CTX-Request-Handler User Guide

Contents

Versions

Document Revisions

The following revisions have been made to this document

Date	Revision	Notes
12/07/2019	1.0	First release

Module Versions

The following revisions have been made to this document

Date	Revision	Notes
12/07/2019	1.0	Creation of: • Request-Handler-Controller • Request-Handler-Management-UI • Request-Handler-Monitoring-UI • Request-Handler-House-Keeping • Request-Handler-Restart-Queues

Preface

About this Manual

This document is a user guide for the CTX-Request-Handler module.

Audience

The audience for this document is those wanting to understand how to use CTX-Request-Handler module.

Related Material

Document
CTX-Request-Handler – Deployment Plan
CTX-Request-Handler.studiopkg

Abbreviations used in this Document

None

Requirements

The CTX-Request-Handler module requires the following:

- Cortex Database OCI

Integration

Integration with Third-Party Systems

Cortex Request Handler Database

For the flows to work in the CTX-Request-Handler module, the Cortex Request Handler database and schema needs to exist on the server containing the Cortex databases. Instructions how to set this up are provided in the 'CTX-Request-Handler – Deployment Plan'.

The tables involved in the Cortex Request Handler schema are:

- Queue – Table containing the properties of the queue
    - QueueName: the name of the queue where requests will be placed
    - MaxSlots: the maximum number of concurrent requests executing in the queue
    - AllowManualCleanUp: allows administration users to set requests executions to Failed (the execution will not stop, but the slot will be freed)
    - AllowTimeOutCleanUp: automatically sets requests executions to Failed (the execution will not stop, but the slot will be freed) after the timeout period has been exceeded by the request execution time
    - TimeOut: the timeout period (in seconds) used for the automatic clean up. If set to 0, it will never timeout
- Request Handler – Stores all the request information:
    - HandlerFlowUUID: the UUID of the Request Handler flow that received the request
    - RequestFlow: the flow the request should be sent to
    - RequestData: the data to be sent to the request flow
    - RequestFlowUUID: the UUID of the request flow execution
    - CreateDate: the date the request was received by the Request Handler flow
    - StartDate: the date the request execution started
    - EndDate: the date the request execution ended
    - EndLog: the request execution end log
    - Status: the request execution status
    - Priority: the request execution priority. This will allow the request handler to prioritise some requests regardless of the creation date
- RequestSequence – Stores links between requests
    - SequenceId: the unique sequence id. This id can be used to find all requests that are related

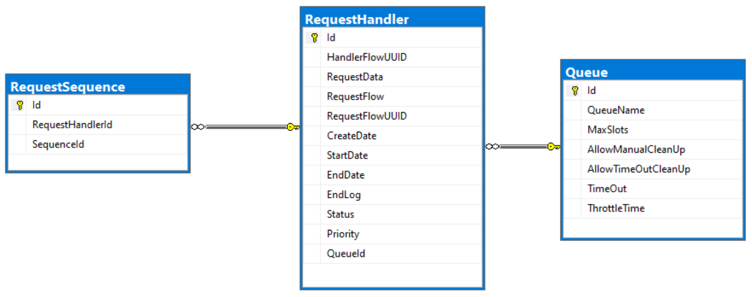


Figure 1 - Cortex Request Handler database schema

Integration with Existing Infrastructure

None

1 Request Handler Overview

The request handler module should be used to manage incoming requests, from third party systems or internally from Cortex flows, which require queueing. The module offers:

- Request handler controller, which automatically saves requests and triggers them as soon as the specific queue requirements are met
- User interfaces to manage and monitor the requests queues

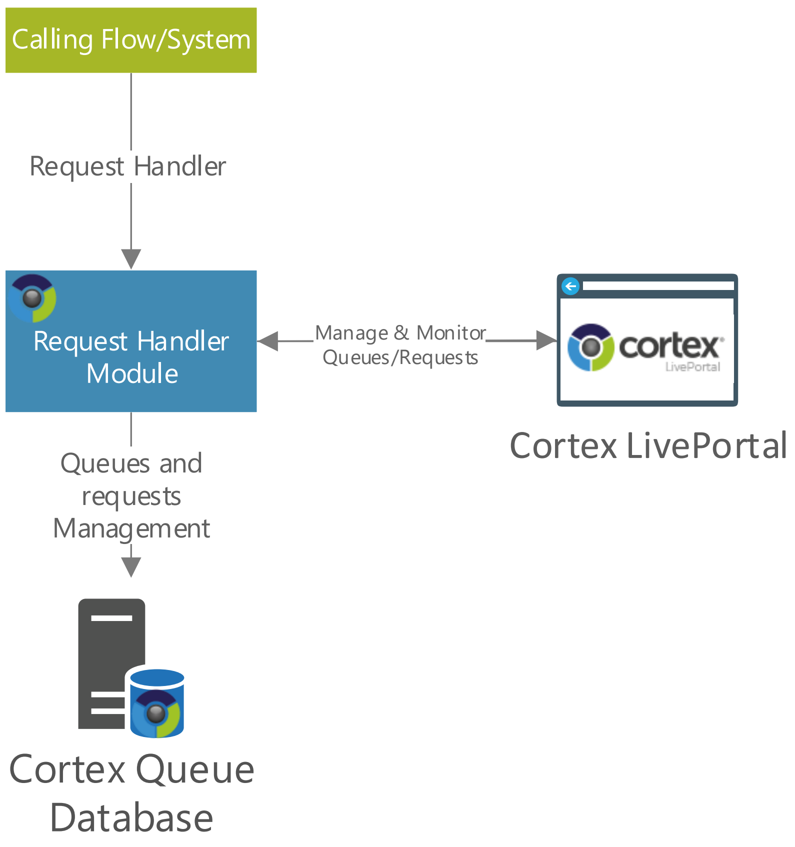


Figure 2 – Cortex Request Handler Module Architecture

1.1 Queue working principles

The queues that control the incoming requests can be configured in a variety of ways to cover for different requirements and scenarios:

1.1.1 Traffic management

- Incoming requests at any rate
- Requests triggered as first-in-first-out
- Maximum slots configured to ensure that only X requests run at each time

⚐ The queue maximum slots can be set to 0 to block the execution of requests. This can be used during maintenance periods.

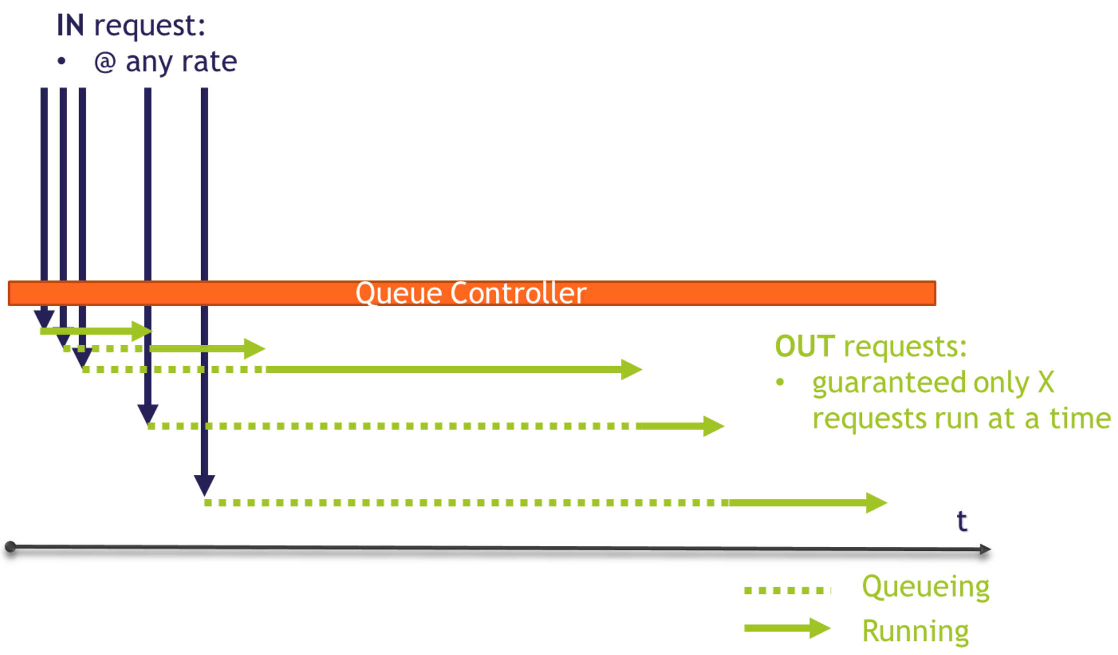


Figure 3 – Traffic queue timeline example with 1 slot

1.1.2 Throttle management

- Incoming requests at any rate
- Requests triggered as first-in-first-out
- Throttle time configured to guarantee a minimum execution time of T between executions start time
- Possibility to set maximum slots so that only X requests run at each time

Even for throttle management cases the queue maximum available slots needs to be defined. If there is no limit, the maximum can be configured with a theorical non-reachable simultaneals request number (example: 10000). As for the traffic example, the maximum slots can be set to 0 to block the execution of requests.

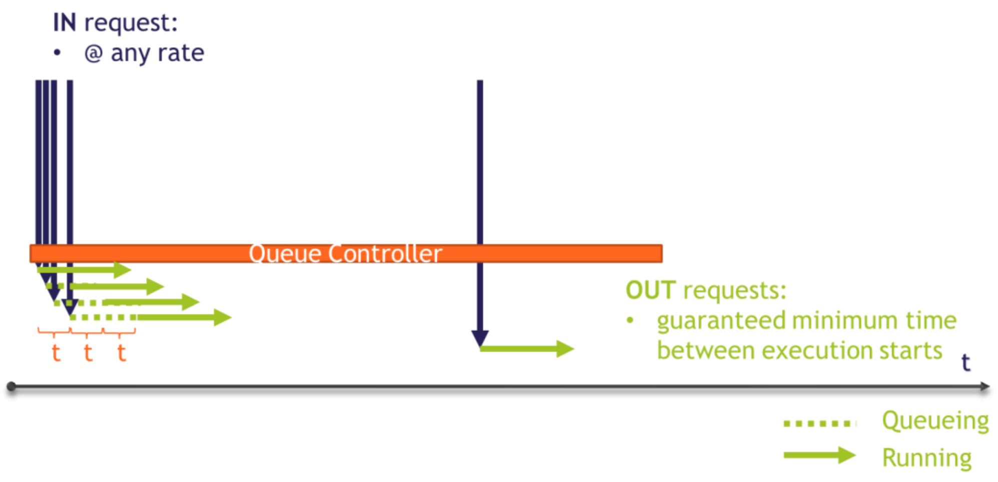


Figure 4 – Throttle queue timeline example

Traffic & Throttle Example

Below an example of a queue configured to manage traffic and throttle at the same time. The executions are triggered with t seconds between starting times and only 2 executions run at the same time.

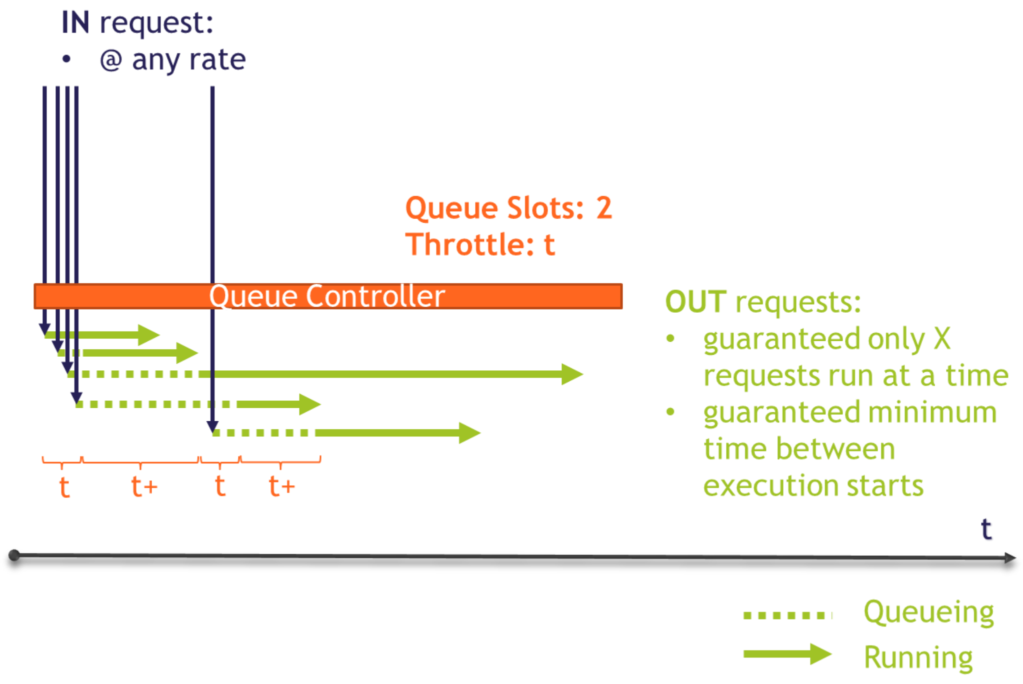


Figure 5 – Traffic and throttle queue timeline example

1.1.3 Sequence management

Queues also support sequence management, which means that specific requests only run if a previous request execution has finished.

This feature is independent of the queue the request belongs to, so each request in a sequencing scenario can belong to a different queue. The execution of each request will still satisfy both the sequence and queue requirements.

More details on the request data required to sequence requests can be found in section 1.3.3.

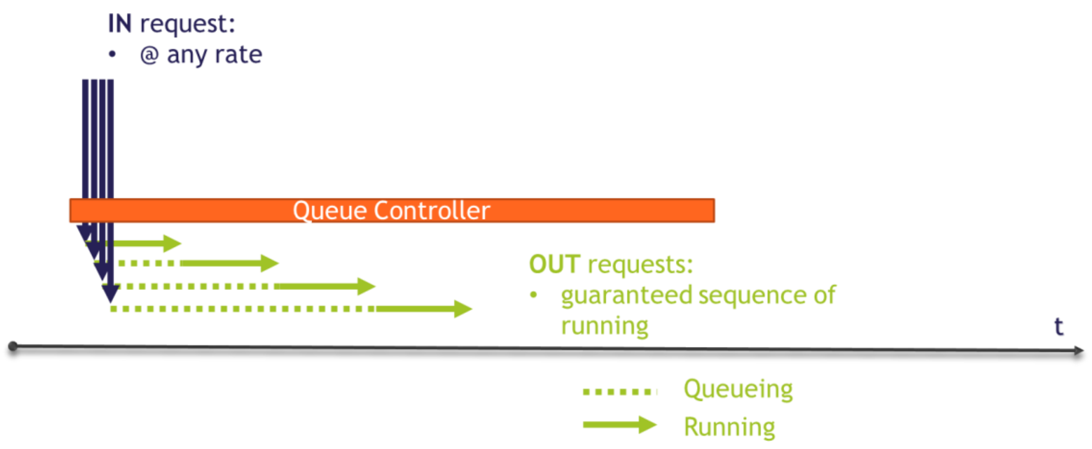


Figure 6 – Sequence management example

1.2 Using the module

As specified in the diagram below, the calling Cortex flows, or external systems, will send a queue request to the Cortex Request Handler module, using the Request-Handler-Controller flow.

The Request-Handler-Controller will evaluate the request and associated queue and if the queue and request requirements are met, start the execution of the request. If the queue is busy or the request has a sequencing dependency, the request will be stored and started as soon as possible.

The module allows for different queues to be specified and used independently.

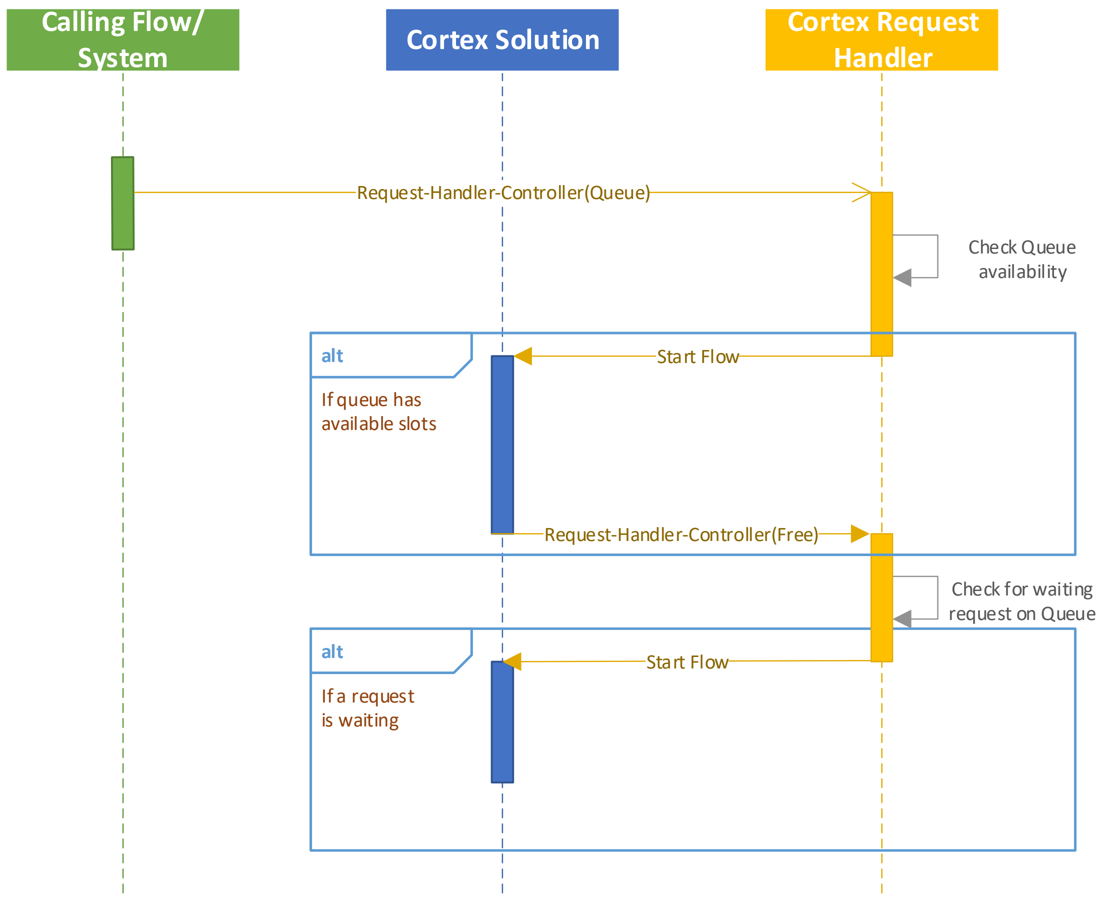


Figure 7 – Request Handler Module queueing requests

1.3 Requests Examples

1.3.1 Queue Request

```
{
   "NAME": "REQUEST-HANDLER-CONTROLLER",
   "ARGUMENTS": {
      "I_QUEUE": "<Queue Name>",
      "I_QUEUE-ACTION": "Queue",
      "I_REQUEST-DATA": "<Request Payload in JSON>",
      "I_REQUEST-FLOW": "<Request Target Flow>"
   },
   "RETURNPARAMETERS": []
}
```

1.3.2 Free Request

```
{
   "NAME": "REQUEST-HANDLER-CONTROLLER",
   "ARGUMENTS": {
      " I_EXECUTION_UUID": "<Request execution UUID>",
      "I_QUEUE-ACTION": "Free"
   },
   "RETURNPARAMETERS": []
}
```

1.3.3 Sequencing Queue Request

First Request

```
{
   "NAME": "REQUEST-HANDLER-CONTROLLER",
   "ARGUMENTS": {
      "I_QUEUE": "<Queue Name>",
      "I_QUEUE-ACTION": "Queue",
      "I_REQUEST-DATA": "<Request Payload in JSON>",
      "I_REQUEST-FLOW": "<Request Target Flow>"
      "I_SEQUENCING-REQUEST": true
   },
   "RETURNPARAMETERS": [
      "O_QUEUE-SEQUENCE-ID"
   ]
}
```

Next Requests into Sequence

```
{
   "NAME": "REQUEST-HANDLER-CONTROLLER",
   "ARGUMENTS": {
      "I_QUEUE": "<Queue Name>",
      "I_QUEUE-ACTION": "Queue",
      "I_REQUEST-DATA": "<Request Payload in JSON>",
      "I_REQUEST-FLOW": "<Request Target Flow>"
      "I_SEQUENCING-REQUEST": true,
      "I_QUEUE-SEQUENCE-ID": "<Sequence Id returned from the previous
request>"
   },
   "RETURNPARAMETERS": [
      "O_QUEUE-SEQUENCE-ID"
   ]
}
```

1.4 User Experience

1.4.1 Request-Handler-Management-UI

The ‘Request-Handler-Management-UI’ is used by the user to create, edit and delete Queues from the system.

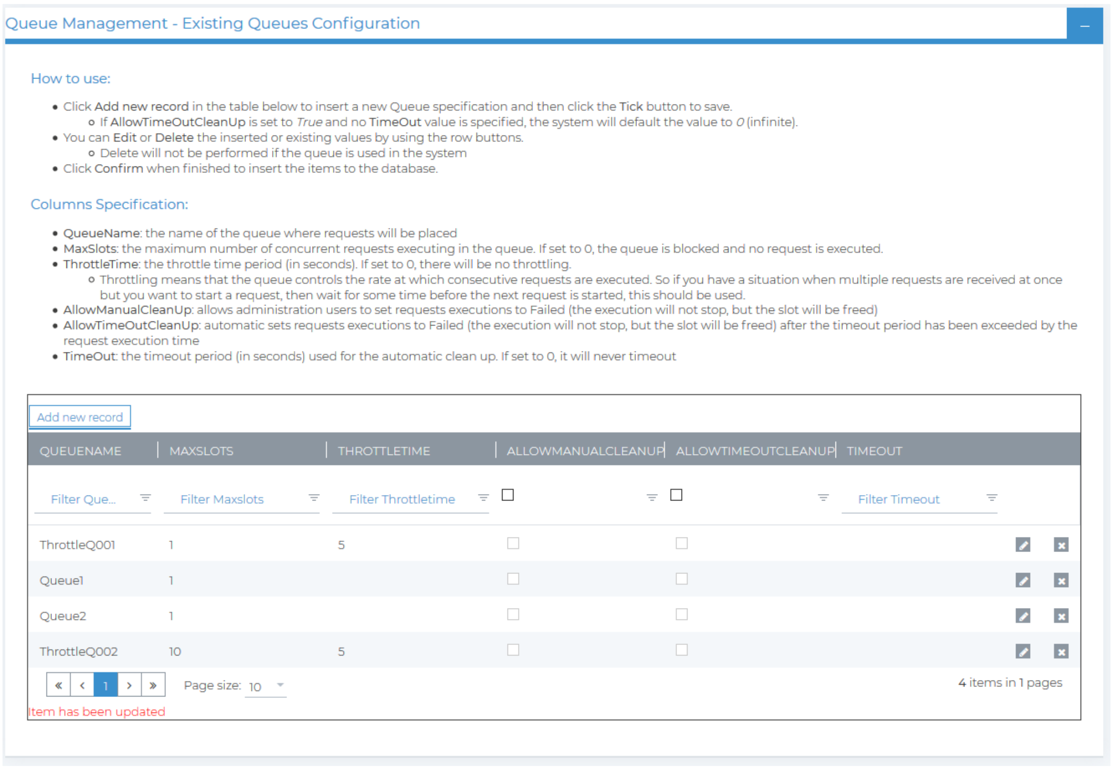
1. When the flow starts, the user is presented with the **homepage**. This page allows the user to view the current Queue configuration and perform one of the following actions:

   a. Add a new Queue

   b. Modify an existing Queue

   c. Delete an existing Queue

An existing queue will only be deleted if there are no associated requests with it


Queue Management - Existing Queues Configuration

How to use:

- Click **Add new record** in the table below to insert a new Queue specification and then click the **Tick** button to save.
  - If **AllowTimeOutCleanUp** is set to *True* and no **TimeOut** value is specified, the system will default the value to *0* (infinite).
- You can **Edit** or **Delete** the inserted or existing values by using the row buttons.
  - Delete will not be performed if the queue is used in the system
- Click **Confirm** when finished to insert the items to the database.

Columns Specification:

- **QueueName**: the name of the queue where requests will be placed
- **MaxSlots**: the maximum number of concurrent requests executing in the queue. If set to 0, the queue is blocked and no request is executed.
- **ThrottleTime**: the throttle time period (in seconds). If set to 0, there will be no throttling.
  - Throttling means that the queue controls the rate at which consecutive requests are executed. So if you have a situation when multiple requests are received at once but you want to start a request, then wait for some time before the next request is started, this should be used.
- **AllowManualCleanUp**: allows administration users to set requests executions to Failed (the execution will not stop, but the slot will be freed)
- **AllowTimeOutCleanUp**: automatic sets requests executions to Failed (the execution will not stop, but the slot will be freed) after the timeout period has been exceeded by the request execution time
- **TimeOut**: the timeout period (in seconds) used for the automatic clean up. If set to 0, it will never timeout

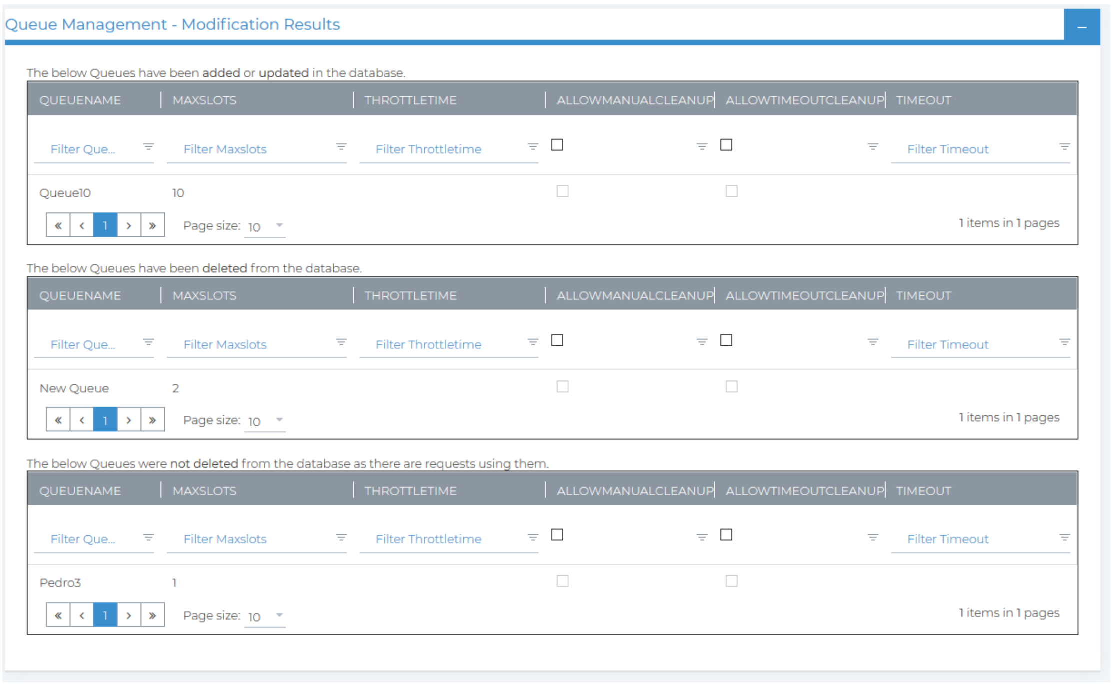
Add new record

QUEUENAME	MAXSLOTS	THROTTLETIME	ALLOWMANUALCLEANUP	ALLOWTIMEOUTCLEANUP	TIMEOUT
Filter Que...	Filter Maxslots	Filter Throttletime	☐	☐	Filter Timeout
ThrottleQ001	1	5	☐	☐	
Queue1	1		☐	☐	
Queue2	1		☐	☐	
ThrottleQ002	10	5	☐	☐	

« ‹ 1 › » Page size: 10   4 items in 1 pages

Item has been updated

2. After performing the required changes and clicking **Confirm** the user will be presented with the results page. This page displays the results in 3 tables with the following data:
   a. New or modified queues
   b. Deleted queues
   c. Queues that were selected to be deleted but were not deleted due to existing requests associated with them


Queue Management - Modification Results

The below Queues have been added or updated in the database.

QUEUENAME	MAXSLOTS	THROTTLETIME	ALLOWMANUALCLEANUP	ALLOWTIMEOUTCLEANUP	TIMEOUT
Queue10	10		☐	☐	

1 items in 1 pages

The below Queues have been deleted from the database.

QUEUENAME	MAXSLOTS	THROTTLETIME	ALLOWMANUALCLEANUP	ALLOWTIMEOUTCLEANUP	TIMEOUT
New Queue	2		☐	☐	

1 items in 1 pages

The below Queues were not deleted from the database as there are requests using them.

QUEUENAME	MAXSLOTS	THROTTLETIME	ALLOWMANUALCLEANUP	ALLOWTIMEOUTCLEANUP	TIMEOUT
Pedro3	1		☐	☐	

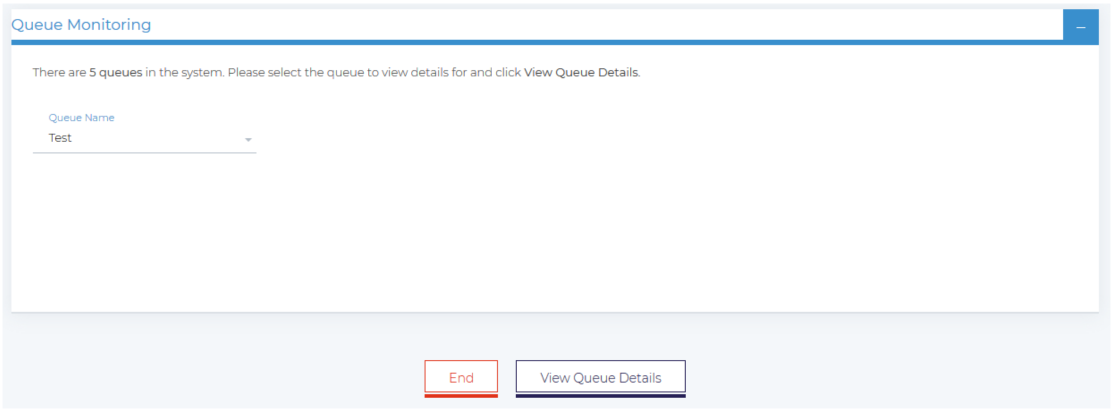
1 items in 1 pages

1.4.2 Request-Handler-Monitoring-UI

The ‘Request-Handler-Monitoring-UI’ is used by the user to view the queues statistics and requests details.

1. When the flow starts, the user is presented with the **homepage**. This page allows the user to select a Queue for which they wished to view the statistics and requests details.

If the system only has one queue configured his page will not be shown

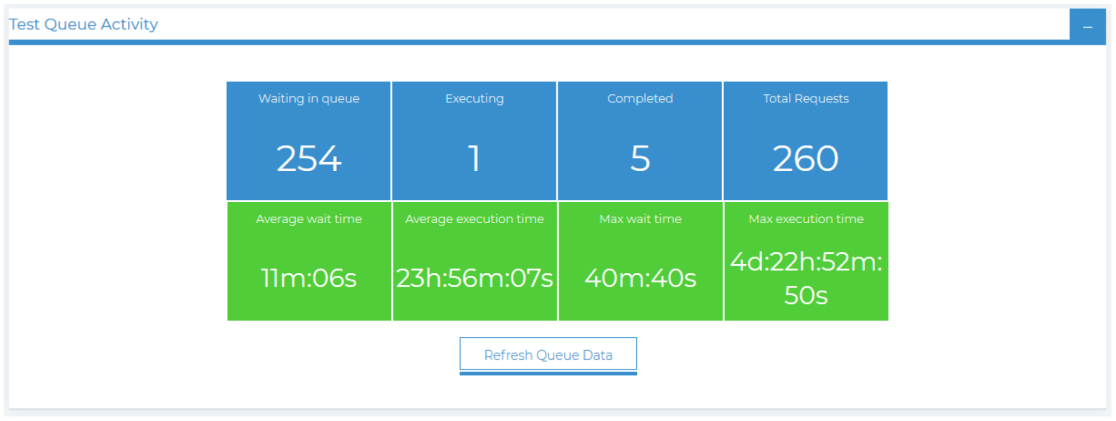

Queue Monitoring

There are 5 queues in the system. Please select the queue to view details for and click View Queue Details.

Queue Name: Test

2. After selecting the Queue and clicking **View Queue Details** the user will be brought to the queue details page. This page displays two panels with the following data:
   a. Queue activity dashboard which shows
      i. the number of requests waiting, executing and completed
      ii. statistics on the executions: average wait and execution time, and maximum wait and execution time

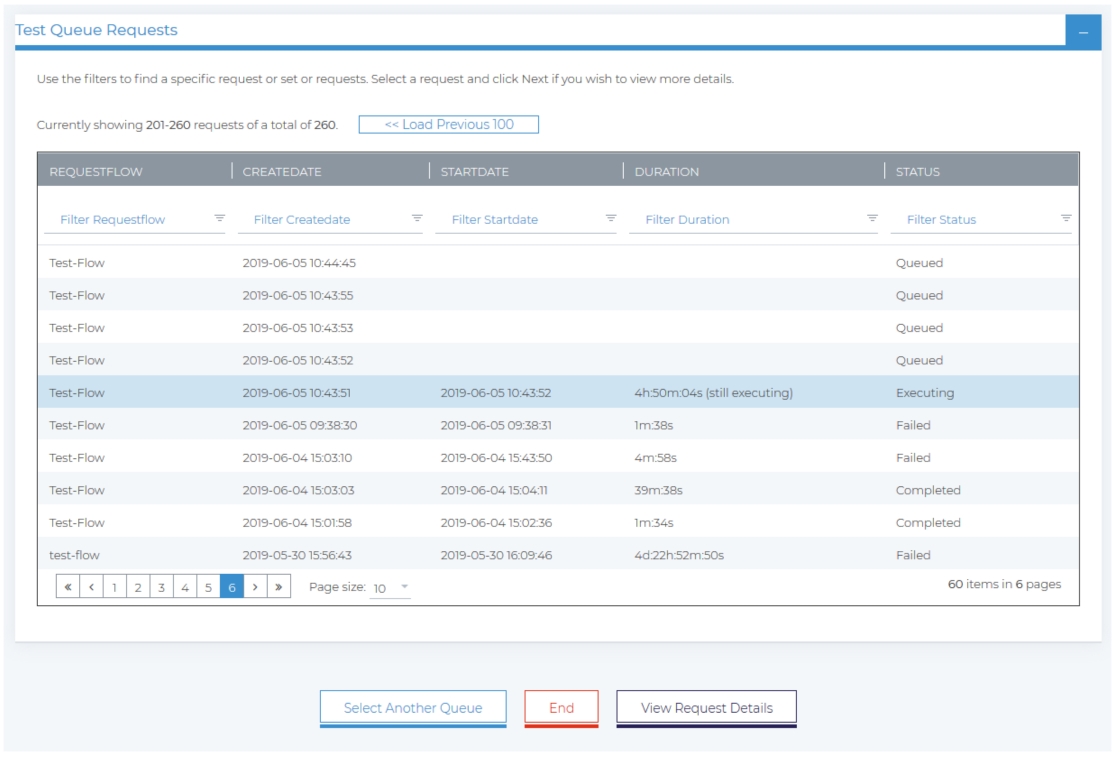
The user can refresh the page data by clicking **Refresh Queue Data**


Test Queue Activity

Waiting in queue	Executing	Completed	Total Requests
254	1	5	260
Average wait time	**Average execution time**	**Max wait time**	**Max execution time**
11m:06s	23h:56m:07s	40m:40s	4d:22h:52m:50s

b. Queue requests which shows the requests of the queue (newer first)

This page only shows batches of 100 requests. If there are more requests the user can use the Load Next/Previous 100 request to view those.


Test Queue Requests

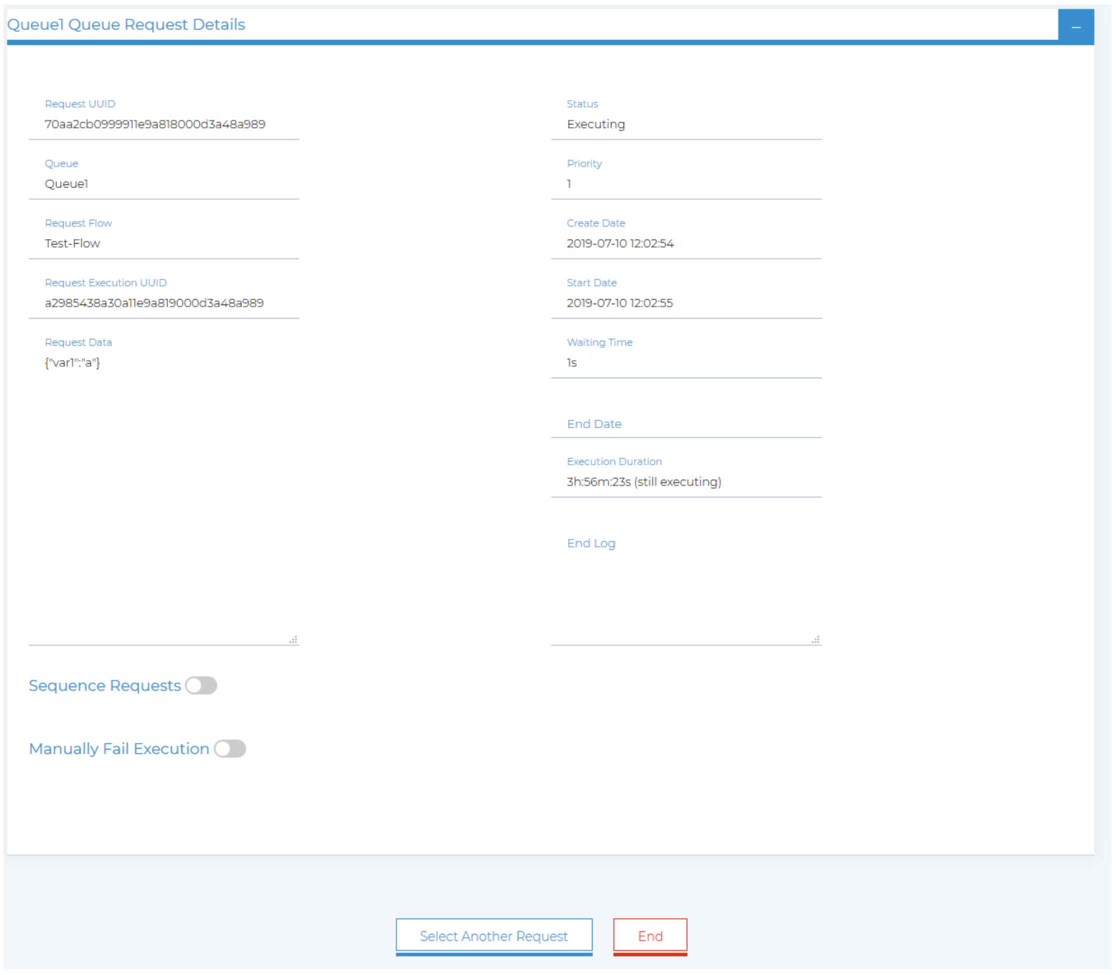
Use the filters to find a specific request or set or requests. Select a request and click Next if you wish to view more details.

Currently showing 201-260 requests of a total of 260.

REQUESTFLOW	CREATEDATE	STARTDATE	DURATION	STATUS
Filter Requestflow	Filter Createdate	Filter Startdate	Filter Duration	Filter Status
Test-Flow	2019-06-05 10:44:45			Queued
Test-Flow	2019-06-05 10:43:55			Queued
Test-Flow	2019-06-05 10:43:53			Queued
Test-Flow	2019-06-05 10:43:52			Queued
Test-Flow	2019-06-05 10:43:51	2019-06-05 10:43:52	4h:50m:04s (still executing)	Executing
Test-Flow	2019-06-05 09:38:30	2019-06-05 09:38:31	1m:38s	Failed
Test-Flow	2019-06-04 15:03:10	2019-06-04 15:43:50	4m:58s	Failed
Test-Flow	2019-06-04 15:03:03	2019-06-04 15:04:11	39m:38s	Completed
Test-Flow	2019-06-04 15:01:58	2019-06-04 15:02:36	1m:34s	Completed
test-flow	2019-05-30 15:56:43	2019-05-30 16:09:46	4d:22h:52m:50s	Failed

3. From the Queue requests table, the user can select a request to view details for, by highlighting a row in the table and selecting **View Request Details**

4. The request details page will display all the data stored with the request

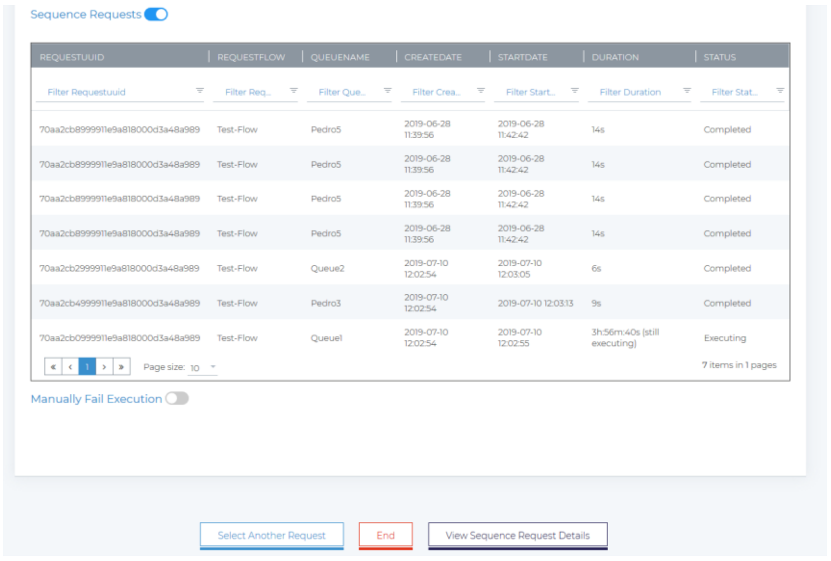


Queue1 Queue Request Details

Request UUID: 70aa2cb0999911e9a818000d3a48a989	Status: Executing
Queue: Queue1	Priority: 1
Request Flow: Test-Flow	Create Date: 2019-07-10 12:02:54
Request Execution UUID: a2985438a30a11e9a819000d3a48a989	Start Date: 2019-07-10 12:02:55
Request Data: {"var1":"a"}	Waiting Time: 1s
	End Date:
	Execution Duration: 3h:56m:23s (still executing)
	End Log:

5. This page will also allow:

    a. The user to view other requests that are linked to the request being viewed (if those exist). This way the user can see the sequence of execution and the status of each request. From the sequence requests the user can also view the details of a request by selecting one row from the table and clicking on **View Sequence Request Details**



REQUESTUUID	REQUESTFLOW	QUEUENAME	CREATEDATE	STARTDATE	DURATION	STATUS
Filter Requestuuid	Filter Req...	Filter Que...	Filter Crea...	Filter Start...	Filter Duration	Filter Stat...
70aa2cb8999911e9a818000d3a48a989	Test-Flow	Pedro5	2019-06-28 11:39:56	2019-06-28 11:42:42	14s	Completed
70aa2cb8999911e9a818000d3a48a989	Test-Flow	Pedro5	2019-06-28 11:39:56	2019-06-28 11:42:42	14s	Completed
70aa2cb8999911e9a818000d3a48a989	Test-Flow	Pedro5	2019-06-28 11:39:56	2019-06-28 11:42:42	14s	Completed
70aa2cb8999911e9a818000d3a48a989	Test-Flow	Pedro5	2019-06-28 11:39:56	2019-06-28 11:42:42	14s	Completed
70aa2cb2999911e9a818000d3a48a989	Test-Flow	Queue2	2019-07-10 12:02:54	2019-07-10 12:03:05	6s	Completed
70aa2cb4999911e9a818000d3a48a989	Test-Flow	Pedro3	2019-07-10 12:02:54	2019-07-10 12:03:13	9s	Completed
70aa2cb0999911e9a818000d3a48a989	Test-Flow	Queue1	2019-07-10 12:02:54	2019-07-10 12:02:55	3h:56m:40s (still executing)	Executing

Page size: 10 — 7 items in 1 pages

b. The user to perform an administrative action (Manual Fail Execution option) if some conditions are met:

   i. The queue allows manual clean-up of executions (Queue configuration flag)

   ii. The request is in executing status

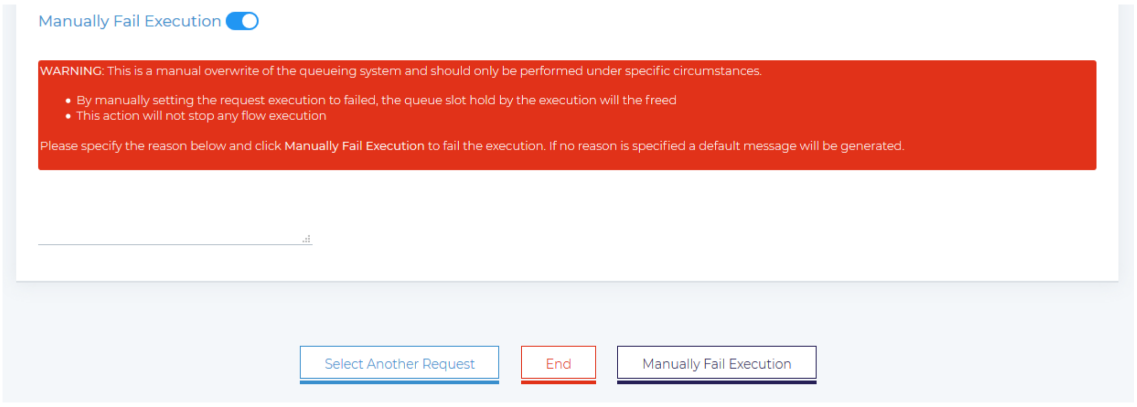
c. When the user clicks on the **Manual Fail Execution** option, some additional UI elements will be displayed:

   i. Warning message

   ii. Text box to fill in the reason why the manual fail is being performed

   iii. Manual Fail Execution button

As the warning message states, this action gives the user the ability to manually overwrite the request status in the queueing system and free the slot being used. This action will not stop any associated flow execution and should only be performed under specific circumstances. For example, if a system catastrophic failure occurs, and the request execution has stopped but the queue database was not updated.



WARNING: This is a manual overwrite of the queueing system and should only be performed under specific circumstances.

- By manually setting the request execution to failed, the queue slot hold by the execution will the freed
- This action will not stop any flow execution

Please specify the reason below and click **Manually Fail Execution** to fail the execution. If no reason is specified a default message will be generated.

2 User Access Management Flows

2.1 Request-Handler-Controller

2.1.1 Overview

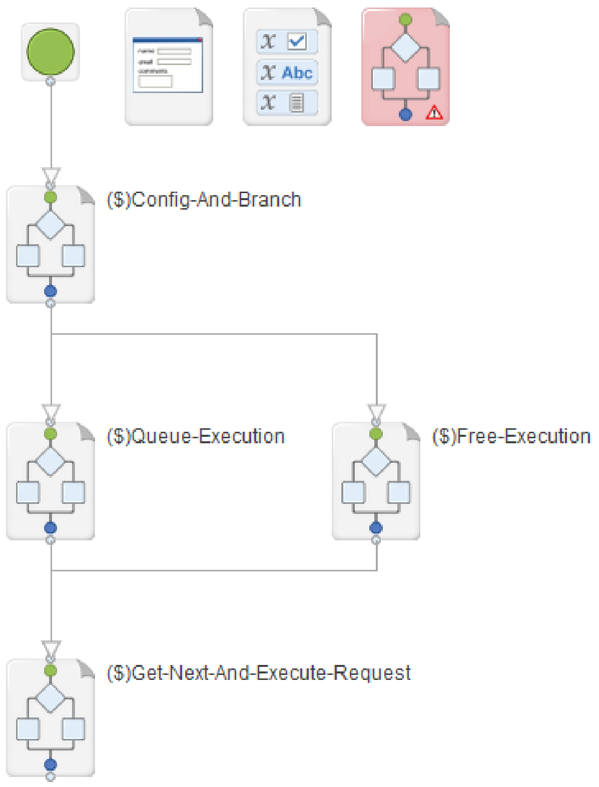
The Request-Handler-Controller flow serves two purposes:

- Queueing requests
- Freeing slots from queued requests

Exceptions will be raised if:

- Queue is not supplied or does not exist in the database
- Queue actions is not supplied or is not Free or Queue
- If the action is Queue and the Request Data and Request flow are not supplied
- If the action is Free and the Execution UUID is not supplied
- The Cortex Request Handler database is not accessible

2.1.2 States



- Config-And-Branch

Validates that the Queue and Queue action inputs are passed into the flow and that the Queue exists in the database. If these checks pass, the flow branches based on the action.

- Queue-Execution

  This state is executed when the action is Queue. Saves the request into the Cortex Request Handler database.

- Free-Execution

  This state is executed when the action is Free. Updates the request details in the Cortex Request Handler and frees the queue used slot.

- Get-Next-And-Execute-Request

  If the queue has the available capacity, gets the next request to be executed and starts the request flow execution.

2.1.3 Inputs

Input Variables	Type	Description
Generic Inputs		
i_Queue-Action	Text	The action to be performed on the queue. This should be either Free or Queue. REQUIRED Example: Free
i_Database-Server	Text	The server where the Cortex Request Handler database is hosted. Default value is set to the same database server as the Reactor database. Example: localhost
i_Database-Name	Text	The name of the Cortex Request Handler database. Default value is set to 'Cortex-RequestHandler' Example: Cortex-RequestHandler
Inputs for "Queue" action		
i_Queue	Text	The name of the queue where requests will be placed. This needs to exist on the database. REQUIRED Example: SwitchAllocation

i_Request-Flow	Text	The name of the flow that the request should execute. This should be an existing Cortex flow in the system. REQUIRED (if action is Queue) Example: Allocation-Flow

i_Request-Data	Text	The data to be sent to the flow that the request should execute. This should be in JSON format.
		
		REQUIRED (if action is Queue)
		
		Example:
		{
		"G_TOKEN": "ABCDEF123456",
		"G_FROMDATE": "2019-02-06",
		"G_TODATE": "2019-02-07",
		"G_ORDERTYPE": "Ticket",
		"G_COMPANY": "Cortex"
		}

i_Queue-Priority	Integer	The priority of the request. The higher the number, the higher the priority of the request. The request handler controller executes requests based on priority and order of arrival. Default value is 1. Example: 9
Inputs for Sequencing "Queue" action		
i_Sequencing-Request	Boolean	A True/False value to indicate if the request is a part of a sequence request. Default value is False. Example: True
i_Queue-Sequence-Id	Text	This input should be used for requests that required to be associated with an existing sequence of requests. As explained in section 1.3.3, after the first sequence request an id is generated which should be used on the subsequence requests. ONLY REQUIRED FOR SUBSEQUENT SEQUENCING REQUESTS Example: c87a972486d911e9a824000d3a2d9202
Inputs for "Free" action		
i_Execution_UUID	Text	The UUID of the flow executing the request. REQUIRED Example: c87a972486d911e9a824000d3a2d9202

i_End-Log	Text	The execution end log. This is a free text which can hold any relevant information for the execution ending. Default value is an empty text Example: Execution successfully ended.
i_End-Status	Text	The end execution status. This value should either be Completed, if the request execution completes successfully; or Failed, if the request execution is not successful. The default value is Completed. Example: Failed

2.1.4 Outputs

Input Variables	Type	Description
o_Queue-Sequence-Id	Text	This output is only generated for the first request of a sequencing request. Example: c87a972486d911e9a824000d3a2d9202

2.2 Request-Handler-House-Keeping

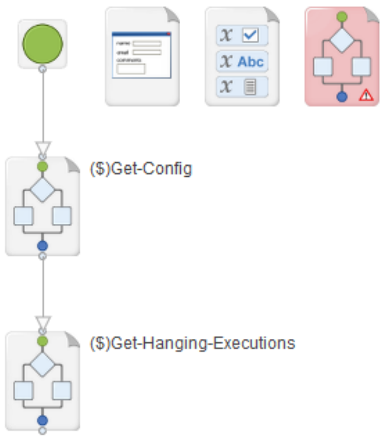
2.2.1 Overview

Connects to the Cortex Request Handler database, finds out which executions have timed out based on the Queue configuration and fails the executions by triggering the Request-Handler-Controller.

Exceptions will be raised if

- The Cortex Request Handler database is not accessible

2.2.2 States



- Get-Config

  Builds up the database connection string.

- Get-Hanging-Executions

  Connects to the Cortex Request Handler database, finds out which executions have timed out based on the Queue configuration and fails the executions by triggering the Request-Handler-Controller.

2.2.3 Inputs

Input Variables	Type	Description
i_Database-Server	Text	The server where the Cortex Request Handler database is hosted. Default value is set to the same database server as the Reactor database. Example: localhost
i_Database-Name	Text	The name of the Cortex Request Handler database. Default value is set to 'Cortex-RequestHandler' Example: Cortex-RequestHandler

2.2.4 Outputs

NA

2.3 Request-Handler-Restart-Queues

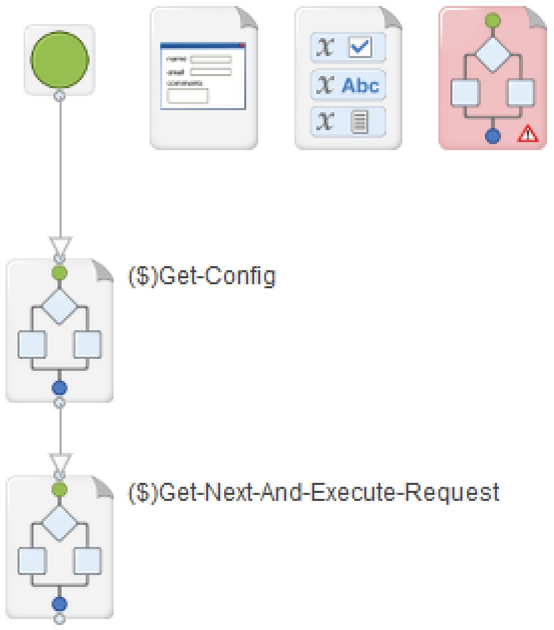
2.3.1 Overview

Restarts requests on a given queue if the queue is active again.

Exceptions will be raised if

- The Cortex Request Handler database is not accessible

2.3.2 States



- Get-Config

  Builds up the database connection string.

- Get-Next-And-Execute-Request

  Connects to the Cortex Request Handler database, finds out the next execution and triggers the execution.

2.3.3 Inputs

Input Variables	Type	Description
i_QueueName	Text	The queue name for which the request should be started. REQUIRED Example: SwitchAllocation
i_Number-Of-Requests	Integer	The number of requests to be started on the queue. The flow will still ensure that the max slots on a queue are not exceeded. REQUIRED Example: 1
i_Database-Server	Text	The server where the Cortex Request Handler database is hosted. Default value is set to the same database server as the Reactor database. Example: localhost
i_Database-Name	Text	The name of the Cortex Request Handler database. Default value is set to 'Cortex-RequestHandler' Example: Cortex-RequestHandler

2.3.4 Outputs

NA