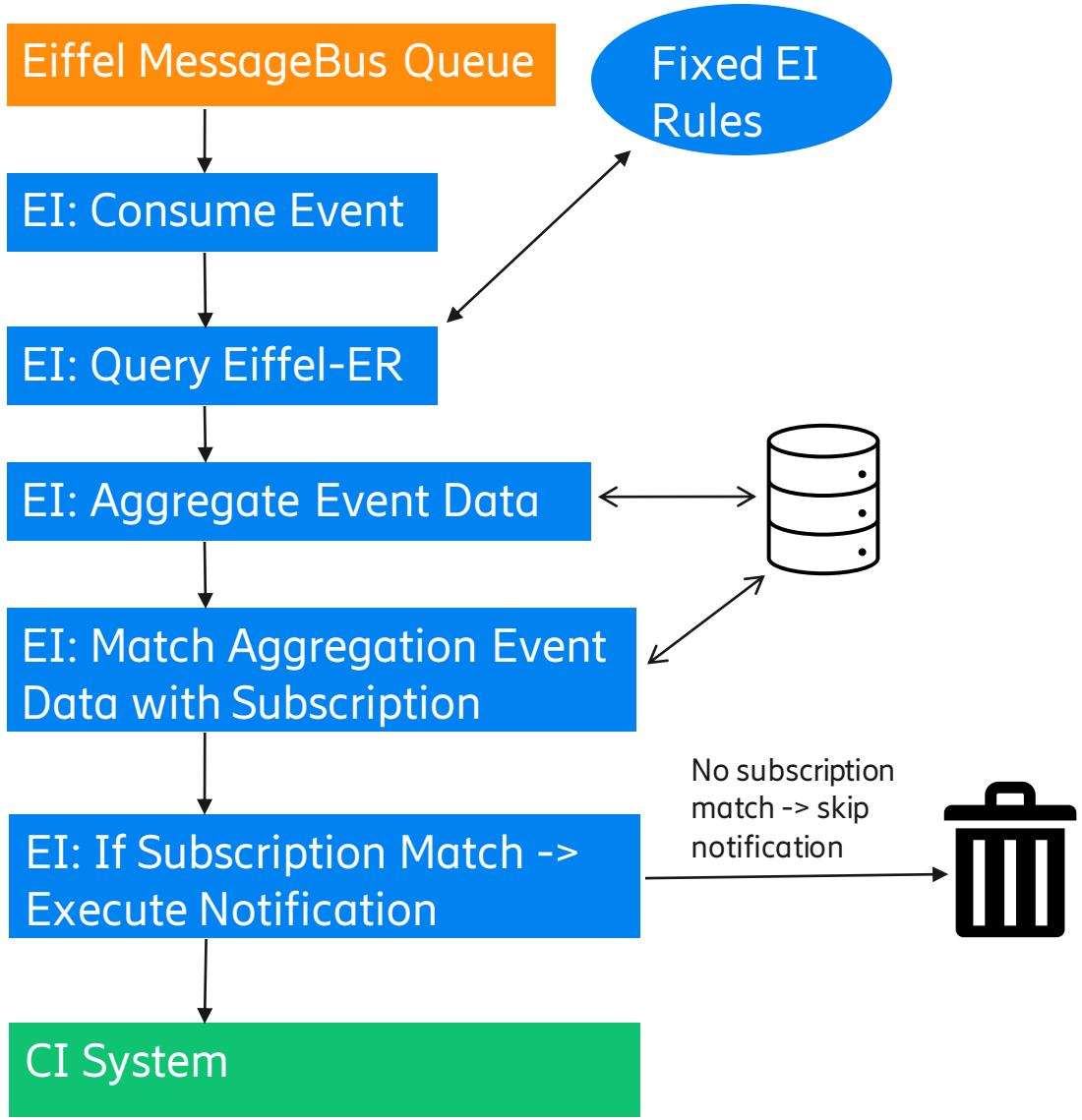
Drawbacks with current EI Solution

- Drawbacks with EI (From an Ericcson perspective):
  - Aggregation hardcoded in each service/instance
  - New Event Aggregations require a new Instance (which require someone to create and maintain the aggregation rules)
  - Every consumed event are aggregated(queries Eiffel-ER) first before matching with subscription which causes huge load on Eiffel-Event-Repository even if Event aggregation never result in a Notification -> A lot data processing without using the data.
  - No easy way for user to follow what events have been aggregated and why no notification have been made.
  - In Ericsson there is one organization that deliver Eiffel services and users do not have access to logs. Only way of debugging for users is to check the EI Rest-Api for aggregation-object and check if there is any failed notifications. Sometimes there is no aggregation-objects and support ticket need to be raised to Eiffel organization for doing troubleshooting.

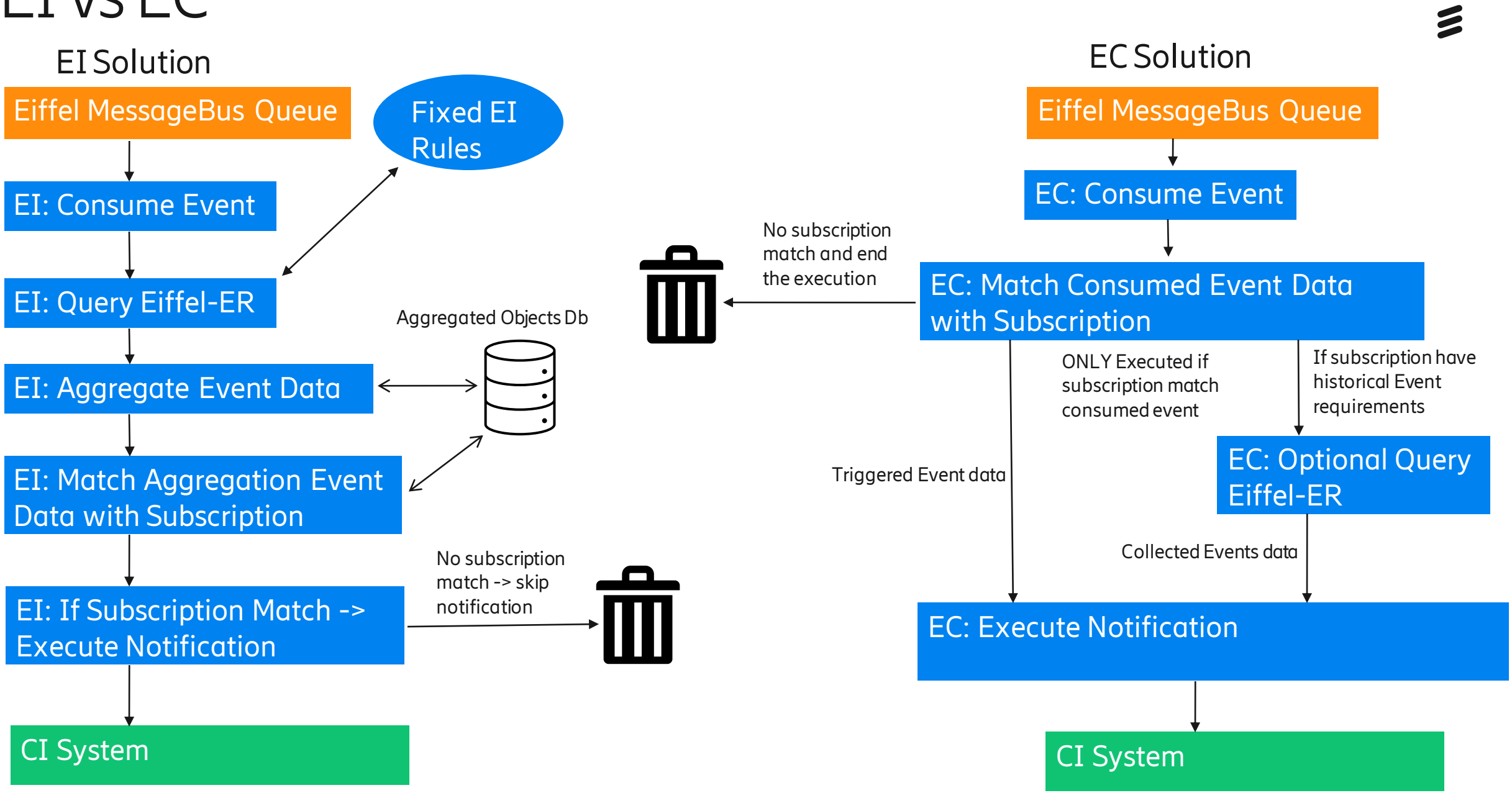
Eiffel-Collector ("EC") services system

- The idea with Eiffel-Collector services is to solve some of the drawbacks with Eiffel-Intelligence and make event data extraction more dynamically.
  - No aggregation rules in service -> Event collector rules defined in subscription
  - Subscription matching is done on consumed event before query Eiffel-Event-Repository, "ER".
  - Query ER is only done if historical events collector requirements are defined in subscription.
  - EC will not aggregate any event data based on rules -> EC will collect all Eiffel events json objects and provide it as a json-list in the notification. Client/User can then decide what Event Data to extract from the Notification events data.
  - Easy to access summarized logs via a Log-Handler service, where logs are stored with triggered event-id as key.
  - Add/Remove BindingKeys on EC RabbitMq(MessageBus) queue without restart of service

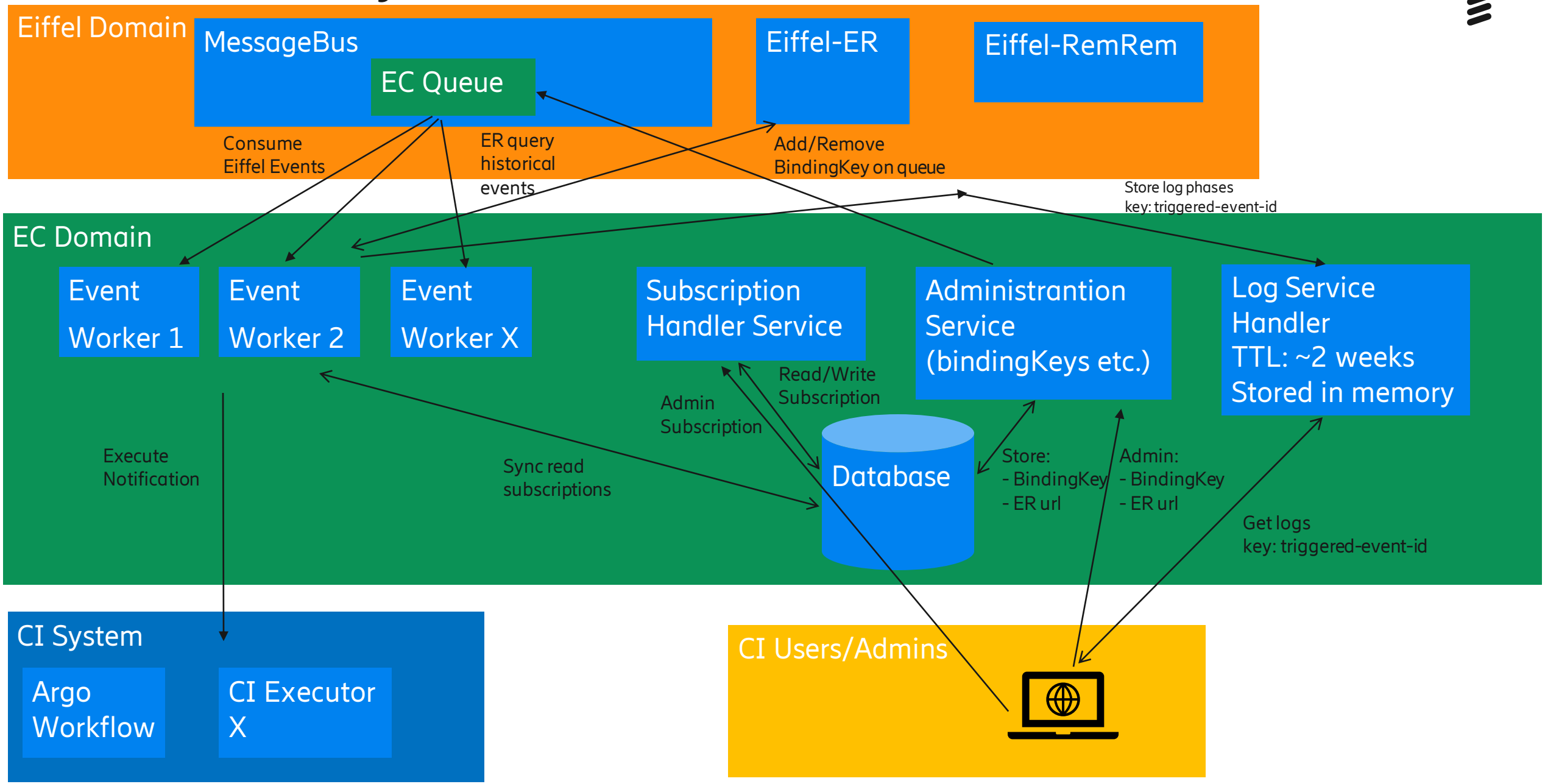
EI "Eiffel-Intelligence" solution



EI vs EC


EI Solution
Eiffel MessageBus Queue
Fixed EI Rules
EI: Consume Event
EI: Query Eiffel-ER
Aggregated Objects Db
EI: Aggregate Event Data
EI: Match Aggregation Event Data with Subscription
No subscription match -> skip notification
EI: If Subscription Match -> Execute Notification
CI System
EC Solution
Eiffel MessageBus Queue
EC: Consume Event
No subscription match and end the execution
EC: Match Consumed Event Data with Subscription
ONLY Executed if subscription match consumed event
If subscription have historical Event requirements
Triggered Event data
EC: Optional Query Eiffel-ER
Collected Events data
EC: Execute Notification
CI System

EC Services System



EC vs EI comparison table

Features	EC (Eiffel-Collector solution)	Ericsson internal Eiffel EI service	OpenSource EI
Downstream Event Collection (DEC) supported	No	Yes	Yes
Update DEC aggregation rules	Not applicable	Hardcoded in file. Requires internal ticket and restart	Hardcoded in file. Requires restart
Upstream Event Collection (UEC) supported	Yes, through ER query	Yes, through ER query	Yes, through ER query
Multiple UEC's in one instance	Yes	No	No
Update UEC conditions	Part of subscription	Hardcoded in file. Requires internal ticket and restart	Hardcoded in file. Requires restart
ER Query Frequency	Each triggering event **	Each triggering event *	Each triggering event *
Collected events reformatted to an object	No, plain json list of all event data	Yes, by Event Extraction Rules	Yes, by Event Extraction Rules
Update Collected events object	Not applicable	Hardcoded in file. Requires internal ticket order and restart	Hardcoded in file. Requires restart
Storing Collected events	No	Yes, stored in MongoDb, TTL configured by internal organization	Yes, stored in MongoDb, default no TTL?
Log Collected events	Yes, in memory (TTL)	Yes, on disk (rotated)	Yes, on disk (rotated)
User accessible event logs	Yes, via REST-Api	No, internal support ticket required	Yes (requires access to read files on node)
Subscription (for Notification) supported	Yes (HTTP, Email)	Yes (HTTP, Jenkins & Email)	Yes (HTTP, Jenkins & Email)
Update Event Trigger	Part of subscription	Hardcoded in file. Requires internal ticket and restart	Hardcoded in file. Requires restart
Update MB BindingKey	Yes, via REST (requires admin account)	Yes, through internal Change request. Requires restart	Yes. Requires restart
ER load from "Empty" service	None, no subscriptions = no ER queries	Equal to non-empty service *	Equal to non-empty service *
Scalable service	Only the event worker pods are scalable	No	No

* If loaded rules file includes no upstream events then no ER queries are performed at all
** If subscription includes no upstream events, then no ER queries are performed at all