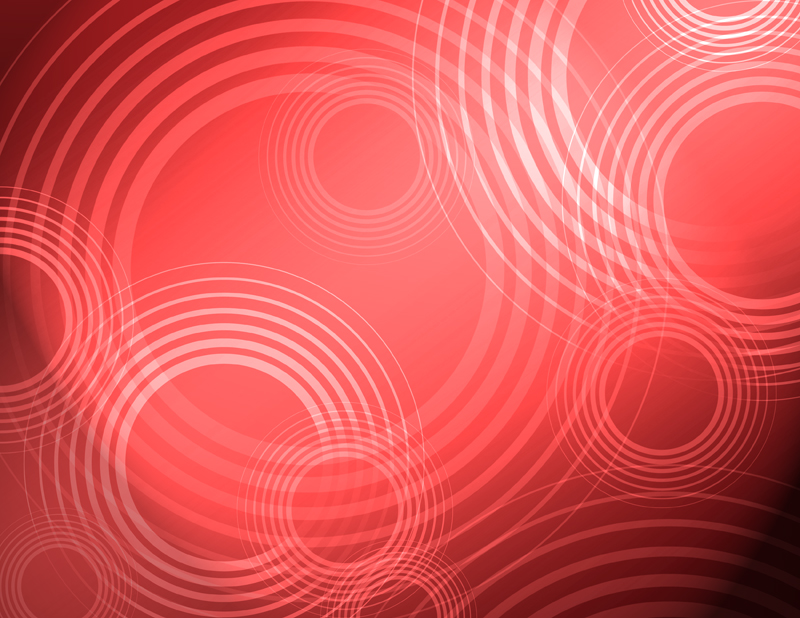


Cable Serviceability API

API Design Document

Rogers Published: 28 July 2021

Owner: Rogers Communication Inc Version Number: 1.0



**Version Control Notice**

This document is a controlled issue that supersedes all previous issues. Please discard any previous copy of this document dated prior to the version and publication date noted on this page.

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Name** |
| 28-Jul-21 | 1.0 | Cable Serviceability Process API | APIGW Team |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Document Sign-off**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Title** | **Email** | | **Phone** |
| Taqiuddin Mohammed | Tech Lead | Taqiuddin.mohammed@rci.rogers.com | |  |
| Amir Segev | BSA | | amir.segev@rci.rogers.com |  |

**Document Owner**

The primary contact for questions regarding this document is:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Title** | **Email** | **Phone** |
| Amir Segev | BSA | amir.segev@rci.rogers.com |  |
| Taqiuddin Mohammed | Tech. Lead | Taqiuddin.mohammed@rci.rogers.com |  |

**Contributors**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Title** | **Email** | **Phone** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**The information contained is confidential and proprietary to Rogers Communications Inc. and may not be used, reproduced or disclosed to others except as specifically permitted in writing by a Senior Officer of Rogers Communications Inc. By receiving and using this information, you agree to protect the same from loss, theft, unauthorized use and dissemination on behalf of Rogers Communications Inc., and to report any such loss, theft or unauthorized use in writing to the General Counsel of Rogers Communications Inc. so that appropriate legal action can be taken.**

**CONTENTS**

[1 Document Details 5](#_Toc78363567)

[1.1 Purpose 5](#_Toc78363568)

[1.2 Document Abbreviations 5](#_Toc78363569)

[2 Solution Design 5](#_Toc78363570)

[2.1 High Level Solution 5](#_Toc78363571)

[2.1.1 Solution Diagram 5](#_Toc78363572)

[2.2 Assumptions 5](#_Toc78363573)

[2.3 Constraints 6](#_Toc78363574)

[2.4 Dependencies 6](#_Toc78363575)

[2.5 Non-Functional Requirements 6](#_Toc78363576)

[3 Solution Implementation 7](#_Toc78363577)

[3.1 Cable Serviceability API 7](#_Toc78363578)

[3.1.1 JIRA Story / Requirement Details 7](#_Toc78363579)

[3.1.2 Low Level Design – Cable Serviceability Exp API 8](#_Toc78363580)

[3.1.3 API Development details 8](#_Toc78363581)

[3.1.3.1 Implementation Details 8](#_Toc78363582)

[3.1.4 Resource Details <if any> 9](#_Toc78363583)

[3.1.5 API Request / Response 9](#_Toc78363584)

[3.1.6 Error Handling 12](#_Toc78363585)

[3.1.7 Policy Details 13](#_Toc78363586)

[3.1.8 Other Requirements <As applicable> 13](#_Toc78363587)

[3.2 Cable Serviceability Process API 14](#_Toc78363588)

[3.2.1 JIRA Story / Requirement Details 14](#_Toc78363589)

[3.2.2 Low Level Design – Cable Serviceability Process API 14](#_Toc78363590)

[3.2.3 API Development Details 15](#_Toc78363591)

[3.2.3.1 Implementation Details 15](#_Toc78363592)

[3.2.3.2 Error Handler 16](#_Toc78363593)

[3.2.4 Resource Details 18](#_Toc78363594)

[3.2.5 API Request / Response 18](#_Toc78363595)

[3.2.6 Error Handling 20](#_Toc78363596)

[3.2.7 Policy Details 20](#_Toc78363597)

[3.2.8 Other Requirements <As applicable> 21](#_Toc78363598)

[3.3 Cable Serviceability System API 21](#_Toc78363599)

[3.3.1 JIRA Story / Requirement Details 21](#_Toc78363600)

[3.3.2 Low Level Design – Cable Serviceability Sys API 22](#_Toc78363601)

[3.3.3 API Development details 22](#_Toc78363602)

[Fig 1.1 22](#_Toc78363603)

[3.3.4 Resource Details <if any> 23](#_Toc78363604)

[3.3.5 API Request / Response 23](#_Toc78363605)

[3.3.6 Error Handling 24](#_Toc78363606)

[3.3.7 Policy Details 25](#_Toc78363607)

[3.3.8 Other Requirements <As applicable> 25](#_Toc78363608)

[4 Resource Requirements 26](#_Toc78363609)

[5 Code Repository 26](#_Toc78363610)

[6 Deployment Automation Requirements 27](#_Toc78363611)

[7 Test Case Details 27](#_Toc78363612)

# ****Document Details****

## ****Purpose****

This is the API Design Document developed by the APIGW (REAL) team for the API that they have developed for internal/external customers. The Design Document, along with the Project Solution Architecture (PSA), are the core artifacts produced in the elaboration phase of the SDLC. In combination, they describe the solution as requested by the project owner as well as the technical response from the IT team.

## ****Document Abbreviations****

List abbreviations and acronyms used in this document:

| **Abbreviation** | **Definition** | **Context   (how/when it’s used)** |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

# ****Solution Design****

## ****High Level Solution****

<< Provide High Level Solution>>

### Solution Diagram

<Please provide the Component diagram along with sequence diagram, explaining the overall solution including the one in scope part (you may use draw.io to create the diagrams). Please delete this line after document finalization.>

## ****Assumptions****

<Please provide the assumptions taken for the overall solution and for all the systems involved (if any). Please delete this line after document finalization.>

Highlight all assumptions for each of the design areas. Include whether the assumption has been validated and the potential impact of each assumption.

| **ID** | **Assumption** | **Application** | **Assumption Validated (Yes/No)** | **Due Date**  **(if assumption is not validated)** | **Impact**  **(if assumption is not validated)** |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |

## ****Constraints****

<Please provide the constrains considered for the overall solution and for all the systems involved (if any). Please delete this line after document finalization.>

A constraint is an external limitation imposed on the project that limits and/or drives the defined solution. It is an uncontrollable or unchangeable fact to which the project solution must abide.

| **ID** | **Constraints** | **Impacted Stakeholders** |
| --- | --- | --- |
|  |  |  |
|  |  |  |

## ****Dependencies****

<Please provide the dependencies considered for the overall solution and for all the systems involved (if any). Please delete this line after document finalization.>

A dependency is a situation where the outcome of another project is materially affected by this project (outbound dependency) or where this project cannot be completed unless a related project provides a certain capability (inbound dependency).  See below for Application Decommissioning dependencies.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Dependency** | **Dependency Type (Inbound/Outbound)** | **Impacted Stakeholders** | **Dependency Owner** |
| 1 |  |  |  |  |
| 2 |  |  |  |  |

## ****Non-Functional Requirements****

<Please provide the Non-Functional requirements for the overall solution and for all the systems involved (if any). Please delete this line after document finalization.>

| **NFR** | **NFR** | **Details** |
| --- | --- | --- |
| Performance | Response Time, Throughput, Utilization, Static Volumetric etc. |  |
| Security | Security Requirement– SSL/WAF |  |
| Error Report requirements | <if error report needs to be generated for this API, capture the required details here> |  |

# ****Solution Implementation****

## ****Cable Serviceability API****

| **API Details** | |
| --- | --- |
| **API/Service Title** | Cable Serviceability api |
| **API/Service Version** | V1 |
| **API Methods** | POST |
| **Required headers** | client\_id, client\_ref |
| **Layer** | Experience |
| **Usage** | Synchronous |
| **URI** | TEST: <https://cableserviceability-api-qa2.rogers.com/api/v1/servicequalification>  Prod: <https://cableserviceability-api.rogers.com/api/v1/servicequalification> |
| **Calling System/s** | SDFC |
| **Backend System/s** | SSG |

### ****JIRA Story / Requirement Details****

<https://reqcentral.com/browse/APGW-11050>

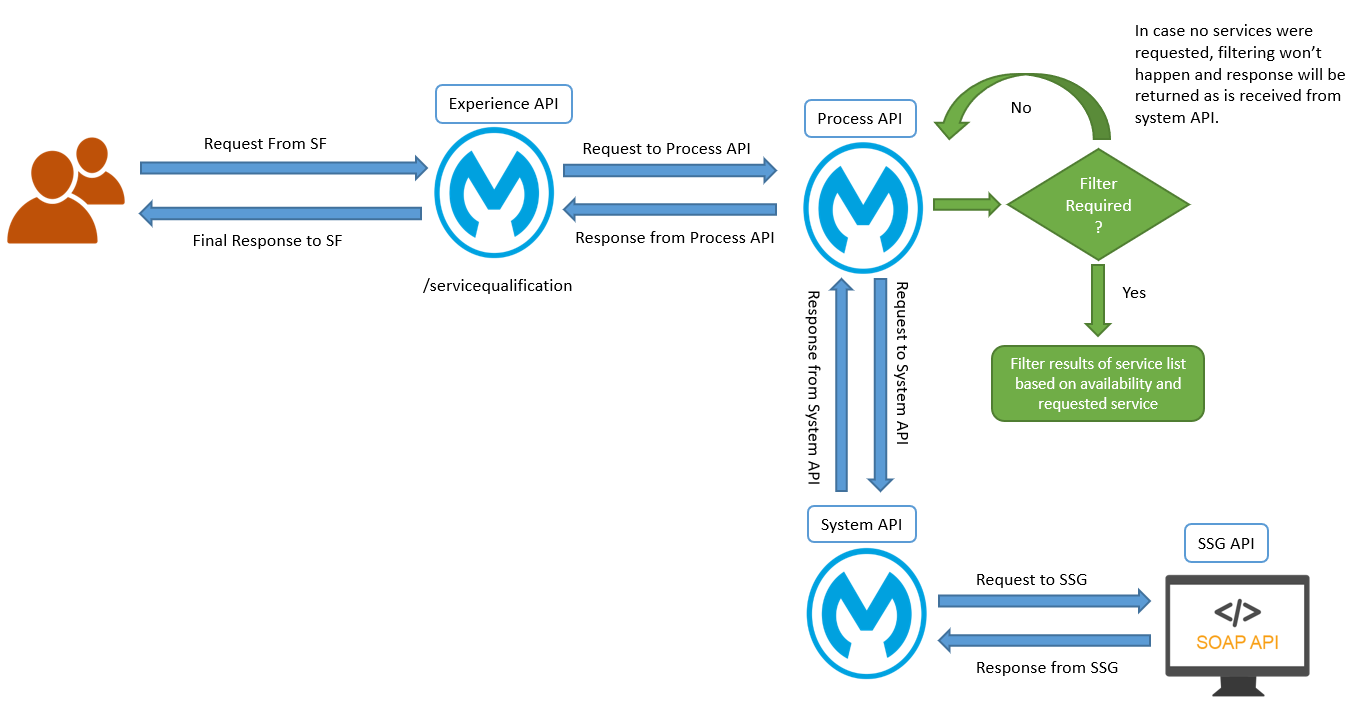
As a tester, I need to validate that an API is available for SFDC for serviceability.

1. Exp API -  mandatory fields: postcode, streetNr, streetName
2. Proc API - with the below filtering, and mapping of fields

Filtering will happen only when the list of Services is populated:

1. Only addresses and services from the requested list that are available (services are not case sensitive). The response should only include the Y available services. if an address is part of a multi-dwelling some unit may not exist in the response if no requested service is available.
2. If the address list is empty of available services an Empty response will be sent.

### ****Low Level Design – Cable Serviceability Exp API****



### ****API Development details****

#### Implementation Details

This section will be discussing the main steps of the implementation. Necessary supporting flows are discussed in separate sections and will be refferred from this section.

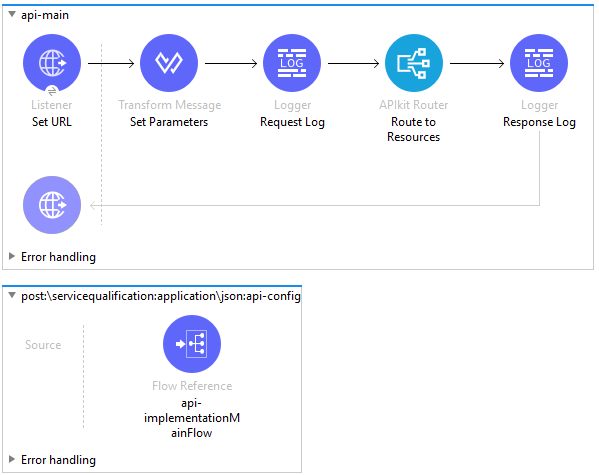


Fig 3.1.3.1.1 Main Flow

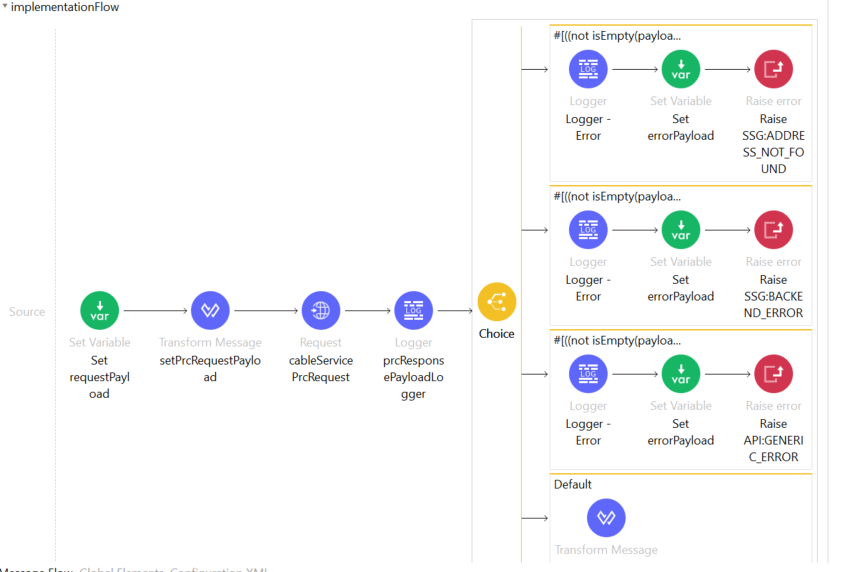
**Step 1:** Request from client is received through HTTP Listener.

**Step 2:** We set few variables from initial request (i.e. transactionId, clientRef etc.). These variables will be used in subsequent flows.

**Step 3:** APIKit Router is responsible for validating the request against the RAML and route to the flow designated for particular endpoint. In this scenario, it will be routing the request to POST \servicequalification which in turn calls the flow that does the main functionality of the implementation.

**Step 4:** At the implementation we for the payload sent to the process APi and call the Process API, all the responses coming for the process is accepted. At the choice block checking for the respone coming from the backend based on the response if the response contains related address not found, raising the error for it and sent to the global error handler error response mapped according to the error, if any backend error coming from the SSG it will be handled at second stage of the choice block and raised error accordingly. At the third block we handle all the process and system API technical errors.

For error handling refer to 3.1.6 section.



### ****Resource Details <if any>****

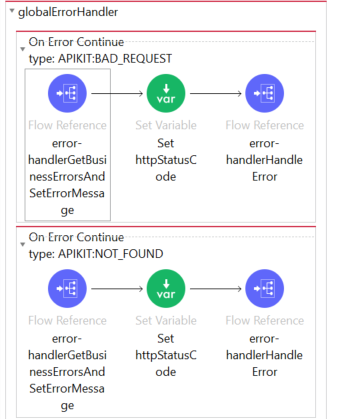
Resource Endpoint: POST /servicequalification

### ****API Request / Response****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **API Name** | **Operations List** | **Backend** | **APIType** | **Consumer** |
| cableserviceability-api | /servicequalification | SSG | EXP | SFDC |
| Sample Request:  {      "address": {          "streetNr": "90",          "streetNrSuffix": "",          "streetSuffix": "",          "streetName": "ADELAIDE",          "streetType": "St",          "subUnitNumber": "",          "postcode": "M5C2R4",          "city": "Toronto",          "stateOrProvince": "ON",          "country": "Canada"      },      "service": ["HDTV","D3"]  } | | | | |
| Sample Response:  {  "transactionId": "123456789",  "serviceName": "cableserviceability-api",  "clientRef": "56789",  "dateTime": "2021-05-03T13:20:49-04:00",  "address": [  {  "streetNr": "90",  "streetNrSuffix": "",  "streetName": "ADELAIDE",  "streetSuffix": "E",  "streetType": "ST",  "subUnitNumber": "202",  "postcode": "M5C2R4",  "city": "TORONTO",  "stateOrProvince": "ON",  "country": "",  "cableCondition": "Hot",  "contractType": "2N1",  "serviceAddressInd": "YES",  "sharedInd": "YES",  "fraudInd": "NO",  "inTerritoryInd": "YES",  "onPlantInd": "YES",  "dwellingType": "",  "dwellingTypeDesc": "",  "franchiseArea": "",  "GPONHub": "",  "GPONNap": "",  "GPONNode": "",  "mapArea": "",  "feedType2": "",  "fiberCondition": "",  "accountNum": "115404201",  "areaCode": "TOR",  "subscriberStatus": "5",  "rateCentre": "TOR",  "municipalityCode": "001",  "channeLineUp": "65",  "serviceAddressID": "2300000443108",  "addressUniqueID": "2035597011",  "service": [  {  "serviceType": "C",  "name": "D3",  "startDate": "2013-01-23",  "dateType": "A",  "availability": "Y"  },  {  "serviceType": "C",  "name": "HDTV",  "startDate": "2012-12-20",  "dateType": "A",  "availability": "Y"  }  ]  },    ]  } | | | | |

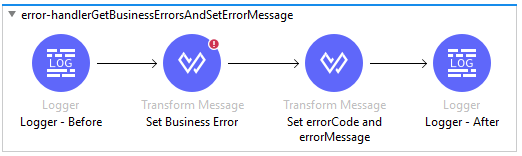
### ****Error Handling****

As like other APIs, this API also contains Global Error Handler in place which is responsible for handling any errors occurred within the experience API or propagated from underlying calling Process API.



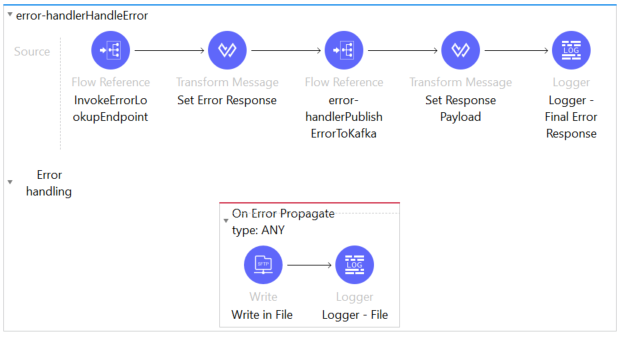
A typical error handler block usually does the following steps in order to create a meaningful error response for the consumer:

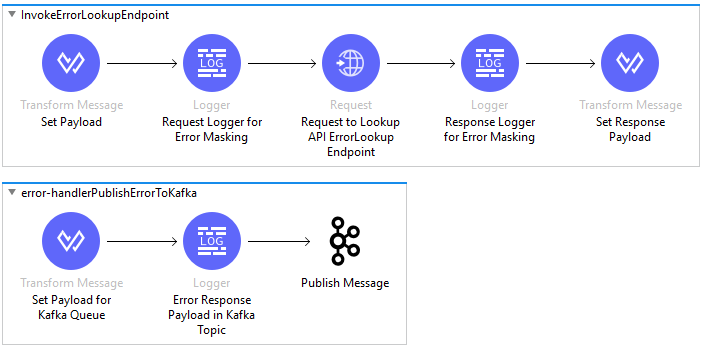
**Step 1:** First step is trying to get the errors from Process API (if any) and set the error message and error code accordingly which involves following flow that does the intended operation.



First thing this flow does is extracting error details from Process API response if applicable and then set the errorCode and errorMessage based on the Process API error.

**Step 2:** Next step is calling errorHandlerHandleError sub flow which does most of the functionalities related to error handler. We will discuss about this flow-





we are calling InvokeErrorLookupEndpoint flow to get the error masking data.

Next thing in the flow is to set the error response based on the available information.

Next step is pushing the error payload to Kafka based on the flag skipKafkaPublish. If that flag is true, then we have to call PublishErrorToKafka flow which will push the error payload to Kafka.

### ****Policy Details****

Client Id enforcement policy is implemented.

### ****Other Requirements <As applicable>****

|  |  |  |
| --- | --- | --- |
| **Item** | **Action** | **Details** |
| Database Table Requirements | N |  |
| Error Definition Table Update | Y |  |
| DNS/Akamai Request details | Y |  |
| Kafka Queue Requirement | Y |  |
| SSL Certificate Requirement | N |  |

## ****Cable Serviceability Process API****

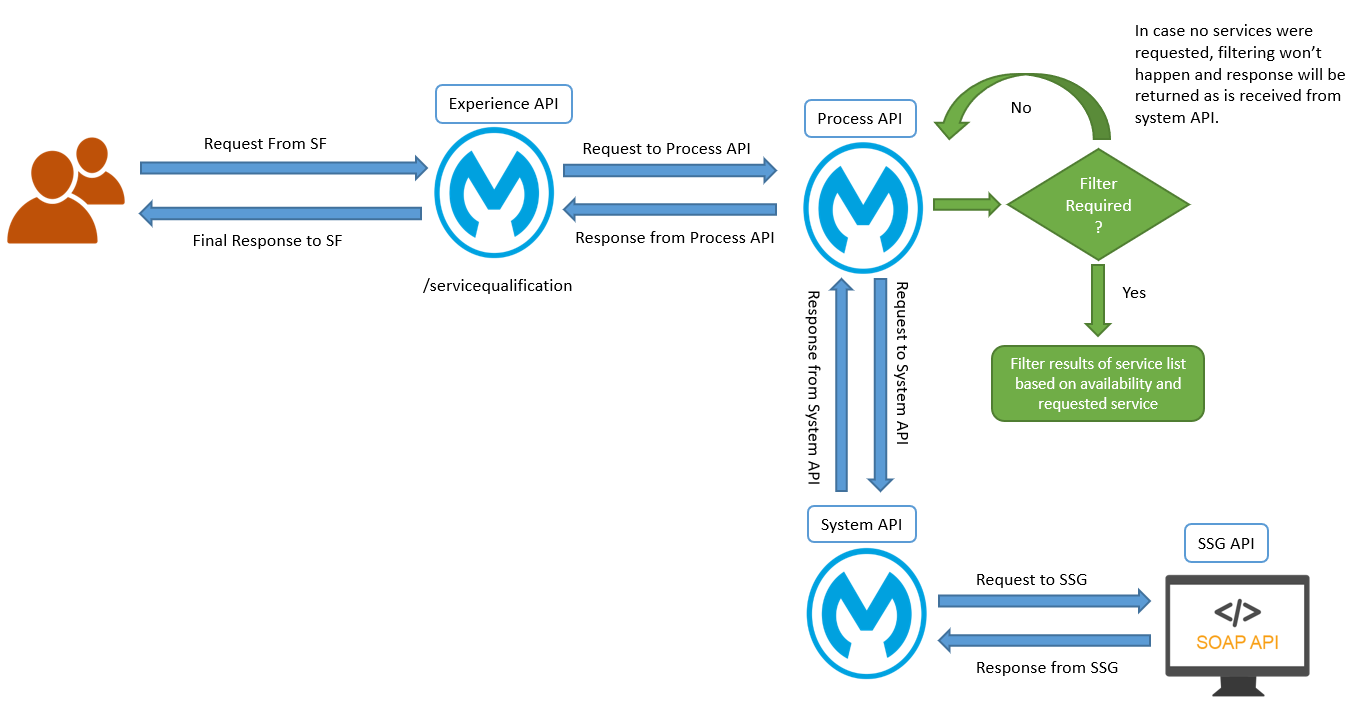
| **API Details** | |
| --- | --- |
| **API/Service Title** | cableserviceability-api |
| **API/Service Version** | v1 |
| **API Methods** | POST |
| **Required headers** | N/A |
| **Layer** | Process |
| **Usage** | Synchronous |
| **URI** | Test: https://cableserviceability-api-qa2.rogers.com/api/v1/  Prod: https://cableserviceability-api.rogers.com/api/v1/ |
| **Calling System/s** | Listener Endpoint |
| **Backend System/s** | SSG |

### ****JIRA Story / Requirement Details****

<https://reqcentral.com/browse/APGW-11051>

### ****Low Level Design – Cable Serviceability Process API****

This API exposes a POST endpoint /servicequalification to the client which in turn calls the process API.



### ****API Development Details****

#### Implementation Details

This section will be discussing the main steps of the implementation. Necessary supporting flows are discussed in separate sections and will be refferred from this section.

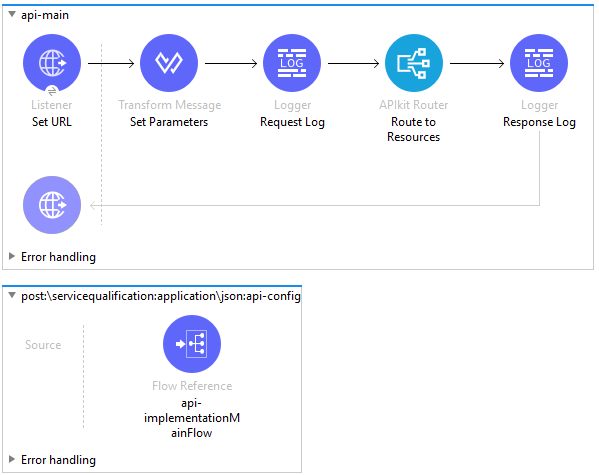


Fig 3.1.3.1.1 Main Flow

**Step 1:** Request from client is received through HTTP Listener.

**Step 2:** We set few variables from initial request (i.e. transactionId, clientRef etc.). These variables will be used in subsequent flows.

**Step 3:** APIKit Router is responsible for validating the request against the RAML and route to the flow designated for particular endpoint. In this scenario, it will be routing the request to POST \servicequalification which in turn calls the flow that does the main functionality of the implementation.

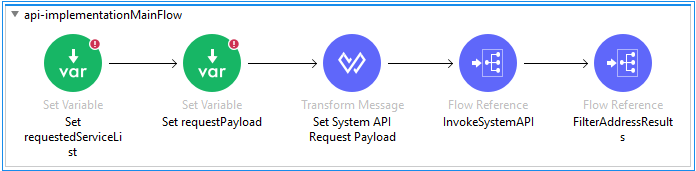


Fig 3.1.3.1.2 Serviceability Implementation

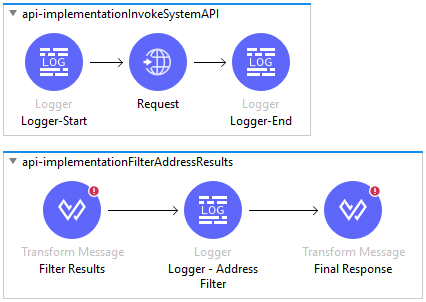


Fig 3.1.3.1.3 Invoke System API and Filter Address Results

**Step 4:** We set requestedServiceList variable to service array from the request. We also set requestPayload variable that contains the actual request payload coming in.

**Step 5:** We set the request for System API

**Step 6:** We call invokeSystemAPI flow which in turn calls the System API.

**Step 7:** If we get the response from System API, we filter the results by calling FilterAddressResults flow if particular service(s) were requested at the time of transaction. This filter logic filters the response and returns only the addresses that has available services which were requested. If no services were requested, response is sent as is without any filtering.

#### Error Handler

As like other APIs, this API also contains Global Error Handler in place which is responsible for handling any errors occurred within the system API or propagated from underlying backend system.

##### **Global Error Handler**

As a part of global error handler, we have two types of error handler block. Typically, an error handler which handles APIKIT errors looks like below and involves the following steps:

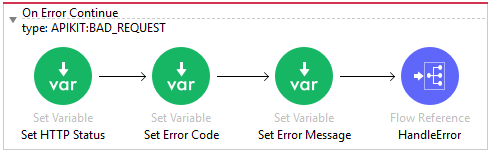


Fig 3.1.3.2.1.1 APIKIT Error Block

**Step 1**: First step is set the httpStatus, errorCode and errorMessage variable which will be used in further steps.

* httpStatus: This variable is set to the http status for that particular error type.
* errorCode: This variable is set to masked error code from the configuration file.
* errorMessage: This variable is set to the error message from the configuration file for this error block.

**Step 2:** Next step is calling HandleError flow which does field mapping related to particular error handler. We will discuss about this flow-

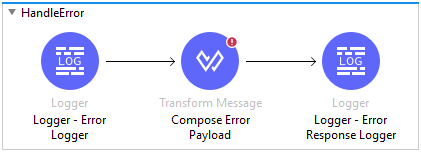


Fig 3.1.3.2.1.2 Handle Error Flow

In this flow, we compose the error response according to the standards based on the information available for operations.

In addition to the APIKIT error handler, we also have error handlers that is focused more into handling HTTP errors. This error handler basically handles the error coming from backend system or any error occurs while calling the backend system. We will discuss about the steps below:

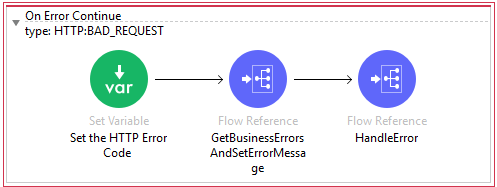


Fig 3.1.3.2.1.3 HTTP Error Block

**Step 1:** First step is setting up http status variable for the particular error type.

**Step 2:** Next step is getting business errors. This calls error-handlerGetBusinessErrorsAndSetErrorMessage flow which tries to extract any errors coming from backend system itself and set proper variables.

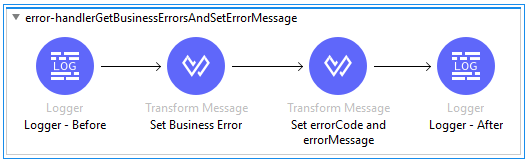


Fig 3.1.3.2.1.4 Get Business Errors Flow

**Step 2:** Next step is setting up errorCode and errorMessage variable leveraging the details from Step 1 (if applicable) or defaults to the defined error code and error message for the particular error type.

**Step 3:** Final step is calling HandleError flow which does field mapping related to particular error handler. We will discuss about this flow-

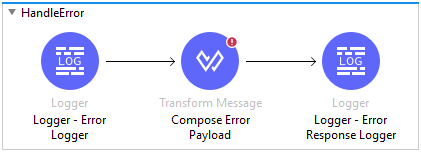


Fig Fig 3.1.3.2.1.5 Handle Error Flow

In this flow, we compose the error response according to the standards based on the information available for operations.

### ****Resource Details****

Resource Endpoint: POST /servicequalification

Implementation is described in Section 3.1.3 **API Development Details.**

### ****API Request / Response****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **API Name** | **Operations List** | **Backend** | **API Type** | **Consumer** |
| cableserviceability-api | /servicequalification | System API | **Experience API** | **R4B** |
| Sample Request:  {      "address": {          "streetNr": "21",          "streetNrSuffix": "",          "streetName": "ARDEN",          "streetSuffix": "",          "streetType": "ST",          "subUnitNumber": "9",          "postcode": "E1C4B5",          "city": "MONCTON",          "stateOrProvince": "NB",          "country": "Canada"      },      "service": []  } | | | | |
| Sample Response:  {      "transactionId": "8a54d598-713c-4110-adde-18b27fd916e5",      "serviceName": "cableserviceability-api",      "clientRef": "REAL Test",      "dateTime": "2021-05-03T11:32:11-04:00",      "address": [          {              "streetNr": "21",              "streetNrSuffix": "",              "streetName": "ARDEN",              "streetSuffix": "",              "streetType": "ST",              "subUnitNumber": "9",              "postcode": "E1C4B5",              "city": "MONCTON",              "stateOrProvince": "NB",              "country": "",              "cableCondition": "Hot",              "contractType": "1A0",              "serviceAddressInd": "YES",              "sharedInd": "YES",              "fraudInd": "NO",              "inTerritoryInd": "YES",              "onPlantInd": "YES",              "dwellingType": "",              "dwellingTypeDesc": "",              "franchiseArea": "",              "GPONHub": "",              "GPONNap": "",              "GPONNode": "",              "mapArea": "",              "feedType2": "",              "fiberCondition": "",              "accountNum": "061544604",              "areaCode": "MON",              "subscriberStatus": "5",              "rateCentre": "MNA",              "municipalityCode": "363",              "channeLineUp": "30",              "serviceAddressID": "2600000417864",              "addressUniqueID": "2036649227",              "service": [                  {                      "serviceType": "C",                      "name": "D3",                      "startDate": "2010-05-13",                      "dateType": "A",                      "availability": "Y"                  },                  {                      "serviceType": "C",                      "name": "DPS",                      "startDate": "2006-07-01",                      "dateType": "A",                      "availability": "Y"                  }              ]          }      ]  } | | | | |

### ****Error Handling****

Please refer to Section 3.1.3.2 Error Handler.

### ****Policy Details****

Client Id policy is enforced.

### ****Other Requirements <As applicable>****

|  |  |  |
| --- | --- | --- |
| **Item** | **Action** | **Details** |
| Database Table Requirements | N |  |
| Error Definition Table Update | N |  |
| DNS/Akamai Request details | Y |  |
| Kafka Queue Requirement | N |  |
| SSL Certificate Requirement | N |  |

## ****Cable Serviceability System API****

| **API Details** | |
| --- | --- |
| **API/Service Title** | Cable Serviceability System API |
| **API/Service Version** | 1.0 |
| **API Methods** | POST |
| **Required headers** | transactionId, client\_ref |
| **Layer** | System |
| **Usage** | Synchronous |
| **URI** | TEST: <https://cableserviceability-sys-api-qa2.rogers.com/api/v1/servicequalification>  Prod: <https://cableserviceability-sys-api.rogers.com/api/v1/servicequalification> |
| **Calling System/s** | Process |
| **Backend System/s** | SSG |

### ****JIRA Story / Requirement Details****

<https://reqcentral.com/browse/APGW-11050>

### ****Low Level Design – Cable Serviceability Sys API****

<Please provide the technical solution details and diagrams/flows for the API. The steps are to be provided for each flow. Please delete this line after document finalization.>

### ****API Development details****

Step 1: Transform JSON to XML

At this step the payload is transformed from a JSON to xml

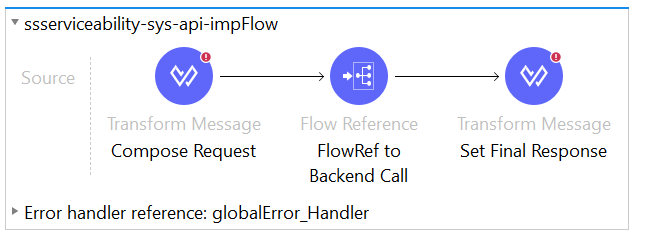
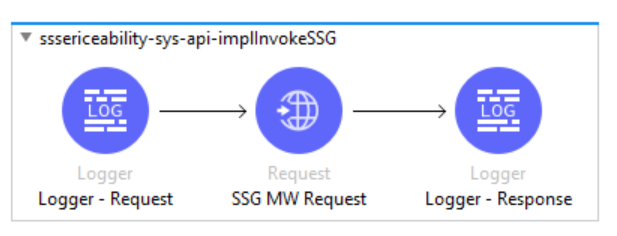


Fig 1.0

Step 2: Make request to SSG MW

A request is sent to SSG MW with the address in XML format



#### Fig 1.1

Step 3: Transform XML to JSON

Transform the first element of the response sent from SSG from XML to JSON

### ****Resource Details <if any>****

<Please provide the resource details and related solution details and diagrams/flows for the API. The steps are to be provided for each flow. Please delete this line after document finalization.>

### ****API Request / Response****

<Please provide the API details in the below format. Please delete this line after document finalization.>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **API Name** | **Operations List** | **Backend** | **APIType** | **Consumer** |
|  |  |  |  |  |
| Sample Request:  {      "address": {          "streetNr": "11",          "streetNrSuffix": "",          "streetSuffix": "",          "streetName": "shiff",          "streetType": "cres",          "subUnitNumber": "",          "postcode": "L6Z0B4",          "city": "Brampton",          "stateOrProvince": "ON",          "country": "Canada"      }  } | | | | |
| Sample Response:  {  "transactionId": "123456789",  "serviceName": "cableserviceability-sys-api",  "clientRef": "werty2345678",  "dateTime": "2021-05-04T15:16:14.083-04:00",  "success": "Success",  "serviceAddressList": [  {  "AddressInfo": {  "StreetNumber": "90",  "StreetNumberSuffix": null,  "StreetName": "ADELAIDE",  "StreetType": "ST",  "CompassDirection": "E",  "Unit": "202",  "PostalCode": "M5C2R4",  "City": "TORONTO",  "Province": "ON",  "CableCondition": "Hot",  "ContractType": "2N1"  },  "Indicators": {  "ServiceAddressInd": "YES",  "SharedInd": "YES",  "FraudInd": "NO",  "InTerritoryInd": "YES",  "OnPlantInd": "YES",  "DwellingType": null,  "DwellingTypeDesc": null,  "FranchiseArea": null,  "GPONHub": null,  "GPONNap": null,  "GPONNode": null,  "MapArea": null,  "FeedType2": null,  "FiberCondition": null  },  "AccountInfo": {  "AccountNumber": "115404201",  "AreaCode": "TOR",  "SubscriberStatus": "5",  "RateCentre": "TOR",  "MunicipalityCode": "001",  "ChanneLineUp": "65"  },  "AddressUniqueKey": {  "ServiceAddressID": "2300000443108",  "AddressUniqueID": "2035597011"      ]  } | | | | |

### ****Error Handling****

Error Handler

Any error coming from backend, mule errors or RAML errors are handled it in the error handler flow and enriched to the standards and sent to calling flow.



Fig 1.2

### ****Policy Details****

<Please provide the policy details in this section. Please delete this line after document finalization.>

### ****Other Requirements <As applicable>****

|  |  |  |
| --- | --- | --- |
| **Item** | **Action** | **Details** |
| Database Table Requirements | N |  |
| Error Definition Table Update | N |  |
| DNS/Akamai Request details | N |  |
| Kafka Queue Requirement | N |  |
| SSL Certificate Requirement | Y |  |

# ****Resource Requirements****

Based on the non-functional requirements (section 2.5above), please provide the following details:

|  |  |  |
| --- | --- | --- |
| ****Env**** | ****Item**** | ****Value**** |
| DEV | Reserved CPU | 0.1 |
| CPU limit | 0.1 |
| Memory | 1.0 |
| TEST | Reserved CPU | 0.1 |
| CPU limit | 0.1 |
| Memory | 1.0 |
| PROD | Reserved CPU | 0.6 |
| CPU limit | 1 |
| Memory | 4 |

# ****Code Repository****

|  |  |
| --- | --- |
| **Repository** | **Details/Location** |
| Github | <https://github.com/RogersCommunications/cableserviceability-api>  <https://github.com/RogersCommunications/cableserviceability-api>  <https://github.com/RogersCommunications/cableserviceability-sys-api> |

# ****Deployment Automation Requirements****

|  |  |
| --- | --- |
| **Automation** | **Details/Location** |
| Jenkins | [http://10.73.18.33:8080/job/<ENV-NAME>/job/cableserviceability-api-<env-name>/](http://10.73.18.33:8080/job/%3cENV-NAME%3e/job/cableserviceability-api-%3cenv-name%3e/)  [http://10.73.18.33:8080/job/<ENV-NAME>/job/cableserviceability-prc-api-<env-name>/](http://10.73.18.33:8080/job/%3cENV-NAME%3e/job/cableserviceability-prc-api-%3cenv-name%3e/)  [http://10.73.18.33:8080/job/<ENV-NAME>/job/cableserviceability-sy-api-<env-name>/](http://10.73.18.33:8080/job/%3cENV-NAME%3e/job/cableserviceability-sy-api-%3cenv-name%3e/) |

# ****Test Case Details****

**Test case report link:**

<https://reqcentral.com/wiki/display/APGW/APGW-11050>