UiPath

Robotic Process Automation

Orchestrator User Guide

Revision History

Date	Version	Author	Description
31st May 2016	2016.1	M.B.	Applied Template

Table of Contents

UiPath Orchestrator - User Guide

UiPath Orchestrator is a collection of Web services that orchestrates UiPath Robots that run repetitive business processes.

Orchestrator lets you manage the creation, monitoring, and deployment of resources in your environment.

UiPath Orchestrator Use Cases

Front Office Server - Robots are used as agent assistants. They are installed on user's workstations and are triggered by human operators on demand. The platform is required only for process deployment and centralized robot logs.

Back Office Server - Robots run unattended in virtual environments and can automate any number of back office processes. On top of the FOS capabilities, the Orchestrator is responsible for remote execution, monitoring, scheduling and provide support for work queues.

Orchestrator Services

- **UiPath.Manager** - Main service of the Server, responsible for Configuration, Deployment and Scheduling
- **UiPath.Log** - Web API forwarder service for Elastic Search and other logging back-ends
- **UiPath.Monitor** - Stateful service based on Akka actor model
- **UiPath.Queue** - SQL based implementation of work queues
- **UiPath.Web** - Web client portal

1. Orchestrator User Interface

Roles

- Authorization based on Active Directory groups mapping

Settings

- General configuration for service endpoints and connection strings

Manage

- **Robots** - Software robots that automate rule-based processes in the same manner as humans do
- **Processes** - Business process definition packages published by developers
- **Assets** - Shared variables or configurations used in processes
- **Queues** - Work queues used to distribute work items to robots

Deploy

- **Environments** - Grouping of robots for deployment purposes
- **Releases** - Distribution of process versions to robot environments

Execute

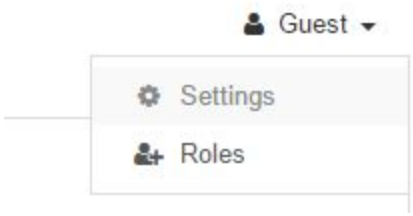
- **Jobs** - Execution instances of processes assigned to robots
- **Schedules** - Triggers for scheduled job execution

Monitor

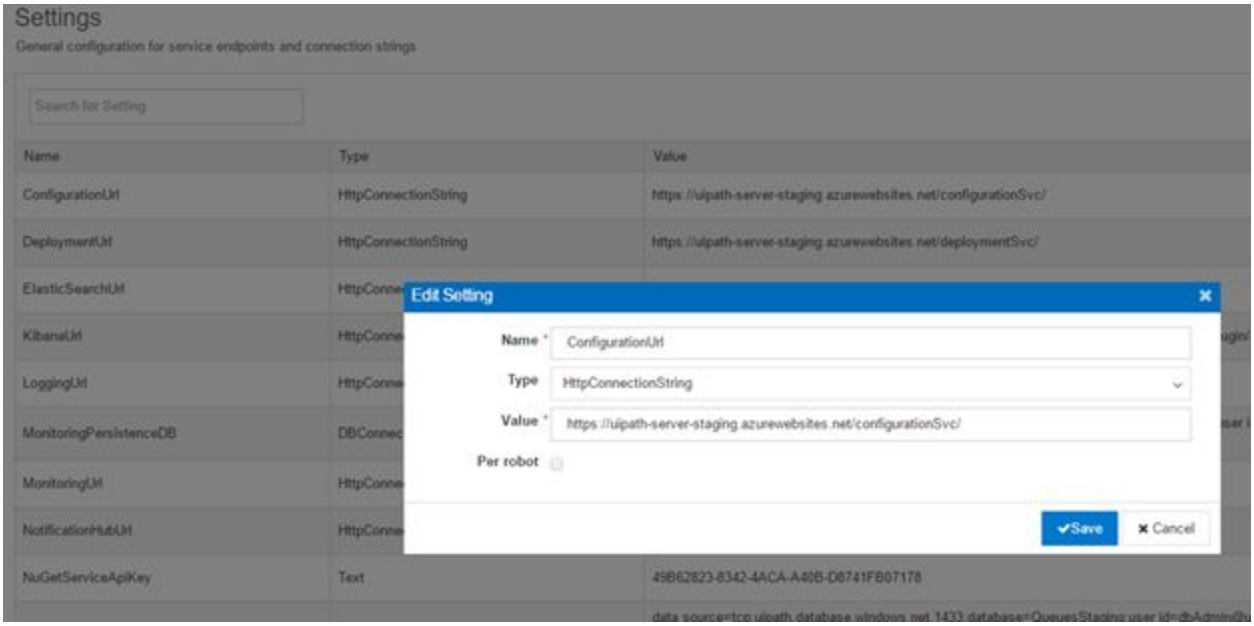
- **Sessions** - Real time monitoring of robots
- **Transactions** - Real time monitoring of work queues
- **Logs** - Logs generated by robots and execution reports

2. Global Settings Management

Settings page is located in user menu:



On this page the administrator is able to configure UiPath Orchestrator settings:

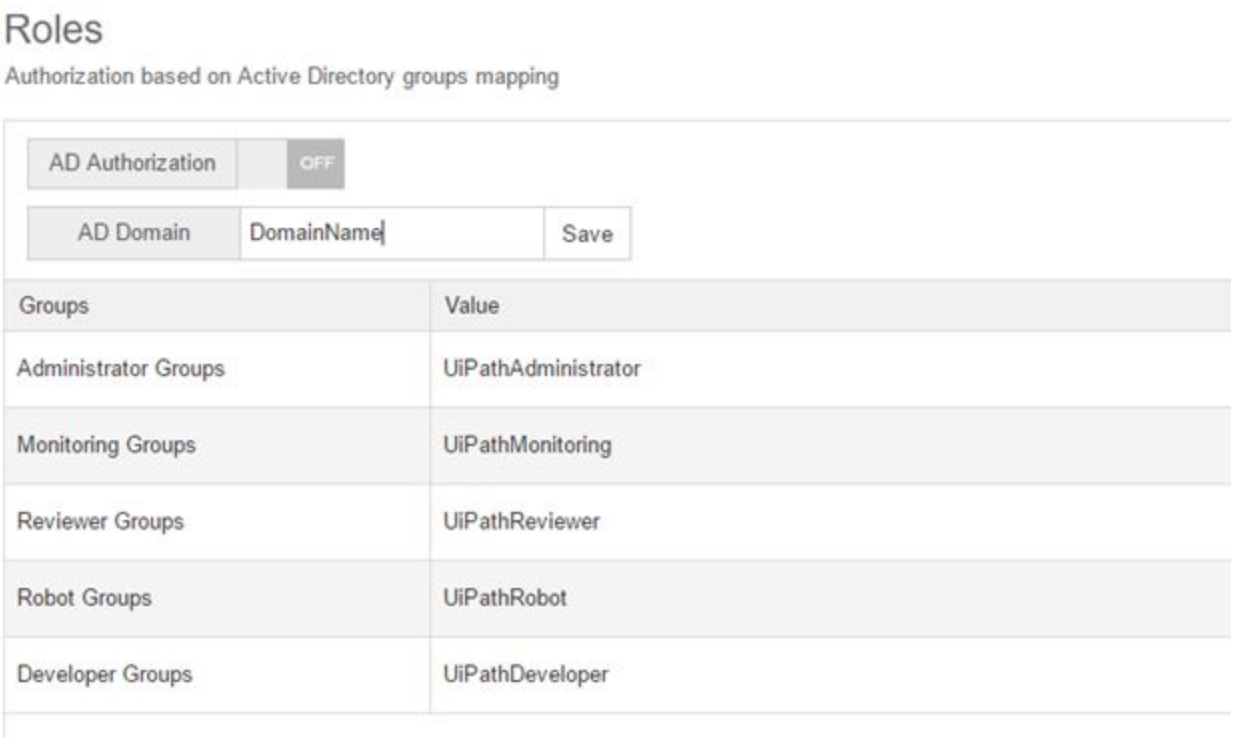


1. Click on the **Edit** Button.
2. Make the necessary modifications and click the **Save** button.

Note: After editing a setting click on the **Reload Services Configuration** button to propagate the modifications. Make sure no robot is connected to the server at the moment. After the services are reloaded, the robots need to register again.

2.1. Roles Page

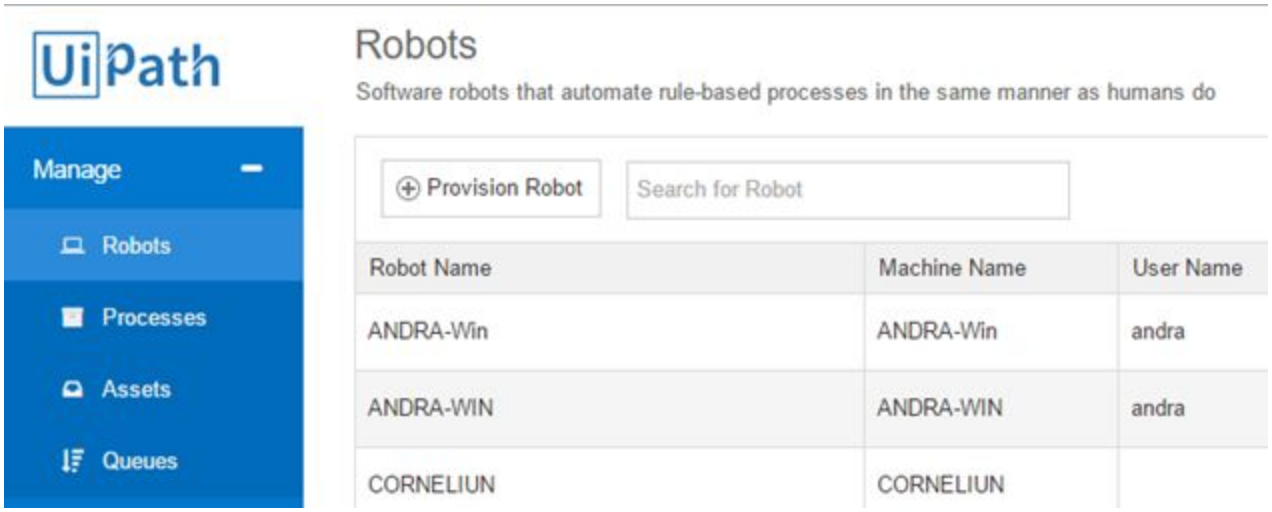
Authorization based on Active Directory groups mapping



3. Managing Robots

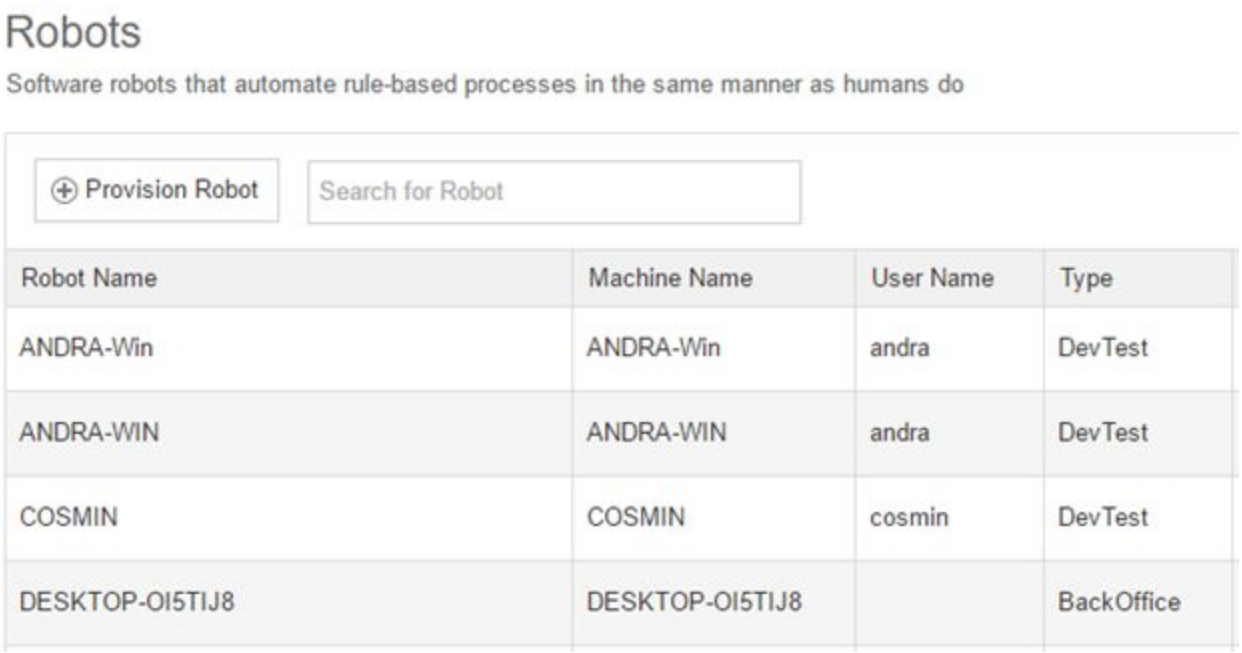
Robots Page - Manage Robots

Software robots that automate rule-based processes in the same manner as humans do.

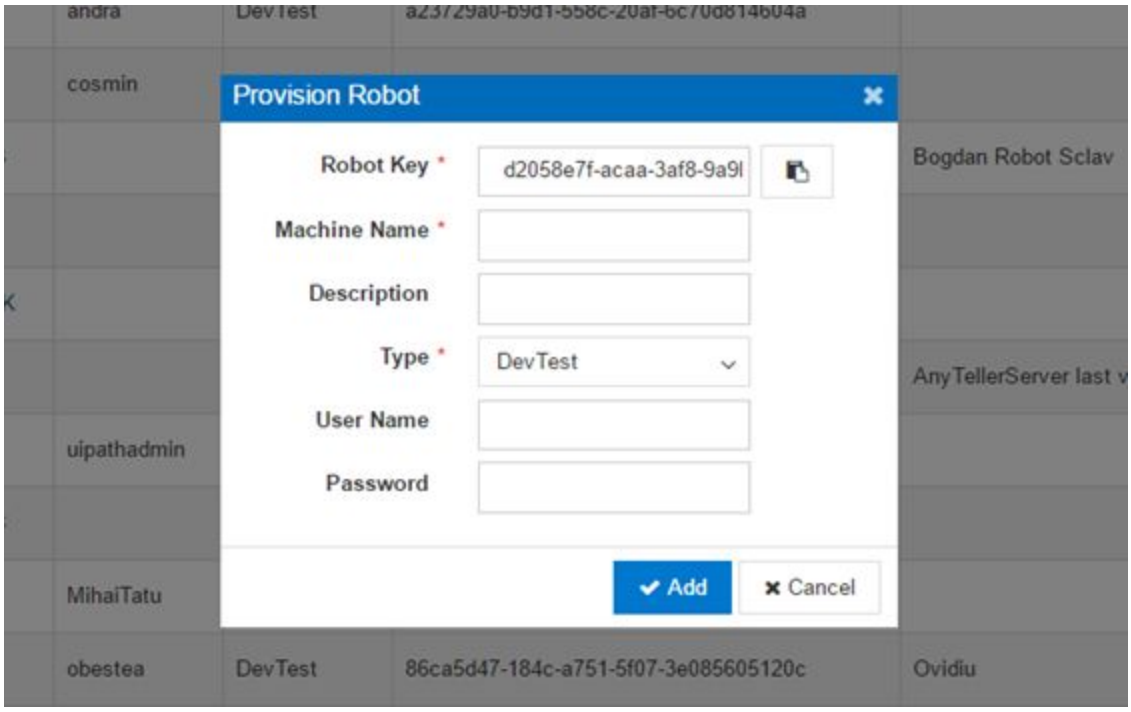


3.1. Adding a New Robot

1. Click the **Provision Robot** button. The **Provision Robot** window is displayed.



2. Complete the form and click **Save**. A robot with your settings has been added.



3.1.1. Field Descriptions for the Provision Robot Window

Field	Description
Robot Key	Mandatory. This key is automatically generated from the server for the robot machine. It is inserted into the client machine. To view the robot key, right-click the UiPath agent -> Configure Server -> copy the Robot Key.
Machine Name	Mandatory. The actual name of the machine.
Description	Optional. Type details about your Robot.
Type	Mandatory. Enables you to choose the type of robot to use. The available options are: ● Development ● Front Office ● Back Office
User Name	Optional. Type the machine username. If the user is a domain, you are required to also specify the domain.
Password	Optional. Type the machine password. ● Back Office - The robot cannot be executed from the server without credentials. ● Front Office – not available

Note: A Back Office robot should request from the server the Login credential. When it receives a Start notification through the SignalR channel, it goes to the Assets service (which shares the access Endpoint with the Configuration service) and gets its configuration assets. It identifies itself with the Robot ID and asks for the Login credential (to open up a Windows session on the client computer). These credentials are stored encrypted in the database. When they are sent, they are sent unencrypted, through an HTTPS channel. After the robot has the login credential, it logs onto Windows, and starts execution.

3.2. Editing a Robot

1. Click the **Edit** button.
2. Make the modification.
3. Click the **Save** button.

3.3. Deleting a Robot

1. Click the **Delete** button.

Note: You cannot delete a robot that is available or running.

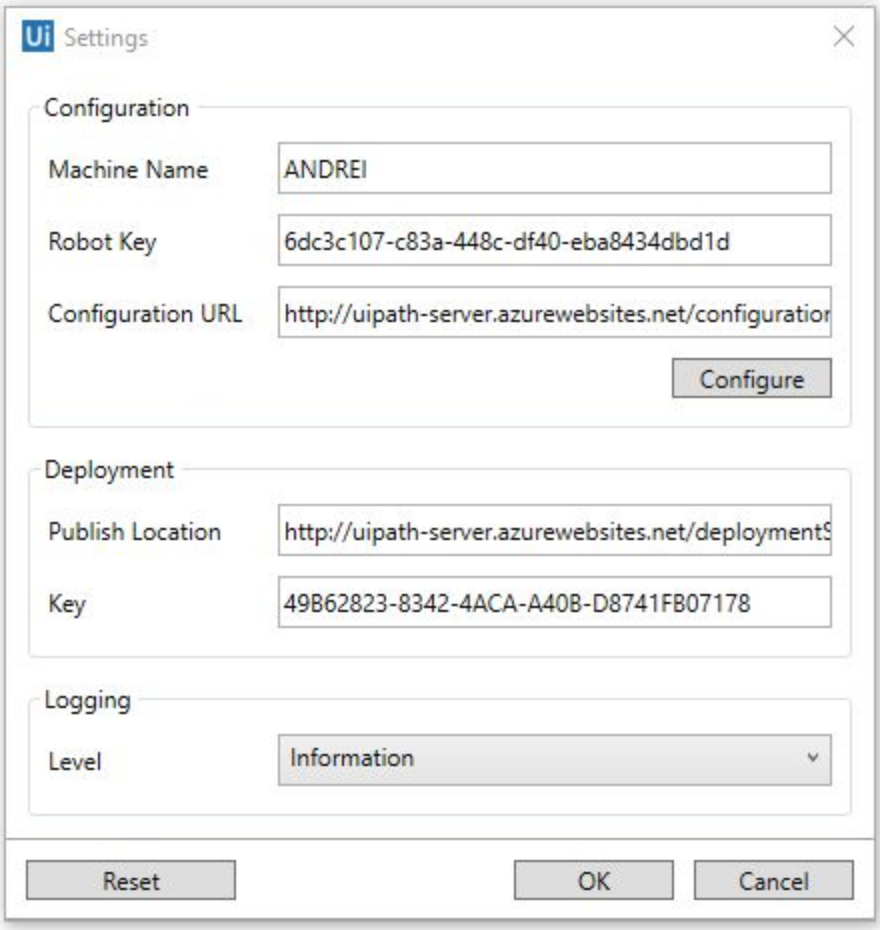
3.4. Connecting a Robot to Orchestrator

1. After you provision the robot on the server and you copy the Robot Key (Adding a New Robot), open **UiPath Robot** agent Tray on the robot machine.

1 - Refresh
2 - Settings
3 - Running or Available processes
4 - Running processes tab
5 - Availble processes tab

2. Click **Settings**. The **Settings** window is displayed.

Note: To open the **Settings** window, you need to have administrator rights.



Note: To save the settings you have to click OK (if you use close or cancel the settings will not be saved).

3.4.1. Field Descriptions for the Settings Window

Field	Description
Machine Name	The name of the computer.
Robot Key	The key generated from the server. Copy paste that key or send it to the administrator.
Configuration URL	The configuration URL. It can be found on the Settings page.
Configure	Click the Configure button after you complete all above.
Publish Location	This is automatically inserted from the server (after you press Configure).
Key	This is the Nuget key. It is automatically inserted from the server, after you press Configure.
Logging	The logging level for the current machine. For more details about logging, see Logging Levels in UiPath.)

3.5. How it Works (Client - Server)

After connection, at server level, in the Monitoring web service, there is a method that fires, called “Register Agent Start”, which gets called when you click “Configure” in the above dialog.

The client goes to Monitoring through an HTTPS channel with this information - machine name, robot id - and requests for registration (Registers with the server).

The Monitoring web service on the server looks it up in the database, to check if such a robot has been defined.

If it hasn’t it throws an error, and notifies the robot that it doesn’t exist on the server.

If the robot name is found, the server generates and sends back:

- a session ID (called Agent Session ID) and
- an Endpoint for the Configuration service.

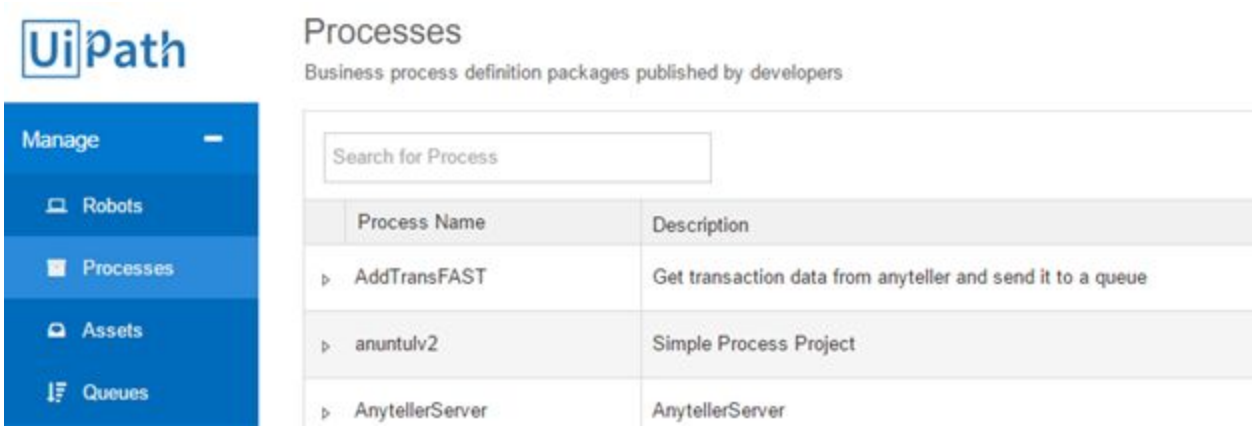
The Agent Session Id is saved by the client in the client’s local settings. The Configuration Endpoint is used by the Robot on the client machine to go back to the server and authenticate itself with Robot ID, in order to retrieve its settings. All is done via HTTPS channels. The Robot retrieves all its associated settings, such as:

- Deployment Endpoint,
- Logging Endpoint,
- a NuGet API key,
- a SignalR hub address (endpoint) and
- a SignalR hub name.

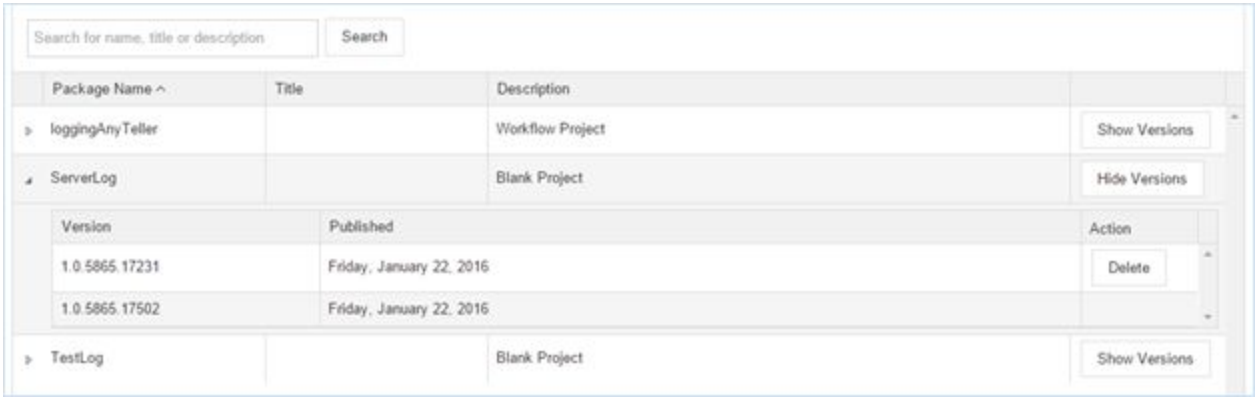
This is when the Robot registers itself to the SignalR channel, after which it starts to send to the Monitoring endpoint, every 30 seconds, a “Heartbeat”. Regardless of whether the robot is idle or is executing a process, if the Windows machine is on and UiRobot service is running, it will send a heartbeat to the server in order for the server to know its status (Available, Running, Busy, Offline). The “heartbeat” message contains information such as Robot ID, Agent Session ID, and name of the running process (or NULL if it’s not running a process). Here, NULL means that the Robot is available. If the Robot fails to send the heartbeat for more than a minute, the server classifies the robot as not available (and sees it as such).

3.6. Processes - page

Business process definition packages published by developers.



On this page, the server administrator can manage all the published processes.



Note: You cannot delete a version that is active (associated with an environment).

4. Assets

Shared variables or configurations used in processes.

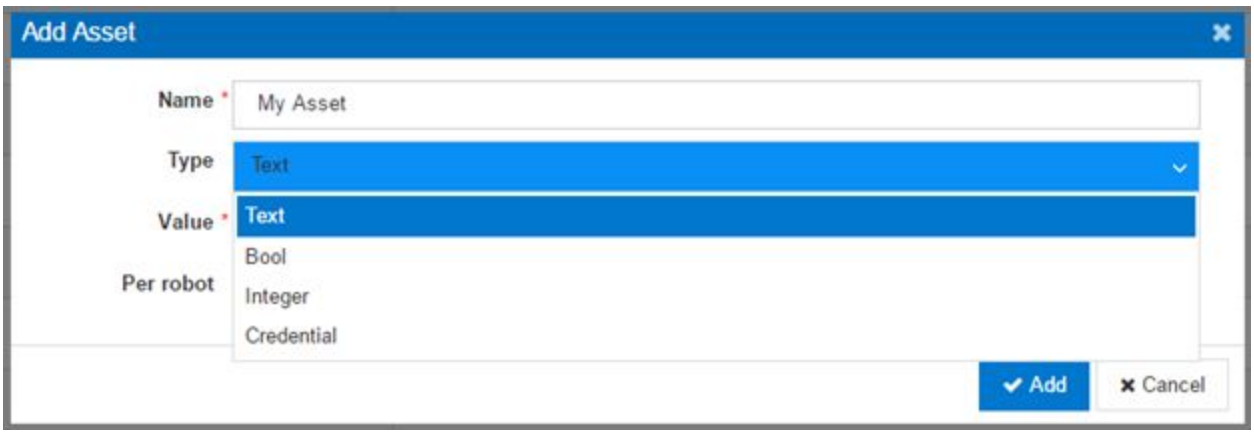
Assets can be requested by the workflow, through special activities. “Get Robot Credential” is one of them, and “Get Robot Asset” is the other.

These have access to the Configuration Endpoint, know the Robot Key, and together with the Robot Key and the Credential names they request, from the Configuration service, the required assets.

4.1. Managing Assets

There are 2 types of assets, global asset (for all the robots) and assets per robot (assets for a specific robot):

- Text - string
- Bool - true or false
- Integer
- Credential - credentials used by robots (e.g. Login details for SAP or Login details for SalesForce)



To add a global asset don't check value per robot.

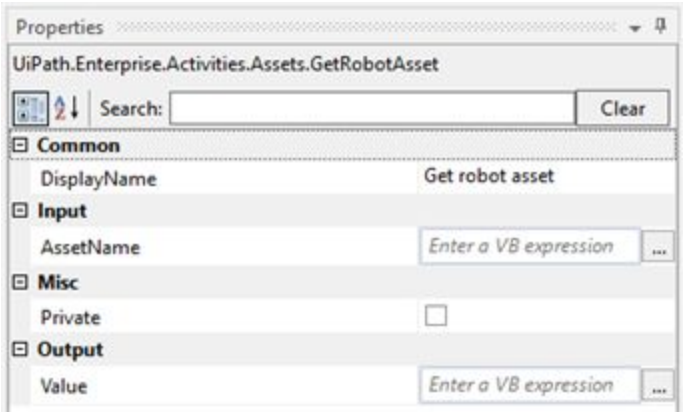
To add an asset for a specific robot check value per robot and choose the robot from the drop down list.

4.2. Using Assets in UiPath Studio

In the activities panel we have 2 activities designed for assets: **Get Robot Asset** and **Get Robot Credential**.

Get Robot Asset

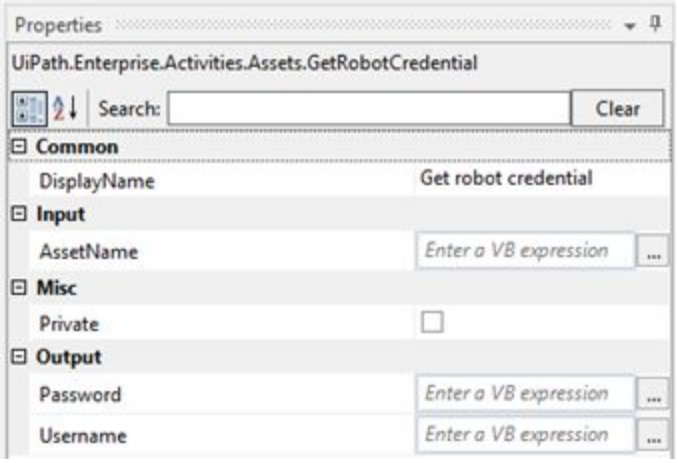
Input > AssetName: This field should have the same name as the asset from the server.



Get Robot Credential

Input > AssetName: this field should be identical with the asset name from the server Output: Password and Username.

Note: The asset name is not case sensitive. For example, “Text” and “text” are the same.



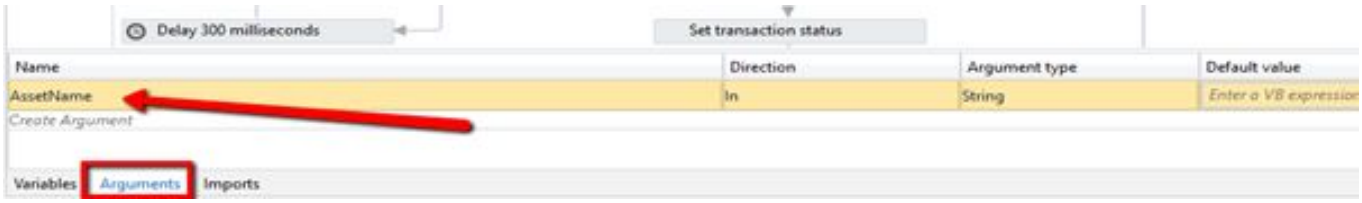
4.2.1. Difference between Get Robot Asset and Get Robot Credential

There is the following difference between **Get Robot Asset** and **Get Robot Credential**:

The asset’s value is sent as a String, Bool or Int variable, while Robot Credential is a 2-field structure: a String representing the username, and a SecureString representing the password. The SecureString is a special .NET Framework type which is encrypted within the framework.

You can access its unencrypted form by using special functions defined in .NET to access its contents.

4.2.2. Automatically Mapping Arguments with Assets



On the **Argument** tab, create an argument with the Asset name. You can use this argument in your workflow similar with a variable.

Note: This functionality works only with back office robots (robots that are started from the Orchestrator).

5. Queues

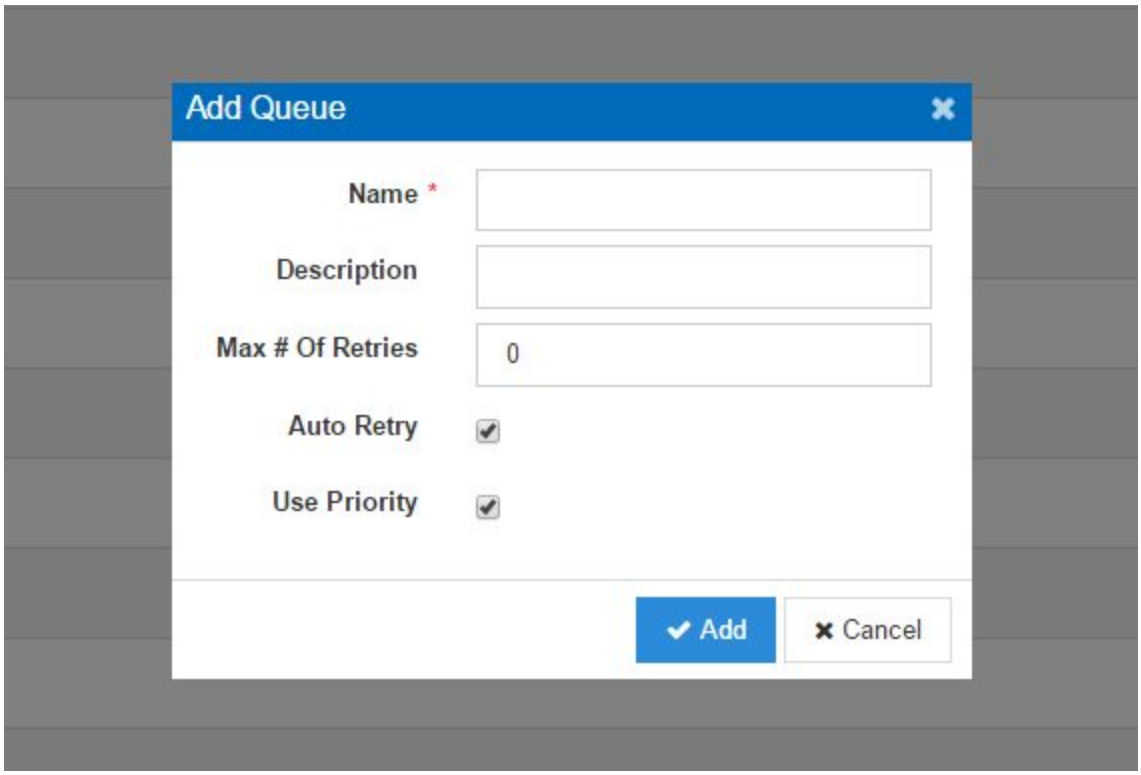
Work Queues - An industry standard platform capable of queuing large volumes of transactions.

5.1. User Interface

5.1.1. Field Descriptions for Queues

Field	Description
Add Queue	Create a new queue.
Search	Search for a specific queue.
Show Item	Displays the queue items.
Delete	Delete a queue
Queue Grid	
Name	The name of the queue.
Description	The description of the queue.
Use Priority	Boolean (True/False).
Max # of Retry	The maximum number of retries.
Auto Retry	Boolean (True/False).

5.2. Adding a New Queue

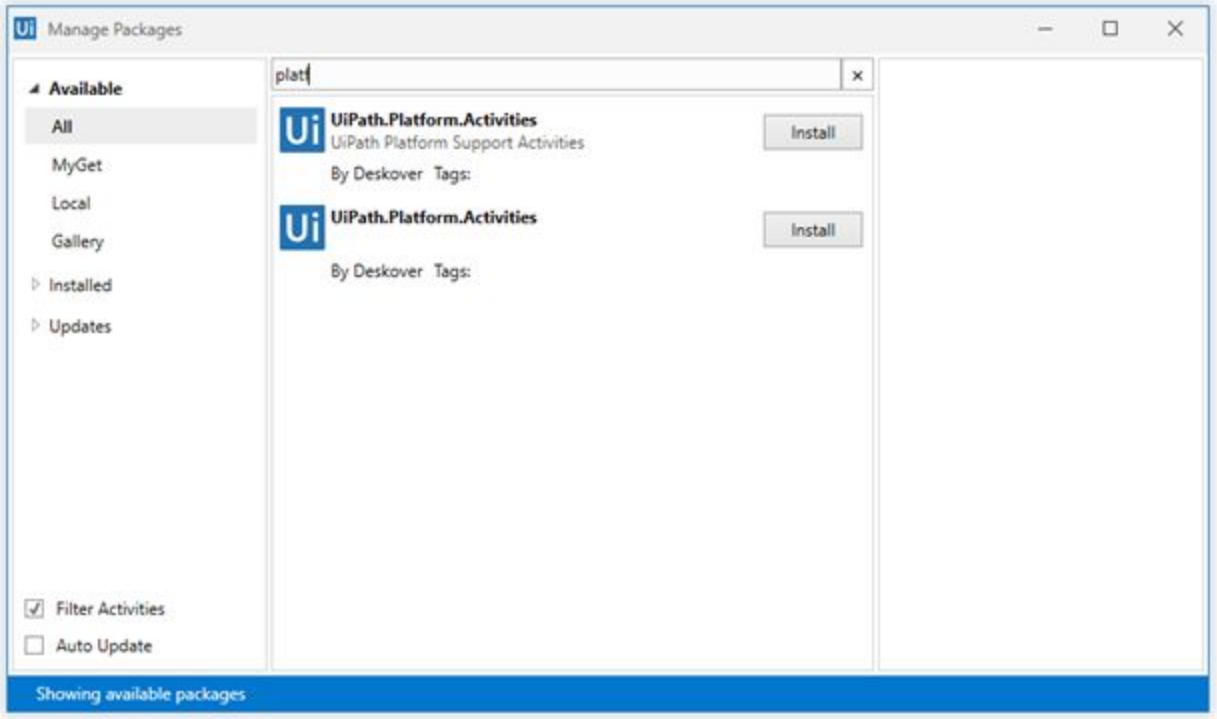


5.2.1. Field Descriptions for Add Queue Window

Field	Description
Name	The name of the Queue (this name is used in UiPath Studio -> QueueName).
Description	An intuitive description for the new queue.
Max # of Retries	How many times server will retry to complete an item in case of Application Exception.
Auto Retry	True or False
Use Priority	If the new queue uses priority (the priority is set per item, by default each item's priority is low. Setting the priority is made in UiPath Studio)

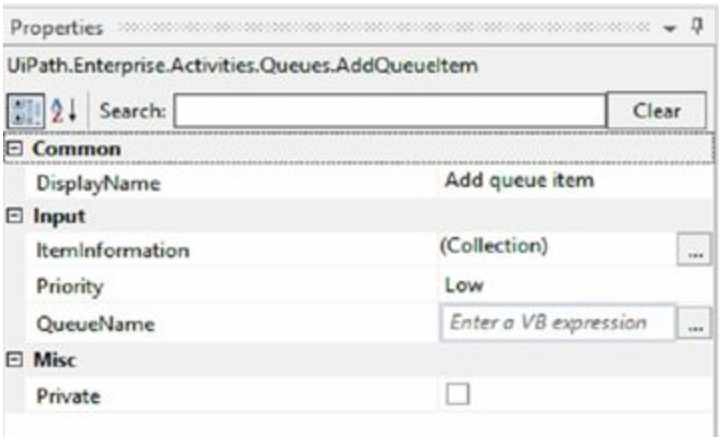
5.3. Adding Items to a Queue in UiPath Studio

1. In UiPath Studio install the UiPath.Platform.Activities package from the package manager.

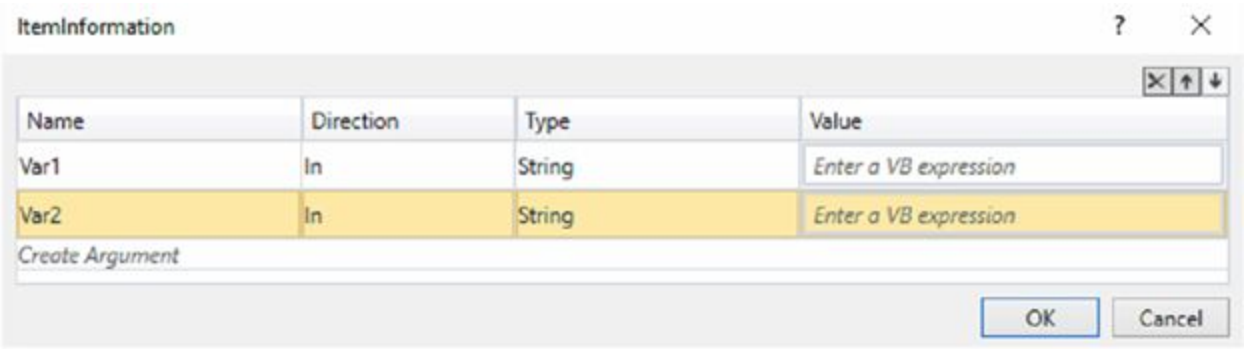


2. From the **Activities** panel drag the **Add Queue Item** activity.

In the activity **Properties** panel:



- **ItemInformation** - the collection of data that will be sent to the server



- **Priority** - Low, Normal and High
- **QueueName** - should have the same name with the queue created on the server side

5.4. Getting Items from a Queue in UiPath Studio

1. From the **Activities** panel drag the **Get Transaction Item** activity.

QueueName - should have the same name with the queue created on the server side

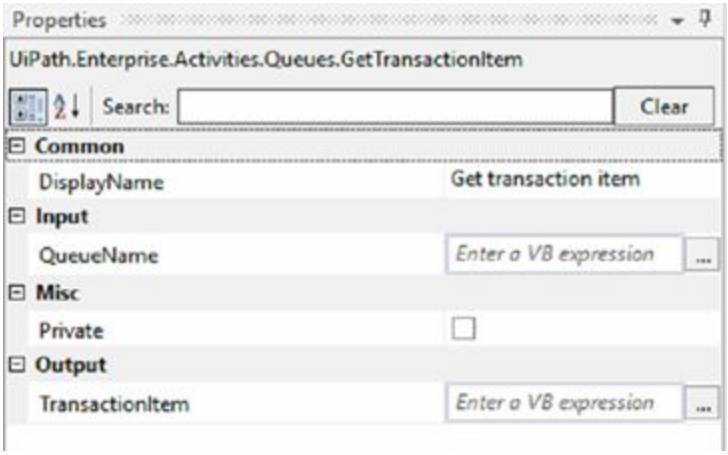
Output, **TransactionItem** - the item from the queue

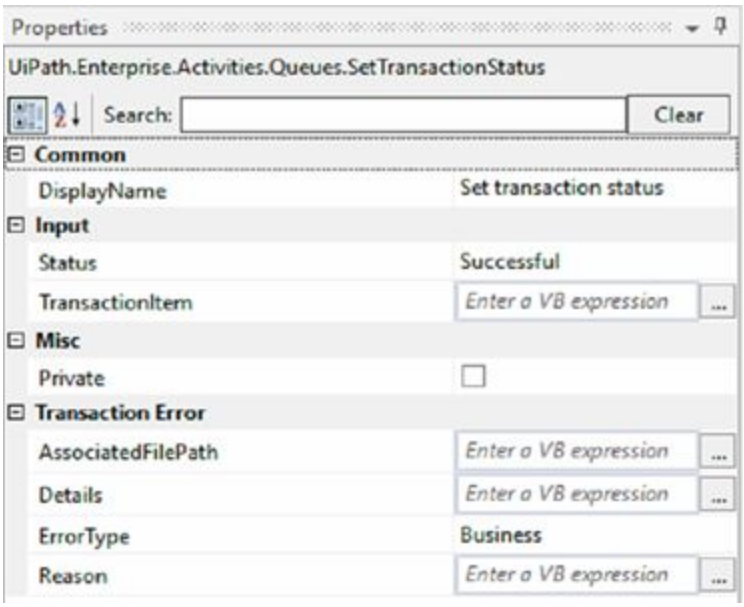
Note: When the Get Transaction activity is used the item will change status on the server with: "InProgress."

2. The next step is to process the item and set the transaction status.
3. From the **Activities** panel drag **Set transaction status** activity.

In the activity properties panel:

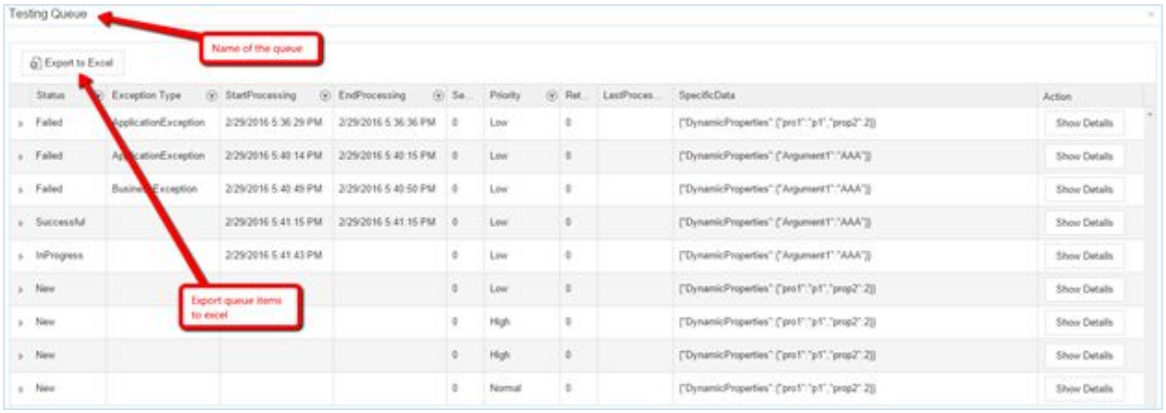
- **Status** - Successful or Failed
- **TransactionItem** - the item from the **Get** activity
- **Transaction Error**
  - AssociatedFilePath
  - Details
  - Error Type: Business or Application
  - Reason



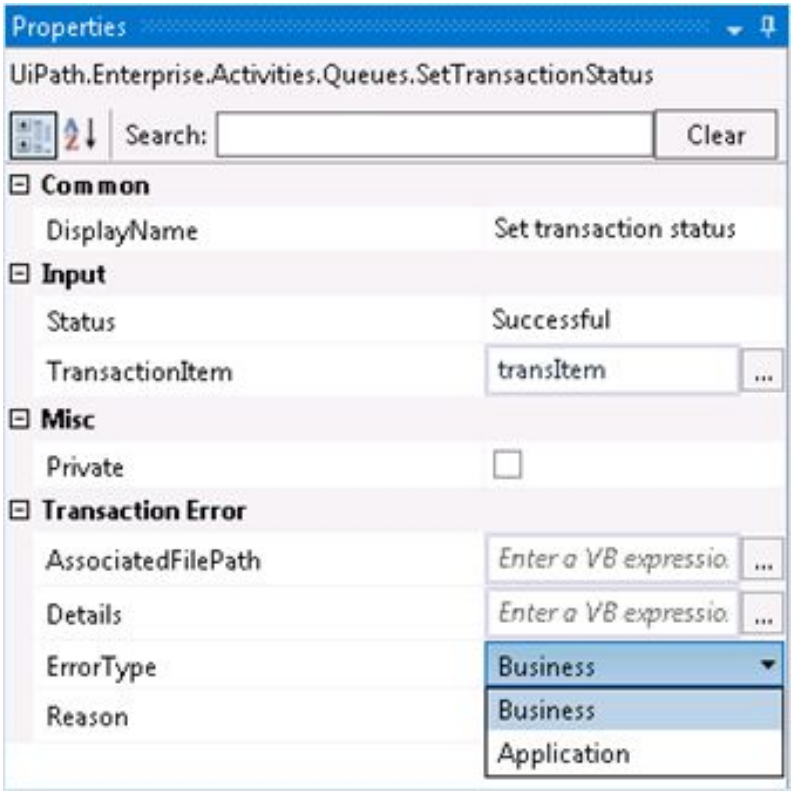


5.5. Show Queue Items

Click on the **Show Items** button.



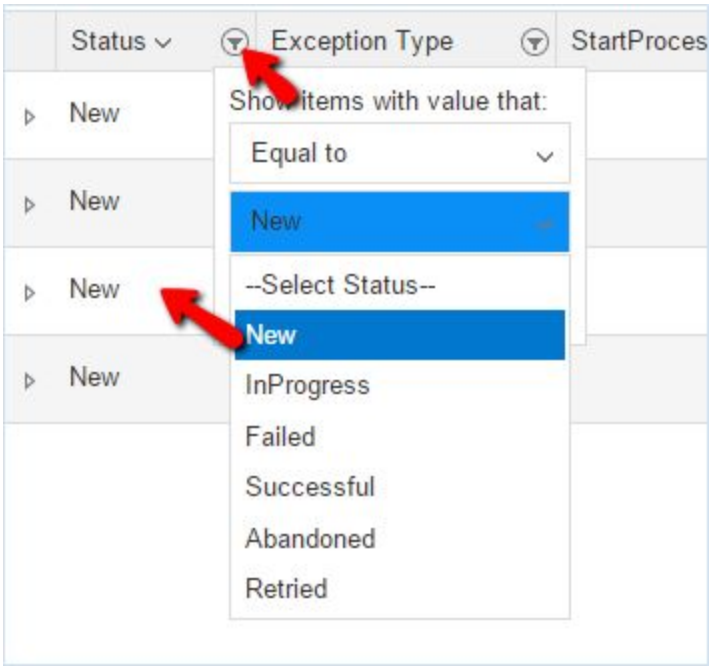
- **Status** - queue item status:
  - Failed - the item was processed and Failed (Business exception or Application exception)
  - Successful - the item was successfully processed
  - InProgress - the item is in progress
  - New - the item is new in the queue
  - Abandoned - there is a job that will transform all the InProgress items in Abandoned
  - Retried - retried item
- **Exception Type** - If the status is Failed this column will display the Exception Type (set from UiPath Studio):
  - ApplicationException - the transaction failed during an application error (e.g. windows crashed and the robot stopped)
  - BusinessException - the transaction failed with business exception



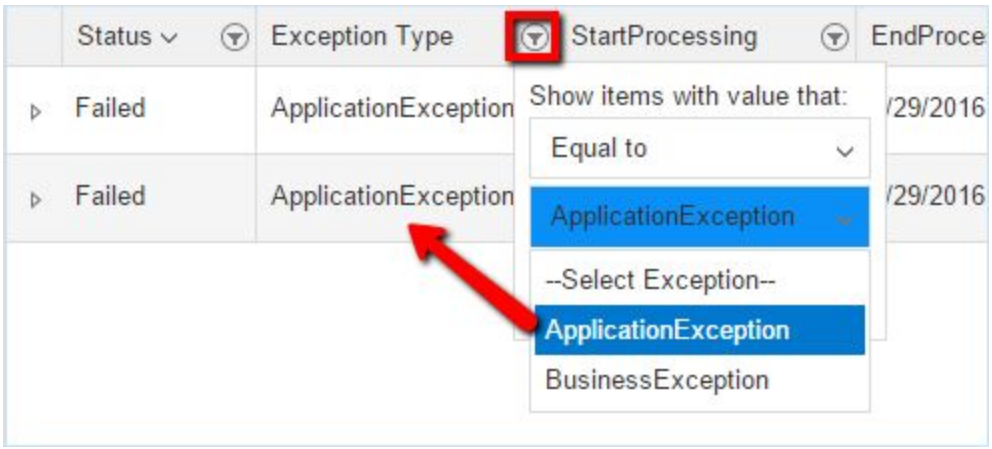
- **StartProcessing** - the start time, item with status InProgress
- **EndProcessing** - after the transaction is finished (successfully or failed)
- **SecondsInPreviousAttempts** - time spent on previous attempts (available for retried items only)
- **Priority** - Low, Normal, High
- **LastProcessingOn**
- **SpecificData** - item data

5.6. Column Filters

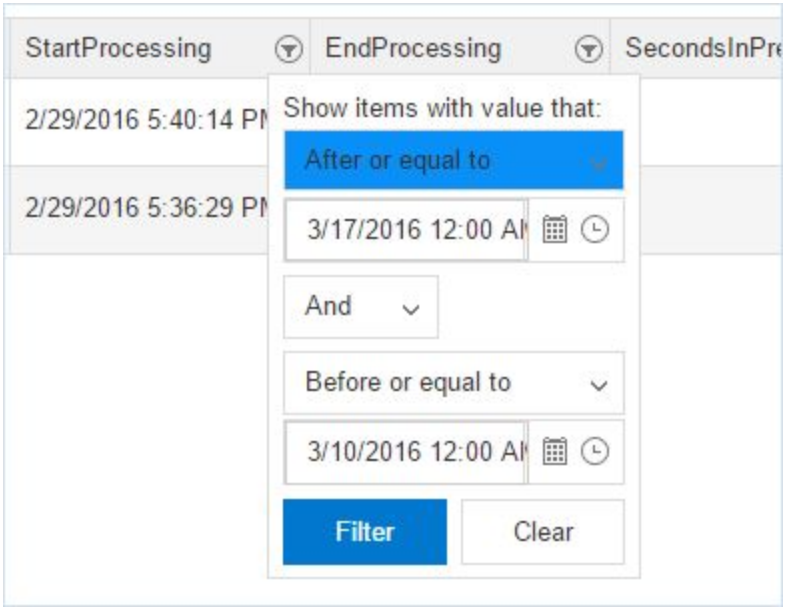
Filter by Status



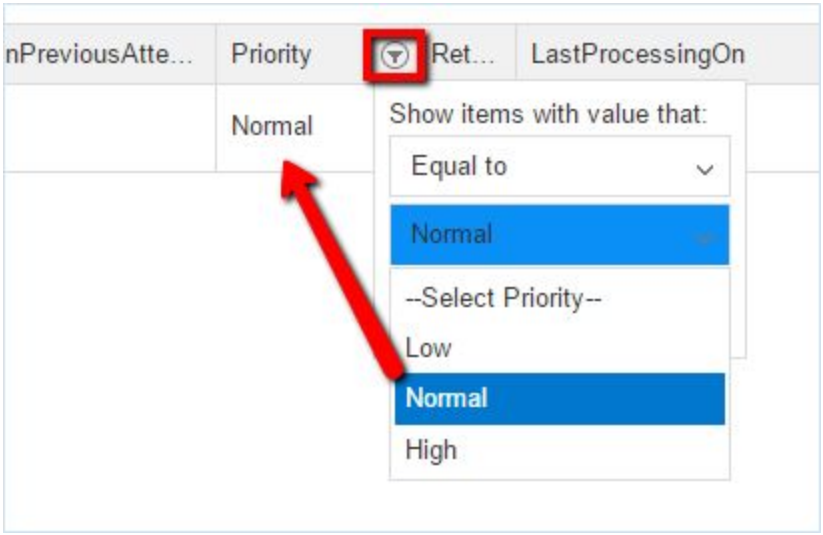
Exception Type



Start/End Processing

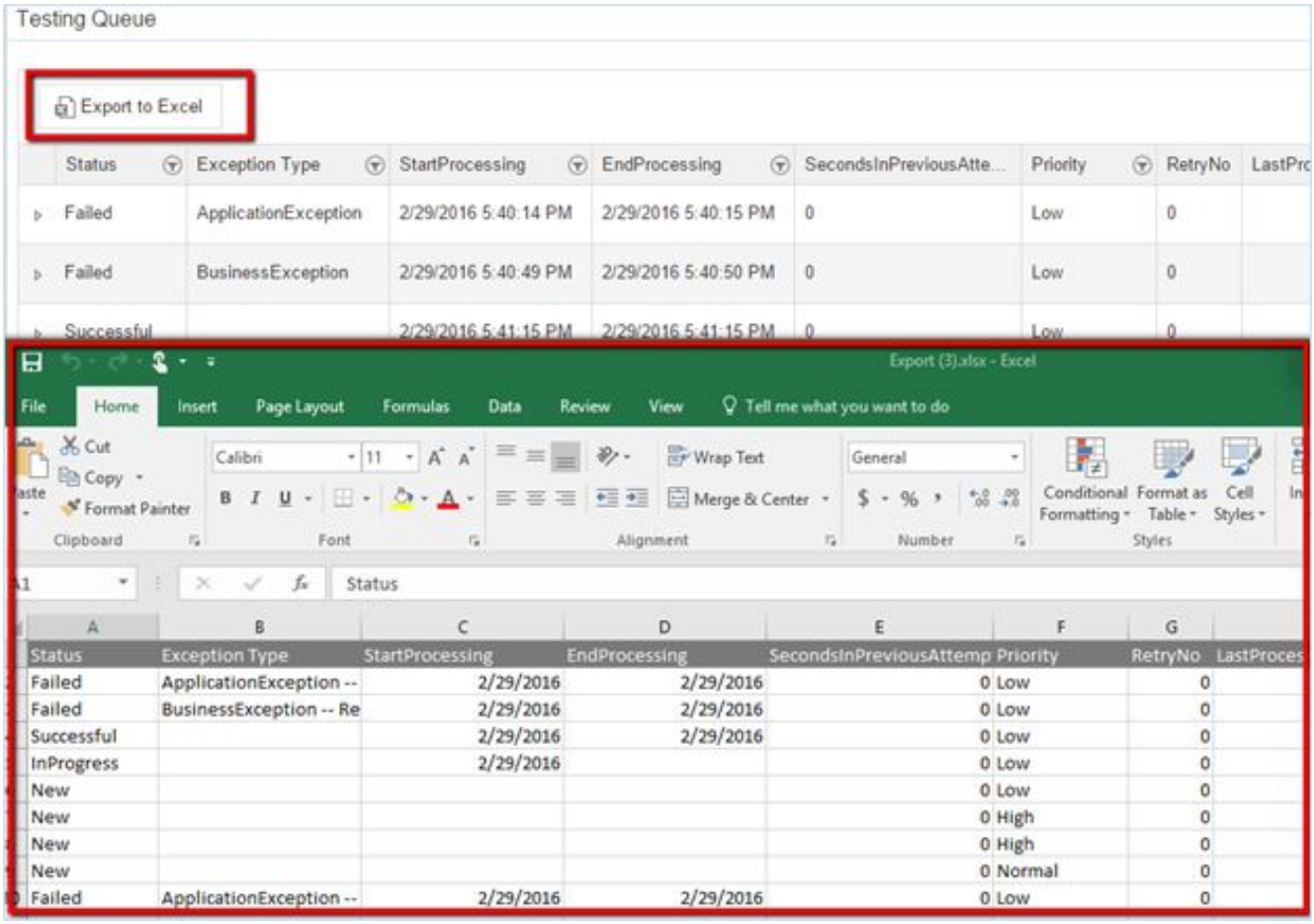


Priority



5.7. Export to Excel

Export the items in an .xls file according to the selected filters.

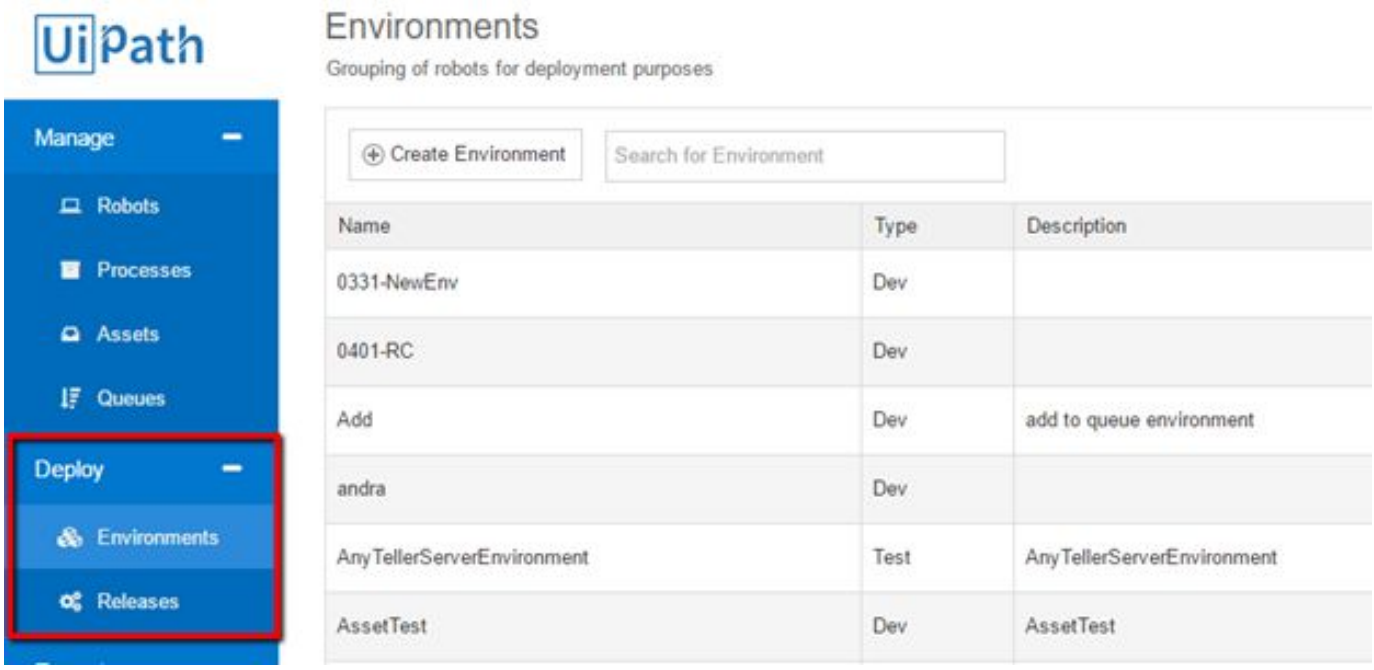


5.8. Delete a Queue

Click the **Delete** button (This should be done only by a reviewer only).

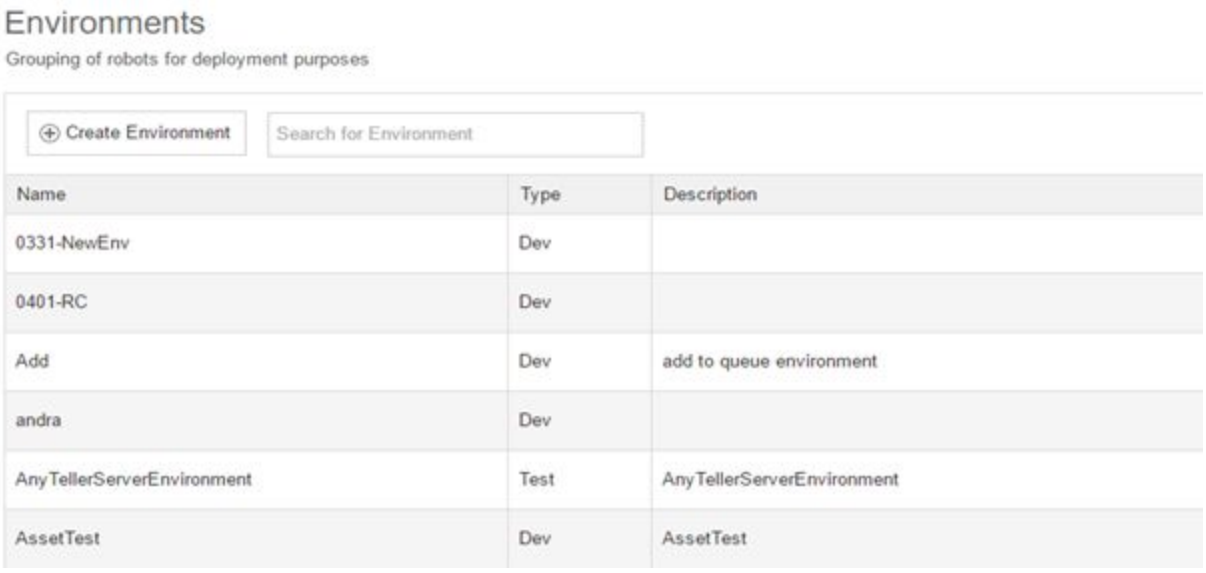
When a queue is deleted all items will also be deleted.

6. Deploy panel



6.1. Managing Environments

Note: The Processes are distributed to all robots from the Environment (e.g. if the Process “X” is associated with the environment “A”, all the robots from environment “B” will not be able to see the Release “X”).

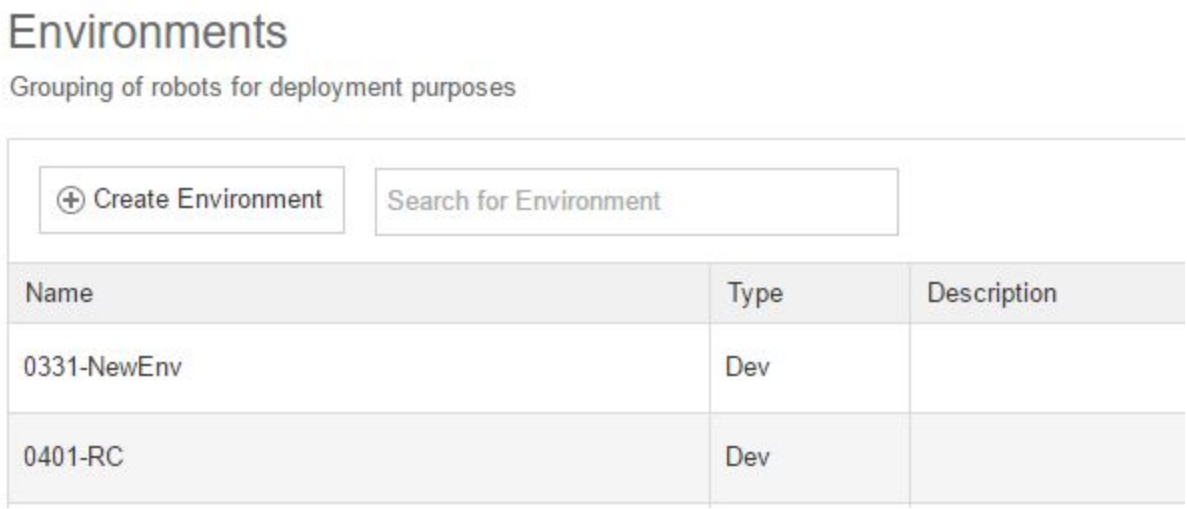


- To sort a grid -> click on the column head

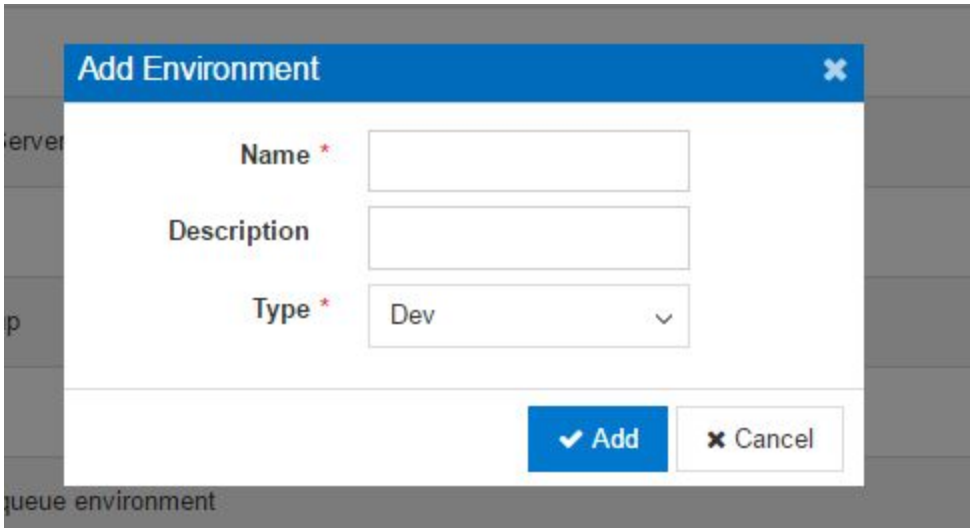
- Use search to find a Environment using a part of his name or description
- Action buttons: Manage Robots, Delete

6.1.1. Creating a New Environment

1. Click **Create Environment** button.



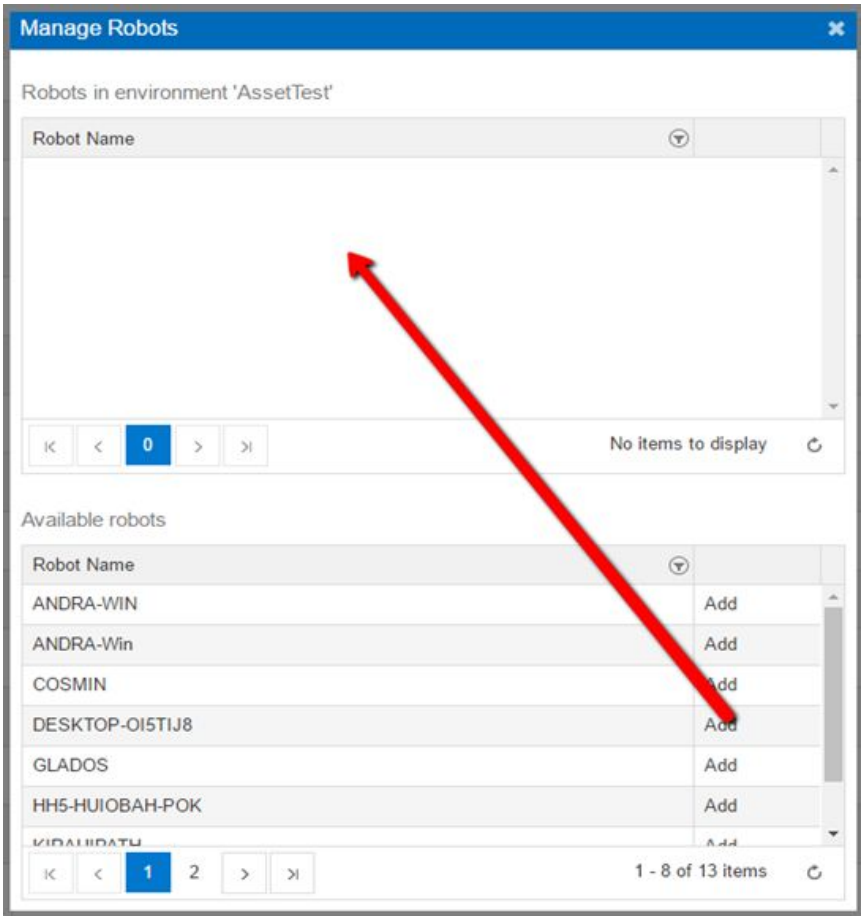
2. Complete the form (name of the environment, description and choose a type)



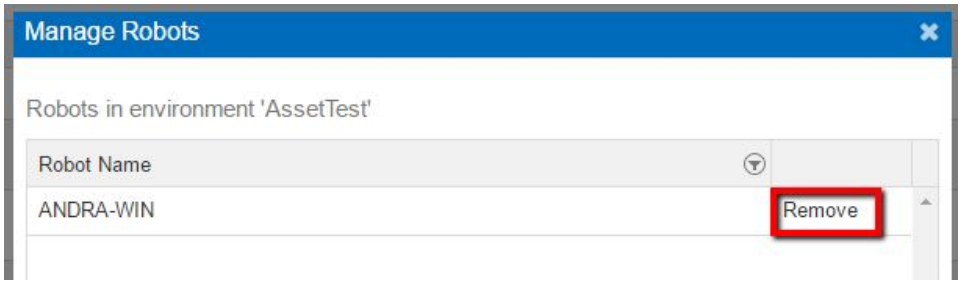
Note: You cannot edit the details for an environment (name, description or type).

6.1.2. Managing Robots and Environments

1. Click the **Manage Robots** button.
2. Click the **Add** button for the robot that you want to add (Available Robots grid, see the screenshot).



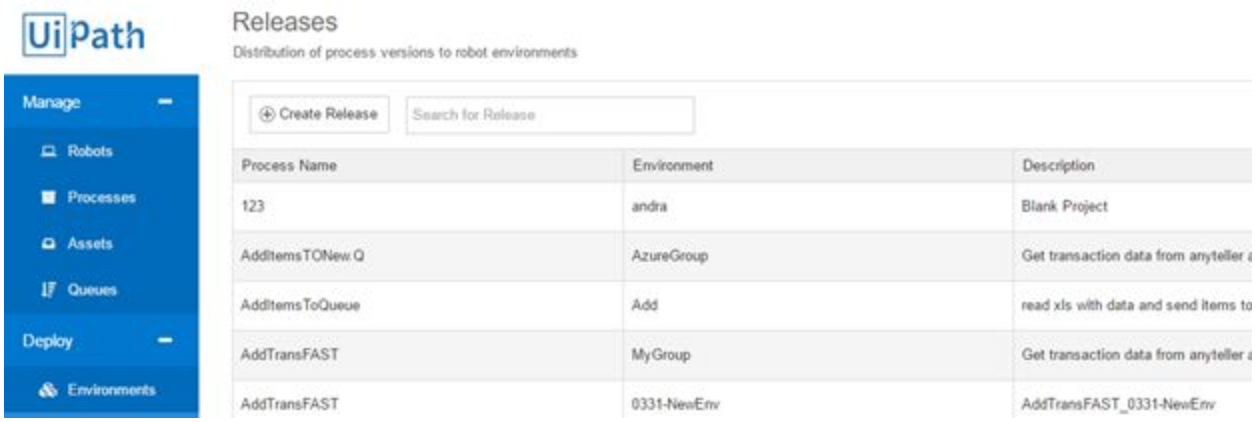
To remove a robot from the **Environment** click the **Remove** button.



Note: To delete a group, just click the **Delete** button from the main grid.

6.2. Releases

Distribution of process versions to robot environments.

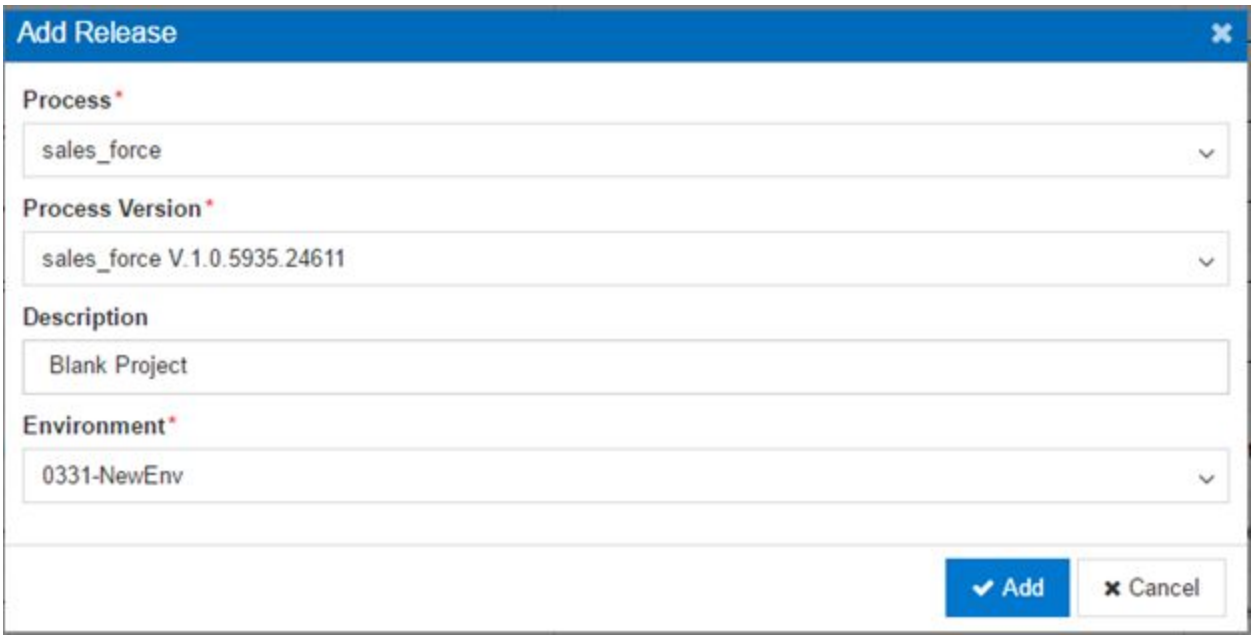


6.2.1. Managing Releases

- Process Name - the name of the process
- Environment - the associated environment for the current process
- Description - description of the process (optional but recommended)
- Current Package Key - published package that is current selected (distributed)
- Search - You can use search to find Processes based on a part of the name, description, robot group or package key.

6.2.2. Create a new Release

1. Click the **Create Release** button, in the following pop-up select:
    - Process - the process that you want to use
    - Process version - the version of the process that you want to use (more details about versions here: Processes page)
    - Description - default is the description of the Process (Project description from UiPath Studio)
    - Environment - select the environment that will use this Process

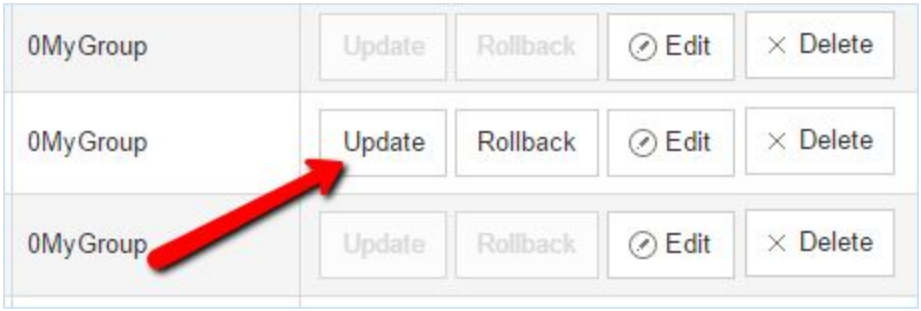


2. Click the **Create** button. The Release is created.

Note: The Process will be deployed to the Robots from the selected Environment. The Environment is mandatory, you cannot create releases without one.

6.2.3. Actions in Releases grid

Update a process with the last version of the package: If a newer version of the package is published the **Upload** button is enabled.

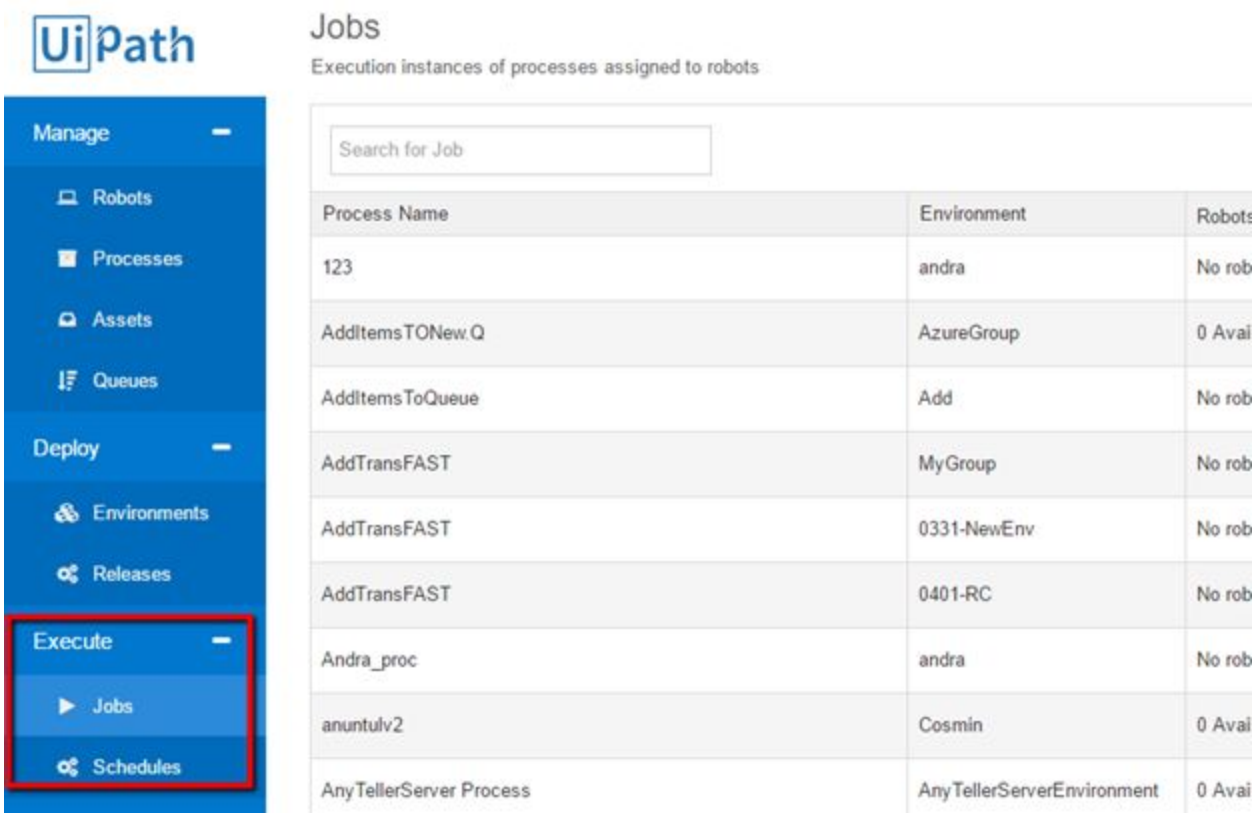


Rollback to previous used version of the process: Click the **Rollback** button.

To delete a **Release**, click the **Delete** button.

Note: Deleting a **Release** is not going to delete the process from the server, only its association with the Environment (to delete the process you have to visit the Processes page).

7. Execute Panel

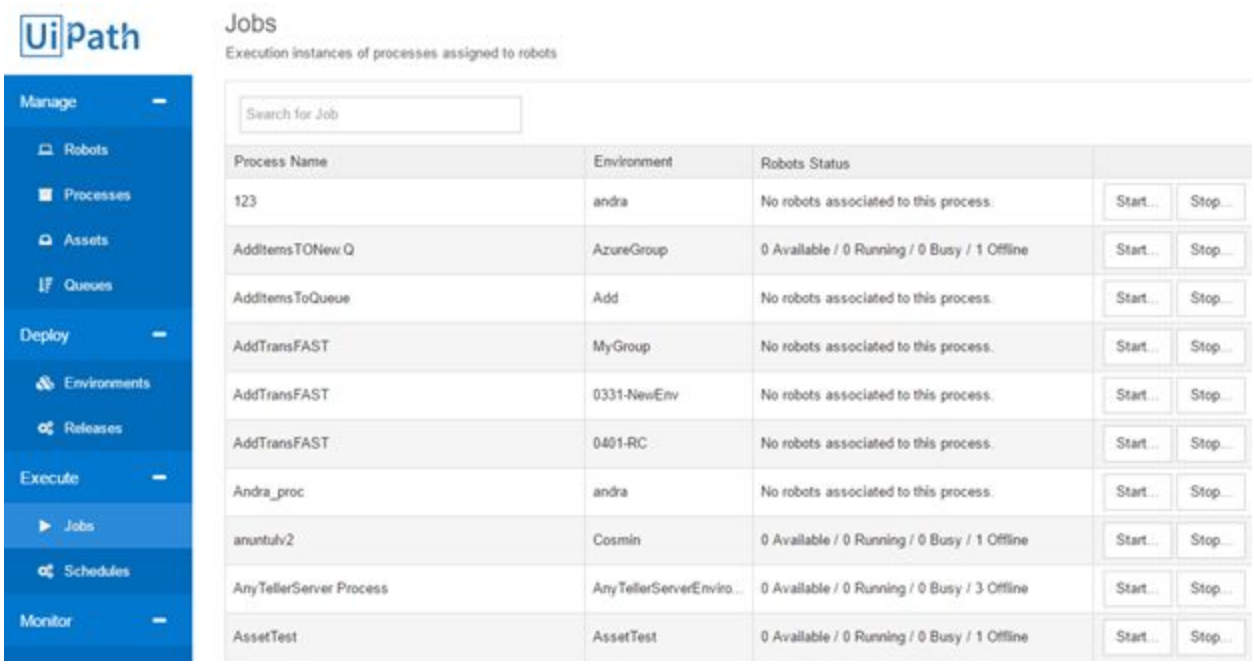


7.1. Jobs

Execution instances of processes assigned to robots

After registration is done (details here: How to connect a Robot to Orchestrator), there are two ways of executing the process: Back Office Robot (or BO robot) and Front Office Robot (or FO robot).

For BO, the server displays all the processes associated with a group of Robots.



- Process Name - the name of the process
- Environment - associated with the process
- Robot Status (the number of the robots with their current status)
    - Available (ready to start)
    - Running (running a process)
    - Busy (the robot is running another process)
    - Offline (the robot is offline, uirobot svc is stopped, the robot machine is offline)

7.1.1. Start

Start the selected robots.

- A robot will run only if the credentials are provided (username and password, more details here: How to add a new Robot)
- You can start a robot only if the status is Available
- The selected robot will start executing the process

7.1.2. How it Works (remote execution for BO robot)

- When you click the “Start...” button on the server, after you choose at least one available robot, a notification is sent on the SignalR hub channel to the Robot. (Each notification has a specific name).
- When you press the Start button, the server sends out a “Start process” notification. This type of notification has multiple parameters: package name, package version, process id.
- The client receives this notification and its associated information, and looks it up locally first. It checks if the package is available and the current version is the requested version, and executes it if

it's so. All this time, the heartbeat is continually sent. When the next heartbeat is sent to the server after the process starts, the client knows - it's executing something, and sends the appropriate Process ID to the server.

- If something is missing - either the package entirely or the requested version, the client contacts the deployment endpoint, the one it received at configuration, together with the NuGet API key - which is used to validate on the NuGet feed that it is authorized to read and write from there - and asks for the missing information on the server (package name and version). It waits for the packet to download, then tries to execute it. If the download process takes longer than one heartbeat period, the client sends, with the next heartbeat, a NULL for the running process. The server re-sends the "Start process" command and displays the robot machine as "Busy". This can happen several times. When the client manages to start the process, it sends out an Acknowledgement to notify the server that it has started to execute the process. The server changes the status of the robot from "Busy" to "Running".

Most common user mistakes

- The robot starts but becomes available very fast
- Username and password are not provided
- Username or password is not correct
- The robot didn't receive the process (package) for the selected release

Note: We recommend that you check the logs for more details about the execution.

7.1.3. Stop and Force Stop

- **Soft stop** the selected robot. The robot will stop when the process is finished or the Should stop activity is executed.
- **Force stop** the selected robot. The robot will stop immediately!

Most common user mistakes

- The robot status is changed in Busy, not in Available
- The Robot didn't finish the process or the process doesn't have a Should Stop activity

7.1.4. How it Works

1. The user at the server's web interface picks a running robot and presses the Stop button.
2. The server sends a SignalR Stop command which tells the client to stop the process with this specific ID. The client checks if it's executing such a process, and if it doesn't, it immediately responds (with Acknowledgment) that it has stopped this process (this happens even if it hadn't had such a process running).
3. If it had been running the process, it first tries to stop execution. When the execution is really stopped it sends the Acknowledgement that it has stopped. While doing so, it goes from "running" to "busy".

Note: A process can also self-stop, when it's not designed to run infinitely. When it terminates the execution, it's sending its own acknowledgement to the server, that it has stopped. A notification is not needed from the server in this case.

For FO robots, the start is manually done on the client-side. After the client starts running a process, when the heartbeat is sent, since it has no process id from the server, the client sends only its process name (it still appears as busy), but can't map its logs to an execution.

You can't see the Remote Execution side of the BOS in FOS, but you will see in the Robot Monitoring section on the server that the robot is executing something, it's not available.

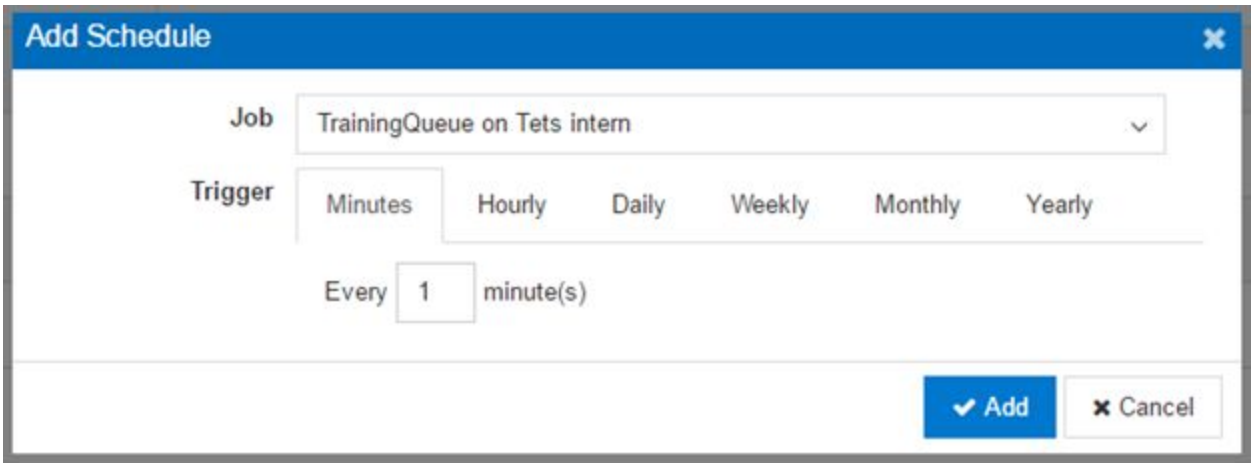
It is still logging for itself but the information logged by the FOS robot is displayed in the Robot Logs page and not in the Process Logs.

Since FO robot is manually started, it only affects the heartbeat. **The server only knows the name of the process that's communicated by the client**.

This is generated from the workflow - you can, for instance define on the server a process named X, and in your local workflow name the same process as Y - the server can't know this, can't match them through process ID.

7.2. Schedules

Triggers for scheduled job execution



- Job - the job name is: Process name + Environment
- Triggers: Minute, Hourly, Daily, Weekly, Monthly, Yearly

8. Monitor panel

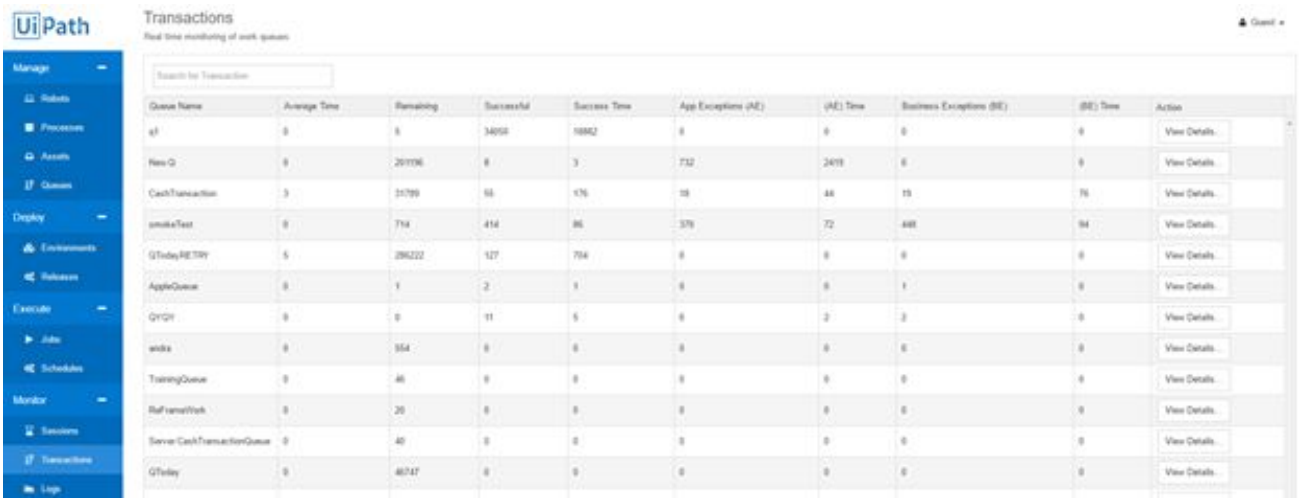
8.1. Sessions

Real time monitoring of robots

- Robot Name - the name of the robot
- Status - the robot status:
  - NotResponding
  - Available
  - Running
  - Offline
  - Starting
  - Stopping
- Reporting Time
- Running Process
-

8.2. Transactions

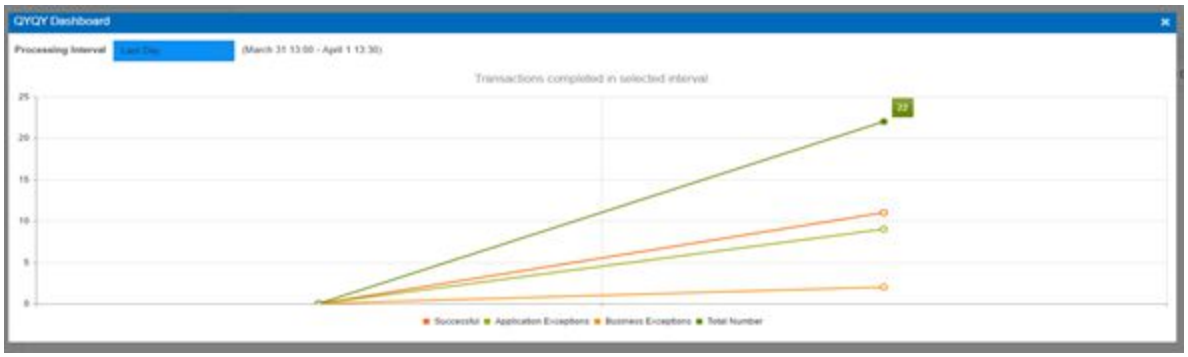
Real time monitoring of work queues



8.2.1. Transactions Grid

- Queue Name - the name of the queue
- Average Time - the mean of the queue item processing time (expressed in seconds);
- Remaining - number of items remaining in the queue
- Successful - number of successful transactions in the queue
- Success Time - total processing time for successful transactions;
- App Exceptions (AE) - total number of transactions failed with application exceptions;
- (AE) Time - total processing time for application exceptions (expressed in seconds);
- Business Exceptions (BE) - total number of transactions failed with business exceptions;
- (BE) Time - total processing time for business exceptions (expressed in seconds);

8.2.2. View transaction details



Processing interval

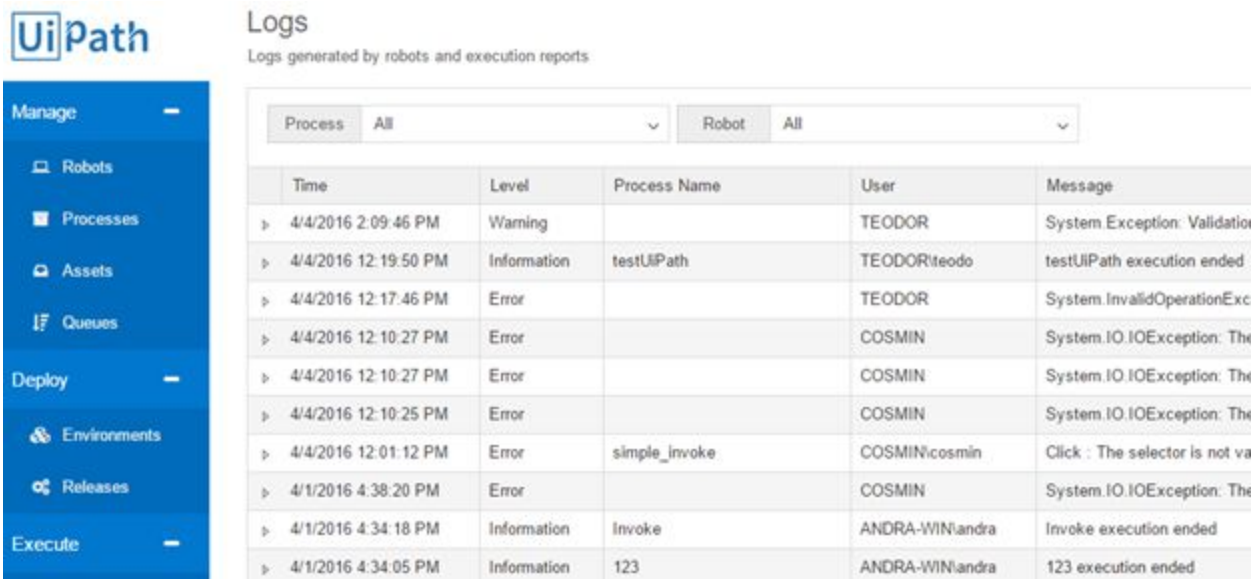
- Last Hour
- Last Day
- Last Week
- Last 30 days

Filters

- Successful
- Application Exceptions
- Business Exceptions
- Total Number

8.3. Logs

Logs generated by robots and execution reports



Filters: Process and Robot

9. Logging in UiPath Studio and Orchestrator

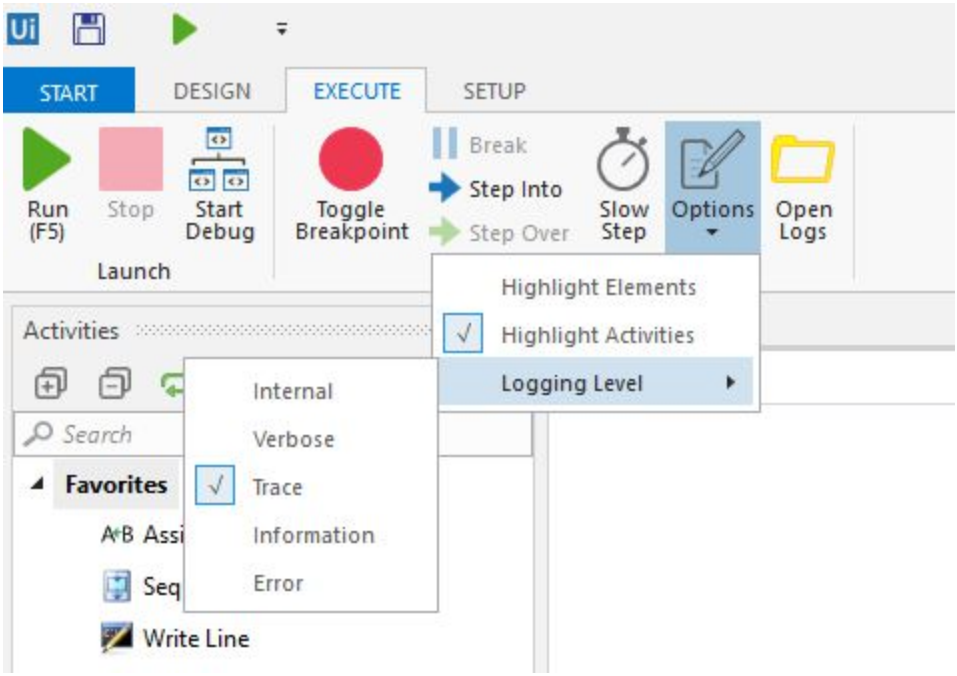
9.1. Logging Levels in UiPath

Logs are sent to server only if the UiRobot service is running.

Logging Levels in UiPath					
Log Level	**Logged**	**Example / Comment**	**Log File**	**Output**	**Server**
Internal	Activities	Trace {"message":{"DisplayName":"Message box",**"State":"Executing","Activity":"UiPath.Dialog.Activities.MessageBox",**"Arguments":{"Caption":"","Text":"String in message BOX"}... Trace {"message":{"DisplayName":"Message box",**"State":"Closed","Activity":"UiPath.Dialog.Activities.MessageBox"**,"Arguments":{"Caption":"","Text":"String in message BOX","ChosenButton":"Ok"}	YES	NO	YES
	Variables	"Variables":{"NewTransaction":"False"}}			
	Arguments (properties)	"Arguments":{"Caption":"","Text":"String in message BOX","ChosenButton":"Ok"}			
	Executor	"WorkflowRunner: OnInvokeJob C:\\Users... "NotifyJobCompleted End","timeStamp":....			
Verbose	Activities	Trace {"message":{"DisplayName":"Message box",**"State":"Executing","Activity":"UiPath.Dialog.Activities.MessageBox",**"Arguments":{"Caption":"","Text":"String in message BOX"}... Trace {"message":{"DisplayName":"Message box",**"State":"Closed","Activity":"UiPath.Dialog.Activities.MessageBox",**"Arguments":{"Caption":"","Text":"String in message BOX","ChosenButton":"Ok"}	YES	NO	YES
	Variables	"Variables":{"NewTransaction":"False"}}			
	Arguments (properties)	"Arguments":{"Caption":"","Text":"String in message BOX","ChosenButton":"Ok"}			

Trace	Activities	Trace {"message":{"DisplayName":"Main","State":"Executing","Activity":"System.Activities.DynamicActivity"} Trace {"message":{"DisplayName":"Main","State":"Executing","Activity":"System.Activities.Statements.Flowchart"}	YES	NO	YES
Info	WriteLine Log Message	Info {"message":"message from activity" **Note:** Except messages logged with Trace level set in activity	YES	YES	YES
Warning	Warnings	Warn {"message":"Warning from log message activity"	YES	YES	YES
	Errors	Error {"message":"Error from log message activity"	YES	YES	YES
	Critical	Critical Errors	YES	YES	YES
Error	Errors	Error {"message":"Error from log message activity"	YES	YES	YES
	Critical + Fatal	Critical Errors	YES	YES	YES
Critical	Critical + Fatal	Critical Errors	YES	YES	YES
OFF	n/a	n/a	NO	NO	NO

9.2. Debug options in UiPath Studio



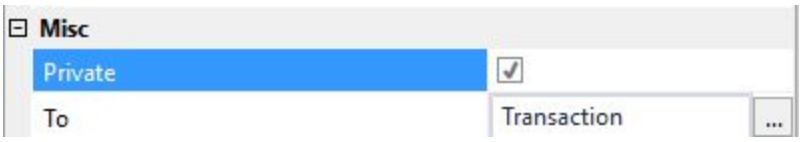
Logging Levels in Debug:

- Internal
- Verbose
- Trace
- Information
- Error

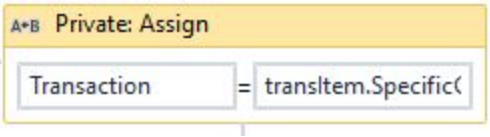
Note: Only for Studio Output and log file.

9.3. Logging Settings in UiPath Studio

Logging Per Activity



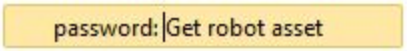
When “Private” is checked, the robot will not log variables or arguments to server. The activity will have in title: Private:



Reserved words in project.json

```
{
  "description": "Get and set transaction status",
  "version": "8.2.5905.32371",
  "main": "Main.xaml",
  "id": "GetAndSetTransFAST",
  "dependencies": {},
  "configurationOptions": {},
  "excludedData": [
    "Private:*",
    "*password*"
  ]
}
```

In the example above all the activities that have “password” in the name will not send logs to server.



9.4. Logging in Event Viewer

The Event viewer displays all the Warnings and Errors logged by the Windows service “UiPath Robot” from system.