

Samsara - Claims Handling - High-Level Design

- 1 Outline
 - 1.1 Title
 - 1.2 References
- 2 Business Objectives
 - 2.1 Product Strategy
 - 2.2 Business Goals
 - 2.3 Business Observability
 - 2.4 Projected Revenue Targets or Cost Savings
- 3 Jira
 - 3.1 Initiative
 - 3.2 Epics
- 4 Description
 - 4.1 Problem Statement
- 5 Estimates
- 6 Dependencies / Related Epics
- 7 Open Issues / Assumptions
- 8 Decision Log
- 9 Design Proposal
 - 9.1 Console Service
 - 9.1.1 Console Service Operations
 - 9.1.2 Create Task
 - 9.1.2.1 Publishing Tasks
 - 9.1.2.2 Subscribing to Task Events
 - 9.1.3 Browse Tasks
 - 9.1.4 Retrieve a Task
 - 9.1.5 Update a Task
 - 9.1.5.1 Update a task status
 - 9.1.5.2 (Re-)Assign Task to a handler
 - 9.1.5.3 Add a Note to a Task
 - 9.1.6 Task Model
 - 9.2 Microservices
 - 9.2.1 Server Micro-services
 - 9.2.1.1 Micro-Service #1
 - 9.2.2 Web Micro-Front-ends
 - 9.2.2.1 Micro-frontend #1
- 10 Non Functional Requirements
 - 10.1 Primary Reviewers
 - 10.2 Non Functional Requirements Table

Outline

<<This is the place-holder header for chapter one. **N.B. Not test goes here.** The sub-chapters below should be completed in full. Delete this text when using the template.>>

Title

<<Add Story Title here. Use best-practice agile convention "As a X I would like Y so that Z.." to demonstrate business value. Alternatively, hyper-link to a Jira issue that contains the same details. Use the title as the text of a hyper-link to the already created epic or story.>>

References

<<Add a numbered list of references covering the research you performed as part of this design. Please use hyper-link text to describe the contents at a particular hyper-link target.>>

Business Objectives

Product Strategy

<<Product Strategy>>

Business Goals

<<List all Business Key Performance Indicators (KPIs) we can use to measure the success of the project>>

Business Observability

<<Enumerate all our business goals>>

Projected Revenue Targets or Cost Savings

<<Detail any projected revenue targets or cost savings>>

Jira

Initiative

<<Add initiative link>>

Epics

<<Add Epic link(s)>>

Description

<<Insert complete description of topic here>>

Problem Statement

<<Insert complete problem statement of topic here. This can be very useful as summary and how to evaluate business value quickly.>>

Estimates

i The Program Evaluation and Review Technique ([PERT](#)) is the recommended technique to be used for estimation of time. Modern agile techniques prefer to use velocity as a basis for commitment as it is more accurate, as estimates are just .. well .. estimates. Also, due to a lack of all information at the time of the estimation process the tendency is to under-estimate. A methodology is suggested here to yield better more accurate estimates, and it is quite simple to apply.

$O = \text{Optimistic Estimate}$

$N = \text{Nominal Estimate}$

$P = \text{Pessimistic Estimate}$

$u = \text{Expected Duration of Task} = (O + 4N + P) / 6$

$v = \text{standard deviation of the probability distribution} = (P - O) / 6$. This is the measurement of uncertainty and risk. Value > 2 is deemed to be high-risk.

$Eu = \text{Sequence of tasks}$. The duration of a set of tasks can be calculated just by adding their u values.

$Ev = \text{The standard deviation of risk of a set of tasks can be calculated: } Ev = \sqrt{\sum(\text{SQUARE}(v))}$

Example:

$O = 1, N = 3, P = 12$

$u = 1 + 12 + 12 / 6 = 4.2 \text{ days}$

$v = (12 - 1) / 6 = 1.8 \text{ days}$

Estimate is (4.2, 1.8), which can be read as this task is likely to be done in 5 days, but could also take up to 6 days. However, it could take more in a worst-case scenario.

It is recommended to create a sub-section under this heading to provide an estimate for each task associated with this design giving a nominal estimate and a standard deviation as executive summary. This allows the project to resource and reason about risk.

Dependencies / Related Epics

Dependency	Jira (if applicable)	Reason	External
------------	----------------------	--------	----------

--	--	--	--

Open Issues / Assumptions

Issue Number	Description	Additional Comments
<<1,2,3, &c.>>	<<Brief Description>>	<<Any other info>>

<<Note: Resolved Open Issues should be removed from here, and moved to the Decision Log>>

<<Note: This LLD cannot be completed until all the Open Issues are resolved and removed or moved to the Decision Log>>

Decision Log

Decision Number	Problem Description	Agreed Resolution	Additional Comments
<<1,2,3, &c.>>	<<Brief Description>>	<<Decision made and how to proceed>>	<<Any other info>>

Design Proposal

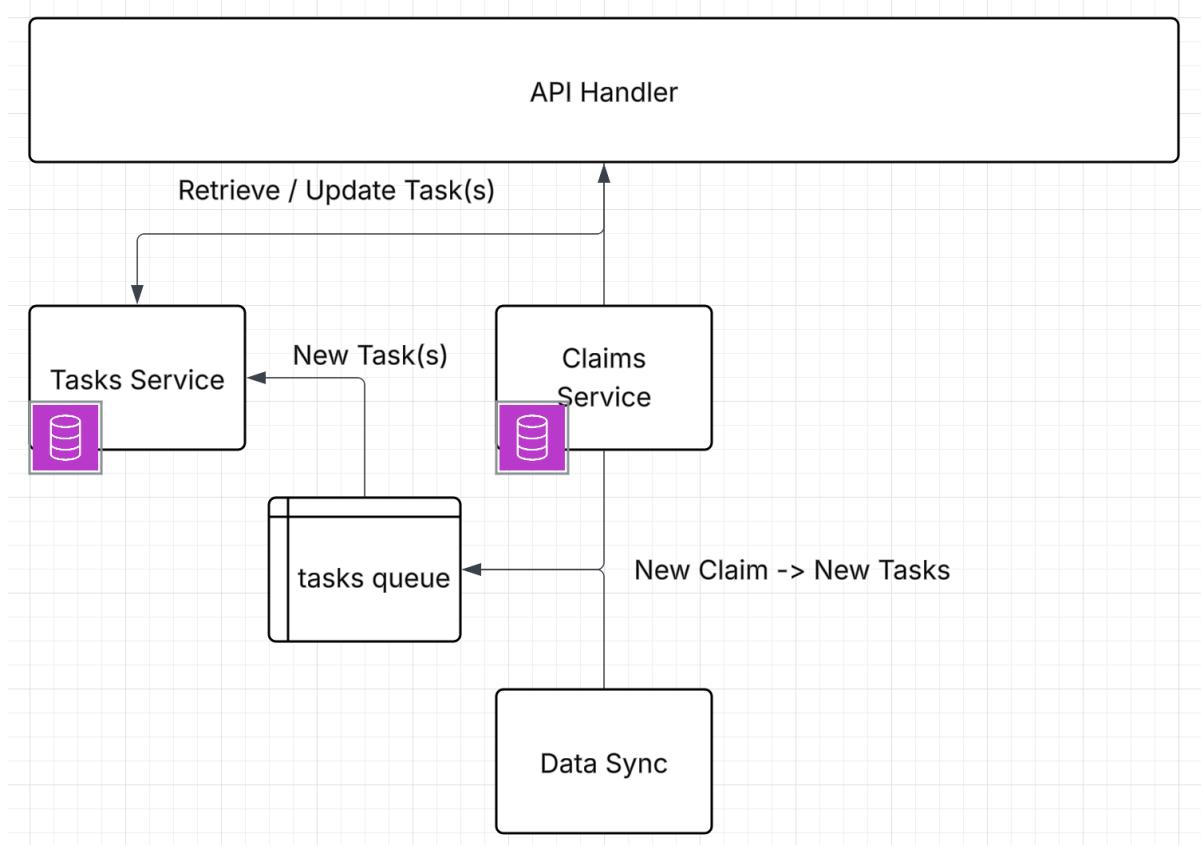
<<Note: This is a place-holder title for chapter 2. **No text should be in this section.** Only in the sub-chapters below.>>

<<Describe the low-level design **in the sub-chapters** of this section. The idea is this chapter is to allow the team sufficient understanding of the design of the story. You can divide all of the chapters below into further sub-chapters as necessary to aid in the overall readability of the document.>>

Console Service

The Console Service shall support the new API version conventions (See *Decision #6*) supporting:

- paging (see *Decision #22*)
- sorting
- filtering (see *Decision #14, #23*)



Console Service Operations

The following operations shall be specified in a new **Console internal open API project**.

Internal Users shall leverage their *One Account* to authenticate (See *Decision #25*)

Operation	Sub-Operations	URI	Notes
Create a Task	N/A	N/A	<p>See section on <i>Create Task</i> section:</p> <ul style="list-style-type: none"> • Event-Driven <i>CreateTask</i> events. The Console Service shall not care where the <i>CreateTask</i> events originate from • <i>CreateTask</i> events shall originate from the <i>Data Sync</i> component initially • Once the <i>Claims</i> micro-service is built and deployed. The <i>CreateTask</i> events shall then originate from the <i>Claims</i> micro-service
Browse Tasks	N/A	GET /console-service/tasks? filter=handler==<handlerId>&li mit=20&offset=0&sort="deadline,desc"	<p>The response should not only include a list of tasks, but also some aggregations related to:</p> <ul style="list-style-type: none"> • Number of completed Tasks in Queue • Number of tasks in Queue • Boolean <i>isSupervisor</i> shall indicate to the UI that this user is a supervisor and should see the management view, and should be able to authorize a payment on a claim <p>Order by deadline, Filter by Assignee.</p> <p>Shall support paging, sorting and filtering.</p>
Retrieve a Task	N/A	GET /console-service/task/ <taskId>	
Update a Task	<ul style="list-style-type: none"> • Update a task status • (Re-)Assign Task - Update the handler and work group assigned to or associated with a task, apply a deadline to the task based on the working days in the country associated with the handler • Add a Note to a task 	PUT /console-service/task/ <taskId> Task Payload	

Create Task

Tasks are currently created for associates, and managed from the business logic in TIA when it is processing policies and claims. The data associated with Tasks is synchronized (in micro-batches) with Sales Force applications using the *Data Sync* component:

<https://github.axa.com/axa-partners-clp/app-convergence-pe-datasync-service>

<https://github.axa.com/axa-partners-clp/app-convergence-pe-datasync-deployment>

The *Data Sync* component works with the following TIA views:

- TASK_MANAGER_MESSAGE_DATA
- TASK_MANAGER_PACKET_DATA

As our first item in the *TIA Modernization* program, we are planning on encapsulating the functionality associated with the *Tasks Screen* into a new Console Service. The Console Service shall replace the existing Sales Force application(s) today, and rolled out per country.

Publishing Tasks

The *Data Sync* component shall be updated to publish in batches new *CreateTask* events to a newly provisioned Rabbit MQ exchange called the *x.clp.task.events* exchange. All events created as part of the TIA modernization project shall adhere to the following contract:

Event Contract

```
/**
 * The <code>Event</code> represents the contract of a standard AXA CLP event.
 */
public interface Event {
    /**
     * Get {@link Meta}.
     * @return {@link Meta}.
     */
    Meta getMeta();
    /**
     * Get {@link EventType}.
     * @return {@link EventType}.
     */
    EventType getEventType();
    /**
     * Get event payload as <code>String</code> representation.
     * Frequently the event payload shall be a JSON serialized Open API object.
     * @return event payload.
     */
    String getData();
}

/**
 * The <code>Meta</code> represents a common event object that can be associated with any event. It is
 * strongly recommended that every event includes a standardised <code>Meta</code> object which includes:
 * <ol>
 *   <li>an event <code>uuid</code> identifier</li>
 *   <li>an event <code>processTimestamp</code> indicating the when the event is processed</li>
 *   <li>an event <code>hash</code> to determine if the event has been tampered with between publishing and
 *       consuming</li>
 * </ol>
 */
@JsonIgnoreProperties(ignoreUnknown = true)
@NoArgsConstructor
@AllArgsConstructor
@Data
@Builder
public class Meta {
    /**
     * A <code>uuid</code> provides the event with an identity.
     */
    @NotNull(message = "UUID cannot be blank")
    private String uuid;
    /**
     * The <code>processTimestamp</code> represents the time that the event starts processing.
     * The event payload representation can encapsulate the create time distinct from the
     <code>processTimestamp</code>.
     */
    private long processTimestamp;
    /**
     * The <code>hash</code> represents the hash of the event payload. This can be convenient to determine if
     the event
     * payload has been tampered with between the time the event is published and the event is consumed.
     */
    @NotNull(message = "Hash cannot be blank")
    private String hash;
}
```

The *EventType* shall be a discriminator Enum which gives information to the subscriber of the event regarding the representation of the data string Event payload. It shall be a common pattern to serialize Open-API types in Event data payloads when communicating with down-stream services. The *EventType* shall provide enough information to a subscriber of the event to deserialize the data. The data payload in the *CreateTaskEvent* could be a representation of a *Customer*, *Partner*, *Case* or *Claim* and the supported representations can be extended in the future. Therefore, we may have supporting *EventType* like the following:

Supported Event Types

```
/**  
 * The <code>EventType</code> outlines the Axa Partner supported event types.  
 */  
public enum EventType {  
    /**  
     * All services shall generate Audit Events to be persisted in an Event Log. The purpose  
     * of the Event Log shall be to support human and machine troubleshooting, and also support  
     * history use-cases.  
     */  
    AUDIT_EVENT,  
    /**  
     * Tasks are propagated to the console service through Create Task Event.  
     */  
    CREATE_TASK_EVENT  
}
```

A *CreateTask* event could then potentially look something similar to the following:

CreateTask

```
/**  
 * The <code>CreateTaskEvent</code> encapsulates an Open API <code>Task</code> type in its data payload.  
 */  
@JsonIgnoreProperties(ignoreUnknown = true)  
@NoArgsConstructor  
@AllArgsConstructor  
@Data  
@Builder  
public class CreateTaskEvent implements Event {  
    @Valid  
    @NotNull(message = "Meta cannot be null")  
    private Meta meta;  
    @NotNull(message = "Event Type cannot be null")  
    private EventType eventType;  
    @NotBlank(message = "Data cannot be blank")  
    private String data;  
}
```

As event-driven solutions become more prevalent in our architecture with the introduction of the TIA Modernization program, this design recommends the introduction of a new **events library**, which can share common event / messaging related functionality across all of our micro-services that use events. The *events* library (<https://github.axa.com/axa-partners-clp/events>) can encapsulate:

- The definition of all AXA CLP events. Each event should adhere to a AXA CLP standard contract, so we can reason consistently about the form of all events being processed in our system.
- The events library can encapsulate a common *Event Publisher*, and *Event Subscription* code as Spring beans. It is better to write this code once, and do it well and then allow micro-services to simply wire in the *Event Publisher* and/or *Event Subscriber* depending on their needs.
- The events library can enforce the invariants associated with events. For example, each Event must contain an identifying Meta UUID, an Event Type and a Data payload.
- The events library can provide de-serialization functions, so that client micro-services can de-serialize data payloads based on the Event Type. To support this, the events library shall include static dependencies on Open-API projects needed to support de-serialization.

The *Data Sync* component already has a solution in place to handle re-try attempts, which would be necessary in the case where the *tasks* topic is unavailable. If the *tasks* topic, is unavailable then the *TASK_MANAGER_PACKET_DATA* shall have the status of its record updated to R.

Subscribing to Task Events

As discussed *CreateTask* event would encapsulate task information propagated from the *Data Sync* component, and contain enough information so that the Console Service can subscribe to the *tasks* topic, and use the *CreateTask* event to create a new *Task* in its data store.

This event-driven interface for *CreateTask* defines the permanent solution for creating tasks. When a *Claims* micro-service is introduced into our architecture – with decomposed business logic from COS – that business logic shall create tasks by sending *CreateTask* events to this topic. That is, the consuming of *CreateTask* events is not expected to change, as further data and functionality is decomposed from TIA.

When processing incoming events in batches it is important that they can be processed in an idempotent manner, so that we are tolerant of duplicate events and we can de-duplicate events. Furthermore, we want to load in multiple tasks into our data-store as part of a single round-trip and we don't want to insert each task into our database in a dedicated round trip. To support bulk *upserting* tasks into our datastore we shall leverage the following PostgreSQL features:

<https://neon.com/postgresql/postgresql-tutorial/postgresql-insert-multiple-rows>

<https://medium.com/@eremeykin/efficient-bulk-data-insertion-into-postgresql-with-java-0487579287ff>

<https://neon.com/postgresql/postgresql-tutorial/postgresql-upsert>

<https://stackoverflow.com/questions/69254548/postgresql-equivalent-of-mysql-on-duplicate-key-update>

Browse Tasks

As we move to more modern Web technologies and paradigms, standard conventions shall be applied to all APIs that provide tabular data or a browse of types. Through a consistent application of these API conventions we can achieve better developer productivity on both the web and server when delivering new features and functionality. As discussed in the *New API Version* section, any browse of types endpoint, or retrieve all tasks in this particular case shall support the following API conventions:

- Standard Paging solution
- Standard Sorting solution
- Standard Filtering solution
- Standard Searching solution

These API conventions can be implemented in a new / existing library, so that individual micro-services don't need to re-implement these conventions and the functionality can be shared. The object shall be for a micro-service to be able to support all above API conventions by adding a dependency on a library and adding one line of code.

It shall be critically important that suitable secondary indexes are created to support filtering and searching requirements for a given UI screen (see *Decision #14* for the precise filtering needs).

A manager browsing their queue of tasks shall get a paged queue of tasks associated with every member on their team ordered by task deadline. When the manager request is authenticated, the group associated with the manager can be retrieved as a scope from the security principal. This group shall then be used to filter the tasks to return, which should include his / her tasks, and the tasks of his / her team.

A regular user browsing their queue shall only get a paged queue of tasks assigned to them ordered by deadline.

The unassigned tab shall return a paged list of **all** unassigned tasks across all teams / groups ordered by the create date of the task.

Retrieve a Task

This is a straight forward operation to retrieve a task by its identifier from the micro-service local database, where there shall be a dedicated secondary index created for said identifier if the identifier is not a primary key.

Update a Task

This operation shall support updating the following elements of a task only

1. Update a task status
2. (Re-)Assign Task - Update the handler associate assigned to a task
3. Add a note to a task

Update a task status

As with a number of model types in our system, tasks shall have a life-cycle represented by a status. This shall be represented by a finite state machine:

- N - New
- H - On-Hold
- C - Closed
- R - Retry

The following are the possible transitions:

- New Closed - Close task business processing shall happen during this valid transition.
- New On-Hold - Request more information from the customer. Remove the task deadline.
- New New - No status change. Can be the case for task (re-)assignment or when adding a note to a task.
- On-Hold Retry - Customer has provided requested information to the associate. Apply a new task deadline.
- On-Hold On-Hold - No status change. Can be the case for task (re-)assignment or when adding a note to a task.
- Retry On-Hold - Customer has provided information, but the associate now realizes a further problem and they need to request more information. Remove the task deadline.
- Retry Closed - Customer has provided information, and now the associate is in a position to make the payment and close the task.
- Retry Retry - No status change. Can be the case for task (re-)assignment or when adding a note to a task.
- Closed Closed - No status change. Can be the case adding a note to a task. We should not allow re-assignment of closed tasks.

This design recommends a finite state machine (FSM) pattern for model types that have a life-cycle represented by a status field. This provides a generic solution of life-cycle status field for all our model types in CLP, that allows us to cleanly apply business logic associated with status transitions. The pattern outlined below leverages an *EnumMap* (See *Reference #25*):

Task Life-Cycle State machine

```
...
/***
 * A <code>TaskStatusHandler</code> handles a task status {@link Transition}.
 */
@FunctionalInterface
public interface TaskStatusHandler {
    /**
     * Execute the {@link TaskStatusHandler}.
     * @param taskJobs a <code>List</code> of {@link TaskJob}s.
     */
    void execute(final List<TaskJob> taskJobs);

    /**
     * Process the passed <code>List</code> of {@link TaskJob}s through the status transition.
     * @param taskJobs a <code>List</code> of {@link TaskJob}s to process.
     * @param taskStatus to update the task in the {@link TaskJob} with.
     */
    default void process(final List<TaskJob> taskJobs, final Consumer<List<TaskJob>> callback, final
TaskStatusEnum taskStatus) {
        taskJobs.forEach(taskJob -> {
            final TaskService taskService = taskJob.getTaskService();
            final Task task = taskJob.getTask();

            taskService.updateStatus(task, taskStatus);
        });
    }
}
...
/**
 * A <code>Transition</code> represents a valid change in a task status.
 */
public enum Transition implements TaskStatusHandler {
    OPEN_TO_CLOSED(TaskStatusEnum.OPEN, TransactionStatusEnum.CLOSED) {
        @Override
        public void execute(final List<TaskJob> taskJobs) {
            process(taskJobs, TaskStatusEnum.CLOSED);
        }
    },
    CLOSED_TO_OPEN(TaskStatusEnum.CLOSED, TransactionStatusEnum.OPEN) {
        @Override
        public void execute(final List<TaskJob> taskJobs) {
            process(taskJobs, TaskStatusEnum.OPEN);
        }
    };

    private TaskStatusEnum src;
    private final TaskStatusEnum dest;

    /**
     * Constructor.
     * @param src {@link TaskStatusEnum}.
     * @param dest {@link TaskStatusEnum}.
     */
    Transition(final TaskStatusEnum src, final TaskStatusEnum dest) {
        this.src = src;
        this.dest = dest;
    }

    private static final Map<TaskStatusEnum, Map<TaskStatusEnum, Transition>> m = new EnumMap<>(TaskStatusEnum.
class);
    static {
        for(final TaskStatusEnum status : TaskStatusEnum.values()) {
            m.put(status, new EnumMap<>(TaskStatusEnum.class));
        }
        for(final Transition transition : Transition.values()) {
            if(transition.src != null) {
                m.get(transition.src).put(transition.dest, transition);
            }
        }
    }
}
```

```

        }

    /**
     * Checks the proposed <code>Transition</code> and if it is valid return it.
     * @param src {@link TaskStatusEnum}.
     * @param dest {@link TaskStatusEnum}.
     * @return a valid {@link Transition}.
     */
    static Transition from(final TaskStatusEnum src, final TaskStatusEnum dest) {
        final Transition toReturn = m.get(src).get(dest);
        if(toReturn == null) {
            throw transitionFailure();
        }
        return toReturn;
    }

    private static AxaPartnersException transitionFailure() {
        return new AxaPartnersException(AxaPartnersError.TECHNICAL_ERROR.getErrorCode(), AxaPartnersError.
TECHNICAL_ERROR.getErrorDescription());
    }
}
...

```

(Re-)Assign Task to a handler

Initially a number of constraints were considered to be applied to the (re-)assign task operation. However, in consultation with the product manager it was decided not to apply any constraints (See *Decision #9*).

This operation shall simply change the handler associated to a task after verifying that the handler identifier passed in the Task payload corresponds to a real valid party associated with the One Account OAuth claim provided as the security principal.

The One Account OAuth claim shall be changed for the operations team members:

- **Supervisor** - Add a supervisor to the OAuth claim of any associates than manage an operations team
- **Group** - Add a group to the OAuth claim to represent the operations team that an associate is a part of
- **Country** - Add a country to the OAuth claim to represent the country where the associate is legally working in

A *Supervisor* shall have elevated access to data dashboards on the tasks screen, and shall be able to view group-level aggregations of data for the group that they supervise.

When an employee is assigned to a task, the *Group* associated with that employee is then also assigned to the task. When a *Supervisor* browses the tasks, they shall get a view of all the tasks associated with all employees in their *Group*.

When an employee is assigned to a task, the *Country* associated with the employee is used to calculate the deadline for the task and the non-working days in the country the associate is working in (e.g. exclude weekend days and public holidays). To facilitate this, we would need to bind the country the associate works in with their OAuth claim / One Account.

<https://github.com/focus-shift/jollyday/blob/main/README.md>

Add a Note to a Task

We shall support the operation to Add a note to a Task. However, we shall not support updating or deleting notes associated with Tasks.

Task Model

The *CreateTask* event shall contain the string representation of a *Task* as its data payload. The *Task* shall be defined in a *Tasks* open-API project and shall be returned by end-points on the Console Service.

A *Task* shall contain the following fields:

Task Property	Task Property Description
country code	https://www.worldometers.info/country-codes/
status	<ul style="list-style-type: none"> • N - New • H - On Hold • C - Completed • R - Retry

entity Id	The entity Id. What this entity Id represents shall depend on the entity Type: <ul style="list-style-type: none">• Case - Request Id• Claim - CLA Case Number• Policy - Policy Number• Customer - Name Number• Partner - Name Number• Customer Sync - Name Number
entity Type	The supported Entity Types shall initially include the following: <ul style="list-style-type: none">• Case• Claim• Policy• Customer• Partner• Customer Sync
entity	A JSON serialized representation of one of: <ul style="list-style-type: none">• Case• Claim• Policy• Customer• Partner• Customer Sync
owner	The identity of the customer associated with the Entity type
handler	The identity of the associate assigned to the Task

The above refers to the *Task* DTO. Persistence and DTO semantics shall not be mixed in the same Java class. A separate *TaskEntity* shall encapsulate the persistence semantics.

Microservices

Server Micro-services

<<Provide a sub-chapter for each collaborating micro-service along with a link to the associated Low-Level design for the micro-service>>

Micro-Service #1

<<Provide link to micro-service LLD details>>

Web Micro-Front-ends

<<Provide a sub-chapter for each collaborating micro-frontend along with a link to the associated Low-Level design for the micro-frontend>>

Micro-frontend #1

<<Provide link to micro-frontend LLD details>>

Non Functional Requirements

<<*Note: The intention of this section is to consider each Non-Functional Requirement and outline your solution to that non-functional requirement, or detail why a non-functional requirement doesn't pertain to your solution.*>>

<<*Note: Ensure you get the correct reviews conducted by the correct team.*>>

Primary Reviewers

Non Functional Requirements Table

Category	Sub-Category	Definition	Solution Details
Data and Security	Data Integrity	<p>The maintenance of, and the assurance of, data accuracy and consistency over its entire life-cycle, so as to limit data corruption.</p> <p>How is data captured, stored, validated, updated, transferred? How will the data be kept accurate and consistent?</p>	
	Data Retention	<p>Determines the retention time, archival rules, data formats, and the permissible means of storage, access disposal and encryption.</p> <p>Must adhere to Axa Partner's internal data retention policy. Which data will be archived and how long will it be kept?</p>	
	Backup	<p>A copy of computer data taken and stored elsewhere so that it may be used to restore the original after a data loss event.</p> <p>Does the system have a requirement for data backup in case of a loss event? How important is it to have this data backed up?</p>	
	Authentication (AuthN) and Authorization (AuthZ)	<p>Authentication is the process that allows a device to verify the identity of someone who connects to a network resource. Authorization is the process of giving someone permission to do or have something.</p> <p>Do you have a list of the hierarchy, user groups & user permissions for your Product? Do you need to have 2 factor or 3 factor authentication?</p> <p>Do you have suitable user and role management?</p> <p>Do you have a requirement to support multi-tenancy users?</p> <p>What authentication mechanism are you going to use?</p> <p>What authorization mechanism are you going to use?</p> <p>What operations need to be allowed / restricted per user role?</p> <p>What data needs to be presented / filtered per user role?</p>	
	Data Confidentiality	<p>Is about protecting data against unintentional, unlawful, or unauthorized access, disclosure, or theft.</p> <p>Does the system have any PCI or PII data? How will you ensure data in transit and at rest is encrypted? How will you ensure that only the people or processes authorized to view and use the contents of a message or transaction have access to those contents? Are there requirements for opting in or out? [for having record created?]</p>	
	Non Repudiation	<p>Provides evidence for the existence of a message or transaction and ensures its contents cannot be disputed once sent.</p> <p>Will these messages be surfaced via the logs? How long should we keep the logs for? Are technical messages shown on Kibana / Loki Grafana? Are there any business operational type messages that would need to be surfaced via another application? Shall there be requests for data deletion?</p>	
	Data Availability	<p>Is about the timeliness and reliability of access to and use of data. Availability has to do with the accessibility and continuity of information.</p> <p>What is the availability requirement E.g. 99.999?</p> <p>What observability do you have in place to measure your availability requirements?</p> <p>What happens if you don't reach your availability targets?</p> <p>How shall your SRE Error Budget be processed?</p>	
	Legal Privacy, Governance Requirements & Certification	<p>For every process or activity related to the processing of Personal Data, products and /or vendors, owner-associates need to follow the relevant assessment process.</p> <ul style="list-style-type: none"> - Information risk assessment: utilized by Global Cyber Security and Fraud to assess risks to Axa Partner's applications. 	
	Logging and Auditing	<p>Logs maintained and easily surfaced for all system events</p> <p>Logs must adhere to Cyber security logging standards.</p>	
	Data Capacity	<p>How much disk space one or more storage devices shall provide. It measures how much data a computer system may contain.</p> <p>How many gigabytes required for data-store capacity? (This will be determined based on data retention requirements and volume)</p>	
Performance and Resource Effectiveness	Volume	<p>The volume of records that will be transmitted to the system daily.</p> <p>What are the current volumes and what it will need to scale to? This is a question for the business, looking at existing clients and associated volumes (daily, weekly, monthly...) and what are the peak times?</p>	
	Efficiency	<p>The amount of resources the system requires - indicates the manner in which the inputs are used by the system.</p> <p>What resources are required by your micro-service?</p> <p>What are the financial considerations regarding your resource efficiency?</p> <p>Are you over-provisioning your micro-service?</p>	

	Reusability	<p>The use of existing assets within the software development process: include code, software components, test suites, designs and documentation.</p> <p>How will you ensure optimal re-use? How much re-use will be required for this Product? (looking at the end-state vision)</p>	
	Scalability	<p>The estimated size of the client/user base and projected growth of the client /user base. What are the peak login times to Clover web dashboard - what's the max number of people that can be logged in at the same time?</p> <p>What is the estimated size of the client base for the MVP? (Number of clients, volumes, which countries etc.)</p> <p>What is the estimated growth of the client base beyond the MVP?</p> <p>Is your service horizontally scalable?</p> <p>If using a micro-service, shall you support auto- scaling up / down?</p> <p>How fast do you need to be able to scale up or scale down?</p> <p>Shall your micro-service receive steady or bursty traffic, and shall it need to respond to significant peak load, followed by periods of idleness?</p> <p>How can you test auto-scaling?</p>	
	Response Time	<p>The total amount of time it takes to respond to a request for service.</p> <p>Is the system batch based or real-time? What's the maximum load for the system, and how will you performance test the response time, at peak load?</p> <p>What average response time does your service need to support?</p> <p>What is your 99th/95th response time percentile requirement?</p> <p>How shall you measure your service's response time in production?</p> <p>What time-out values can you safely set based on your services dependencies?</p>	
	Latency	<p>Measures the time it takes for some data to get to its destination across the network and back again. It is usually measured as a round trip delay.</p> <p>How many milliseconds does this need to be? (Ties in with response time and volume/load)</p> <p>Do you have any SLA requirements?</p> <p>What SLA requirements does your services dependencies support?</p> <p>AWS Services shall be co-located in the same AWS zone as the micro- services using them. The expectation is that latency will be on the order of 10's of ms and not 100's of ms.</p>	
	Throughput	<p>How much data can be transferred from one location to another in a given amount of time.</p> <p>How many bits per second does this need to be?</p> <p>How many requests per second does your service need to support for certain request size, or traffic patterns?</p> <p>How many transactions per second does your service need to support?</p> <p>How many concurrent requests can your service support?</p> <p>How many concurrent transactions can your service support?</p>	
	Reliability and Resilience	<p>An ability of the system to withstand a major disruption within acceptable degradation parameters and to recover within an acceptable time.</p> <p>Is this application a critical application?</p> <p>Are SLAs either in place or proposed for the new system w.r.t.: High availability (four 9s, three 9s)? Recovery Time Objective (RTO) - The time it takes to get back up and running Recovery Point Objective (RPO) - The age of the recovery point (i.e. an hour, a day, a week) or in other words the time elapsed since the most recent reliable backup.</p> <p>Does your micro-services have suitable circuit breaker design to protect up-stream processes?</p> <p>Does your micro-services need to provide a rate limiting solution?</p> <p>How does your reliability solutions at application level work with the reliability solutions operating at an infrastructural level (E.g. How does service mesh time-outs / circuit breakers work when there are also solutions applied at the application level)</p>	
System Lens	Interoperability	<p>Other systems that the proposed system is dependent on to form an overall solution</p> <p>What other system or applications does your system need to integrate with in the overall solution? What technologies are being used? Are they legacy/new? What factors need to be considered here?</p>	
	Portability	<p>The usability of the same software in different environments (an abstraction between the application logic and system interfaces)</p> <p>Are the solutions being proposed locked to a particular vendor? Is this a risk?</p>	

	Compatibility	The capacity for two systems to work together without having to be altered to do so. Compatible software applications use the same data formats. JSON is the preferred data format being used for all our interfaces which is expected to be compatible with all interfacing systems now and in the future. Another option is to use the AVRO data interchange format, but we've decided to go with JSON for the moment as the API development, event producing, consuming and storage is much simpler to support and maintain.	
	Integrability	The process of bringing together the component sub-systems into one system and ensuring that the subsystems function together as a system. A measure of the ability to extend a system and the level of effort required to implement the extension.	
	Extensibility	A measure of the ability to extend a system and the level of effort required to implement the extension.	
Operations	Maintainability	The ease with which a Product can be maintained.	
	Supportability	The ability of technical support personnel to install, configure, and monitor computer products, identify exceptions or faults, debug or isolate faults to root cause analysis.	
	Testability	The degree to which the system supports testing in a given test context.	
	Configuration Management	The process for maintaining computer systems, servers, and software in a desired, consistent state. Usage of standardized Kubernetes Config Maps delivered in a standardized approach per environment as advocated by the CLP team best practice. All micro-service / project secrets are managed correctly.	