The initial approach to come up with four individual functions for each rule

Contents

[Abstract 1](#_Toc113278839)

[Four Functions per rule approach 1](#_Toc113278840)

[Pseudocode approach 2](#_Toc113278841)

[c4c\_itsetstatus\_update 2](#_Toc113278842)

# Abstract

This document covers the key aspects to refactor the POC’s under C4C-SBS-Extensions Atlas MongoDB app services. This design aims at three different POC’s to have four functions to be created for update, get, patch, and execute rule function. Individual functions will be created specific to the POC or a rule.

# Four Functions per rule approach

**Feature functions for every POC.**

* The **update** method for every POC/rule will act as the main function. The update method first fetches required information from a ticket record using a get call to c4c, form the required request object using the fields from ticket record and fires the rule engine execution. The rule engine give back match found/not match found responses. If the rule execution identifies a match found, then the designated field will be updates to the required value.
* The **get** method is used to fetch the ticket record for an entity id passed in the request and pick only required fields from ticket record to send to rule engine.
* The rule engine **execute** function will first fetch the rule object from MongoDB collection for a rule feature. After that map the required fields for a rule object by matching from ticket record and for the request object to send to rule engine for execution. The rule engine response will be returned to execute function.
* The **patch** method using **context.http.patch** depending on the data to be patched for a POC when the rule match found.
* Once the patch operation succeeded make sure log the status based on the status of patch operation completed
* An example pseudocode for a POC is given below.

# Pseudocode approach

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Parameters** | **Route** | **Psuedocode** |
| c4c\_itsetstatus\_update | request, response | /c4c\_itsetstatus\_update | 1. This will be the only function which takes care of get the ticket record from c4c APIM endpoint and execute the rule for the data availed and patch the delta changes back to c4c for a specific column depending upon a POC functionality.  2. Fire the context.http.get call to APIM get endpoint using ticket\_root\_entityIdIn value and for further JSON iteration use the c4cResponse.d.results[0].  3. After getting the ticket response and only required fields the same will be passed to execute rule function.  4. Based on the rule engine match found status, the Patch function will be called to update the ticket record field for a rule. |
| c4c\_itsetstatus\_get | ticket\_root\_entityIdIn | /c4c\_itsetstatus\_get | This function should implement the get call using await context.http.get to fetch the ticket record and send the response back to update function. |
| c4c\_itsetstatus\_executerule | Request, Response | /c4c\_itsetstatus\_executerule | 1. Form the input data from the ticket record response availed through the get method as follows . c4cResponse.d.results[0] object like below which can be dynamic as below.  let inputData = {};  if (data && data.attributes && data.attributes.length > 0) {  data.attributes.map((element) => {  if (element.name === 'CountryText') {  const ServiceRequestUsedAddress = request['ServiceRequestUsedAddress'];  console.log("ServiceRequestUsedAddress", JSON.stringify(ServiceRequestUsedAddress));  inputData[element.name] = ServiceRequestUsedAddress[element.name] ? ServiceRequestUsedAddress[element.name] : '';  } else {  inputData[element.name] = request[element.name] ? request[element.name] : '';  }  console.log("inputData", element.name, request[element.name]);  });  }  2. Execute the rule engine against the input and respond with rule match found status. |
| c4c\_itsetstatus \_patch | Request, Response | /c4c\_itsetstatus \_update | 1. Based on the rule engine match found status, the update function will make a call here. Execute the patch method using context.http.patch depending on the data to be patched for a POC when the rule match found.  2. Once the patch operation succeeded make sure log the status based on the status of patch operation completed |
| Other POC’s will have same four functions | ---DO----- | Define routes | Define implementation of function. |