# Belangrijkste aanpassingen t.o.v. framework

## Config

* ***Config ALGEMEEN***
  + Wanneer er verschillen in variabelen zijn tussen Acc- en Prod-omgeving, kan je beide variabelen in resp. kolom E en F weergeven. Laat vervolgens kolom D verwijzen naar de actieve kolom. De daadwerkelijke waarde in kolom B kan je vervolgens verwijzen naar kolom D. Op die manier pas je makkelijk de omgeving aan zonder de waarden weg te gooien.
  + Er is een extra Sheet *LogMessageHtml* met een aantal variabelen die worden gebruikt in het logmailtje aan het einde van de run. Hier kan je nog de procesnaam invullen en e.v.t. wat elementen aanpassen.
* **Config: *ConfigFileLocation*** – Heeft geen value in config zelf, wordt in workflow zelf ingevuld en e.v.t. aangepast. Kan gebruikt worden wanneer je op een ander punt in de workflow bijv. een andere sheet uit je configfile wilt aanroepen.
* **Config: *Production*** – True/False, kan gebruikt worden wanneer in de flow bepaalde instellingen wisselen in acc t.o.v. prod. Kan je in de flow omzetten middels Boolean.Parse()
* **Conifg: *UseOrchestratorTransactions*** – Het gebruik van Orchestrator Transactions kan worden in- en uitgeschakeld
* **Config**: ***MaxTransactions*** *–* Om het maximaal aantal transacties in één run te beperken. ALL / INPUTDIALOG / of een getal
* **Config: *db\_\****
  + ***UseConnection*** – True/False – Biedt de mogelijkheid om de databasefunctionaliteit van het framework uit te schakelen. De overige db\_-instellingen vervullen dan geen functie meer.
  + ***ConnectionString*** – De ConnectionString om te verbinden met de database
  + ***GetCredentials*** – Voor wanneer in de ConnectionString een user/pwd nodig zijn. Deze nooit handmatig in de string uitschrijven, maar onder deze configvariabele een Credentialnaam opgeven. Bij het eerste gebruik op een machine wordt om deze credentials gevraagd. In de ConnectionString kan je dan de elementen [user] en [password] opnemen.
  + ***ProviderName*** – Deze instelling kan je vrijwel altijd laten staan op System.Data.SqlClient
  + ***JobLogTable*** – De tabel waar (voor meedere processen) de verschillende runs worden gelogd.
  + ***WorkFlowLogTable*** – De tabel waar (voor meerdere processen) de logregels worden weggeschreven aan het einde van de run.
  + ***TransactionStatusTable*** *–* De tabel waar (specifiek voor dit proces!) alle transacties worden gelogd. Deze naam moet je dus aanpassen aan de *procesnaam­\_transactions*
* **Config: *Folder*\_*Temp*** – Een tempfolder die aan het begin/eind van de workflow automatisch wordt geleegd. Dit kan een relatief/hard pad zijn. Vaak is een hard pad (buiten je projectfolder) het meest praktisch, wanneer je deze map op meer punten in je proces gaat gebruiken.
* **Config: *Folder\_Log*** – Een map waar de logbestanden in worden weggeschreven. Ook de screenshots bij een SystemError worden hierin geplaatst. Ook hiervoor is het handiger om een hard pad buiten de projectmap te bepalen.
* **Config: *LogMessageAddress*** – Een mailadres waar de logmailtjes naar toe mogen. Bij ontwikkelen een mailadres van jezelf invullen (om de klant niet te overspoelen). Meerdere adressen kan, gescheiden door puntkomma (;), maar let er wel op dat je de hyperlink die excel er dan van maakt weer verwijdert, anders krijg je een foutmelding.

## Main – INIT

* **Use ConfigFileCopy** – Dit gedeelte (standaard commented out) kan je enablen. Er wordt dan eerst een kopie van het configfile gemaakt. Dit voorkomt de foutmelding die je krijgt als je de config nog open hebt staan tijdens het bouwen. Zo kan je gemakkelijk aanpassingen doorvoeren.
* **Ask for MaxTransactions** – Geeft een dialoogvenster (als dit is ingesteld in de config) met de vraag hoeveel transacties er maximaal moeten worden verwerkt.
* **Get Database Credentials**  - Voegt de databasecredentials toe aan de ConnectionString (als dit is ingesteld in de config).
* **Retrieve JobDetails** - Voor de logging wordt een dictionary ‘JobDetails’ aangemaakt, die op basis van het (automatische) NLogbestand een aantal waarden uitleest en verwerkt.
  + Een onderdeel hiervan is de flow ProcessNLogFile.xaml. Wanneer je een enorm groot logbestand hebt, doordat je TRACE-logging (log activities) aan hebt staan geeft dit problemen. Geef dan het logbestand een andere naam (dan wordt er vanzelf een nieuw aangemaakt) en zet TRACE-logging uit.
* **CleanUpTempFolder.xaml** – de tempfolder wordt (geprobeerd) te legen

## Main – Get Transaction Data

* **TransactionItem** – variabele type is in dit framework standaard String
* **Check for MaxTransactions** – De configinstelling wordt hier gecheckt. Ook wordt er aan de dictionary JobDetails een EndProcessTrigger Key toegevoegd met de reden waarom de flow beëindigt (voor o.a. het logmailtje).
* **GetTransactionData.xaml**
  + Hierin wordt een dictionary TransactionDetails aangemaakt. Hierin kunnen specifieke waarden voor een transactie worden toegevoegd, die weer in Process kunnen worden gebruikt. Dit scheelt steeds veel variabelen en argumenten aanmaken. In het bestand Documentation/Dictionaries.xlsx kan je bijhouden welke keys je in de dictionary stopt.
* **Queue- / TransactionItems** – Wanneer Orchestrator wordt gebruikt i.c.m. een Queue, dan is dit de plek om in te stellen hoe je deze gebruikt. Wanneer er al QueueItems in Orchestrator aanwezig zijn (vanuit bijv. een dispatcher), dan gebruik je het gedeelte Get Queue Items. Als je de transactie-items ín de flow ophaalt, gebruik dan Add Transaction Item. **Altijd maar een van beide!**

## Main – Process Transaction

* **Add Transaction Log Fields** – De transactie-specifieke logvelden worden hier toevoegd voor alle transactie-gerelateerde logregels.
* **Update Database (start)** – Voor iedere transactie wordt een nieuwe regel in de database tabel TransactionStatusTable aangemaakt. Bij een retry wordt dezelfde regel weer gebruikt. Wanneer je specifieke waarden uit je TransactionDetails dictionary óók wilt wegschrijven in de database, dan kan dat worden toegevoegd aan de argumenten van UpdateDatabaseTransactionRow.xaml
  + Deze flow kan je op meer punten in Process aanroepen. De argumenten in\_SET\_Values en in\_SET\_Columns kan je verrijken (list of strings) met de columnname en de bijbehorende variabelen. De volgorde maakt niet uit, maar Columns en Values moeten uiteraard altijd precies corresponderen. Ook moet de desbetreffende kolom in de database wel zijn aangemaakt.
* **Process.xaml** – Deze flow heeft het extra in-argument (dictionary) *in\_TransactionDetails*, die al gevuld wordt in o.a. GetTransactionData.
* **Update Database (end)**  - Als Process.xaml zonder problemen is doorlopen, wordt de eindtijd van de transactie ook toegevoegd in de database.
* **Try Catch – BusinessRuleException / SystemException** – In het geval van Technische of Functionele uitval wordt de uitvalmelding ook nog weggeschreven in de database.
* **Try Catch – Finally** – **Remove Transacton log fields** – Het mee-wegschrijven van de transactie-specifieke variabelen aan de logregels wordt hier beëindigd.

## Main – End Process

* **CloseAllApplications.xaml** – Deze workflow bevat nu ook het argument *in\_Config*, voor het geval dat handig/nodig is bij deze workflow.
* **CleanUpTempFolder.xaml** – De temp-folder wordt weer geleegd.
* **ProcessWorkflowLogs** – Alle logregels van de huidige run worden uitgelezen en verwerkt. Dit kan lang duren / problemen geven wanneer je TRACE-logging (log activities) aan hebt staan.
* **SendLogEmail**.**xaml** – Op basis van de logregels wordt het logmailtje opgebouwd. Er wordt een html-mailbericht opgesteld. Elementen hiervoor komen uit de config (sheet LogMessageHtml) en uit Data\LogMessageHtmlBody.
  + In de basis bestaat het bericht uit:
    - Header.txt
    - Body1.txt
    - Table1
    - Body2.txt
    - Table2
    - Body3.txt
    - Footer.txt
  + De verschillende variabelen in deze tekstbestanden worden vervangen door elementen uit o.a. de config.
  + De Html-tabellen worden gegenereerd met behulp van de workflow **LogEmailBuildHtmlTable.xaml**. Deze heeft als in-argument een datatable die wordt omgezet naar een html-tabel. Ook de gewenste tabelbreedte (in px of %) en de breedteverdeling van de kolommen is instelbaar. (array of strings, aantal moet corresponderen met de kolommen in de datatabel)
  + Het **Create Attachment** gedeelte bouwt een excel-bestand op met meerdere sheets. De sheet voor TransactionsLast100Days kan daar worden aangepast / gedisabled indien ongewenst.
* **SendLogEmail – Try Catch – Exception** – Wanneer er een foutmelding optreedt in SendLogEmail.xaml, probeert UiPath nog een algemeen mailtje te versturen (Send Outlook Mail Message), eveneens in een try-catch
* De **Mail-activity** voor het logmailtje omzetten naar bijv. SMTP i.p.v. Outlook moet dus op twee plekken gebeuren (try, catch). Hiervoor zullen wat extra config-variabelen moeten worden aangemaakt.
* **Update Job Endtime**  - De eindtijd van de job en de end-trigger wordt nog weggeschrevne in de database.

# Werkwijze / handleiding

* Pas allereerst in het **project.json** bestand via notepad o.i.d. de Projectnaam en de Description aan. Anders is e.e.a. in de logging niet meer uit elkaar te houden.
* Geef in de config al direct de juiste variabelen op, waaronder die voor de TransactionStatusTable (als je database gebruikt).
* De tabellen moeten uiteraard wel eerst worden aangemaakt. De workflow **CreateDatabaseStructure.xaml** kan je gebruiken om de database in te richten ***nadat de config juist ingesteld***. Bedenk goed of prod/acc verschil op databaseniveau of op tabelniveau wordt gemaakt en hoe dus de tabellen moeten heten.
  + Wanneer je deze flow eenmalig runt worden alle nog niet bestaande tabellen aangemaakt.
  + De basis hiervoor zijn de bestanden in Data\Scripts\....sql. Extra benodigde kolommen voor je TransactionStatusTable kan je hier aan toevoegen.
  + Wanneer je voor de prod-database alleen via UiPath Robot de database kunt benaderen, kun je dus éénmalig een package van je project runnen waarin deze workflow als ‘main’ is ingesteld en de config is ingesteld op prod.