Renaud HENRY

Les clichés de renaud

Guide d’utilisation de Robot Entreprise Framework Custom

Table des matières

[1) Présentation du ROBOT ENTREPRISE FRAMEWORK (REF) 2](#_Toc86560562)

[2) Présentation du ROBOT ENTREPRISE FRAMEWORK CUSTOM (REFC) 3](#_Toc86560563)

[a. Lacunes du REF 3](#_Toc86560564)

[b. Représentation et détailles premier niveau du REFC 3](#_Toc86560565)

[c. Structure des dossiers Modernes dans l’Orchestrateur 4](#_Toc86560566)

[Figure 1 : le ROBOT ENTREPRISE FRAMEWORK (REF) 2](#_Toc86523734)

[Figure 2: le ROBOT ENTREPRISE FRAMEWORK CUSTOM (REFC) 3](#_Toc86523735)

# Présentation du ROBOT ENTREPRISE FRAMEWORK (REF)

Le ROBOT ENTREPRISE FRAMEWORK (REF) proposé par UIPATH est un outil de base pour implémenter des processus.

Les tâches des processus robotisés sont atomiser en TransactionItem dans un file d’attente. Cela permet d’avoir une granulosité fine dans l’analyse du comportement des robots.

Le REF se décompose de 4 états :

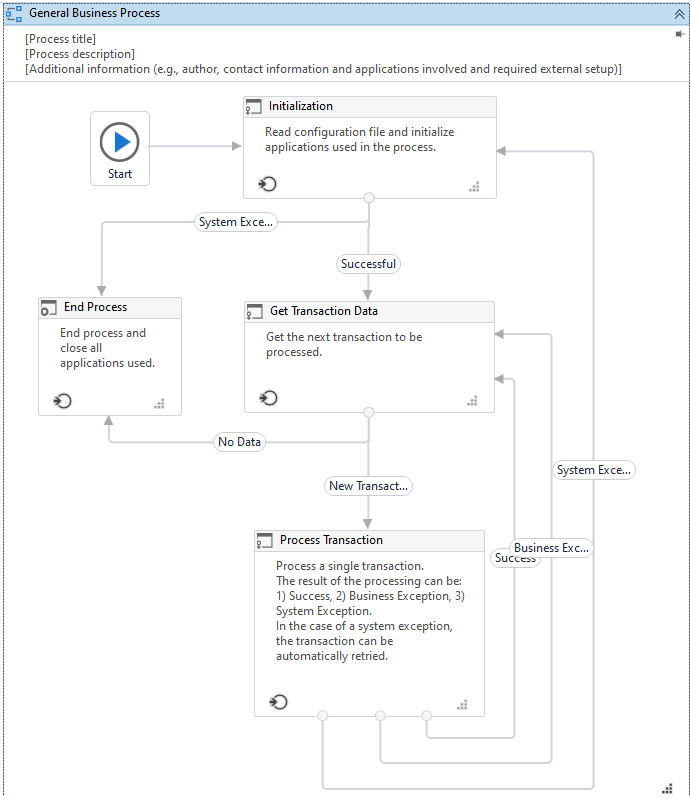


Figure 1 : le ROBOT ENTREPRISE FRAMEWORK (REF)

Etat 1 : Initialization

Cette état permet de charger la configuration et d’initialiser les applications

Etat 2 : Get Transaction Data

Cette état permet récupérer une TransactionItem de la file d’attente

Etat 3 : Process Transaction

Cette état permet réaliser la travail issus de la TransactionItem récupérée à l’état 2

Etat 4 : End Process

Cette état permet clore le programme

Dans ce contexte la file d’attente est alimenté par un second programme UIPATH nommé Dispatcher.

# Présentation du ROBOT ENTREPRISE FRAMEWORK CUSTOM (REFC)

## Lacunes du REF

Le REF souffre de quelques lacunes :

* Il faut maintenir 2 programmes UIPATH, le REF et le Dispatcher séparément
* Si on veut exécuter une procédure au lancement et à la fin du robot, cela n’est pas possible.
* Si on souhaites la gestion de la notion de groupe de TransactionItem, cela est complexe à obtenir.
* La gestion des erreurs via un email avec capture d’écran n’est pas proposé par le REF
* Si on dispose d’une base de données, on serait tenté d’enregistrer les log dedans.
* Le dictionnaire qui contient sur un seul niveau tous les paramètres

## Représentation et détailles premier niveau du REFC

C’est pour cela que je vous proposes le REFC en 6 états

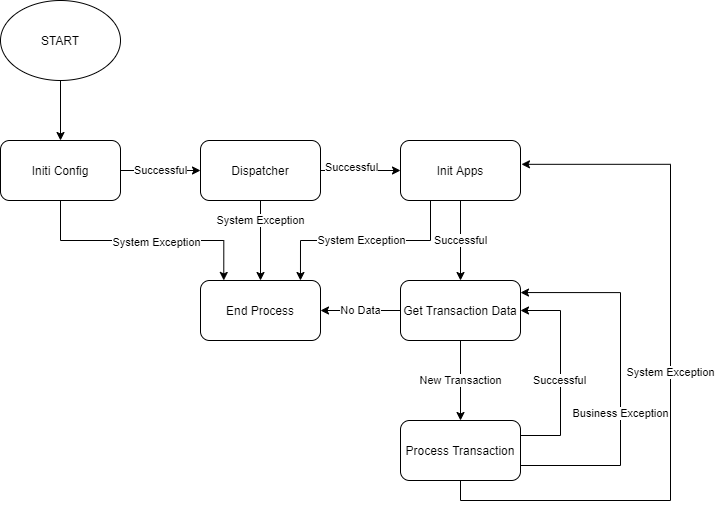


Figure 2 : le ROBOT ENTREPRISE FRAMEWORK CUSTOM (REFC)

Etat 1 : Initialisation de la configuration

* Cette état permet de charger la configuration et d’exécuter une procédure à lancement du robot

Etat 2 : Dispatcher

* Cette état permet le générer les TransactionItem de la file d’attente, avec une fonction Mise à zéro de la file d’attente.

Etat 3 : Initialisation des applications

* Cette étape permet Initialiser les applications du robots

Etat 4 : Get Transaction Data

* Cette état permet récupérer une TransactionItem de la file d’attente

Etat 5 : Process Transaction

* Cette état permet réaliser la travail issus de la TransactionItem récupérée à l’état 2

Etat 6 : End Process

* Cette état permet clore le programme ,d’exécuter l’envoi de l’email récapitulatif et de lancer une procédures à la fin du robot

## Structure des dossiers Modernes dans l’Orchestrateur

Afin de faire fonctionner le REFC il faut une structure de dossiers modernes tel que :

1. Un dossier Racine ‘’CONFIG’’ avec:
   1. une ressource (Asset) nommé ‘’TemplateFolder’’ :
      * Cette ressource permet au REFC de savoir l’architecture des dossiers de l’Orchestrateur
      * Exemples :
        + {$Env$}/{Projet} => UAT/RPA0010\_BOT-FACTURE
        + {Unite}/{RegionMonde}/{Pays}/{ Projet }/{$Env$} => FINANCE/EUROPE/FRANCE/RPA0010\_BOT-FACTURE/UAT
   2. des sous-dossiers, un par environnement avec dans chaque sous- dossiers des ressources :
      * CCEmailError => destinataire en copie des mails d’erreurs (obligatoire)
      * CCEmailRecap => destinataire en copie des mails des récapitulatifs (obligatoire)
      * DestinataireEmailError => destinataire direct des mails d’erreurs (obligatoire)
      * DestinataireEmailRecap => destinataire direct des mails des récapitulatifs (obligatoire)
      * emailError => Email émetteur des mails d’erreurs (obligatoire)
      * emailRecap => Email émetteur des mails des récapitulatifs (obligatoire)
      * NameEmailError => Nom de l’émetteur des mails d’erreurs (obligatoire)
      * NameEmailRecap => Nom de l’émetteur des mails des récapitulatifs (obligatoire)

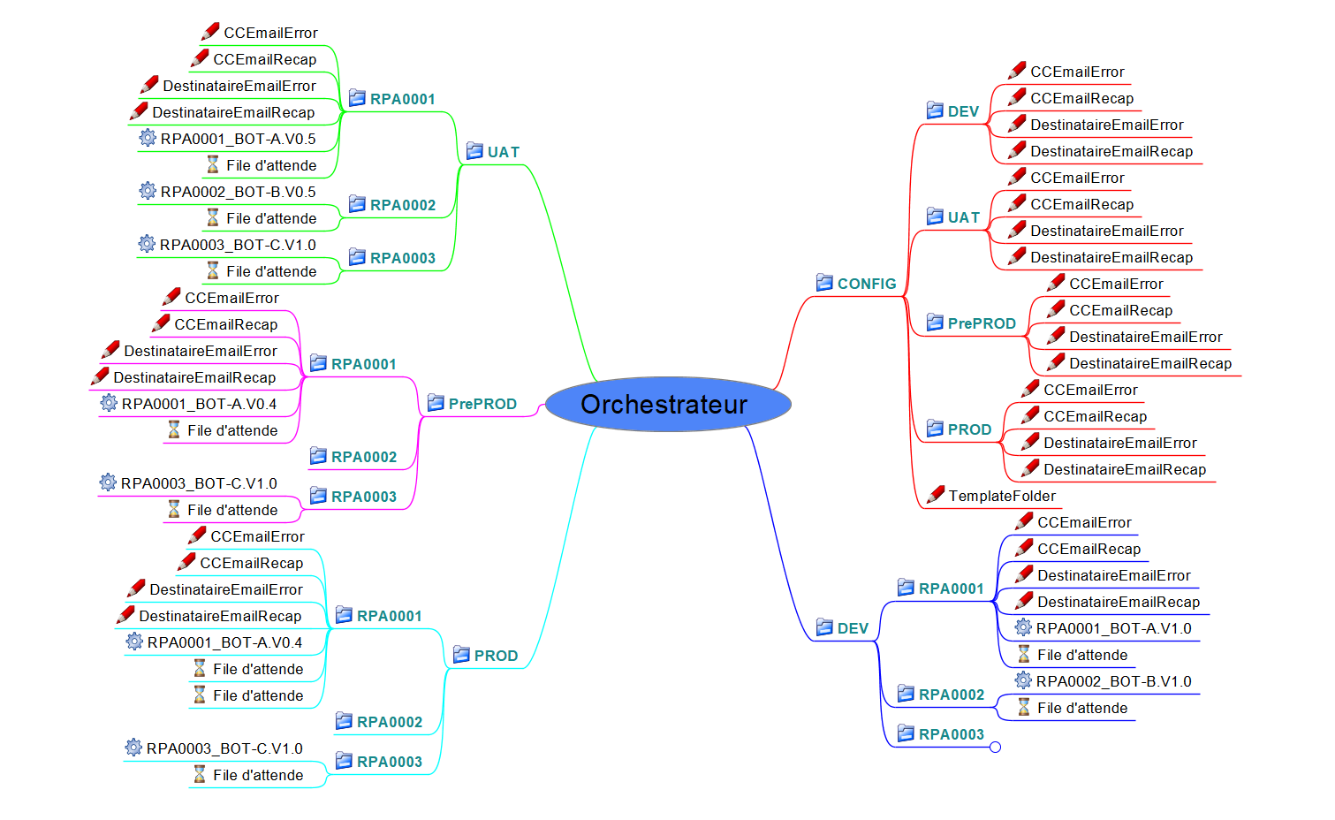


Figure 3 : Structure simple du REFC

 : Dossier Orchestrateur

 : Resource Texte

 : File d’attende

La Figure 3 n’est pas optimale avec le Template {$Env$}/{Projet} car les ressources du BOT A se répéter dans les différents dossier donc je propose un nouveau Template, { Projet}/{$Env$}, ce qui donne la Figure 4.

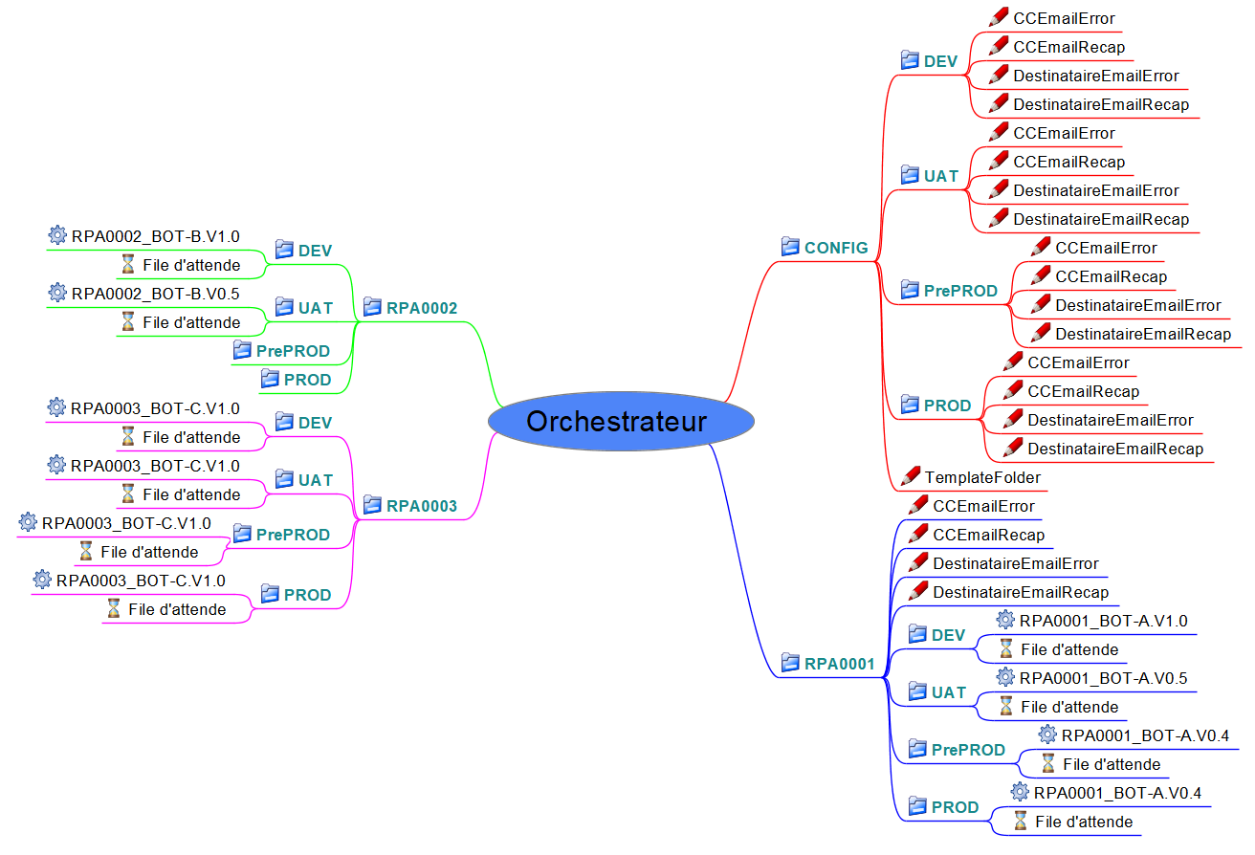


Figure 4 : Structure simple optimisé du REFC

Dans la Figure 4, les ressources du dossier « RAP00 » supplante les ressources génériques (pour tous les ROBOTS) du dossier « CONFIG/[DEV, UAT, PrePROD, PROD] ». Ce mécanisme permet de mettre des valeurs générales à tous les robots, mais aussi de spécifier pour certains robots des ressources uniquement pour eux.