**US\_299203\_Cognizant: Get transactions/orders by day from RentTrack**

**Understanding of the requirement**

Get the latest transaction reported status for each user So that it can be compared to the users credit report and highlight the impact it had on their credit .

**Assumptions**:

* VPN connectivity between Heroku and Amazon MQ has been established
* Sandbox Access will be provided
* GitLab Apiary access will be provided

**High level Technical design and solution**

* 1. Impacted Modules

bcm-service

bcm-messaging

bcm-jobs

renttrack-plugin

* 1. Impacted Workflows
     1. Overview:

Get the latest transaction reported status for each user.

On daily basis, the CronM will invoke BCM job API to generate the transaction report for each user for the configured number of cycle in days

* + 1. APIs Involved:

GET /api/tenant/leases.{\_format}

* + - * + BCM is the consumer
        + Error Codes:

200 - Returned when successful

204 - No content with such parameters

500 - Internal Server Error

GET /api/tenant/leases/{leaseId}/order.{\_format}

* + - * + BCM is the consumer
        + Payload : leaseId
        + Filter: representation=detailed
        + Error Codes:

200 - Returned when successful

204 - No content with such parameters

409 - Lease not found

500 - Internal Server Error

GET /api/tenant/utilities.{\_format}

* + - * + BCM is the consumer
        + Error Codes:

200 - Returned when successful

204 - No content with such parameters

500 - Internal Server Error

GET /api/tenant/utility/{utilId}/order.{\_format}

* + - * + BCM is the consumer
        + Payload : utilId
        + Filter: representation=detailed
        + Error Codes:

200 - Returned when successful

204 - No content with such parameters

409 - Lease not found

500 - Internal Server Error

iii. Messaging Infrastructure:

Amazon MQ is a managed message broker service for Apache ActiveMQ that makes it easy to set up and operate message brokers in the cloud. Message brokers allow different software systems–often using different programming languages, and on different platforms–to communicate and exchange information. Amazon MQ reduces your operational load by managing the provisioning, setup, and maintenance of ActiveMQ, a popular open-source message broker.

ActiveMQ Brokers have been created already for Dev and Stage (as part of Rewards Module’s setup), and BCM would be using those brokers to host the relevant Queue(s). There is no separate broker for Test environment - we would be using the one in Dev.

BCM Job module publish messages to BCM messaging module through Active MQ for each user for them the report to be generated.

BCM Messaging module retrieve the each message and call rentTrack plugin to fetch both the lease and utility orders and save it in the BCM database.

iv. UI Screens: N/A

* 1. DB Changes:

1. Auditing Tables such as **WebService\_logs, External\_log**

2. Table Name : **customer\_data**

The transaction\_pull\_date column added in the customer\_data table to check the last transaction pull date of each BCM customer.

Table Description :

* ucid
* brand
* customer\_data\_id
* customer\_email
* created\_date
* modified\_date
* access\_token
* refresh\_token
* token\_expiry
* token\_created\_date
* transaction\_finder\_id
* payment\_account\_id
* resident\_status
* **transaction\_pull\_date**

3. Table Name : **jobs**

BCM job wills be stored in jobs table

Table Description :

* job\_id
* job\_name
* brand
* created\_date
* modified\_date
* period\_start\_date
* period\_end\_date

3. Table Name : **job\_events**

Job Events will be used to store for tracking the status of each individual job

Table Description :

* job\_event\_id
* job\_id
* job\_event\_code
* brand
* details
* event\_date
* created\_date
* modified\_date
* job\_id
* job\_name
* brand
* created\_date
* modified\_date
* period\_start\_date
* period\_end\_date

3. Table Name : **customer\_order\_process**

Customer process Order status will be stored in customer\_order\_process table to track the message processing status for each customer.

Table Description :

* id
* job\_id
* customer\_data\_id
* status
* created\_date
* modified\_date

4 .Table Name : **customer\_orders**

Customer lease and utility orders will be stored in customer\_orders table

Table Description :

* id
* rt\_order\_id
* customer\_data\_id
* order\_type
* status
* rent
* other
* total
* fee
* paid\_for
* created\_date

5. Table Name : **customer\_orders\_reported\_info**

BCM will save the reported back info of each order in the this table

Table Description :

* id
* order\_id
* bureau\_type
* reported\_at
* created\_date

6. Table Name : **failed\_transaction\_pull**

When exception occurred during the customer order process, BCM will save the exception details in failed\_transaction\_pull with the detailed error message

Table Description :

* id
* customer\_data\_id
* job\_id
* error\_message
* created\_date
  1. Dependencies

RentTrack API and Amazon MQ

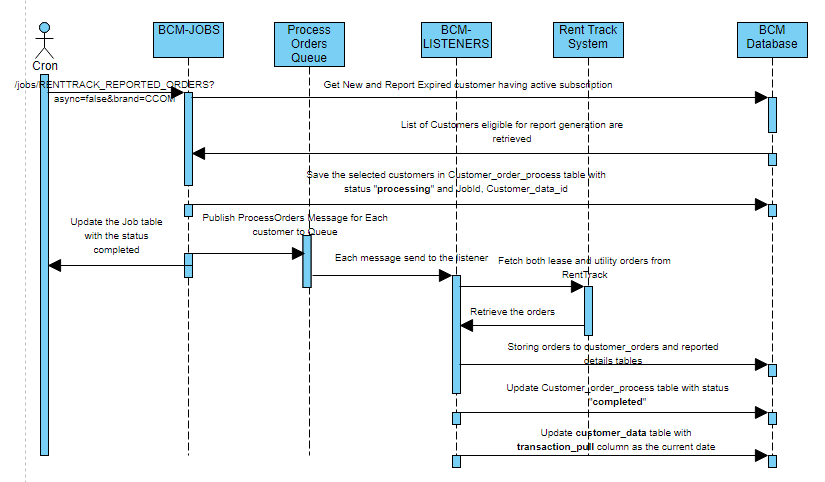
* 1. Blocker Inputs needed

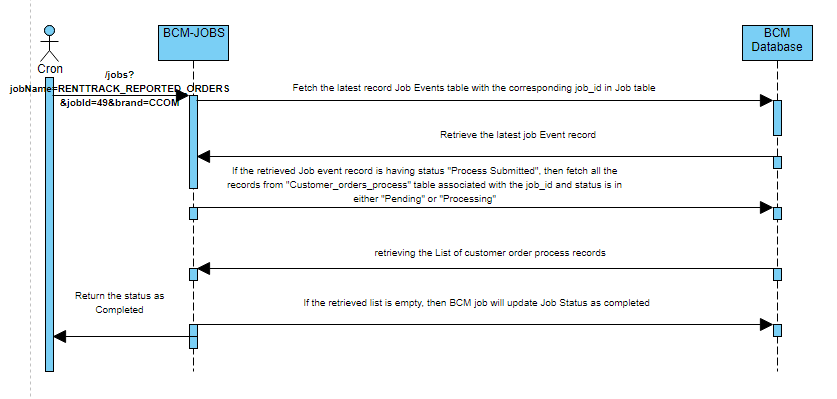
The sandbox access need to be provided

* 1. Design Patterns considered and the chosen one

SOA

g) Sequence Diagram





**Different State of Job**

|  |  |
| --- | --- |
| **State** | **Description** |
| STARTED | The job has been created, but not started |
| CREATED | The job has been started, but is not yet completed |
| COMPLETED | The job completed successfully |
| INTERRUPTED | An interrupt has been registered for the job, but the job has not yet processed the interrupt (the job will soon stop). |
| INTERRUPT REQUESTED | An interrupt has been registered for the job, but the job has not yet processed the interrupt (the job will soon stop). |
| ERRORED | The job did not complete because of an error |
| PROCESS\_SUBMITTED | Order Process Submitted for all the customers |
| ALREADY RUNNING | It will not be captured in BCM database. It will be shown only in the API response if the job has already been running, but is not yet completed. |

**Different State of Customer Order Process**

|  |  |
| --- | --- |
| **State** | **Description** |
| PENDING | The customer order process message in in queue |
| FAILED | Customer order process has been failed |
| ERROR | Error occurred while BCM processing the customer order process message |
| PROCESSING | BCM started processing the customer order process message |
| COMPLETED | Customer order process has been completed successfully |

**Business** :

**BCM Job Module**

**Start Job Service**

The below BCM API will be used to start the Job

POST /v1/jobs/RENTTRACK\_REPORTED\_ORDERS?async=false&brand=CCOM

**Step 1 :** Once BCM Job module received to start job through API call from CronM, BCM will check the last triggered job status in the BCM job table. If there is no active job ( with the job status PROCESS SUBMITTED) in BCM jobs table, then BCM will create and save the new job with the job status as CREATED.

**Step 2** : If the BCM found any active job(with the status PROCESS SUBMITTED), then BCM will validate all the records in the customer\_order\_process table associated with that active job to make sure whether the job has been completed. If BCM founds the job is still active, then no new job will be triggered. BCM will respond back to the invoker with the status that the job is ALREADY RUNNING.

**Step 3:** Once the new job has been started, then the job will be updated with the new status as STARTED.

**Step 4:** BCM will fetch the list of active customers who are all new users to BCM and the users whose transaction pull date is expired. (Expiration duration is configurable value in days)

**Step 5:** Once BCM fetched the list of customers from the previous step, then each Customer will be processed individually. BCM will insert the record into the Customer order process table with the status as PENDING

**Step 6:** Before processing each customer record, BCM Job module will check the job status whether the stop service has been initiated. If the job event is INTERRUPT\_REQUESTED, then BCM job Module stop the Process and add the job Event as INTERRUPTED.

**Step 7:**BCM Job will generate and publish the Customer process order message to the Amazon MQ with the queue name **BCM.Private.BCMProcessOrdersQueue**

Example Message Content :

{

“customerOrderProcessId”:120,

“customerDataId”:10,

“jobId”:1

}

**Step 8:** BCM will capture any exception occurred while processing each customer order and store it in the failed transaction pull table.

**Step 9:** Once all the customer order process messages have been processed, then the job status will be updated as PROCESS\_SUBMITTED in BCM job table.

**Stop Job Service**

The below BCM API will be used to stop the Job

DELETE /v1/jobs?JobName=RENTTRACK\_REPORTED\_ORDERS&JobId=71&brand=CCOM

**Step 1 :** Once BCM received the stop request, then BCM Job module will fetch the job record based on the JobId and Brand details.

**Step 2 :** If the Job is in active state (such as STARTED, CREATED), then the BCM job Module will add the job event status INTERRUPT REQUESTED

**Step 3:**  If the Job is in inactive state (such as INTERRUPT REQUESTED, INTERRUPTED, PROCESS SUBMITTED, COMPLETED), then the BCM job Module will respond back to the invoker with current status of the job.

**Step 4 :** If the job Id is unavailable then BCM Job Module will process the stop request for the latest job and respond back to the invoker with all the jobs.

**Get Status Job Service**

The below BCM API will be used to get the status the Job

GET /v1/jobs?JobName=RENTTRACK\_REPORTED\_ORDERS&JobId=71&brand=CCOM

**Step 1 :** Once BCM received the get status request, then BCM Job module will fetch the job record based on the JobId and Brand details.

**Step 2 :** If the Job is in running state (PROCESS SUBMITTED), then the BCM job Module will fetch the list of customer order process records from the customer\_order\_process table associated with that job and in the customer order process status either in PENDING or PROCESSING.

**Step 3:**  If the BCM job Module is retrieved an empty list, then BCM job Module will add the Job event as COMPLETED and send the response to the invoker.

**Step 4** : If the BCM job Module is retrieved the list of records, then BCM job Module will consider the job is still running and send the current JOB status to the invoker.

**Step 5:** If the job status is other than PROCESS SUBMITTED state, then BCM job Module will respond back to the invoker with the current status of the job.

**Step 5 :** If the job Id is unavailable then BCM Job Module will process the status request for the latest job and respond back to the invoker with all the jobs.

**BCM Messaging Module**

**Step 1:** Once BCM Message Module Job listeners received the customer process order message, then BCM will validate that message.

**Step 2:** BCM Messaging Module will update the customer order process status to PROCESSING for the valid message.

**Step 3:** BCM Messaging Module will get the lease details of the customer by using the below RentTrack API call

GET /api/tenant/leases.{\_format}

**Step 4:** BCM Messaging Module will get the lease orders of the customer for the active lease by using the below RentTrack API call. if lease order founds, then BCM Messaging Module will save the lease orders in the customer\_orders and customer\_orders\_reported\_info tables accordingly

GET /api/tenant/leases/{leaseId}/order.{\_format}

**Step 5:** BCM Messaging Module will get the utility details of the customer by using the below RentTrack API call

GET /api/tenant/utilities.{\_format}

**Step 7:** BCM Messaging Module will get the utility orders of the customer for the active utilitites by using the below RentTrack API call. If any utility order founds, then BCM will save the utility orders in the customer\_orders and customer\_orders\_reported\_info tables accordingly.

GET /api/tenant/utility/{utilId}/order.{\_format}

**Step 8 :** BCM will update the customer process status as COMPLETED once the process has been completed successfully.

**Step 9:** If any exception occurred during the order process, it will be captured in both

failed transaction pull table with the detailed error message and in customer order process table.

**Clarifications**:

As per the demo comments, we have included the meesaging system in job processing.

**Code / Framework References**

**Misc:**