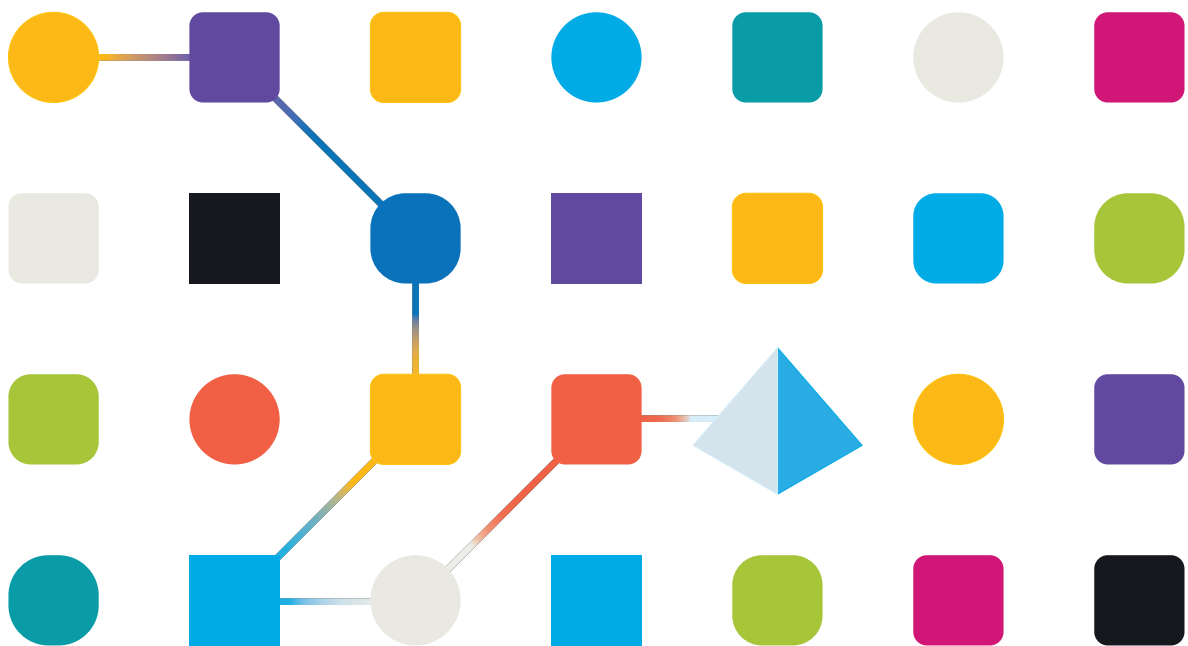# blueprism®

Blue Prism Cloud 4.1

User Guide

Document Revision: 1.0

Trademarks and copyrights

The information contained in this guide is the proprietary and confidential information of Blue Prism Cloud Limited and/or its affiliates and should not be disclosed to a third-party without the written consent of an authorized Blue Prism representative. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying without the written permission of Blue Prism Cloud Limited or its affiliates.

© 2021 Blue Prism Cloud Limited. "Blue Prism", the "Blue Prism" logo and Prism device are either trademarks or registered trademarks of Blue Prism Limited and its affiliates. All Rights Reserved.

All other trademarks are hereby acknowledged and are used to the benefit of their respective owners.

Blue Prism Cloud Limited and its affiliates are not responsible for the content of external websites referenced by this help system.

Blue Prism Cloud Limited, 2 Cinnamon Park, Crab Lane, Warrington, WA2 0XP, United Kingdom.

Registered in England: Reg. No. 8534024. Tel: +44 870 879 3000. Web: www.blueprism.com

Contents

IADA Orchestration – Evolution Edition

This user guide covers Blue Prism Cloud Intelligent Automated Digital Assistant (IADA). This specific guide is for Release 4.0 of the Blue Prism Cloud platform.

IADA allows automation developers to align business metrics to their varied workloads, driving priorities and service level agreements (SLAs) to determine the order that individual queue items should be executed in.

This guide outlines functionality and usage of IADA and how to apply orchestration across your workforce and into your automations. Whether this is existing automations or new automations.

Overview

When we build process automation's we tend to only consider the process design/build at hand. We are taught and are proficient in being able to create a process automation to run well. When testing we test this single process automation for any errors or inconsistencies in performance.

What we are less likely to consider, or certainly spend not enough time considering, is how our process automation ecosystem will work as a whole; as a combined unit. An organization may employ a controller or task one of the developers with acting as a controller. The controller's role is to check on the automations, setup schedules and ensure that work items are being processed based on SLAs, priorities and demand. This is an inefficient use of resource and adds additional expense. Would it not be better to autonomously make decisions on what process automations to run and when to run them?

Sometimes, the perception of implementing the concept of orchestration is that there is a plug-and-play solution available or a solution that looks up against a dictionary of defined process rules. Unfortunately, to create a successful and effective solution which will return the desired benefits, there is a level of bespoke customization and design consideration that needs to be undertaken.

Scheduler

The scheduler available in Blue Prism is great for small to medium volume implementations of process automations. However, after a period of time you may find that the scheduler does not fully fit the need of your automations entirely and you may notice some limitations:

- Can I add additional complex logic to trigger schedule runs?
- Can I chain schedules together to remove general down time traditionally seen at the end of a scheduled process run?
- How can I react to changes in priority and SLA across queue items, without manual intervention?
- How can I get the more out of my digital workforce?

These, in addition to many more, are questions that clients and partners have asked us. A continued implementation of the traditional scheduler by itself may lead to a level of inefficiency, which in turn may stop your ability to grow scale without having to add further to your previous investment e.g. purchase more digital worker licenses when there is still more efficiency to be gained from the exiting workforce.

That being said, in some very specific circumstances the scheduler should be used exclusively or in conjunction with IADA orchestration, some of these are detailed below.

When to use the scheduler

There are instances where the scheduler is still the best solution for a given process:

- When it is imperative that a process needs to be run at a specific point in time and nothing can interfere with this;
- If a digital worker needs to be in a specific state (special permissions, access, abilities etc.) to run a process but other digital workers cannot be given the same access. In this situation it would be best to dedicate sections of the day (if not all day depending on the volumes associated with the process) specifically to schedule time for this process. An example of this may be when different windows accounts are required to run specific automation processes; and
- When software licensing restricts the digital workers all having the same software build installed on them.

When looking at utilizing orchestration it is imperative that each process should be assessed individually and a decision made as to whether or not it is fit to be included within orchestration or to use the traditional Blue Prism scheduler. Ultimately the best way to think of orchestration is as another tool to be utilized in conjunction with the scheduler; not to replace it in all instances.

Orchestration

The way Blue Prism Cloud utilizes orchestration is by leveraging the IADA web service within a Blue Prism process called Orchestrator, the Orchestrator process will perform all the logic and intelligent decision making on which process to run; and when to run that process. The Orchestrator process will be running throughout the day, all day (aside from a small automation maintenance window), consuming all your digital workforce or as much as you want to allow it to utilize.

Implementing orchestration will allow you to use more advanced decision making, leveraging Blue Prism Cloud IADA web service and Orchestrator process will ultimately allow you effectively utilize your Blue Prism Cloud Digital Workforce in the best way possible.

What is orchestration?

As we explained, within an organization not all process automations and their associated items have the same value to the organization, some business processes need to be prioritized over others. By default, items will simply be addressed and processed in a First-In-First-Out (FIFO) methodology.

This combined with the fact that there can be multiple queues with the Blue Prism Cloud Digital Workforce that need to be managed, means that work can get stalled in handling an individual queue or stuck processing automations with work items which are of a lower value.

IADA maintains, in effect, its own view, looking across not a single queue but all the queues and assessing assigned values (such as business priorities and SLAs), it will determine the next automation item that needs to run, whilst queuing the next best value item ready for execution by the next available digital worker.

For an automation to be given a priority over another automation, it has to be configured with a business priority and an SLA. There are multiple ways of adding these parameters into your automations and we will show you how in more detail later. This is the key element so when items are added to the work queue, the business metrics are added along with the process.

For clarification, lets define some key terms:

Business priority

The business priority value must be above zero. The lower the value, the higher the business priority (e.g. a business priority value of 1, will be processed before an item with a business priority value of 2). This business priority value will override the SLA value.

IADA accepts a business priority value between 1 to 999.

Business SLA

The business SLA value is a timespan. This should be the amount of time that the work queue item should be completed and processed within. The SLA timespan value will then be used within the IADA algorithm to produce an SLA Datetime.

IADA accepts a business SLA in the format of HH:MM:SS (Hours, Minutes, Seconds) e.g. 043000 = 4 Hours, 30 Minutes, 0 Seconds. The minimum SLA is 1 second, the maximum SLA would be 99 Hours 59 Minutes and 59 Seconds.

Tag

The tag is a combination of the Priority, SLA and Process Name as per the example below, it will be recognized and accounted for by IADA.

```
Tags
001240000Testing
```

In the example the Priority is set to '001', the SLA is '24 hours, 0 minutes and 0 seconds' and the Process Name is 'Testing'.

Orchestration configuration queue

If a digital worker has started running a process automation, how does it notify the remainder of the digital workforce of this action, or how does it set a flag that the remainder of the workforce can check? As with any human workforce, people communicate and talk to each other, sometimes making each other aware of their actions. Your digital workforce needs to do the same.

Unfortunately, there isn't a concept of global variables that are read/write accessible within the Blue Prism RPA designer. There are a number of different solutions that can be used to create communication channels. For the purpose of this documentation we will use the concept of a separate specific Blue Prism queue containing only one item. The data required will be stored in the 'Item Data' of the queue item, though there are other options which could be considered, for example:

- SQL database table
- JSON file
- Custom .config file with parser
- Excel document
- XML
- CSV file

We have chosen to create and maintain a queue item (in the 'Orchestration Configuration Queue') that digital workers can read and write to. Now when we want to log an action that the digital workforce needs to be aware of, we can open the item, read back the contents, update the contents and write the updated contents back to the queue item. This is also known as the Orchestration 'States' file and you will see the term 'States' in the process templates.

We now have a mechanism of logging when we are processing an automation and when an automation has been completed.

In order to ensure that there are no conflicts when reading from or writing to the queue item we will use environment locks.
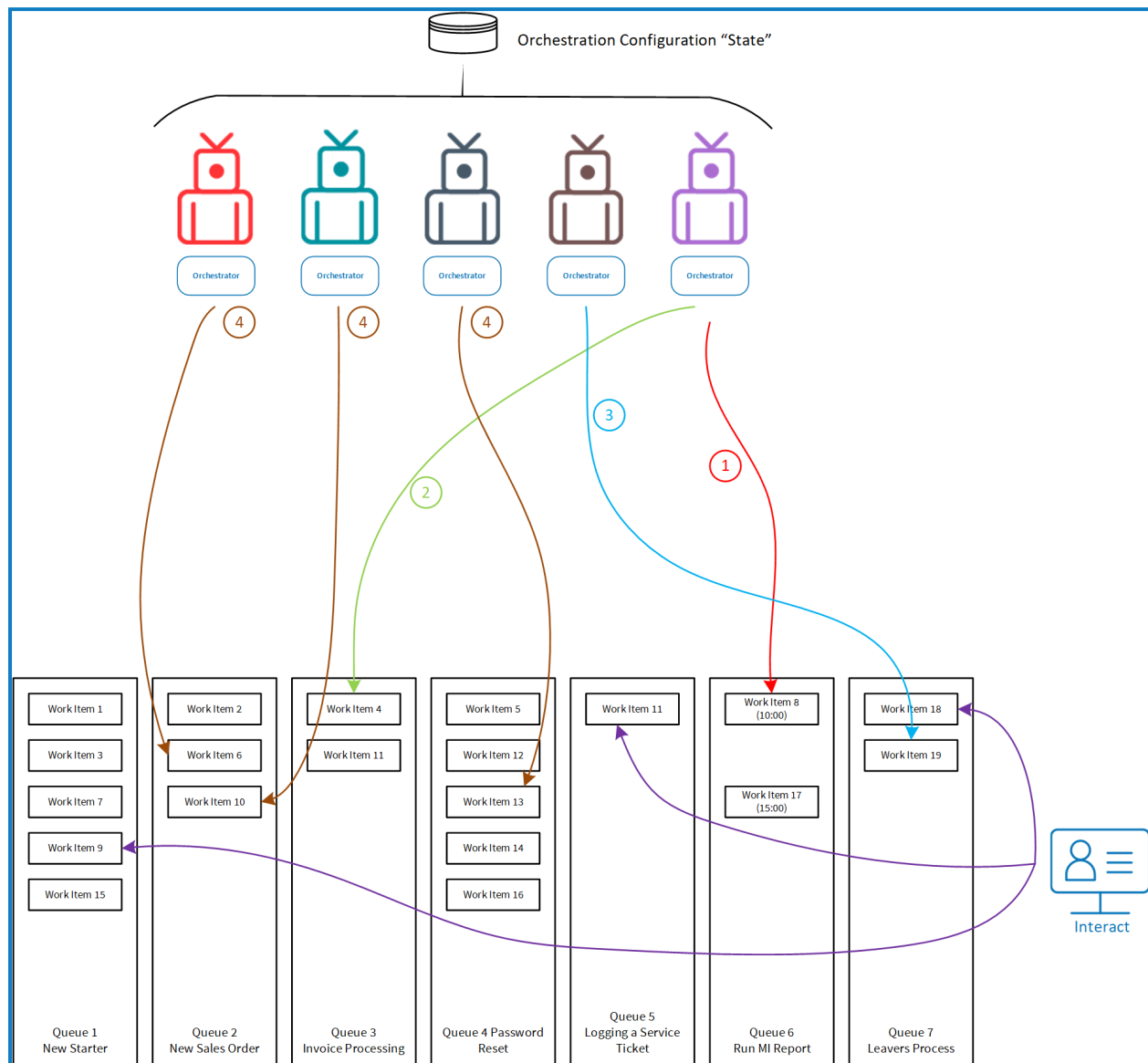
Environment locks

One of the final pieces of the puzzle to create to enable orchestration to occur is a locking and token release mechanism. We already have this in the form of 'Environment Locks'. It is recommended that you are familiar with environment locks and understand how they function. We use environment locks quite heavily within orchestration as this concept is key to adding a control mechanism around:

- Decision making for getting the 'Next Best Item' to be worked via IADA;
- Information gathering – reading from the Orchestration Configuration Queue item; and
- Writing to the Orchestration Configuration Queue item.

How does orchestration work?

Let us consider the following five digital worker platform as illustrated below.

On each digital worker the Orchestrator process would run. The Orchestrator process runs and operates the same way across the digital workforce environment.

The digital worker will first check to see if there is a scheduled process that needs to be actioned, this is determined by the datetime and when the schedule should occur, '1' in the diagram.

If the process is not scheduled to run it will then go and reassess across all the queues the 'Get Next Best Item' to process.

> If there was a scheduled task to perform it would initiate that process and complete that task before looking for new work.

In our example the highest business priority is 'work item 4' in the Invoice Processing queue, '2' in the diagram.

The first task the Orchestrator process does is write to the Orchestration Configuration Queue to notify the rest of the digital workforce, 'I am handling this item'.

The digital worker would have either already called its Collector process to retrieve any associated data items, if required, along with the associated process required to be run against the work item, the business priority and the SLA. This information is then used by the Executor process within the Orchestrator process to perform the required automation.

> The Collector would not be run if the item was added to the queue using Interact as the associated tag would contain all the necessary information.

This routine is then repeated with the other digital workers, checking for scheduled tasks, before identifying the 'Next Best Prioritized Item' for processing, in our example, '3', this is work item 19 in the Leavers Queue.

The other digital workers will source their own work, '4' in the diagram calling their Collectors, if required and Executors to perform the work task on the next business critical process identified.

This also includes items that can be added into the queues on an ad-hoc basis from users using the Interact interface.

It is recommended that the Orchestrator process is run across the whole of the digital workforce.

It is recommended that the digital workers are built identical with each digital worker capable of running every process.

> If there is no tag or orchestration status assigned to an item in a queue the Orchestrator process will not process the task and it will be ignored. The Orchestrator process looks across all queues, but only processing items with tags.

What are the benefits of orchestration?

To demonstrate and explain how an Orchestrator process could be designed and in turn show the benefits, we have provided an example use case below.

A company, 'Xample Mortgages', have asked Blue Prism Cloud to create nine processes comprised of four Executor processes (referred to as Executors as these will work/process items in a queue), four respective Collector processes (referred to as Collector processes as these are tasked with collecting the data and populating the respective queues to be worked by the Executors) and one monthly reporting process. These process automations are required to be ran as effectively as possible throughout the day. Their parameters and requirements are outlined below:

1. Redemption Payments Received – Collector:
   - **CONTEXT**: The data to be collected is only available following money received from Bank transfers on weekdays. All the data for the day will be available by 12:00:00. Only one data dump is received per day so once the Collector has been run successfully it will not need to be run again that day .
   - **SCHEDULE**: Once a day on Monday / Tuesday / Wednesday / Thursday / Friday.

2. Redemption Payments Received – Executor:
   - **CONTEXT**: Each payment received must be applied to the customer's account within 24 hours. Due to financial regulations this process has a short SLA and must be processed as a higher priority item. If items are not processed within the timeframe then a fine is applicable.
   - **SLA**: 24 Hours 00 Minutes.
   - **PRIORITY**: 010.

3. Mortgage Applications – Collector:
   - **CONTEXT**: The client is a larger mortgage provider and as such receives thousands of mortgage applications per day. These can and do arrive at any time of day and the automation must account for that. Xample Mortgages aims to respond to the customer application (by providing confirmation/rejection of an offer) within 2 days.
   - **SCHEDULE**: To be run every day (including weekends) every 30 minutes.

4. Mortgage Applications – Executor:
   - **CONTEXT**: The successful mortgage applications that arise from this process are the main source of income for Xample Mortgages. There are no fines associated with late application responses, but the aim is for an internal SLA of 2 days; therefore, it receives 2nd priority for automation.
   - **SLA**: 48 Hours 00 Minutes.
   - **PRIORITY**: 020

5. New Starters – Collector:
   - **CONTEXT**: The company has several large call centers that have relatively high rates of attrition. There are generally new starters joining the company every week.
   - **SCHEDULE**: This Collector should be run once in each of the following periods (every weekday): 10:00:00-11:00:00, 13:00:00-14:00:00, 16:00:00-17:00:00, 20:00:00-21:00:00.

6. New Starters – Executor:
   - **CONTEXT**: This is an internal process that aims to complete all onboarding activities for new starters to Xample Mortgages. In the past the activities required to onboard new starters (system access, payroll, HR notifications) was disjointed and often completed late. Sometimes employees were unable to begin work on the confirmed start date. This automation aims to speed up the process to ensure that the new joiner will have everything at their disposal once they start working for Xample Mortgages. The new joiner instruction is sent ahead of time and there is an SLA of 2 days associated with this process.
   - **SLA**: 48 Hours 00 Minutes.
   - **PRIORITY**: 030.

7. Leavers – Collector:
   - **CONTEXT**: As previously mentioned Xample Mortgages has a high rate of attrition and the leaver Collector must be handled in a timely manner (although not as high a priority as the New Starters). So as not to interfere with working week tasks Xample Mortgages would like the Collector for this process to only run on weekends ready for HR to pick up the post automation activities (if any).
   - **SCHEDULE**: This Collector will only run on Saturday and Sunday from 03:00:00 every 3 hours.

8. Leavers – Executor:
   - **CONTEXT**: The leaver related activities have a longer internal SLA and lower priority level.
   - **SLA**: 72 Hours 00 Minutes.
   - **PRIORITY**: 040.

9. Monthly Reporting Process:
   - **CONTEXT**: On the 1st of each month a summary report will be produced covering a selection of queues and providing Management information. This report must only be run once and should be run in respect of the preceding month.
   - **SCHEDULE**: Run on the 1st of each month (regardless of day of the week).

If you consider how this may be achieved using a scheduler, you will quickly identify potential issues and limitations. For example:

- Can you effectively perform all the necessary checks, and deploy the necessary management functionality within the scheduler?
- Can you confidently provide the business with 100% assurance that the digital workforce can meet and delivery on these requirements?
- What happens where one schedule processes over runs and a trigger for another key time dependent process is missed or delayed?
- Can a scheduler determine which items to process next based on business priorities and SLA?
- Would a controller have to take the following actions (only during their standard work hours):
  - Continually monitor queues?
  - Stop various processes across digital workers in order to free up resources?
  - Diverting said resource to a specific process?
  - Keep an eye on this process and then change all the updated digital workers back to their original schedules?
- Will items or processes be missed if a human must continually do this throughout the day across 100 digital workers?

The last question is rhetorical, but the above scenarios are a prime example of how orchestration can be implemented to manage and control all process automations to meet the necessary requirements and business metrics.

Designing for orchestration

When implementing orchestration, it is common to feel apprehensive when considering how to approach the design, and it is sometimes thought that complex cognitive systems and Artificial Intelligence (AI) are required to complete the design.

Although cognitive systems like Machine Learning (ML) and AI implementations are important, it should be a secondary consideration after implementing a Logical and Algorithmic design. These technologies need to overlay a base foundation of logical decision making.

Logical decision making

Once you break down the requirements, you can start to understand what checks the process might want to perform when deciding if and when to execute an automation. If we just take one process, the Redemption Payments Received – Collector process automation, the following considerations need to be understood:

- Has it already been run?
  - This is important and should be one of the first checks that we perform. If the process automation has already ran, then we don't and should not run this again for that day.
- Is it after 12pm?
  - Easy enough, we should check the current time and ensure that it is after 12pm. If it is not after 12pm (on the current day) then we should not run this process automation yet.
- Is today one of the respective days it can be worked on?
  - As previously noted, this Collector process should only be run on weekdays. This means that we need to check the current day and determine if it is a weekday.
- Is it 'running' already by another digital worker?
  - This automation will not be recognized as complete until after the automation has ended. This means that we need to check to see if the process is in progress.
- What is the current average execution time?
  - This has already been given in the requirements, but we should be careful. Is this an average execution time based on a human completing the process? When was this average execution time noted? Was this noted 6 months ago, and since we have upgraded the process to be more productive and quicker? We could check the latest figures by analyzing previous execution times.
- Can we delay?
  - Ultimately, we don't need to run this process at 12PM, or at least there isn't anything in the requirements above that would suggest we can't. The only requirement is that it must be completed at some point today after 12:00:00. The answer to this question may be determined by what we think the average execution time of the automation might be.

The same level of understanding and questioning should be performed for all the remaining processes in our example.
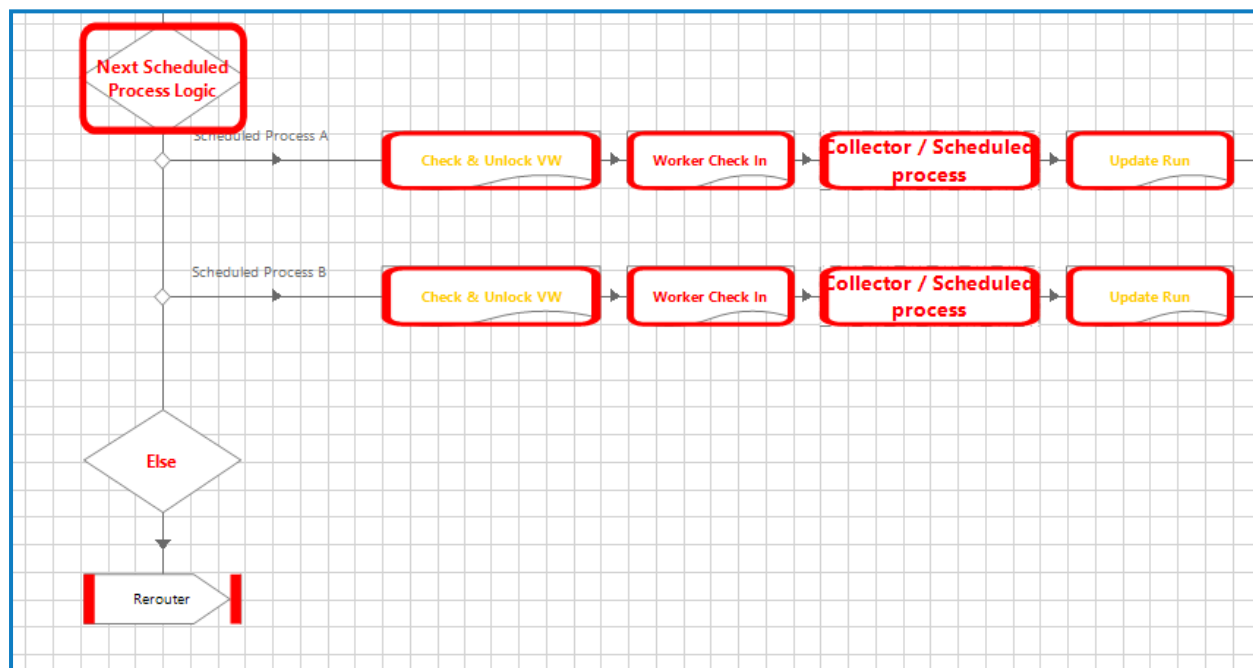
Orchestration design

There are two key pages within the orchestrator template:

- Get next best item
    - Using a combination of SLA and priority (as noted on the tag added at the point where the case was created) IADA will reference applicable queues and determine the next best prioritized item to be worked and initiate that process.
    - This page will utilize the IADA webservice action 'Find Next Prioritized Item'.
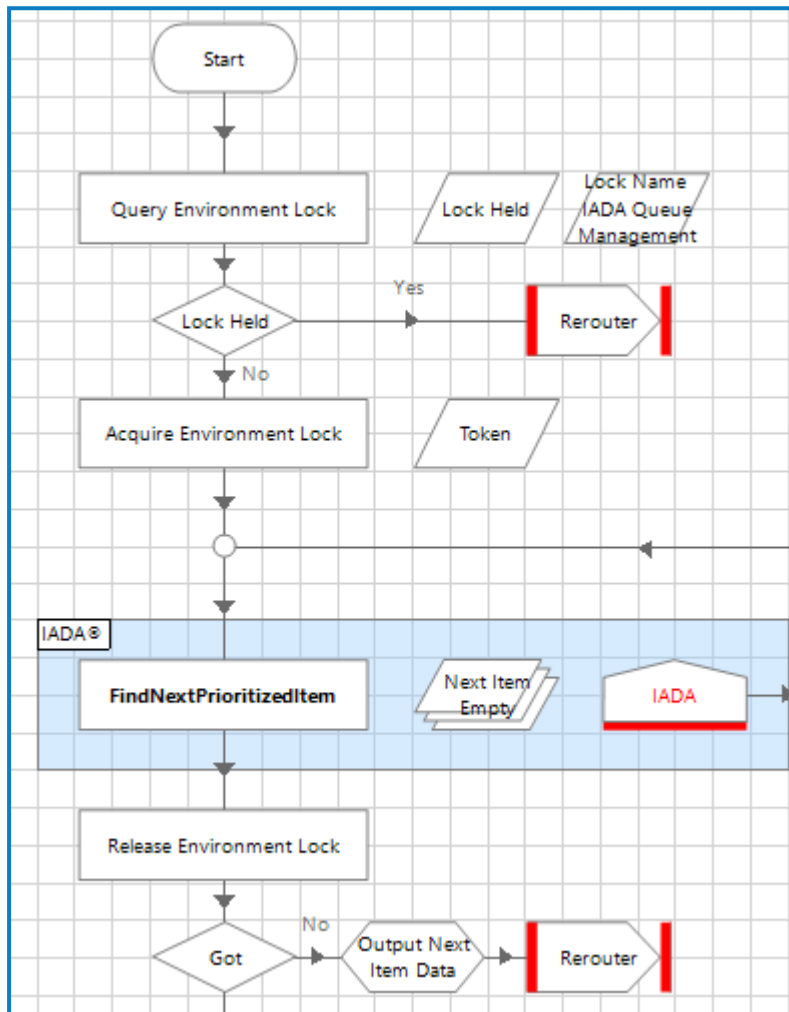
- Attempt Scheduled Processes
    - Any processes outside of the standard Executor processes will be handled by this page. This includes Collectors, report processes (etc.) that are due to run on a schedule. The schedule logic will be included here.
    - The key part of this page is the 'Next Scheduled Process Logic' choice stage. This is where the logic outlined for our example, Xample Mortgages, feeds in.
    - Each requirement must be broken down into its logical parts using a combination of logic expressions, pre-built logic objects and data related to the current state of the automation. This would be stored in the previously mentioned Orchestration Configuration Queue item and output as a collection in the orchestrator referred to collectively as 'States' and would include items such as:
        - Which weekdays should the automation run?
        - When was the automation last run?
        - Has it been run today?
        - Can it only be run once day?
        - Can it only be run during certain periods?

Pre-built orchestration logic object

As part of orchestration solution an accompanying object 'Component – Orchestration – Check States Criteria' is available. This component contains actions that can be utilized to output Boolean responses based on data input including:

- Run the process only on a set of defined days;
- Run the process after every 'X' minutes has passed;
- Run the process only within agreed defined timeframes;

As an example, we will walk through the 'Process on Defined Days' action.

Input(s):

- 'CSV Text'
  - **Type**: Text
  - **Context**: In the form of text input of comma separated list of weekdays i.e. Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday.

Output(s):

- 'Run Process'
  - **Type**: Flag
  - **Context**: Output the Boolean decision as to whether or not a process should currently be run.
- 'CSV Text'
  - **Type**: Text
  - **Context**: In the form of text input of comma separated list of weekdays i.e. Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday.

> With the day that was just worked now removed from the list.

If a process must only be run once a day on 3 specific days of the week (e.g. Monday, Wednesday, Friday) the input would be the list separated by commas (with no spaces). Once the action is run it will determine if today's day (e.g. Wednesday) is one of the specific days and output:

- 'Run Process': True
- 'CSV Text': Monday, Friday

You will notice that the item that was found (Wednesday) has now been removed. This will be written back to the Orchestration Configuration Queue item so that when the check is performed again today it will not be found again and therefore not be worked.

Expressions

Next, let us take a look at the expressions we can use to start our decision-making process. We will use each of the Xample Mortgages example processes.

Redemption Payments Received – Collector

Logic:

```
[Collector RPR - Run Today] = True
AND
LocalTime() > [States.Collector RPR - Start Time]
AND
[States.Collector RPR - Active Workers] = ''
```

Using the above expressions, we are checking that:

- The Collector could be run today ([Collector RPR - Run Today] = True)

  This is determined using the action 'Process on Defined days' from the Object 'Component – Orchestration – Check States Criteria'.

  The input is:

- If the current time is greater than the earliest time the Collector can be run (LocalTime() > [States.Collector RPR - Start Time])

  This is obtained from the field created in the States Collection called 'Collector RPR – Start Time'

- Whether or not there are currently any other active digital workers processing this piece of work ([States.Collector RPR - Active Workers])

  This is obtained by checking the text within the States Collection field 'Collector RPR - Active Workers] and confirming that it is currently blank

If all of the above conditions are met, then this schedule process would be due to run on the digital worker at this present moment in time.

Mortgage Applications – Collector

Logic:

```
[Collector Mortgage Applications - Run Now] = True
AND
[States.Collector Mortgage Applications - Active Workers] = ''
```

Using the above expressions, we are checking that:

- The Collector could be run now ([Collector Mortgage Applications - Run Now] = True)

  This is determined using the action 'Process Every X Minutes' from the Object 'Component – Orchestration – Check States Criteria'. Checking if the current time is greater than the start time. Once run it will add 30 minutes to the current start time to produce a new start time.

The inputs are:

- Whether or not there are currently any other active workers processing this piece of work ([States.Collector Mortgage Applications - Active Workers] = ")

  This is obtained by checking the text within the States Collection field 'Collector Mortgage Applications - Active Workers' and confirming that it is currently blank

If all of the above conditions are met, then this schedule process would be due to run on the digital worker at this present moment in time.

New Starters – Collector

Logic:

```
[Collector New Starters - Run Now] = True
AND
[States.Collector New Starters - Active Workers] = ''
```

Using the above expressions, we are checking that:

- The Collector could be run now ([Collector New Starters - Run Now] = True)

  This is determined using the action 'Process Within Timeframes' from the Object 'Component – Orchestration – Check States Criteria'. Checking if the current time is within defined timeframes. Once run it will remove the existing timeframe from the configuration queue item.

The input is:

Collector New Starters - Time periods (Text)

10:00:00-11:00:00,13:00:00-14:00:00,16:00:00-17:00:00,20:00:00-21:00:00

- Whether or not there are currently any other active workers processing this piece of work ([States. Collector New Starters - Active Workers] = ")

  This is obtained by checking the text within the States Collection field 'Collector Mortgage Applications - Active Workers' and confirming that it is currently blank

If all of the above conditions are met, then this schedule process would be due to run on the digital worker at this present moment in time.

Leavers – Collector

Logic:

```
[Collector Leaver - Run Today] = True
AND
[Collector Leaver - Run Now] = True
AND
[States.Collector Leavers - Active Workers] = ''
```

Using the above expressions, we are checking that:

- The Collector could be run today ([Collector Leaver - Run Today] = True)

  This is determined using the action 'Process on Defined days' from the Object 'Component – Orchestration – Check States Criteria'.

The input is:

- The Collector could be run now ([Collector Leaver - Run Now] = True)

  This is determined using the action 'Process Every X Hours' from the Object 'Component – Orchestration – Check States Criteria'.

The inputs are:

- Whether or not there are currently any other active workers processing this piece of work ([States. Collector Leavers - Active Workers])

  This is obtained by checking the text within the States Collection field 'Collector Leavers - Active Workers' and confirming that it is currently blank

If all of the above conditions are met, then this schedule process would be due to run on the digital worker at this present moment in time.

Monthly Reporting Process

Logic:

```
Left(Today(), 2) = [States.Monthly Reporting - Run X Of The Month]
AND
[States.Monthly Reporting - Run Today] = False
AND
[States.Monthly Reporting - Active Workers] = ''
```

Using the above expressions, we are checking that:

- The Monthly reporting process could be run today (Left(Today(), 2) = [States.Monthly Reporting - Run X Of The Month])

  The value can be found in the States collection.

  Monthly Reporting - Run X Of The Month (Text)

  01

- Whether or not the monthly reporting process has already been run today ([States.Monthly Reporting - Run Today] = False).

  The value can be found in the States collection.

  Monthly Reporting - Run Today (Flag)

  False

- Whether or not there are currently any other active workers processing this piece of work ([States. Monthly Reporting - Active Workers] = ")

  This is obtained by checking the text within the States Collection field 'Monthly Reporting - Active Workers' and confirming that it is currently blank

If all of the above conditions are met, then this schedule process would be due to run on the digital worker at this present moment in time.

Getting and setting values

You will notice that we are using a lot of data items (variables) in our expressions and we haven't chosen to 'hard-code' any values, other than the expression itself. This is very important. It provides us with some benefits that, certainly in the medium to long-term, will be applicable. Some examples of these benefits are below:

- If the business changes any rules on when they want this process to be ran, we can easily change this;

- If we as developers or designers decide that we need to change some of the rules and configurations, again, we can easily do so;

- Most importantly, if we algorithmically or intelligently recognize that these values need to be changed, we are able to programmatically do so.

We've already spoken about using the Orchestration Configuration Queue item to hold these configuration values in addition to some shared updates that digital workers can read and write to. Let's include some stages in our design to help build this out.
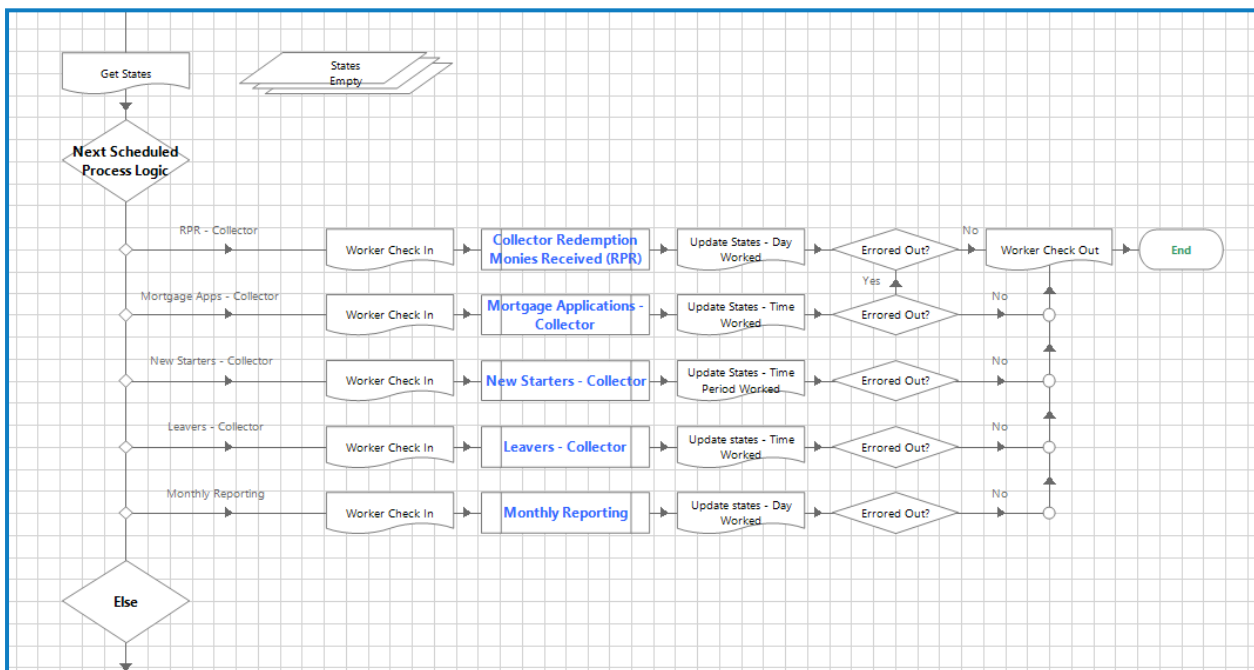
Now the design is taking form. Don't be alarmed at the amount of stages that we have added. Let's walk through this change.

> Another accompanying object 'Component – Orchestration Configuration' has been created to handle the reading/writing to and from the configuration queue item. This object is referenced in the stages below. We will touch on this object in more detail later.

- **Get States** – The digital worker will call the 'Orchestrator – Read From Configuration' action and parse the data appropriately.

- **Worker Check In** – Once a digital worker has determined that it is going to attempt a specific schedule process it will 'check itself in' to the respective field on the Orchestration Configuration Queue item; to prevent others from working this item. This means that subsequent digital workers will recognize the respective process is currently being executed.
 - Old State: "
  - New State: 'PVW001'

- **Update States - Day Worked** – The updated variable will now have the current day removed. This will be written back to the Orchestration Queue Configuration Item using the page 'Update State' which in turn uses the action 'Orchestrator – Write To Configuration'. The next time that this variable is referenced it will not run today. If the current day = Wednesday this is how it would appear:
  - Old State: 'Monday, Wednesday, Friday'
  - New State: 'Monday, Friday'
- **Update States - Time Worked** – Hopefully by this point, you are following and understand what this stage will do. It will create an update in the Orchestration Queue Configuration Item that will log the next time that process is due to run i.e. Previous time = 14:30:00 + 30 minutes = 15:00:00.
  - Old State: '14:30:00'
  - New State: '15:00:00'
- **Update States - Time Period Worked**– It will create an update in the Orchestration Queue Configuration Item that will remove the applicable time period that the digital worker just worked in i.e. if the digital worker just processed the schedule process during 13:00:00-14:00:00.
  - Old State: '13:00:00-14:00:00, 16:00:00-17:00:00, 20:00:00-21:00:00'
  - New State: '16:00:00-17:00:00, 20:00:00-21:00:00'
- **Update states - Time Worked** – It will create an update in the Orchestration Queue Configuration Item that will log the next time that process is due to run i.e. Previous time = 06:00:00 + 3 hours = 09:00:00.
  - Old State: '06:00:00'
  - New State: '09:00:00'
- **Update states - Day Worked** – Here is where we will set the flag that recognizes that the Monthly Reporting process automation has completed.
  - Old State: 'False'
  - New State: 'True'

Configuration file

As referenced earlier another accompanying object 'Component – Orchestration Configuration' has been created to handle the reading from and writing to the Orchestration Configuration Queue item. This object also contains locks within it to handle the possibility of conflicts between multiple digital workers attempting to access the queue item at the same time. This object has three key pages called by the Orchestrator process. We will describe each of these actions in detail below:
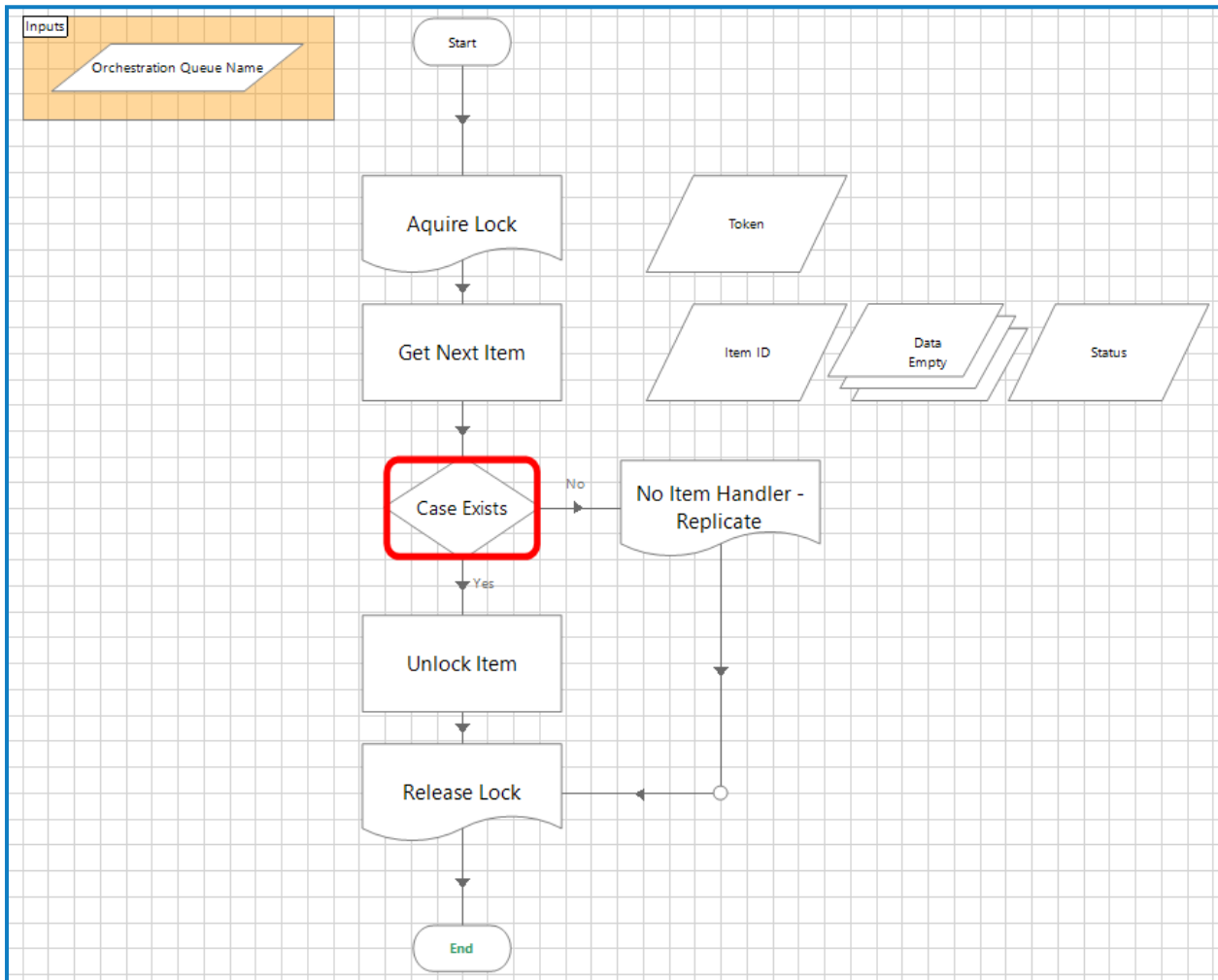
User – Create Or Update Configuration:

- Acquire the lock (to avoid conflicts of other processes trying to obtain the one item in the Orchestration Configuration Queue).

- Get the one queue item.

- Set the item data equal to the item data that was input.

- Update and then release the queue item – it does not complete, defer or exception the item.

- The lock is then released. In the event that no queue item exists (either because it has yet to be created for the first time or an exception caused it to not be pending) a new queue item is created with the new Item Data.
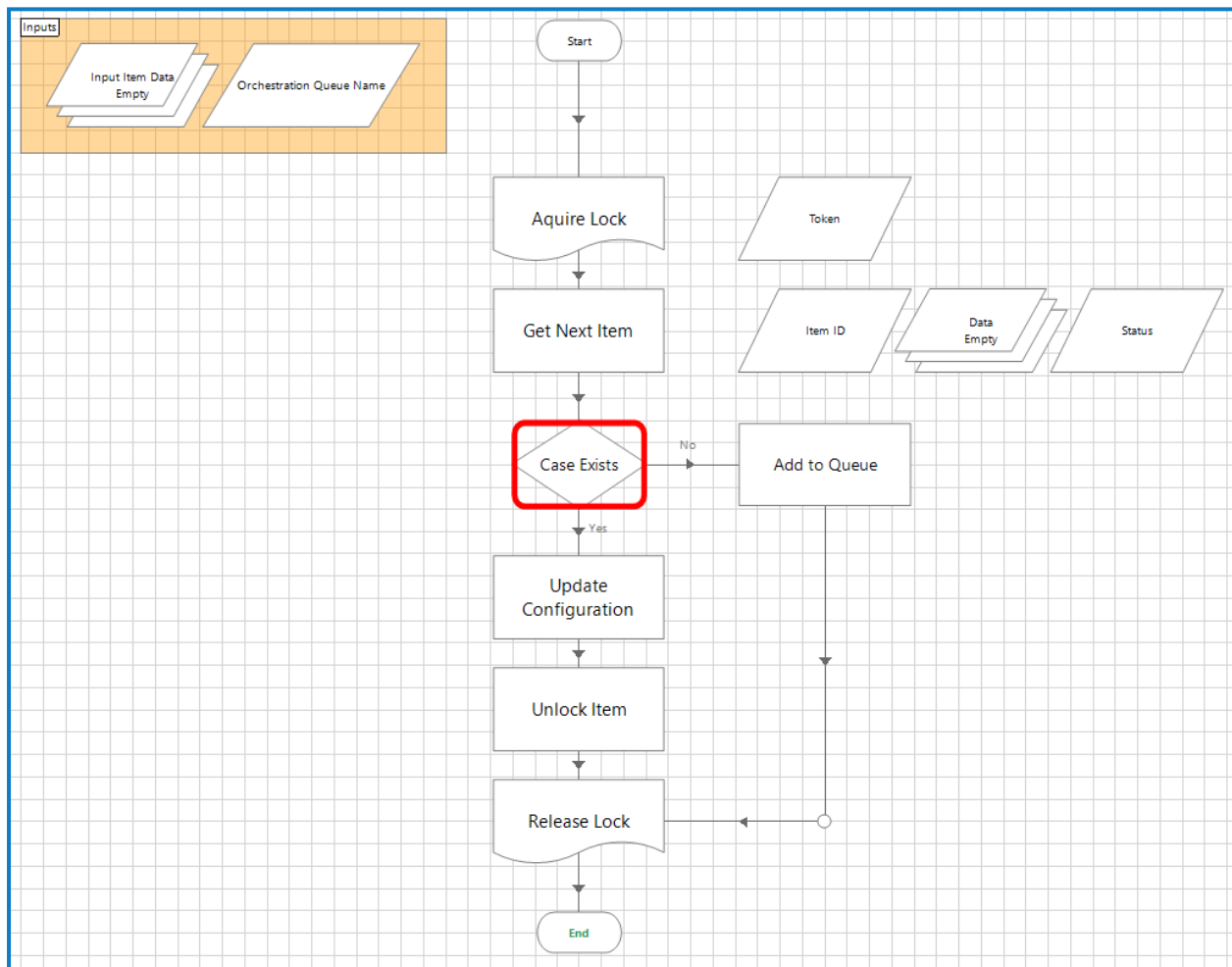
Orchestrator – Read From Configuration

- Acquire the lock (to avoid conflicts of other processes trying to obtain the 1 item in the Orchestration Configuration Queue).

- Get the one queue item.

- Read from and output the queue item data.

- Then release the queue item – It does not complete, defer or exception the item.

- The lock is then released. In the event that no queue item exists (because an exception caused it to not be pending); the most recent exceptioned case is obtained, the item data copied and a new queue item is created with the most recent item data.

Orchestrator – Write To Configuration

- Acquire the lock (to avoid conflicts of other processes trying to obtain the 1 item in the Orchestration Configuration Queue).

- Get the one queue item.

- Read from and output the queue item data.

- Then release the queue item – It does not complete, defer or exception the item.

- The lock is then released. In the event that no queue item exists (because an exception caused it to not be pending); the most recent exceptioned case is obtained, the item data copied and a new queue item is created with the most recent item data.
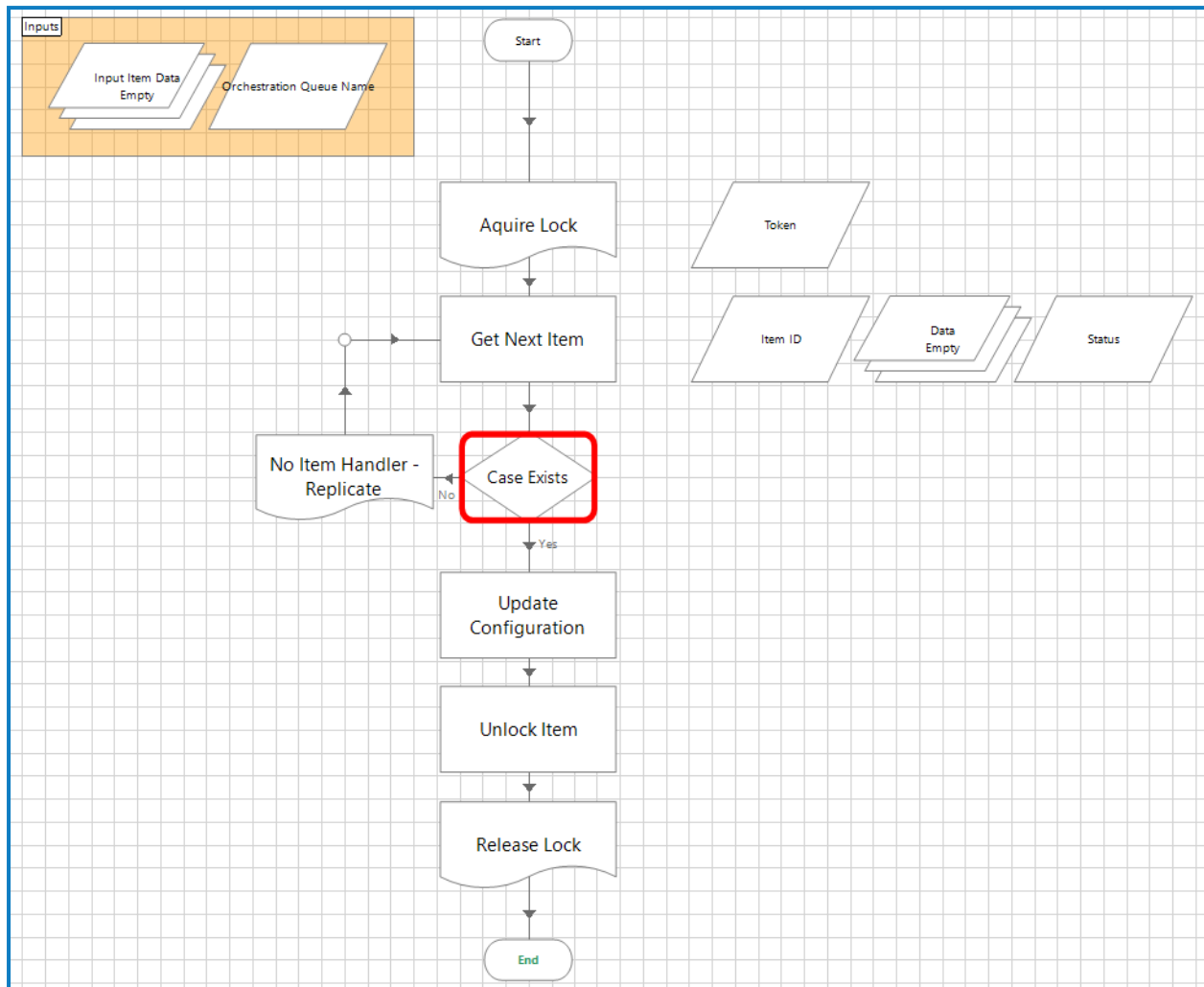
Moving to orchestration

The benefits of orchestration have now been explained and you may already have in mind some existing automated processes that you want to integrate into orchestration.

Considerations

As mentioned previously if a queue item has the correctly formatted tag, i.e. Priority, SLA and Process Name it will be recognized and accounted for by IADA.

Any queue items that do not follow this format will be excluded from the IADA 'Get Next Best Item' check each time it is performed. Therefore, for items belonging to a process, to be integrated into orchestration, they must be added with the correct tag. Items are added to queues in one of two ways:

- Added via Interact, submitting a request into a Blue Prism queue, will add the correct IADA formatted tag; or
- Using a Collector process to collect data and add this to the queue.

Blue Prism Cloud has built a Collector template designed to be integrated with orchestration. There is also an accompanying Executor template which we will cover later.

Collector template

The Collector process forms a framework into which the components of the Collector process can be added.
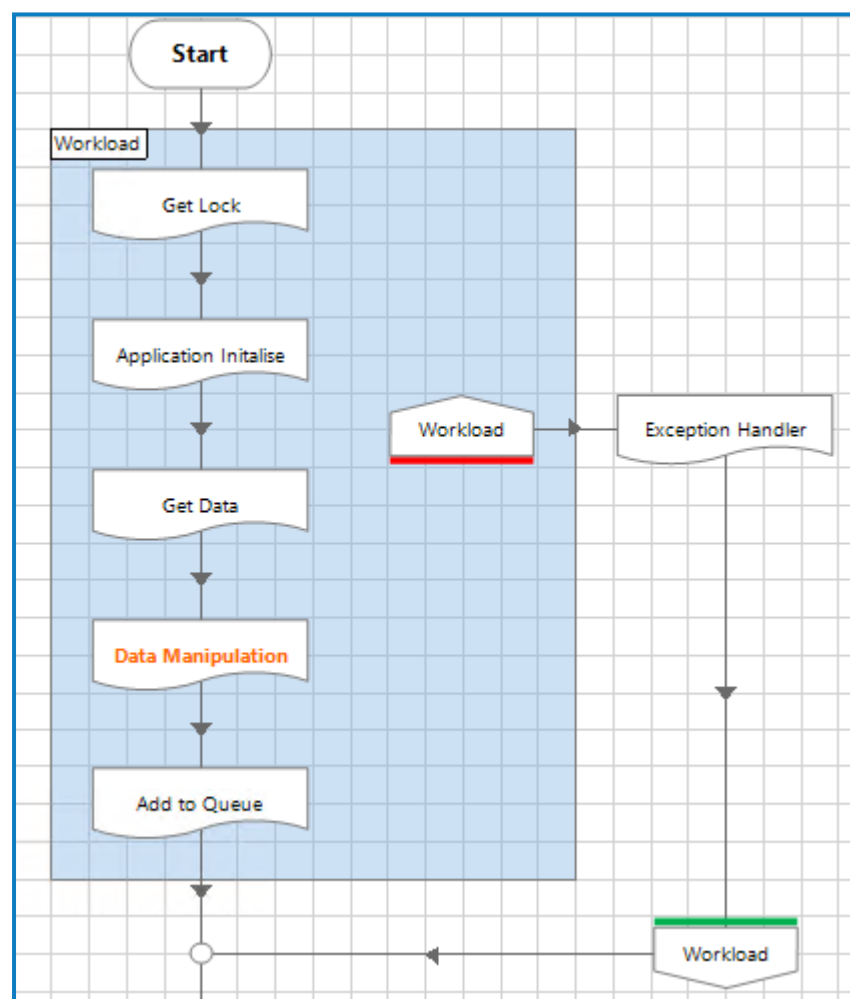
- **Page: Application Initialise**
  - The Collector process will open any initial applications it requires to complete its work.
- **Page: Get Data**
  - Get Data will use the application(s) to extract the raw data required to determine what cases will be added to the queue.
  - The data isn't yet ready to be added to the queue in its raw form.
  - This is also where we typically enter our steps/super components to carry out our application interactions/logic in line with what is defined in the Process Definition Document.
  - The goal of this page is to obtain a collection of raw data ready to be added to a queue (if no Data Manipulation is required).
- **Page: Data Manipulation**
  - Any steps required to format the data will occur here.
  - This page is optional and can be removed from the flow if not required.
- **Page: Add to Queue**
  - The data is ready and will be added to the queue with the input Priority, SLA and Process Name.

In order to integrate an existing process used to load data to a queue, the process needs to be separated into these key pages.

Orchestration – integrating the Executor processes

We have described how we add items to the queue using the Collector template. We now need a process integrated into orchestration that will work these queue items, this is the Executor.

Before we get to the Executor we need to see where these processes will be housed in the Orchestrator process itself, this is 'Page: Process Control'.
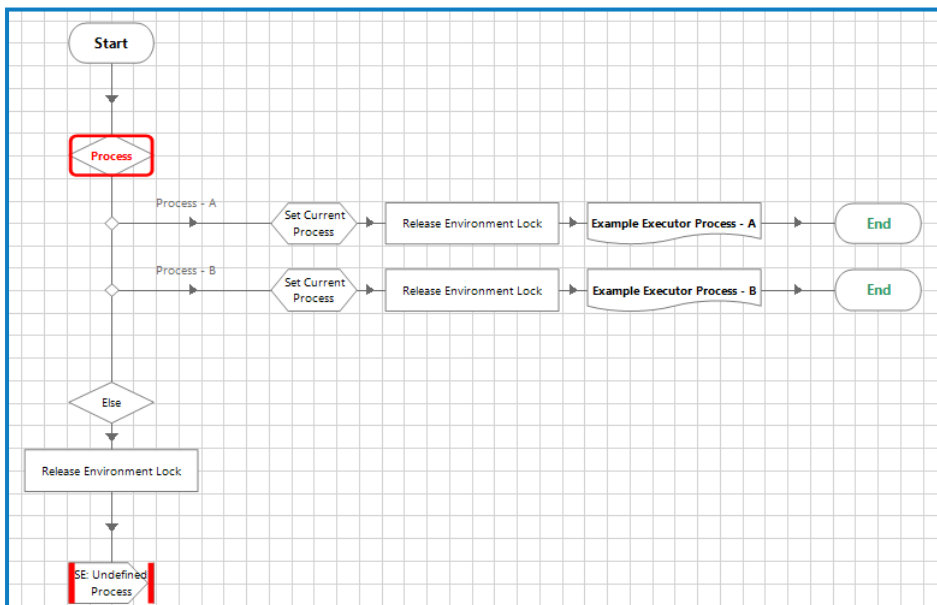
The process name, derived from the 'Get Next Best Item' page, is fed into the multi choice stage. Each Executor process will require a path on this Choice stage as well as an accompanying Process Page.

The Executor Process will replace 'Process A' on the diagram below.
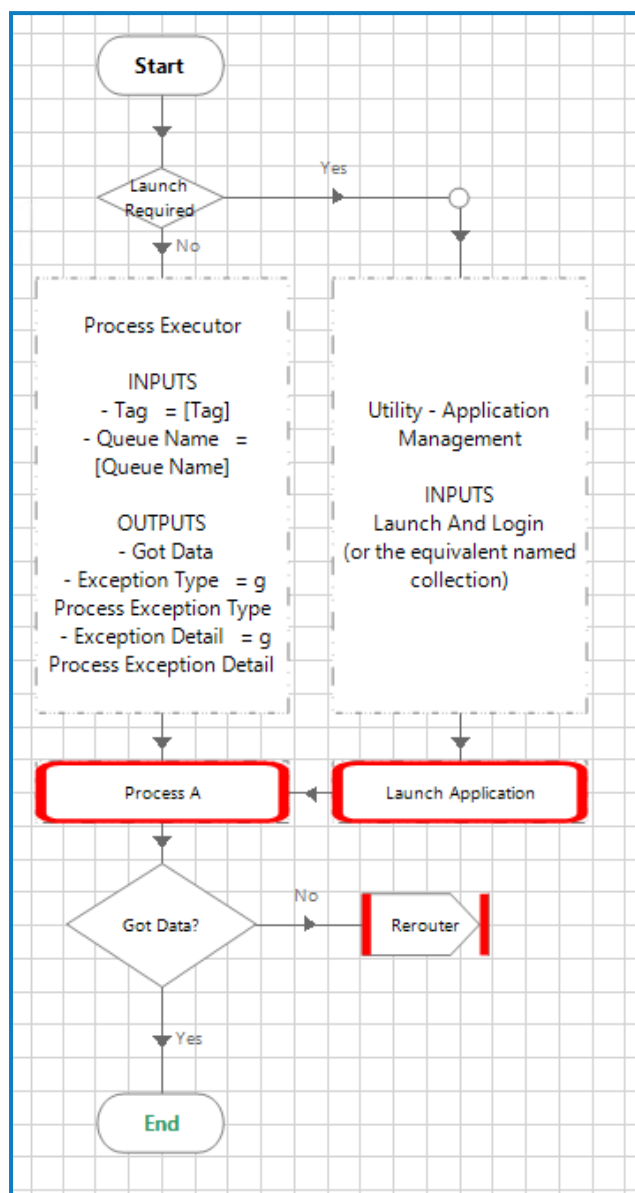
The list of required inputs and outputs for orchestration have been outlined in the text narrative stage above.

The Component required to start up the application(s) used by the process should also be included here and replace 'Launch Application'.

Executor template

The Executor template is a framework into which the workflow steps required for each individual case's work should be added. The main page of the Executor contains the 'Get Next Item', which uses the IADA tag to be able to pick up the correct case. Then the Workflow page into which the key steps will be added.
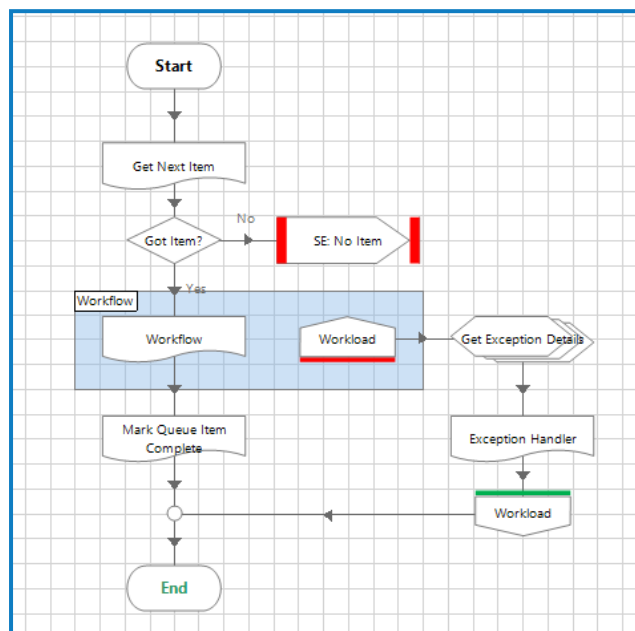
The template page contains 3 default steps that should be replaced with all of the steps required for an individual case to be worked by the Executor.

Once the steps have been completed successfully the Executor will mark the case as completed and exit.

The Orchestrator will then continue its loop and proceed to determine if it needs to run a schedule process or pick up another 'Get Next Best Item' and so the loop continues.

> The Executor is used in conjunction with the Orchestrator so that application management, the opening and closing of initial applications, is handled at the Orchestrator level.
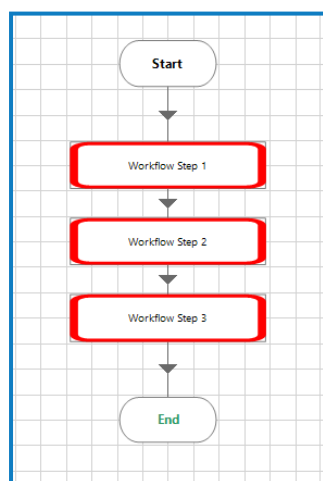
Moving to an orchestration process – example

Using the above descriptions and concepts we will update existing automated processes for "Xample Mortgages" to make them orchestration ready:

- Pre-Payroll Checks – Collector
- Pre-Payroll Checks – Executor

As we mentioned, the process has already been automated and is running in production. The process falls under a different department (Finance) and Xample Mortgage's CTO wanted to bring all their processes together using the orchestration model.

Considerations – Pre-Payroll Checks process

The following items need to be considered:

- The "Pre-Payroll Checks" requires an individual case per company employee;
- The cases should be created and worked on the 25th of the month;
- The confirmed Priority = 025 and SLA = 12 Hours:
  - The priority sits between the priorities of two of the existing processes:
    - Mortgage Applications (Priority: 020); and
    - New Starters (Priority: 030).

Steps – Pre-Payroll Checks process

Following the design and build of the Collector and Executor the first step to begin the integration would be to update the Collection ('inputs') within the 'Process - Orchestration - Configure States Queue'.
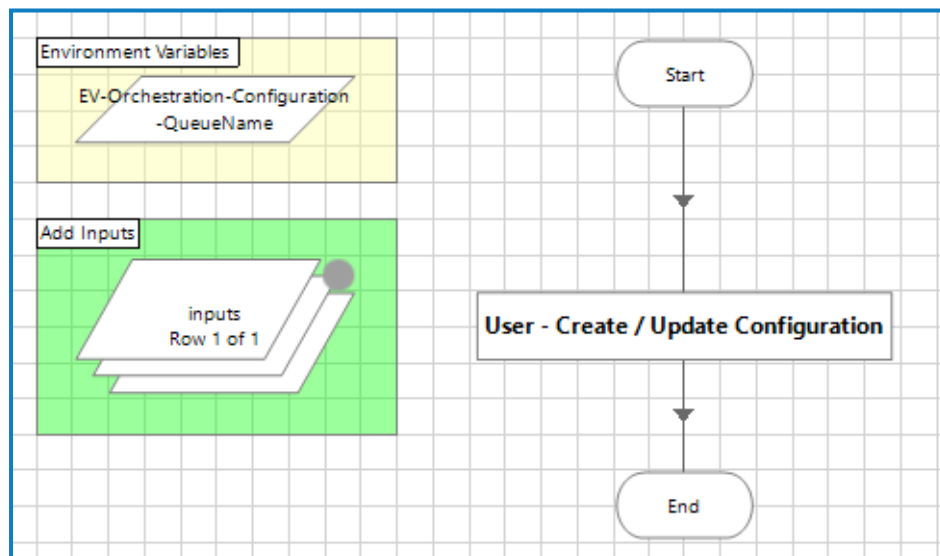
We have identified that we need 3 new fields to capture:

- The active digital workers that are currently working the Pre-Payroll Checks Process
  - Default population = Blank
- The day of the month in which the Pre-Payroll Checks Process should run
  - Default population = 25th of every month
- Whether or not the process has already been run today
  - Default population = False

The new fields have been added to the collection:

During the automation maintenance window the Pre-Payroll Checks process will be run to reset the 'states' ready for the next day. Therefore, the day before the Pre-Payroll Checks process is due to be implemented into production, these fields should be added to the 'inputs' collection in the 'Process - Orchestration - Configure States Queue' and this version of the process should be moved into production.
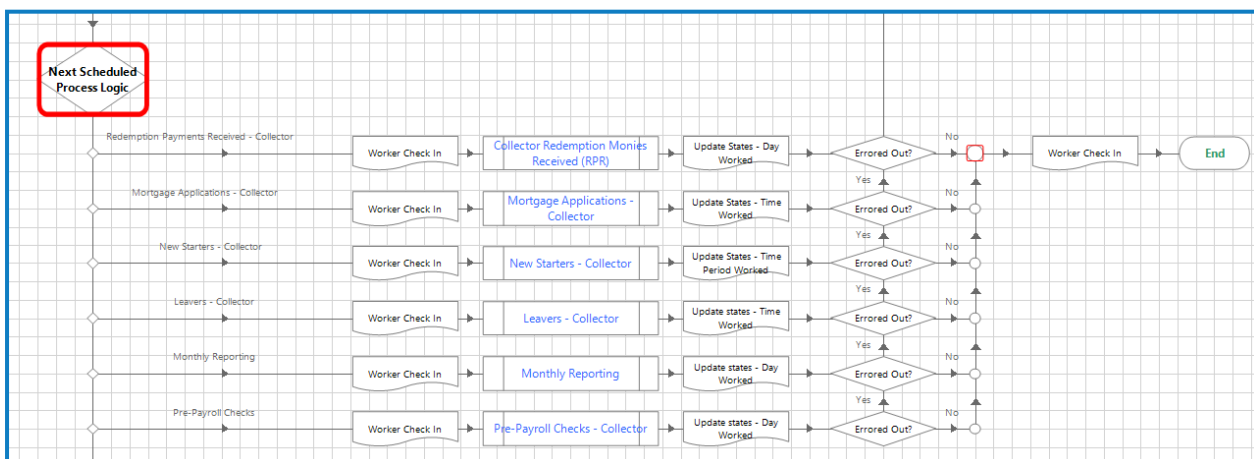
We can now update the 'Attempt Scheduled Processes' page to reference these items in the 'states' collection by updating the 'Next Scheduled Process Logic' to include a new path for Pre-Payroll Checks:

The choice criterion has been updated with the following logic to ensure that the collector is only run once, and by one worker, on the 25th of each month:

Once we have the new path, we can add in the Collector process:

We will look at each of the new additions on this path:

- Digital worker Check In:
  - Inputs

Name	Data Type	Value
Name	Text	"Monthly Reporting - Active Workers"
State	Text	[States.Monthly Reporting - Active Workers] & [g Machine Name]& ";"
States	Collection	[States]

  - Outputs – N/A
- The Collector Process itself – Pre-Payroll Checks – Collector
  - Inputs – N/A
  - Outputs

Name	Data Type	Store In
Errored out	Flag	✓ Errored Out

> This data item flag is an example – depending on the process you may wish there to be additional outputs with more information.

- Update states- Day Worked
  - Inputs

Name	Data Type	Value
Name	Text	"Collector - PrePayroll Checks - Ran Today"
State	Text	"True"
States	Collection	[States]

  - Outputs – N/A
- Errored Out Decision
  - Decision – referring to the flag which confirms whether or not the process encountered an error that forced it to stop whilst running.
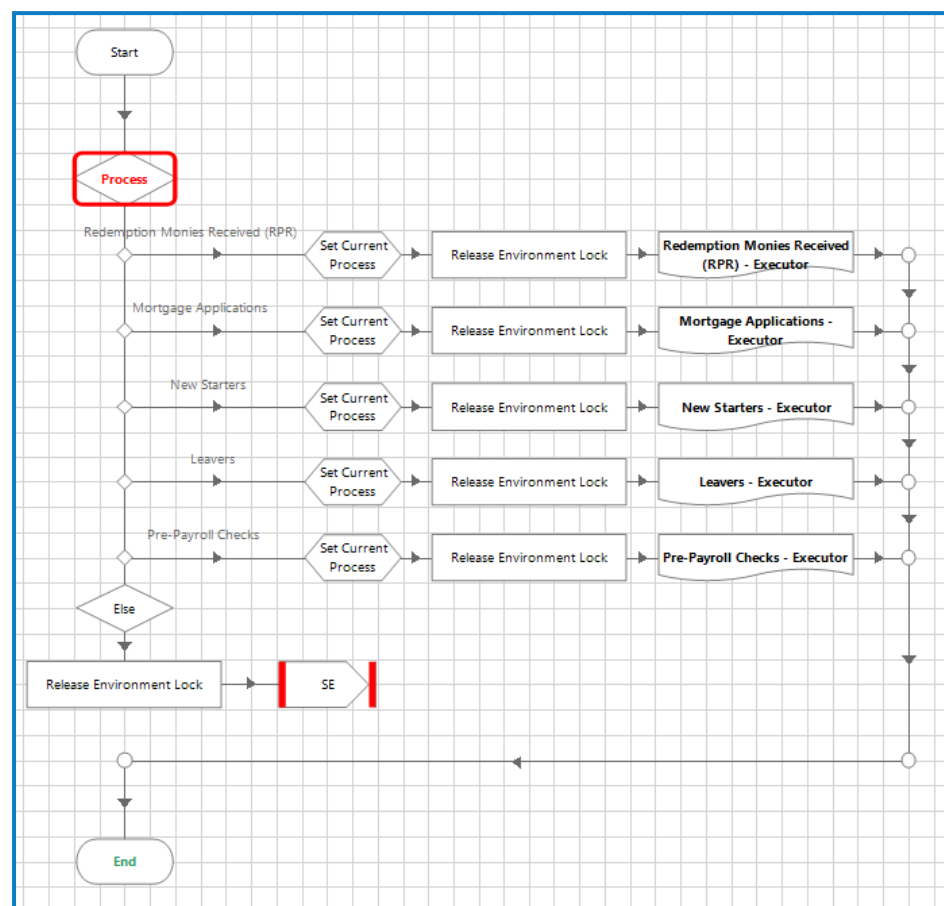
Decision Properties

Name: Errored Out?

Description:

Expression

[Errored Out] = True

The Pre-Payroll Checks Collector process is now integrated. The next step would be to integrate the Executor to process the items produced by the Collector.

The orchestration process is updated to include the new 'Process' choice stage for the 'Pre-Payroll Checks'.

Choice Name	Choice Criterion
Redemption Monies Received (RPR)	InStr([Process Name], "Redemption Monies Received (RPR)") > 0
Mortgage Applications	InStr([Process Name], "Mortgage Applications") > 0
New Starters	InStr([Process Name], "New Starters") > 0
Leavers	InStr([Process Name], "Leavers") > 0
Pre-Payroll Checks	InStr([Process Name], "Pre-Payroll Checks") > 0

The path refers to a page called 'Pre-Payroll Checks – Executor'. There is a template page on the Orchestrator called 'Process Template'. We copied this page and amended it to include the new Executor.

The inputs for this page are as follows:

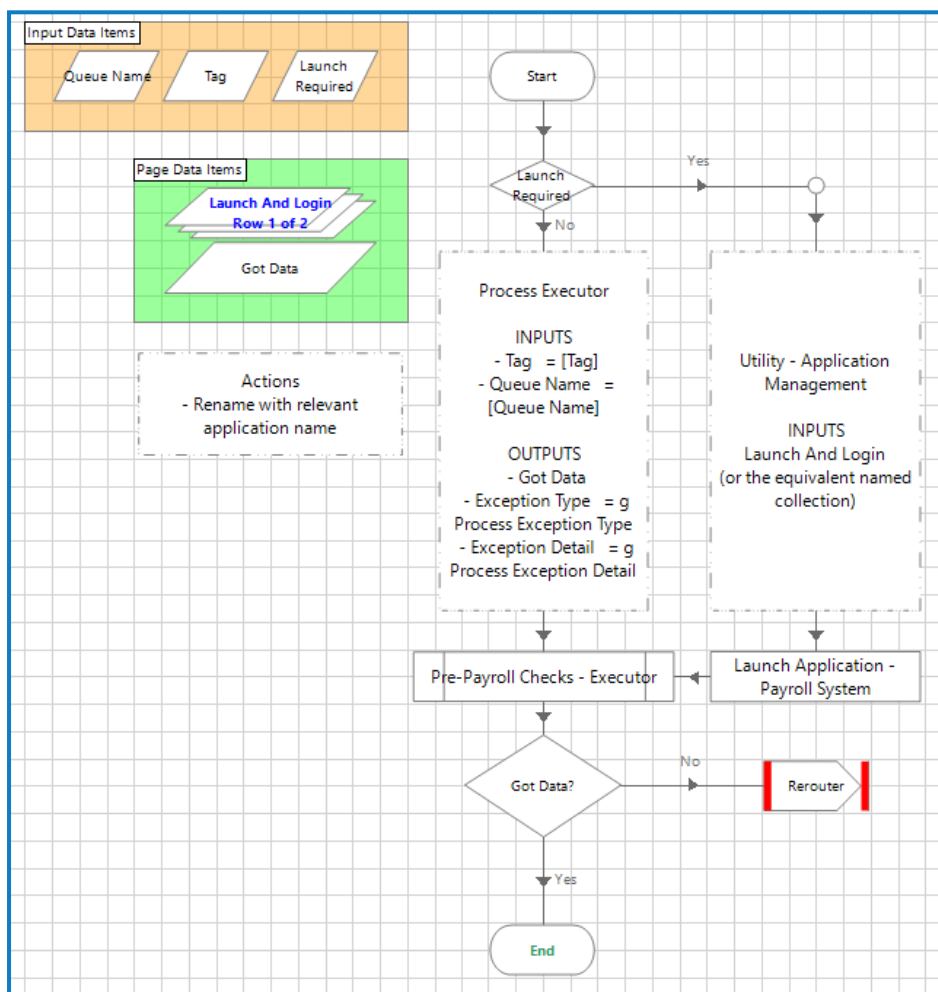These will never change as they are fed from the IADA Get next Best item call.

The Process Executor page is illustrated below.

Pre-Payroll Process Checks:

- Inputs (these will not change)

- Outputs (these will not change)

The final step on the page is to include the BPC 'Utility - Application Management' and the associated Controller Component setup for each of the applications required for this process. In this case the Payroll system used is Xayroll.

The 'Controller Component - Xayroll' will have already been setup as part of the application and process build. The Controller Component is designed to be fed a list of actions that it will carry out, in sequence, to reach the desired outcome for the application.
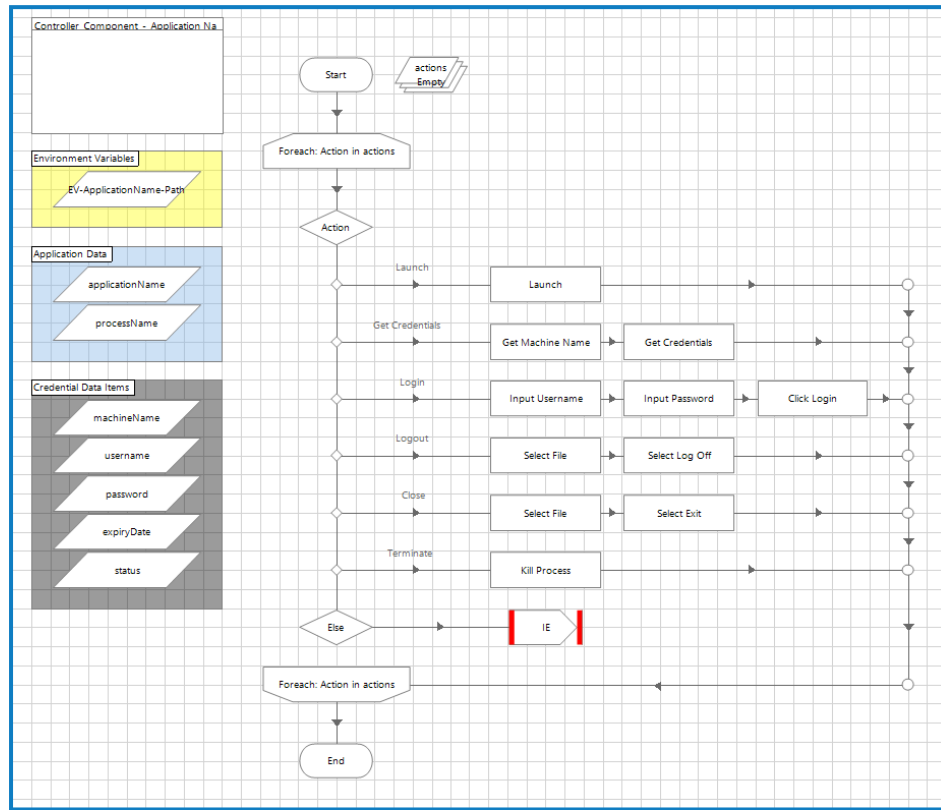
The actions required to open the application are fed into the component by way of the 'Actions' collection. In this instance the actions to feed in would read:

The loop within the component will then take each 'Action' in turn; thereby opening Xayroll, obtaining the respective credentials for the virtual machine and logging into Xayroll with those credentials.

The new 'Pre-Payroll Checks' process has now been fully integrated into Xample Mortgages Orchestration process and will be ready to go for the next working day.