Human Assisted Resolution Process

Studio Template

Revision History....4
Release Notes:....5
Overview....6
Key Features....6
Generic Human In The Loop Flow....7
Solution Architecture....9
Human Assisted Resolution, Part of End-to-End Business Processes....9
Studio Project Overview....10
Settings for Unattended Processes....10
Example Implementation....10
Project Files....11
Data\Config.xlsx....11
Data\Files\HART.pdf....11
Data\Templates\BusinessRuleException_ProcessName.json....11
Data\Templates\SystemException_ProcessName.json....11
Data\Templates\Exception_AssignedUsersProcess.json....11
Data\Templates\CategorizeDocument_ProcessName.json....11
Data\HttpRequest\Asset.json....11
Data\HttpRequest\StorageBucket.json....12
Data\HttpRequest\TaskCatalog.json....12
Main.xaml....12
InitializeOrchestrator.xaml....12
Framework\00_ReadConfigFile.xaml....12
Framework\10_InitializeProcess.xaml....13
Framework\20_GetTransactionItem.xaml....13
Framework\30_SelectTemplate.xaml....13

Framework\40_DataPostProcessing.xaml....13
Framework\50_EndProcess.xaml....13
Framework\ERR_AbortProcess.xaml....13
Framework\ReusableWorkflows\GenerateId.xaml....13
Framework\ReusableWorkflows\GenerateKey.xaml....14
Framework\ReusableWorkflows\OrchestratorRequest_GET.xaml....14
Framework\ReusableWorkflows\OrchestratorRequest_POST.xaml....14
Framework\ReusableWorkflows\SetTransactionStatus.xaml....14
Quick Start Guide....15
Orchestrator Configuration....15
Unattended Automation....15

Revision History

Date	Changes
Jul-2023	2023.7

Release Notes:

2023.7:

Initial features:

- Production-ready: Built-in logging, exception handling and retry-mechanisms.
- Follows RPA, Orchestration Processes and Long-running workflows best practices.
- Automatic generation of assets, storage buckets and task catalogs.
- New feature of the ‘Create Form Task’ activity: directly loading the form JSON file at runtime.

Overview

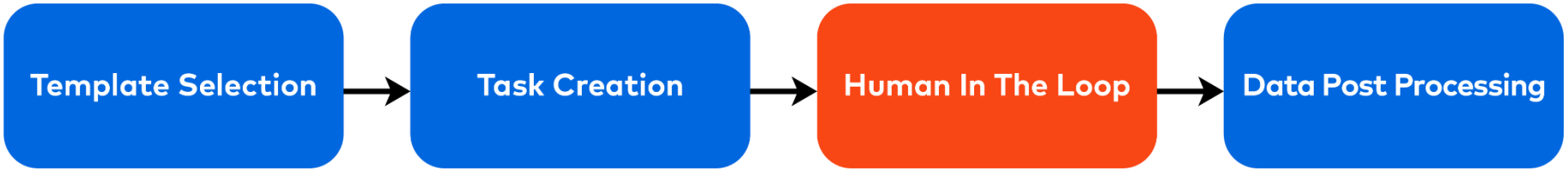
When a workflow step stops running, regardless of the reason, it should be recovered from its error condition. Some projects may have as requirement the manual error handling or any human intervention, therefore, Action Center tasks, corresponding to each type of error, need to be created. To have an optimized and scalable version of this, it would be more suitable to have a separate error handling/human intervention framework.

Key Features

- Everything can be generated automatically on the back end (including assets, storage buckets and task catalogs)
- Simple process, usable in all cases, from small processes to complex solutions
- Easy to integrate into larger automation flows.
- Production-ready: has built-in logging, exception handling and retry mechanisms.
- Follows the best practices pertaining to RPA, Document Understanding, Orchestration Processes, and Long-running workflows.
- Based on the Document Understanding Framework - Meant to make development, deployment and debugging much easier.
- HART processes will not run as batch jobs. Instead, an individual job should be started for each file to be processed. For this reason, it is much easier to search in the Orchestrator for an individual job or to perform debugging.

Generic Human In The Loop Flow

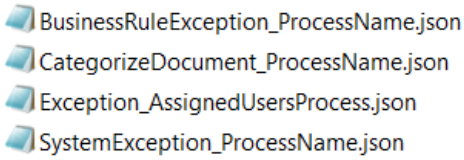
Processing flow for each case



Keep in mind that the diagram above shows that the most detailed logical flow is split into the smallest possible modules. In practice, it is to be expected that some parts could be merged or might be completely excluded, as they are not required in a particular implementation.

Template Selection

All templates are saved in the Data\Templates folder. The selection is based on the exception/task type and process name.



Task Creation

Creation of the Action Center Task and waiting for it to be resumed. With the latest form task feature, the forms JSON file can be directly loaded at run time.

Human In the Loop

The human operator completes or rejects the created task.

Data Post Processing

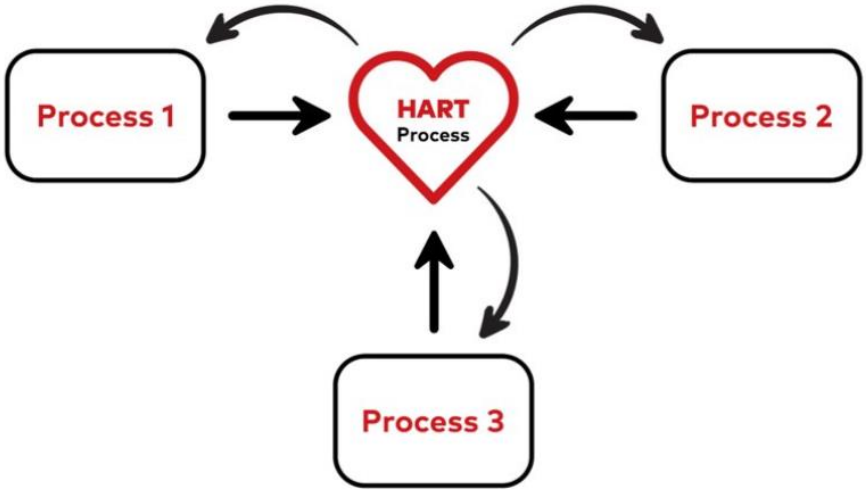
Processing the Action Center input of the human operator, in case there is any. (E.g. Performing an Orchestrator HTTP request in order to get the assigned task user)

Solution Architecture

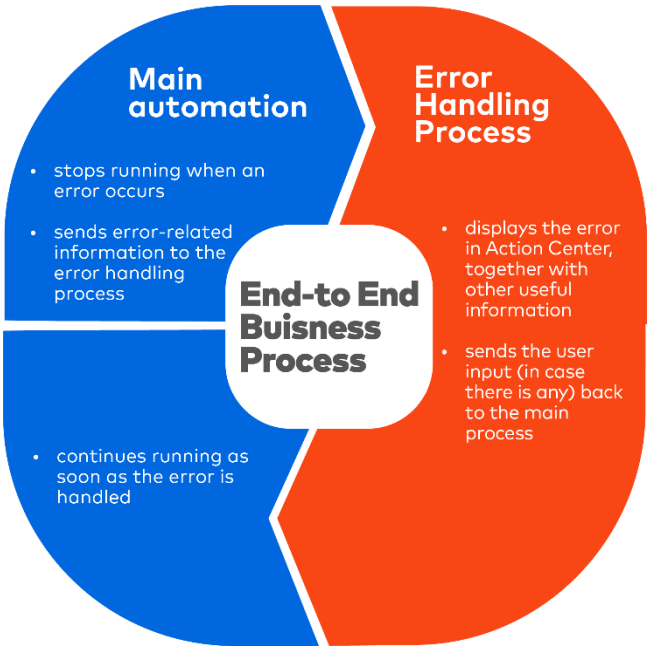
Human Assisted Resolution, Part of End-to-End Business Processes

The HART process is part of bigger business processes to be automated.

A single process, HART, handles the creation of Action Center tasks for every process in that specific use case. After the completion of each task, the processes will continue from where they left off, with the corrected data. Like this, both time and cost will be saved.



The architecture for an end-to-end Business Process involving HART consists of:



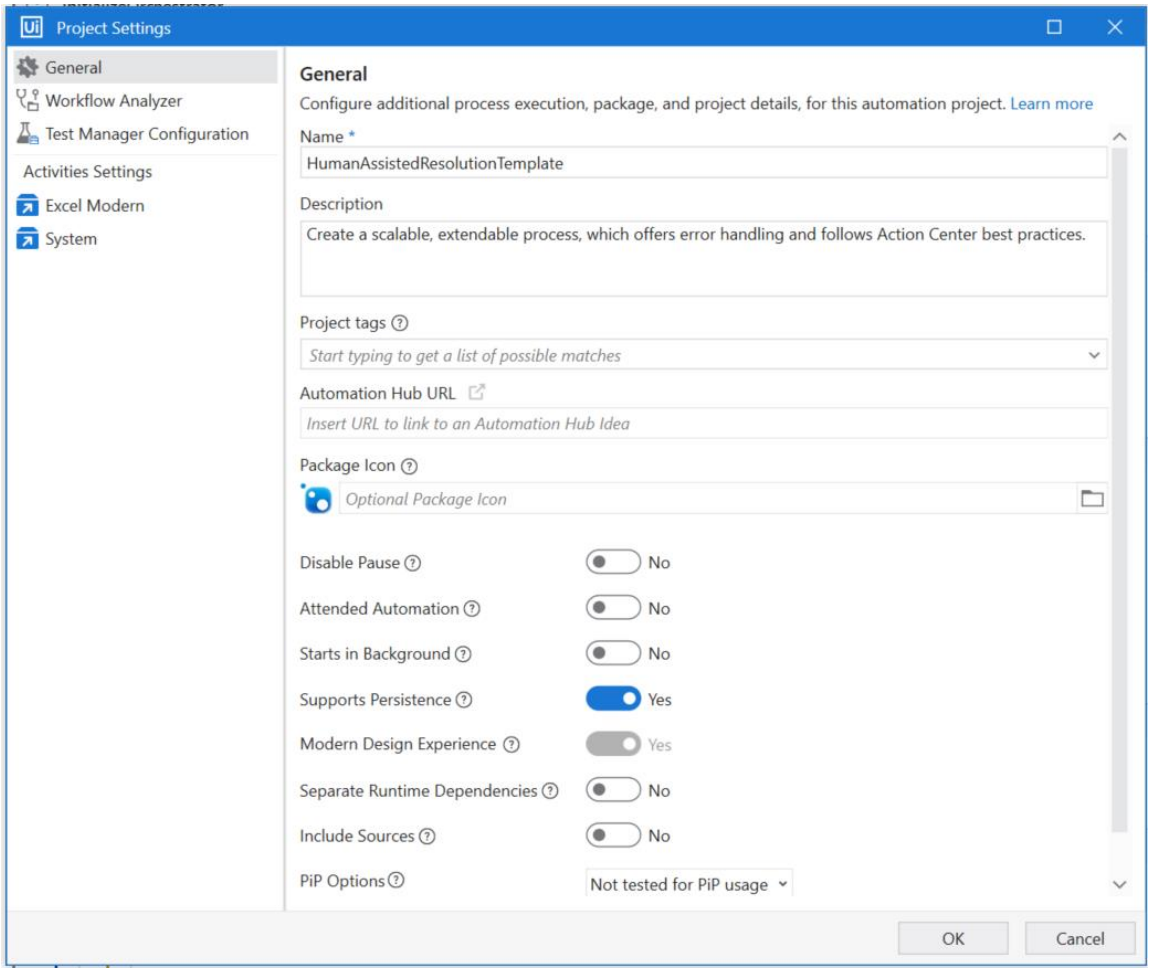
Studio Project Overview

The HART Process is available as a UiPath Studio project template and the projects created using it automatically include all files.

Settings for Unattended Processes

Enable Persistence Support

Open the Project Settings and make sure that **Supports Persistence** is set to **Yes**.



Example Implementation

To facilitate understanding the Human Assisted Resolution Process template, it comes with a pre-implemented example. It showcases the processing of 4 types of error handling/human in the loop use cases. Workflow-specific details can be found in the next chapter.

Project Files

Data\Config.xlsx

Configuration file for project settings. Minimum configuration required:

- Configure the **OrchestratorFolder**
- Configure the **StorageBucketName** under the **Settings** sheet.
- Configure the **TaskTitle**, **Title**, **Description**, **SolutionDescription** and **FileName** under the **Settings/Assets** sheet. The configuration of the **TaskCatalog** is optional.
- Configure the **OrchestratorQueueName** if using queues.

Data\Files\HART.pdf

File, used as a component in one of the Form JSON files. The **Files** folder can contain any pdf/image/documents to be displayed in Action Center.

Data\Templates\BusinessRuleException_ProcessName.json

An example of form JSON file for the business rule exception case.

Data\Templates\SystemException_ProcessName.json

An example of form JSON file for the system exception case.

Data\Templates\Exception_AssignedUsersProcess.json

An example of form JSON file for the case where the user that has been assigned to a task should be retrieved.

Data\Templates\CategorizeDocument_ProcessName.json

An example of form JSON file for the case in which a user submitted data, regarding a specific document, is displayed together with the robot classification of that document and the RPA rules performed on it, for a reviewer to check. The form outputs are the remarks given by robot, after being reviewed and possibly changed (the reviewer could deem an RPA rule as passed, can change the category/subcategory of the document) by the reviewer, and the custom remarks given by the reviewer.

Data\HttpRequest\Asset.json

Payload for asset generation in Orchestrator (the payload is used as parameter in the Http Request activity)

Data\HttpRequest\StorageBucket.json

Payload for storage bucket generation in Orchestrator (the payload is used as parameter in the Http Request activity)

Data\HttpRequest\TaskCatalog.json

Payload for task catalog generation in Orchestrator (the payload is used as parameter in the Http Request activity)

Main.xaml

Workflow to be set and used as Main for unattended HART processes that use Action Center for Human-in-the-Loop.

Arguments:

- in_Template (default **Nothing**): specifies what form JSON file should be used during the task creation step.
- in_UseQueue (default **True**): specifies whether Orchestrator queues are used. If set to True, the value of the in_TargetFile argument is ignored and the file to be processed is fetched from the Transaction Item.

InitializeOrchestrator.xaml

Workflow should be executed before Main.xaml.

Checks if the storage bucket and the assets from the settings sheet have been manually created in Orchestrator. If not, they are being created through Orchestrator HTTP requests. The creation of the task catalog is optional.

Arguments:

- in_ConfigFile (default **Data\Config.xlsx**): specifies the path of the configuration file.
- in_ConfigSheets (default **Settings**): specifies from what sheet the contents (storage bucket, assets, task catalog) should be extracted.

Framework\00_ReadConfigFile.xaml

Reads the contents of the Config file into a Config dictionary at runtime.

Note: Does not load Orchestrator assets!

No custom code was added here for the purpose of creating the example implementation.

Framework\10_InitializeProcess.xaml

Workflow that loads the Orchestrator assets. Any process-specific initialization code belongs here.

No custom code was added here for the purpose of creating the example implementation.

Framework\20_GetTransactionItem.xaml

Gets the next Transaction Item when using Orchestrator queues. The target file to be processed is expected to be found under the **TargetFile** key of the Transaction Item's **SpecificContent**.

Also loads all the Transaction Item's **SpecificContent** into the Config dictionary for ease of use.

No custom code was added here for the purpose of creating the example implementation.

Framework\30_SelectTemplate.xaml

Workflow for selecting the correct path of the form JSON file, which is used by the **Create Form Task** activity.

Framework\40_DataPostProcessing.xaml

Workflow for processing the task data.

No custom code was added here for the purpose of creating the example implementation.

Framework\50_EndProcess.xaml

Workflow for Post-Export Processing and Process Cleanup logic.

No custom code was added here for the purpose of creating the example implementation.

Framework\ERR_AbortProcess.xaml

Workflow that is executed if the process is aborted due to a terminating exception. Code for error cleanup or for sending error notifications belongs here.

No custom code was added here for the purpose of creating the example implementation.

Framework\ReusableWorkflows\GenerateId.xaml

Generates an ID, which is used in the payload creation of the HTTP requests.

Framework\ReusableWorkflows\GenerateKey.xaml

Generates a key, which is used in the payload creation of the HTTP requests.

Framework\ReusableWorkflows\OrchestratorRequest_GET.xaml

Executes an Orchestrator HTTP request to retrieve data from Orchestrator.

Framework\ReusableWorkflows\OrchestratorRequest_POST.xaml

Executes an Orchestrator HTTP request to create different entities, like assets/storage buckets/task catalogs into Orchestrator.

Framework\ReusableWorkflows\SetTransactionStatus.xaml

Sets and log the transaction's status. The approach is like the one used by the RE-Framework.

Quick Start Guide

Orchestrator Configuration

- If using Action Center, create a **Storage Bucket** for your process.
- If needed, create a **Queue** for your process.

Unattended Automation

- If using Action Center, configure the **StorageBucketName** under the **Settings** sheet.
- If using queues, configure the **OrchestratorQueueName**.
- Configure the **Settings for Unattended Processes**.