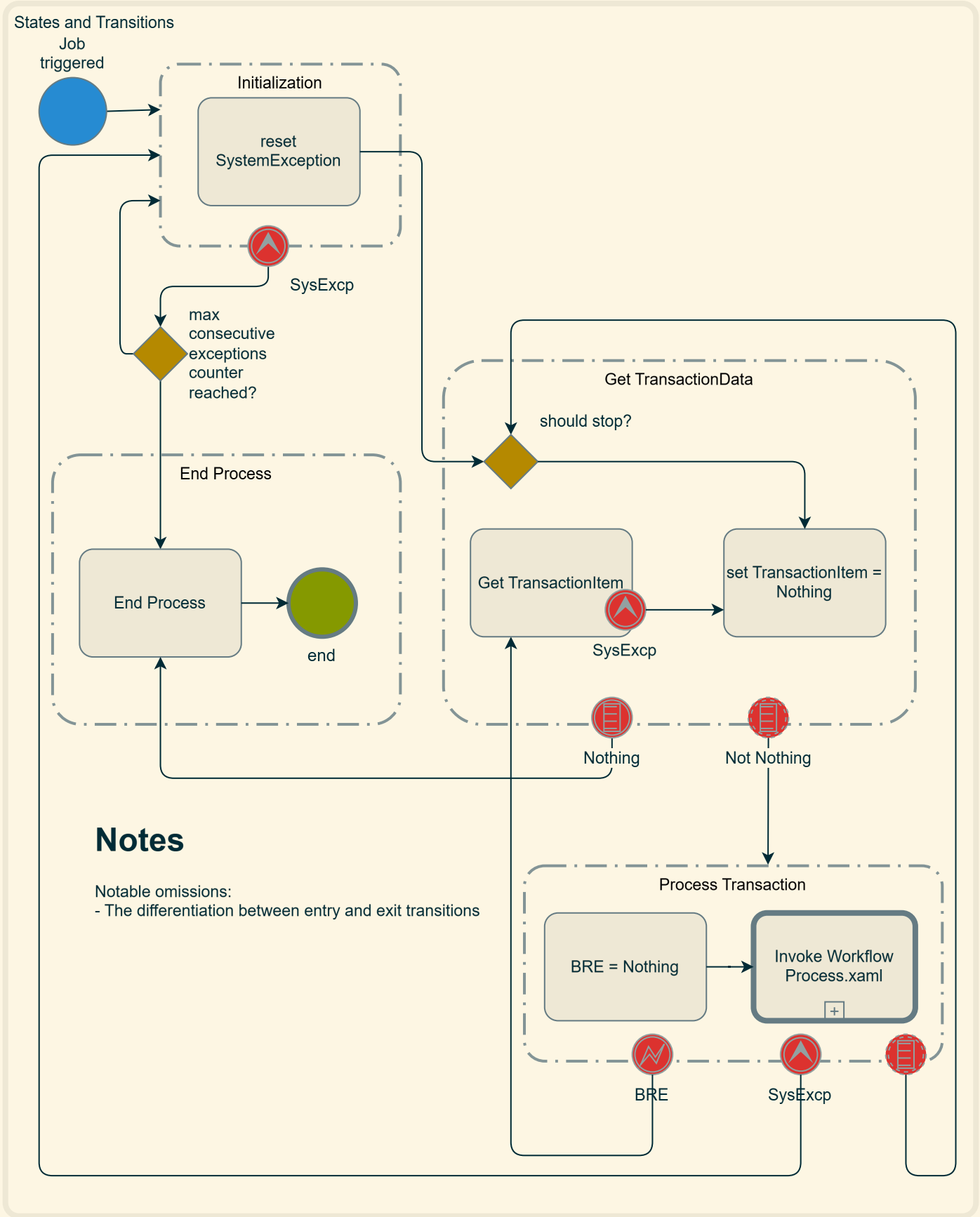# RPA UiPath Core 301

UiPath REFramework demystified
States and Transitions
Retry Behaviour
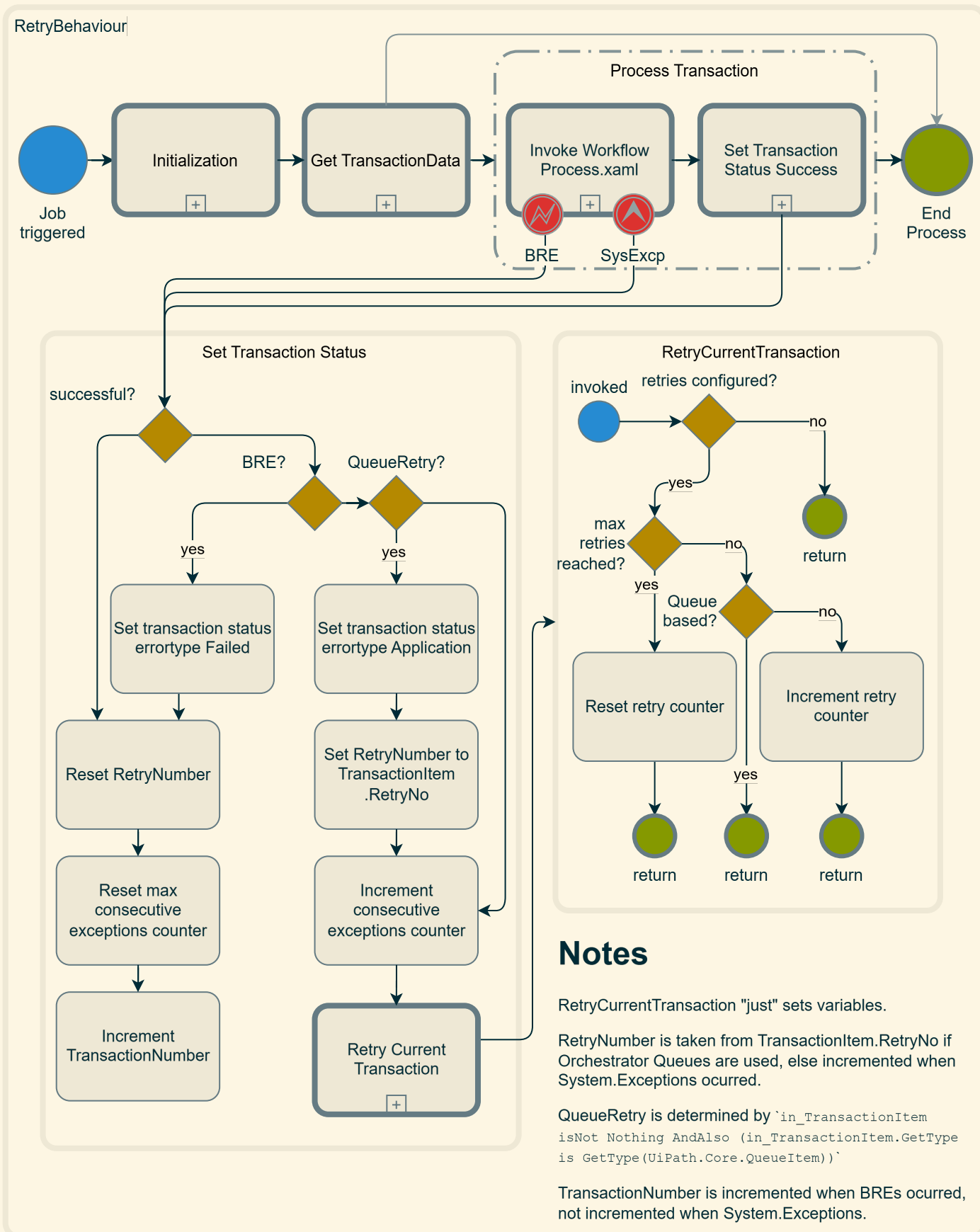Adaptions
InProcessQueue

RPA UiPath REFramework

The vendor-provided REFramework project template is feature-rich and inherently complex. While UiPath Studio's graphical editor offers a flowchart-like representation of the program structure, creating simplified, targeted diagrams can significantly enhance understanding by focusing on specific aspects of the workflow.
The transitions between the states are based on TransactionItem being Nothing or not, and variables holding a UiPath.Core.BusinessRuleException or System.Exception.
The file Main.xaml contains the state machine and is also defined as the main entrypoint. It implements the Performer functionality of the Dispatcher-Performer model for scaling automations.
Other entrypoints could be added and triggered from UiPath Orchestrator, e.g. to implement the Dispatcher.

States and Transitions

Job triggered

Initialization
reset SystemException
SysExcp

max consecutive exceptions counter reached?

End Process
End Process
end

Get TransactionData
should stop?
Get TransactionItem
SysExcp
set TransactionItem = Nothing
Nothing
Not Nothing

Process Transaction
BRE = Nothing
Invoke Workflow Process.xaml
+
BRE
SysExcp

Notes

Notable omissions:
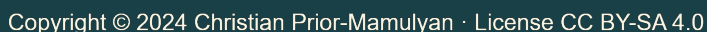- The differentiation between entry and exit transitions

RPA UiPath REFramework

The vendor-provided REFramework project template is feature-rich and inherently complex. While UiPath Studio's graphical editor offers a flowchart-like representation of the program structure, creating simplified, targeted diagrams can significantly enhance understanding by focusing on specific aspects of the workflow. The retry behavior is based on config items from the Config.xlsx, variables and exceptions. Retries can happen at runtime within the project code or as a result of the setup of a project-external UiPath Orchestrator Queue. The major distinction is that UiPath.Core.BusinessRuleExceptions are not retried, all other exceptions conditionally as depicted in the diagram.

RetryBehaviour

Job triggered → Initialization → Get TransactionData → Process Transaction [Invoke Workflow Process.xaml (BRE, SysExcp) → Set Transaction Status Success] → End Process

Set Transaction Status

successful? (diamond)
- BRE? → yes → Set transaction status errortype Failed → Reset RetryNumber → Reset max consecutive exceptions counter → Increment TransactionNumber
- QueueRetry? → yes → Set transaction status errortype Application → Set RetryNumber to TransactionItem.RetryNo → Increment consecutive exceptions counter → Retry Current Transaction

RetryCurrentTransaction

invoked → retries configured?
- no → return
- yes → max retries reached?
  - yes → Reset retry counter → return
  - no → Queue based?
    - yes → return
    - no → Increment retry counter → return

Notes

RetryCurrentTransaction "just" sets variables.

RetryNumber is taken from TransactionItem.RetryNo if Orchestrator Queues are used, else incremented when System.Exceptions ocurred.

QueueRetry is determined by `in_TransactionItem isNot Nothing AndAlso (in_TransactionItem.GetType is GetType(UiPath.Core.QueueItem))`

TransactionNumber is incremented when BREs ocurred, not incremented when System.Exceptions.

RPA UiPath REFramework

The vendor-provided REFramework project template is feature-rich and inherently complex. While UiPath Studio's graphical editor offers a flowchart-like representation of the program structure, creating simplified, targeted diagrams can significantly enhance understanding by focusing on specific aspects of the workflow. The overall control flow of the REFramework has remained stable since its inception, but its documentation has evolved considerably. Recent updates provide detailed guidance on adapting the REFramework to handle rows in an Excel sheet or MailMessage objects instead of the default QueueItem. These modifications, while powerful, necessitate careful updates across many parts of the framework, as changes to the data type propagate through various interconnected components, including activities, variables, and arguments.

Main.xaml of the standard REFramework / big picture

End Process

CloseAllApplications → KillAllProcesses

End Process

Job triggered

Initialization

InitAllSettings

InitApplications

SysExcp

Get Transaction Data

Get transaction data / item
[+]

Set TransactionItem = Nothing

SysExcp

Process Transaction

Invoke Workflow Process.xaml
[+]

Set Transaction Status

invoked successful?

no → BRE?

no → QueueRetry?

no

yes

... [+]   ... [+]   ... [+]   ... [+]

return

SysExcp

Success

BRE

Notes

Cyan-colored border lines indicate necessary changes to migrate away from Orchestrator Queues.

When in Main.xaml, GetTransactionData.xaml and SetTransactionStatus.xaml the datatype of TransactionItem is changed, then Workflow Analyzer reports for the project 24 errors.

Notable omissions:
- how the transaction item with the standard datatype UiPath.Core.QueueItem is passed from Main.xaml
- the differentiation between entry and exit transitions

to be continued

Corrections? Ideas?
Feel free to repost,
to start a discussion with your colleagues