**Steps to debug RPA configuration**

Table of Contents

1. [**RPA overview** 2](#_RPA__Overview)
2. [**Debugging connectivity from RPA to SapphireIMS server** 2](#_Toc52992180)

* [Steps to debug 2](#_Toc52992181)
* [RPA connectivity configurations 4](#_Toc52992182)

1. [**Debugging ticket/asset operations (delegate expression tasks)** 4](#_Toc52992183)

* [Steps to debug 4](#_Toc52992184)
* [Tables involved 5](#_Toc52992185)

1. [**Debugging query tasks used in RPA 5**](#_Toc52992186)

* [Steps to debug 5](#_Toc52992187)
* [Configuration tables 6](#_Toc52992188)

1. [**Debugging webhook tasks** 6](#_Toc52992189)

* [Steps to debug 6](#_Toc52992190)
* [Configuration tables 7](#_Toc52992191)

1. [**Debugging DMN and RPA process** 8](#_Toc52992192)

* [DMN overview 8](#_Toc52992193)
* [Steps to debug 8](#_Toc52992194)
* [Configuration table 8](#_Toc52992195)

1. [**Debugging RPA Process** 9](#_Toc52992196)

* [Steps to debug 9](#_Toc52992197)
* [Configuration table 9](#_Toc52992198)

# RPA overview

RPA is an automation framework by using which various automation use cases like ticket/asset operations, workflow management, eventual data synchronization, integrating third party APIs / data, etc., can be achieved.

RPA framework is built with following key technologies

* BPMN engine ([More about BPMN](https://camunda.com/bpmn/))
* DMN engine ([More about DMN](https://camunda.com/dmn/))
* Apache velocity engine ([More about velocity engine](https://velocity.apache.org/engine/1.7/getting-started.html))
* JSON path [(More about JSONPath)](https://jsonpath.com/)
* Cron expressions for timer definition ([More about Cron expressions](http://www.quartz-scheduler.org/documentation/quartz-2.3.0/tutorials/tutorial-lesson-06.html))

RPA is available as an executable plug-in which will install an independent dedicated server and separate schema (rpa). The installation and configuration steps for RPA is already shared as a separate document.

The RPA process has to be configured with series of tasks which are to be executed in sequential flow. The tasks and connectors to define the flow has to be configured as a BPMN notation diagram and any decision-making tasks have to be configured with DMN and gateway to define the segregated flow. The vital configurations like ticket/asset operations are available as delegate expressions (built-in beans).

# Debugging connectivity from RPA to SapphireIMS server

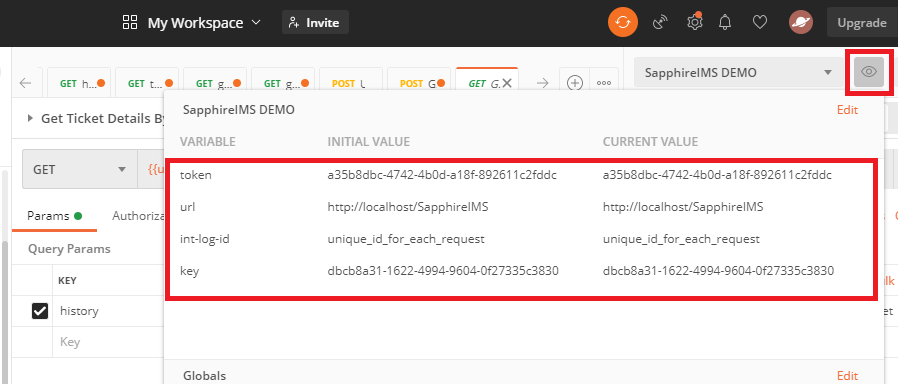
The RPA connectivity is the key factor to be validated once after installing it. The RPA connectivity failure can be due to any one of the reasons from the following.

## Steps to debug

1. Failure to complete the proper tenant registration steps.

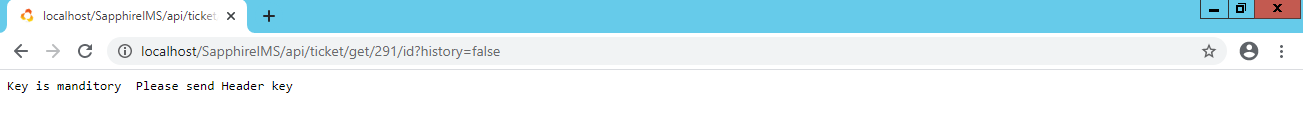
* The tenant registration will generate the key, token and tenant Id which has to be configured into application.properties file ( below 5003 patches) and rpa-authentication.properties file (5003 and above patches).
* The encrypted key and token have to be configured into the properties app.authKey and app.authToken. The separate document is already available for encrypting key and token.
* While registering the tenant, the valid SapphireIMS URL (portal URL) has to be passed as one of input and which is must to make a communication layer from rpa to SapphireIMS.
* The registered key, token, tenant Id & SapphireIMS URL can be retrieved from automation\_tenant table once after the registration for verification.
* If the tenant Id is configured wrong, we will get **Invalid tenant Id exception** in rpa log and if the key/token is wrongly encrypted and configured, we will get **401 unauthorized** exceptions.

 Configuring tenant Id, key & token is must to contact the SapphireIMS API. After doing so, we can cross verify it by calling the get ticket/ get asset API from post man. The SapphireIMS APIs have to be integrated to postman and one-time authentication details have to be configured as an environment variable before making a post man request as attached below.



Note:  Separate detailed documents on RPA installation and configuring key & token and integrating APIs to post man are shared already.

To check the connectivity of API URLs, we can call some sample GET method API URL like**get ticket API** url in browser ( eg: [http://localhost/SapphireIMS/api/ticket/get/291 /id?history=false](http://localhost/SapphireIMS/api/ticket/get/291%20/id?history=false)) and response will ask us to pass the key, token as attached below. The validation on key & token response will ensure that the configured API URL is accessible.



## RPA connectivity configurations

|  |  |  |  |
| --- | --- | --- | --- |
| **Configuration type** | **Name** | **Path (File) / Schema (table)** | **Column (table) /Property ( Property file)** |
| Property file | Application.properties | SapphireIMS.war\WEB-INF\classes\resource | **app.authKey –** registered key. **app.authToken** – registered token. **tenant.id** – registered tenanted. **rpa.basePath –** RPA application path. |
| Property file | Conf\_RPA\_Application.properties | Sapphire Installation path\ \Plugins\SapphireIMSRPA\conf | This property files has various properties to configure RPA port, RPA schema connection details, Attachment path (RPA generated), SSL connectivity parameters (RPA URL), SMTP setting parameters (RPA mail tasks)> |
| Table | Automation\_tenant | Rpa (schema) | **app\_auth\_key** – column where key is stored **app\_auth\_token** - where token is stored **ims\_url –** Sapphire application portal URL path |

# Debugging ticket/asset operations (delegate expression tasks)

The RPA configurations related to ticket/asset operations (pre-defined delegate expressions) are internally the API calls to Sapphire application server. The RPA configuration for any asset/ticket operation will make a communication layer to corresponding SapphireIMS API in a webhook mode. The RPA server will transfer the required request with parameters based on RPA configuration to Sapphire application server and the desired action will be taken place in Sapphire server and it will send back the response to RPA.

## Steps to debug

Each request and response from RPA will be logged into **RPA log** and the table **automation\_integration\_log**in rpa schema. The sample RPA log trace on update ticket API request and response is attached below.



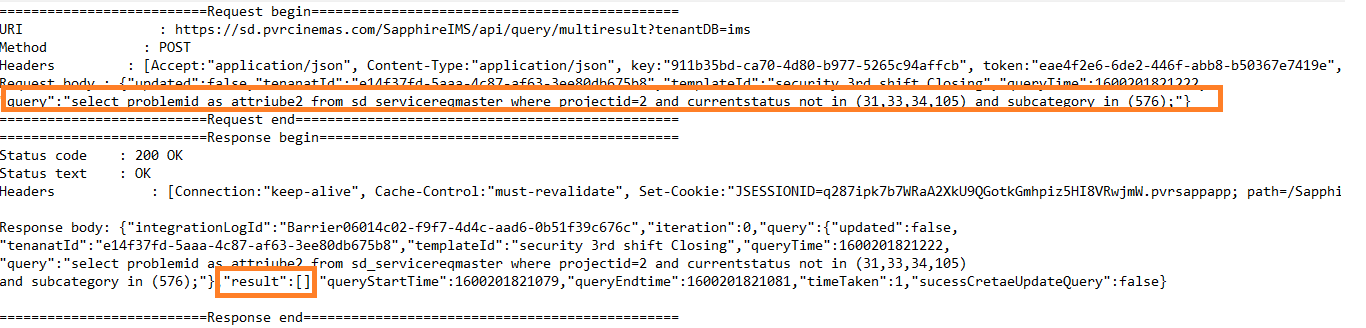
* The request handling failure in Sapphire application server can be verified by copy pasting the above API request body and API details into post man and calling it manually.
* Any error in response from Sapphire API can be found in server.log.
* The preliminary step is to validate the request body parameters for any invalid parameters and to validate whether the connectivity is proper. Any valid issue found after the validation of API request has to be reported for product bug fixing.

## Tables involved

|  |  |  |
| --- | --- | --- |
| **Table Name** | **Schema** | **Table columns to debug** |
| automation\_integration\_log | Rpa | **URL**  - ticket / asset API url **request –** request data **response** - response data **tenant\_id –** tenant Id which made the request |

# Debugging query tasks used in RPA

The query task to be used in RPA configuration has to be added from RPA add-ons on need. The query task is again another API call to Sapphire server which will execute the query and send the response back as JSON. The details such as the executed query, response time, query result can be traced from RPA log as attached below.



The failure in query task can be validated by copy pasting the traced query from the log manually into the query browser and verified. Also, the log will indicate the query result at the time of execution of query which is vital to validate as the results may be different in different time. For Mysql functions used in the rpa query task, mysql general log can be enabled to trace the individual queries involved.

## Steps to debug

The following things to be verified or debugged on query task issue 

1. The single/multi result queries have to be configured as appropriate query result type in RPA configuration (SINGLE/ MULTI, CREATE). Mismatch in configuration would lead to query exception.
2. The appropriate alias name has to be used for select queries as per the column data type in the query used. The alias name to data type column mapping has to be done as follow.

|  |  |
| --- | --- |
| **Query column data type** | **Alias name range** |
| Integer | attribute1 (to) attribute5 |
| String | attribute6 (to) attribute14 |
| Date | attribute15 (to) attribute20 |

## Configuration tables

|  |  |  |
| --- | --- | --- |
| **Table Name** | **Schema** | **Table columns to debug** |
| Sd\_template | ims | **Name** – Configuration name **Template –** query added through RPA add-on **Type –** should be SQL\_QUERY for query task  **Project\_id** – configured project id |
| Automation\_template | rpa | **external\_ref\_Id** - configuration name **template –** configured query **type –** should be SQL\_QUERY for query task **tenant\_id –** requested tenant Id |

# Debugging webhook tasks

The webhook tasks are used to integrate real time data to sapphire server using external (or) outbound API. The webhook takes the connection parameters like API URL, request body, headers, HTTP Method, API authentication details etc., and connect to external API and return the response back to next consecutive tasks.

## Steps to debug

Following are the recommendation to validate the status of webhook task.

* Each request parameters (API connection parameters mentioned above) & response data for it will be captured in RPA log and the table automation\_tenant.
* The postman call has to be made with HTTP method, request API, JSON request body (Rest - POST Method), XML request body (SOAP post method), request headers, authentication type (if set at external API), authentication parameters ( for new requirement, it is recommended to get a postman JSON template from customer for external APIs to import directly and validate the response before the implementation).
* The response to the API call will be the response from the external API to the specific request. For any failure in response, the API request details has to be cross verified and then to check with customer to get it validated from their end.

## 

## Configuration tables

|  |  |  |
| --- | --- | --- |
| **Table Name** | **Schema** | **Table columns to debug** |
| Sd\_template | ims | **Name** – Configuration name **Template –** request body added through RPA add-on **Type –** should be WEBHOKS\_BODY for webhook task  **Project\_id** – configured project id |
| Sd\_integration\_config | ims | **URL –** API url **http\_method –** GET/POST method **header –** Headers to be sent  **project\_id –** Configuration project Id **auth\_handler –** Custom class file path to handle if any specialized authentication (rpa jar file custom class path) **response\_handler -** Custom class file path to handle if any specialized response handler to handle response in required format **request\_handler -** Custom class file path to handle if any specialized pre request handler to manipulate requests before sending **auth\_key –** key to connect to Inbound or outbound (if required) **auth\_value -** token to connect to Inbound or outbound (if required) **error\_handler –** Custom class file path to handle if any error (or) send appropriate error messages |
| Automation\_template | rpa | **external\_ref\_Id** - configuration name **template –** request body added through RPA add-on **type –** should be WEBHOKS\_BODY for webhook task **tenant\_id –** requested tenant Id |
| automation\_integration\_config | rpa | **URL –** API url **http\_method –** GET/POST method **header –** Headers to be sent  **Web\_hook\_type –** API type / API name  **auth\_handler –** Custom class file path to handle if any specialized authentication (rpa jar file custom class path) **response\_handler -** Custom class file path to handle if any specialized response handler to handle response in required format **request\_handler -** Custom class file path to handle if any specialized pre request handler to manipulate requests before sending **auth\_key –** key to connect to Inbound or outbound (if required) **auth\_value -** token to connect to Inbound or outbound (if required) **error\_handler –** Custom class file path to handle if any error (or) send appropriate error messages |

# Debugging DMN and RPA process

## DMN overview

The DMN can be used when some decision has to be carried out with input parameters to define the flow of tasks. It is generally used to define outputs for various possible inputs. For example, service 1 -> Approval logic 1 to be applied and Service 2 -> Approval logic 2 has to be applied. Each row in DMN is to define each input and its output mapping.

## Steps to debug

Following are the things to be validated for DMN :

* The desired output for the current input being passed to DMN should be validated on row basis tracking.
* The data type of each input and output has to be configured strictly. The expected single/multi result has to be configured strict in RPA process and failure in it will lead to process failure.
* Each DMN table columns has a refresh icon option which is applicable to specific column mapped parameter. If service is mapped to a specific DMN column, after adding a new service from configuration, we have to do the Project refresh cache and DMN column refresh cache to reflect in the column data picker.
* The DMN output will usually be passed to gateway task which would accept a Boolean expression (true/false expression) and define segregates gateways or flows.

## Configuration table

|  |  |  |
| --- | --- | --- |
| **Table Name** | **Schema** | **Table columns to debug** |
| Rpa\_process | ims | **processed –** RPA process Id  **project\_id** – Project Id of the configuration  **type** – Should be DMN **process** – XML content of the DMN process **active** – To enable/disable RPA process |

# Debugging RPA Process

RPA involves series of service tasks, connectors, expression based gateways, DMN, sub process (loop), start & end event. In a typical case, there can be time based start events (or) intermediate timer events, user tasks, email tasks, etc,.

## Steps to debug

Following are the steps to be verified to debug RPA process:

* The preliminary check is to validate as if RPA server is running.
* RPA log has to be checked to trace the cause and task which is failed. The failed task Id will be available in RPA log which can just be copy pasted into RPA process configuration screen (ctrl + f) to get the exact failed task.
* The response body which shows below execption should be verified in Sapphire application’s server log for exact cause.   
  **Response body: {"status":"Runtime Exception","code":1000,"message":"Please Contact Sapphire Support","developerMessage" :null,"moreInfoUrl":null,"throwable":null}**
* The sequential flow of the RPA has to be traced in each task level to trace any logical issue in RPA configuration. Each task (API tasks) will be recorded with request and response parameters which have to be validated sequentially.
* The inappropriate use of result names, result type (query & DMN task), conditional checks, finite loops, expressions (or) timer delay can cause logical, performance and syntax error to the process. These things have to be taken care each time we configure/debug the RPA process.
* The failure in a specific task can lead to process failure unless error handling is implemented in RPA process.
* The project refresh cache is must for any new project configuration changes in service desk and asset refresh cache is must for any asset configuration change.
* The possible null handling and empty checks has to be done in RPA using velocity template if condition check or query level to avoid any data loss / wrong data storage.
* The timer start events in RPA will launch independent of project with configured time interval and so disabling these RPA processes from UI project level will not disable this unless we revert the timer start event to normal start events.
* The inappropriate use of time delay or loop may cause slowness and performance impact. The delay has to be set only when it is required.

## Configuration table

|  |  |  |
| --- | --- | --- |
| **Table Name** | **Schema** | **Table columns to debug** |
| Rpa\_process | ims | **processed –** RPA process Id  **project\_id** – Project Id of the configuration  **type –** Should be BPMN **process** – XML content of the BPMN process **active** – To enable/disable RPA process |