© Andy Menon

Advanced Enterprise RPA Process Template

A Database Configurable RPA Process Flow Template

Contents

Introduction....3
The Configuration Database....3
Tenant....4
Process....5
Setting_Category....5
Setting_Type....6
Config....6
VW_Load_Config- The Public Interface View....7
Framework components to access the View....8
Connecting to the Configuration Database....9
Justifying the use of ODBC DSNs....9
To be fair, some flip sides to ODBC DSNs....10
Conclusion....10
Setting up a Microsoft Access DSN on the Robot Machine....10
Locking the MS Access Database....11
Centralizing the MS Access Database File....11
Setting up a Remote Database DSN on the Robot Machine....14
The MasterConfig.xlsx Framework Configuration File....16

Importing Additional Database Libraries 18
Configuring a New RPA Process Flow in the Configuration Database 19
Configuring a New RPA Process Flow 23
Main.xaml 23
Init State Machine 24
Loading Configuration Settings – Most Significant Change 25
KillAllFolders.xaml - New Component to Drop Folders 26
Move_or_Delete_Directory.xaml 27
Move_or_Delete_File.xaml 27
InitAllApplications.xaml 27
Create_Directory.xaml - New Component to Create Folders 28
KillAllProcesses.xaml 29
Utility Component - Identify_Processes_By_Name.xaml 29
Send_Email_Notification.xaml 31
Validating the Project before Deployment 31
Test Results and End Note 32

Introduction

This User guide documents the set up and configuration of an Advanced Enterprise RPA Process Template. This framework template enables enterprises to configure settings for their RPA Processes in local or remote databases such as MS Access, SQL Server, MySQL, PostgreSQL or AWS RDS.

Configuring the framework in databases enables enterprises to better organize, secure and port their RPA processes across the enterprise.

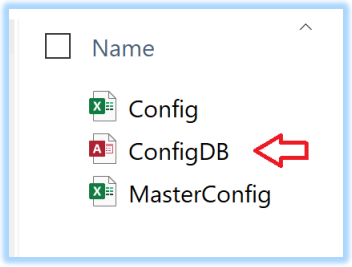
This framework is built upon an already existing UiPath component published here at the UiPath Go! Marketplace. Consequently, this advanced enhancement retains the capability of the original to organize configurations in a local Excel file.

The Configuration Database

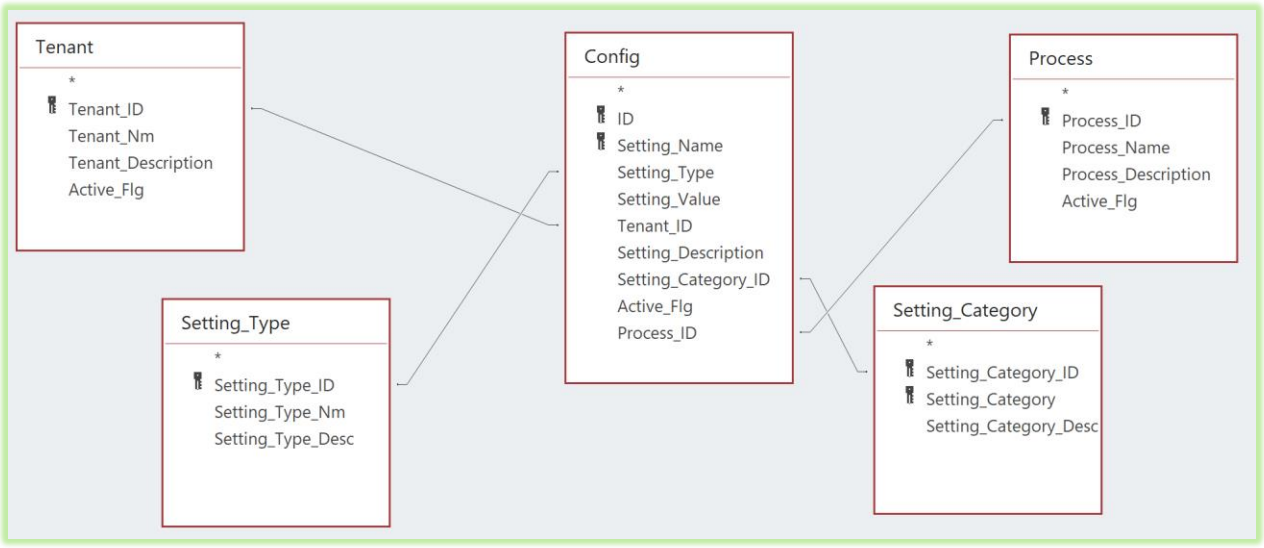
In its current form an Excel file is used to configure settings of an RPA Process built using the *current framework*. While this is great for small scaled operations, managing individual Excel files becomes logistically difficult especially when there are several tens, if not hundreds of integrations involved. Besides, Excel files are documents and are prone to getting loosely dispersed across an enterprise via emails, messengers and shared drives. This may result in an outdated or wrongly configured file deployed to a production environment by accident.

The key advancement in this template is its ability to consume configuration settings from a local database (*such as Microsoft Access*) or a remote, full-featured enterprise SQL database such as Microsoft SQL Server, MySQL, AWS RDS, or PostgreSQL.

In order to make it easier to get started, a templated Microsoft Access Database is included as part of the framework template within the **\Data** folder. The schema from this database could be easily extracted and recreated on any common enterprise database if you choose to.

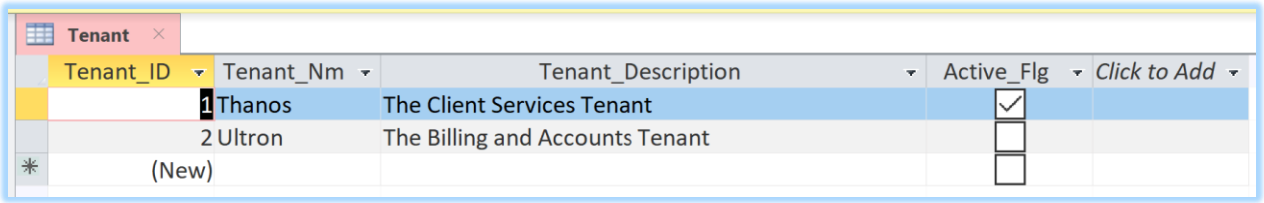


The database is very simple and consists of five tables as shown in the design schematic below:



Tenant

The *Tenant* table maintains a logical record of the tenant(s) where an RPA process has been deployed. An enterprise deployment could consist of one or more Orchestrators that are either on-premise or remote, single or multi-tenant. Therefore, the Tenant is not a technical attribute of a given RPA process. Rather, it helps enterprises to better recognize and keep track of where their RPA processes are deployed to, especially if there many of them across several groups across the organization.



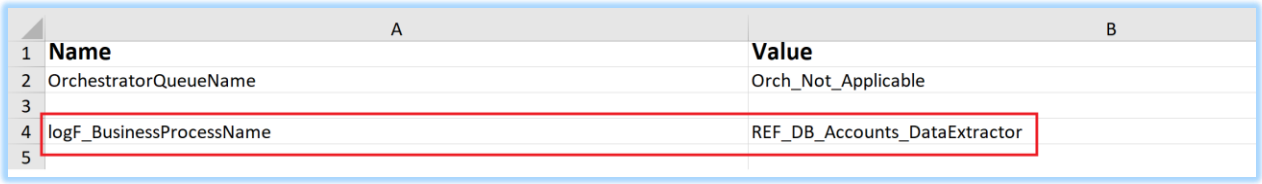
The *Active_Flg* indicates that the first tenant is currently active and in service.

NOTE

The Tenant could be further broken down to Tenant and Service. However, the config database and the template could be easily modified during implementation based on how an enterprise would prefer to organize their RPA processes. And for this reason, such depth is not covered in the current scope of this template.

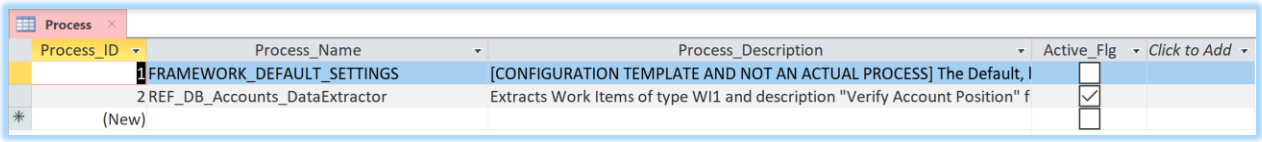
Process

Maintains a record of the names of the RPA Business processes deployed across the enterprise. Ideally, this would be the value that would be set for the *logF_BusinessProcessName* key in the Config.xlsx document:



	A	B
1	**Name**	**Value**
2	OrchestratorQueueName	Orch_Not_Applicable
3		
4	logF_BusinessProcessName	REF_DB_Accounts_DataExtractor
5		

In the Process table, a business process is recorded as follows:



Process

Process_ID	Process_Name	Process_Description	Active_Flg	Click to Add
1	FRAMEWORK_DEFAULT_SETTINGS	[CONFIGURATION TEMPLATE AND NOT AN ACTUAL PROCESS] The Default, I	☐	
2	REF_DB_Accounts_DataExtractor	Extracts Work Items of type WI1 and description "Verify Account Position" f	☑	
(New)			☐	

As before, the *Active_Flg* indicates that a process is in active service.

NOTE

The Access database comes pre-loaded with a "shell process" named *FRAMEWORK_DEFAULT_SETTINGS*. This process will make it easier to configure actual business processes in the database as we will see in subsequent sections.

The shell is used to aid in the configuration setup of new RPA processes and is not a deployable process by itself. **Therefore, do not delete entries for this shell process from the Config table!**

Setting_Category

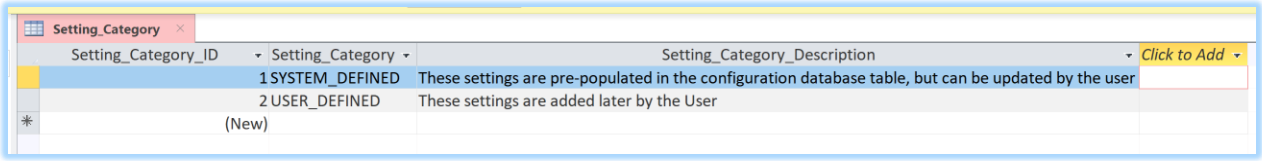
The configuration broadly recognizes two categories of settings - SYSTEM_DEFINED and USER_DEFINED. As the names suggest, the system-defined settings are the 18 standard settings across the three tabs of the Config.xls configuration document.

The user-defined settings on the other hand are those that are specific to a given RPA process and are explicitly added by the users over and above the system-defined settings.

Organizing settings in this manner makes it easier to distinguish them for better maintenance.

For Example: Users would exercise caution before going in and changing any of the settings if they knew that they are SYSTEM_DEFINED. Similarly, USER_DEFINED settings would make it easier to copy configurations over to a new processes that resembles an existing one.

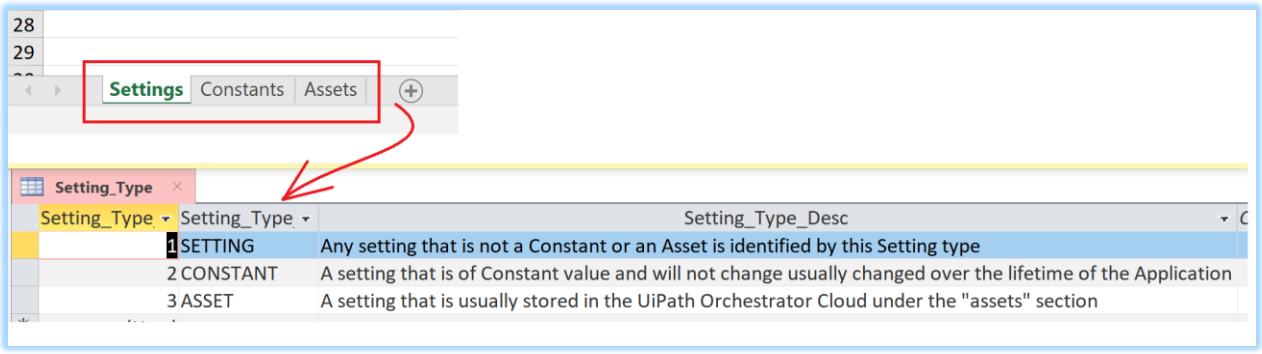
Each setting in the Config table (*described later below*) is associated with a setting category.



Setting_Type

Easily defined, these are the names of the three worksheets from the traditional Config.xlsx configuration file. Each setting in the Config table is associated with a setting type.

The figure below shows the parallels between Config.xlsx and the setting types in the *Setting_Type* table



Config

This is the central configuration table that drives an RPA Process. An Advanced RPA Process flow built using this template consumes settings from the *Config* table via an interface view (*described in the next section*). The table has been named so to easily recognize the parallels between the configuration database and the traditional excel file named Config.xlsx

Each entry in the Config table is comprised of three columns:

Setting_Name – counterpart to the "Name" column from the Config.xlsx

Setting_Value – counterpart to the "Value" column from the Config.xlsx

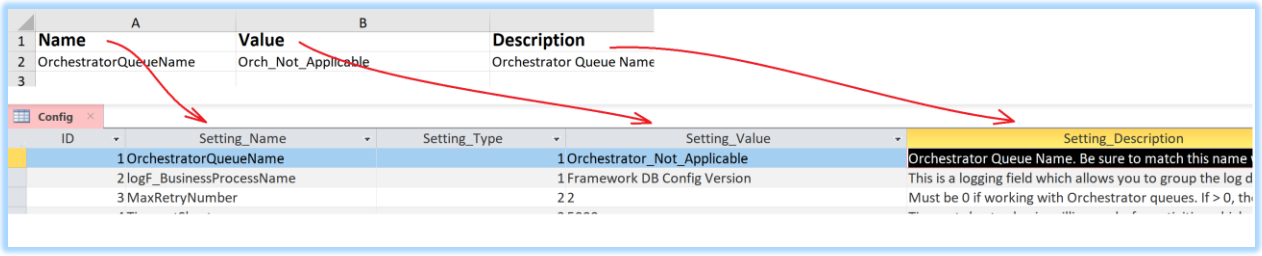
Setting_Description- the "Description" counterpart from the Config.xlsx

NOTE

Unlike the Config.xlsx the names of the columns have been changed because "Name" and "Value" are reserved keywords in some database platforms.

For Example: PostgreSQL does not support a column named "Name"

Figure below shows the parallels between the Config.xlsx document and the Config database table:



Each entry in the Config table is associated with the following attributes from the tables described earlier:

1. The Tenant ID from the Tenant table
2. The Process ID from the Process table
3. The Setting_Category ID from the Setting_Category table
4. The Setting_Type ID from the Setting_Type table

VW_Load_Config- The Public Interface View

Regardless of the underlying structure of the configuration database, the RPA process will consume settings from a public interface view that must remain largely unchanged. The view must expose a set of mandatory columns to work seamlessly irrespective of where the database has been deployed to.

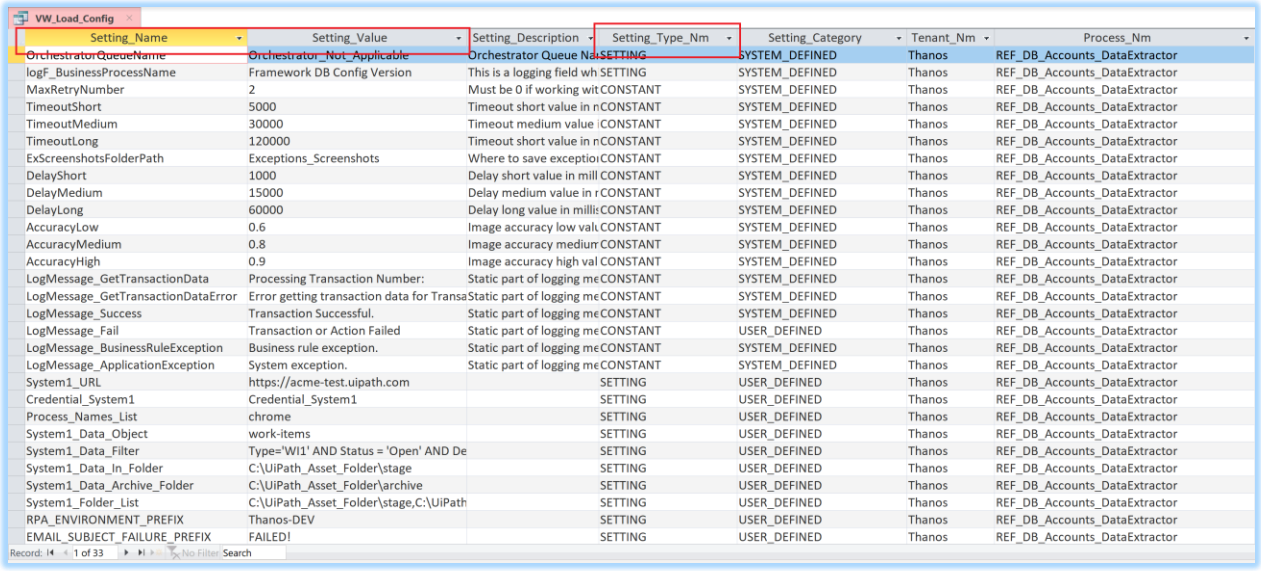
In line with UiPath's Security Best Practices, it's recommended to implement the following:

1. Tightly regulate edit permissions to the database and underlying tables
2. Expose just the view to the RPA application using a non-Admin database login
3. If the database is MS Access, then lock the database down using a complex password

The view must be visible to the RPA process in the format shown below. This template is by default designed to load settings as demonstrated by this example SQL:

```
SELECT Setting_Name, Setting_Value, Setting_Type_Nm
FROM VW_Load_Config
WHERE Tenant_Nm = 'Thanos'
AND Process_Nm  = 'REF_DB_Accounts_DataExtractor';
```

Note that all the three columns in the SELECT statement are required. The template will load settings that are associated with a pre-defined Tenant and Process names organized in the Tenant and Process configuration tables respectively, as discussed in the previous sections.



VW_Load_Config

Setting_Name	Setting_Value	Setting_Description	Setting_Type_Nm	Setting_Category	Tenant_Nm	Process_Nm
~~OrchestratorQueueName~~	~~Orchestrator_Not_Applicable~~	Orchestrator Queue Na	SETTING	SYSTEM_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
logF_BusinessProcessName	Framework DB Config Version	This is a logging field wh	SETTING	SYSTEM_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
MaxRetryNumber	2	Must be 0 if working wit	CONSTANT	SYSTEM_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
TimeoutShort	5000	Timeout short value in n	CONSTANT	SYSTEM_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
TimeoutMedium	30000	Timeout medium value i	CONSTANT	SYSTEM_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
TimeoutLong	120000	Timeout short value in n	CONSTANT	SYSTEM_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
ExScreenshotsFolderPath	Exceptions_Screenshots	Where to save exceptio	CONSTANT	SYSTEM_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
DelayShort	1000	Delay short value in mill	CONSTANT	SYSTEM_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
DelayMedium	15000	Delay medium value in r	CONSTANT	SYSTEM_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
DelayLong	60000	Delay long value in millis	CONSTANT	SYSTEM_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
AccuracyLow	0.6	Image accuracy low val	CONSTANT	SYSTEM_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
AccuracyMedium	0.8	Image accuracy mediur	CONSTANT	SYSTEM_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
AccuracyHigh	0.9	Image accuracy high val	CONSTANT	SYSTEM_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
LogMessage_GetTransactionData	Processing Transaction Number:	Static part of logging me	CONSTANT	SYSTEM_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
LogMessage_GetTransactionDataError	Error getting transaction data for Transa	Static part of logging me	CONSTANT	SYSTEM_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
LogMessage_Success	Transaction Successful.	Static part of logging me	CONSTANT	SYSTEM_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
LogMessage_Fail	Transaction or Action Failed	Static part of logging me	CONSTANT	USER_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
LogMessage_BusinessRuleException	Business rule exception.	Static part of logging me	CONSTANT	SYSTEM_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
LogMessage_ApplicationException	System exception.	Static part of logging me	CONSTANT	SYSTEM_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
System1_URL	https://acme-test.uipath.com		SETTING	USER_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
Credential_System1	Credential_System1		SETTING	USER_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
Process_Names_List	chrome		SETTING	USER_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
System1_Data_Object	work-items		SETTING	USER_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
System1_Data_Filter	Type='WI1' AND Status = 'Open' AND De		SETTING	USER_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
System1_Data_In_Folder	C:\UiPath_Asset_Folder\stage		SETTING	USER_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
System1_Data_Archive_Folder	C:\UiPath_Asset_Folder\archive		SETTING	USER_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
System1_Folder_List	C:\UiPath_Asset_Folder\stage,C:\UiPath		SETTING	USER_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
RPA_ENVIRONMENT_PREFIX	Thanos-DEV		SETTING	USER_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
EMAIL_SUBJECT_FAILURE_PREFIX	FAILED!		SETTING	USER_DEFINED	Thanos	REF_DB_Accounts_DataExtractor

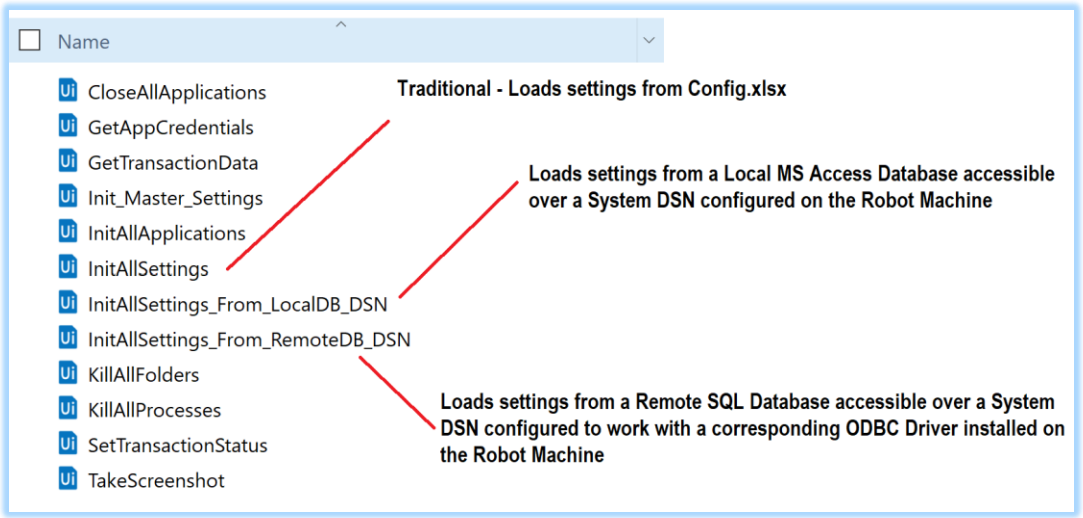
Record: 1 of 33 No Filter Search

NOTE

The steps to point the template to the Configuration database will be described in the later sections of this document.

Framework components to access the View

The advanced framework comes with a couple of additional sequence components to access the view and load the configuration settings. These components have been added to the **Framework** folder and will not require any modifications unless you plan to customize access to the database.



NOTE

Although identical, the components to access a Local and Remote databases have been provided separately to consider the possibility of how you might want to set up access to the Configuration database.

For Example: You may opt to configure a process to originally load settings from a local MS Access database via a System DSN set up on the Robot machine. However, when you migrate the configurations to a Remote database, you may opt to connect to that database via an OLE DB driver. In this case you may have to rewrite the component **InitAllSettings_From_RemoteDB_DSN.xaml** to build an OLEDB connection string in memory before connecting to the configuration database.

Connecting to the Configuration Database

In the interest of reducing complexity and risks of managing connection strings, ODBC Data Source Names (DSNs) will be the default and preferred method to gain access to the configuration database from the Robot machines.

Justifying the use of ODBC DSNs

DSN connection strings are source agnostic – the client will need to know only the DSN name without having any knowledge about the actual database.

DSNs minimize or eliminate the need to manage passwords and lengthy connection strings – a connection string will expose details such as server names, database names and usernames to the connecting clients

If passwords are stored as *SecureString* in an Orchestrator credential asset, there is no easy way to decrypt the *SecureString* password at runtime and integrate it into a connection string – not all connection providers have an easy way to do so.

The other option is to store passwords using standard encryption methods – however, a decryption routine can most likely expose a password in plaintext format when the code is being tested in a debugger. Besides, if encryption standards change, then this code will require to be updated!

In most enterprise environments, system administrators or DBAs will own the responsibility of managing DSNs and therefore absolving developers from directly having knowledge of any username or passwords to access a production database server.

Finally, a DSN will abstract the RPA process from an outright change in the underlying database technology – for example, if the configuration database is moved from SQL Server to Oracle, the

process will have no idea that this change occurred as long as the DSN name and the interface view structure remain unchanged.

To be fair, some flip sides to ODBC DSNs

ODBC DSN set up is natively integrated into the Windows platform – other platforms have no concept of setting up a DSN in the same manner as it's set up on a Windows machine.

Most commercial setups use JDBC drivers to connect to databases. This is as easy as copying a JDBC library file into their applications without having to perform a formal driver installation on the Robot machine.

DSNs will need formal installation of ODBC drivers suited to the target database platform – *For example*, the MySQL ODBC driver or the PostgreSQL ODBC driver will need to be installed on each Robot machine based on the target database this machine is going to connect to. If there are several Robot machines, all of them will be forced to go through this tedious installation process!

Conclusion

If other attributes of the enterprise architecture (*database products, servers etc.*) remain largely constant over a span of time, the positives of DSNs will outweigh the negatives and therefore this template will use the DSN approach to set up access to the configuration databases.

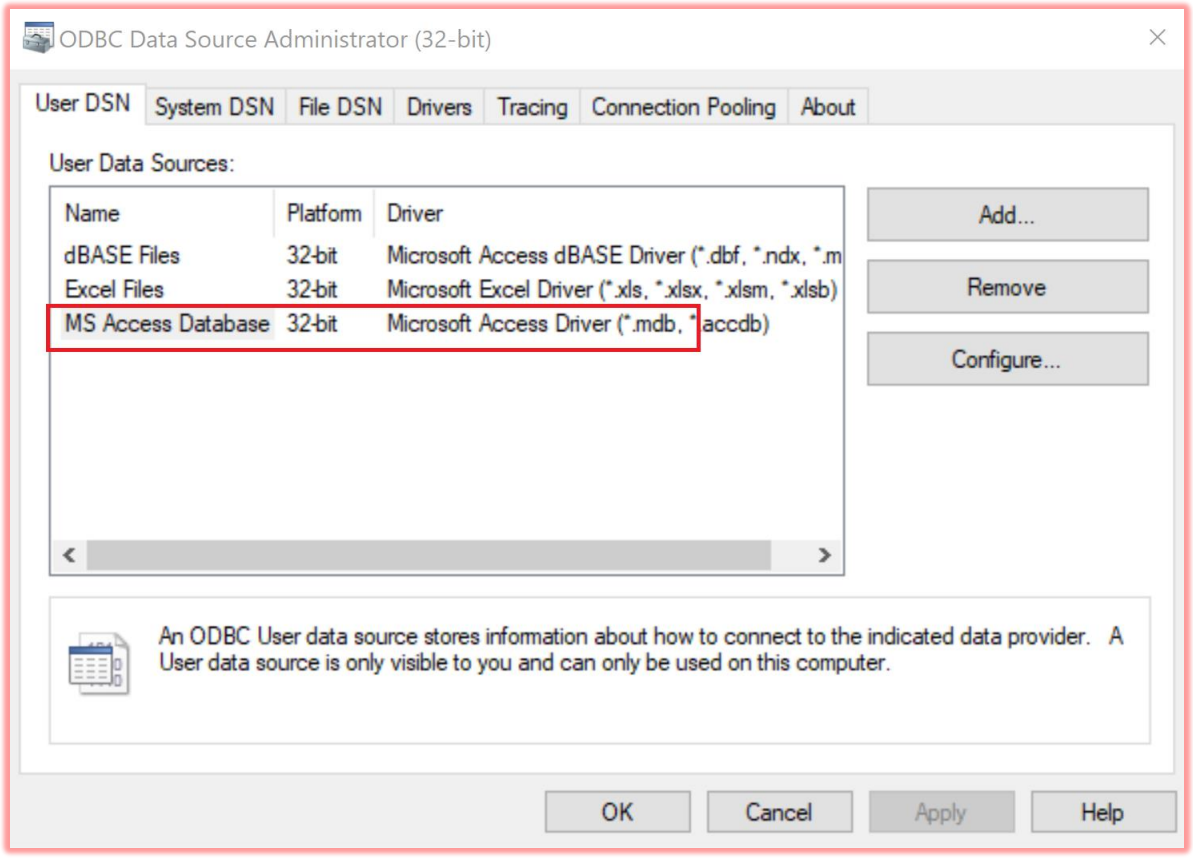
Setting up a Microsoft Access DSN on the Robot Machine

This template comes with a 2016 version of a Microsoft Access Database. The file name has the extension **.accdb** unlike previous versions that have the **.mdb** file extension.

Depending on which version (*32 or 64 bit*) of Office you have installed, you may have to download and run *AccessDatabaseEngine.exe* or *AccessDatabaseEngine_X64.exe* on the Robot machine. You can install only one of them on a host machine!

NOTE

If you have both versions of office installed, then the 32-bit version will not allow you to connect to Access because the 64-bit office is installed, and vice-versa. This is a catch-22 situation and the easiest workaround would be to save the MS Access database as a **.mdb** file and use the default MS Access driver that is provided in the 32-bit ODBC Manager in Windows.



Locking the MS Access Database

The MS Access database in this template is by default locked with a password. The password is:

ChangeYourPassword!

It's recommended that you open the database in Exclusive Mode, remove this password and lock the database with a new one before configuring the DSN.

Centralizing the MS Access Database File

The DSN for MS Access points directly to the location of the database file. Therefore, it isn't a good idea to keep the database under the **Data** folder of the RPA Process. Keeping the Access database under the **Data** folder will mean that you will need to have a separate DSN for each application. This could soon become as painful as managing a separate Config.xlsx file for each RPA process!

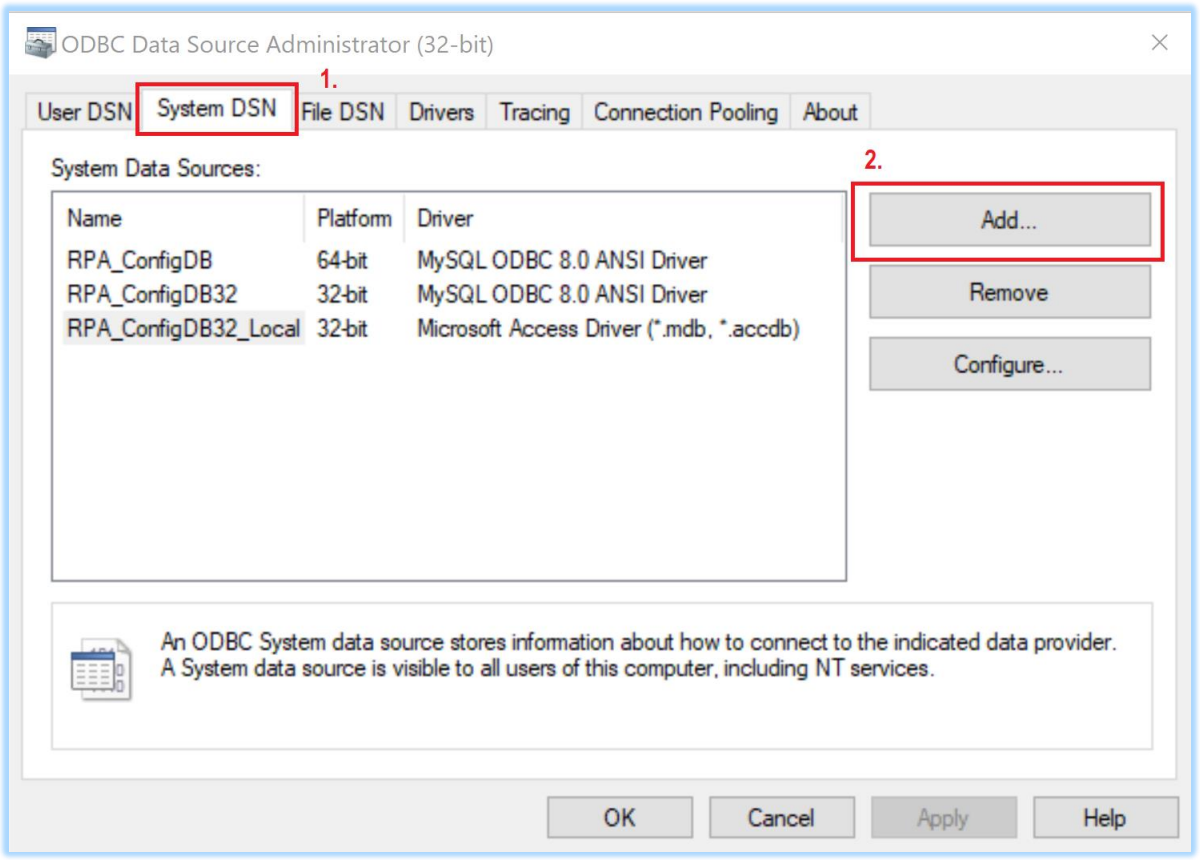
A better solution would be to save the database file to a central location on the Robot machine before configuring the DSN.

This has two advantages:

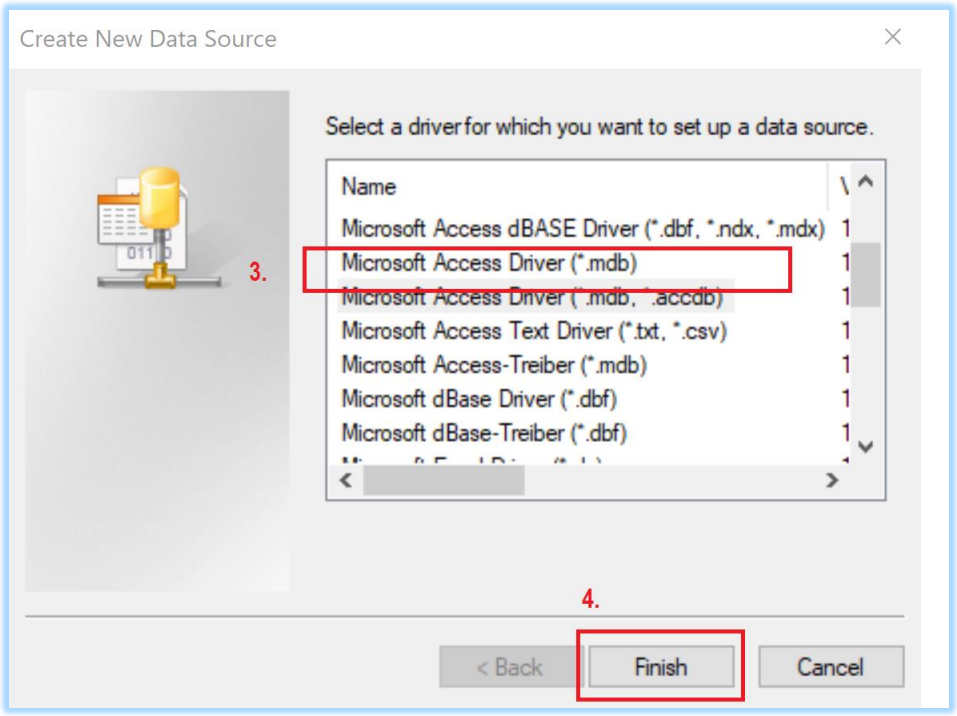
1. Several RPA processes can make use of a single, centrally managed database
2. The file location will remain constant, and only one DSN can be used by multiple processes to load their configuration settings

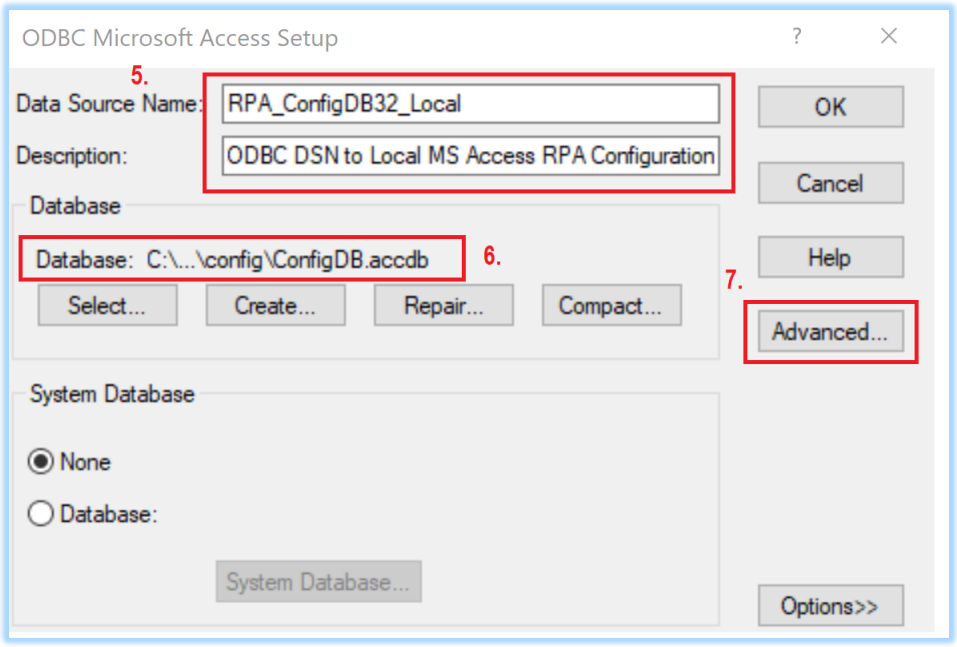
Configuring the DSN is a pretty simple process – make sure that you point to the database, and then configure the Username as **Admin** (default) and the password you used to lockdown the database with. Give the DSN a meaningful name for ease of use in the next steps.

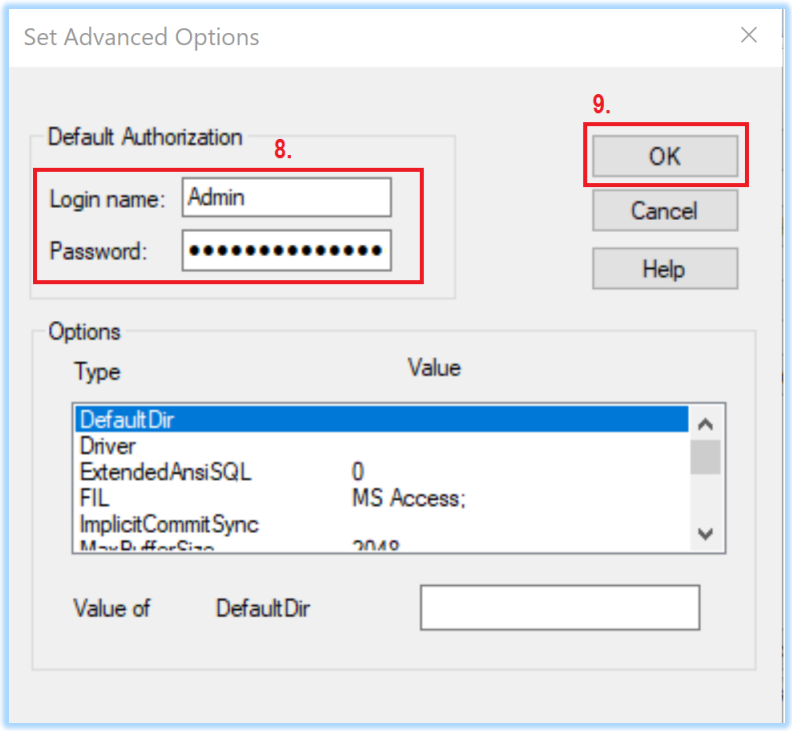
Follow steps **1 through 9** to set up the DSN as shown below. Note down the name of the DSN as we will be needing it in when setting up the *Master configuration* file later during the framework configuration.



© Andy Menon

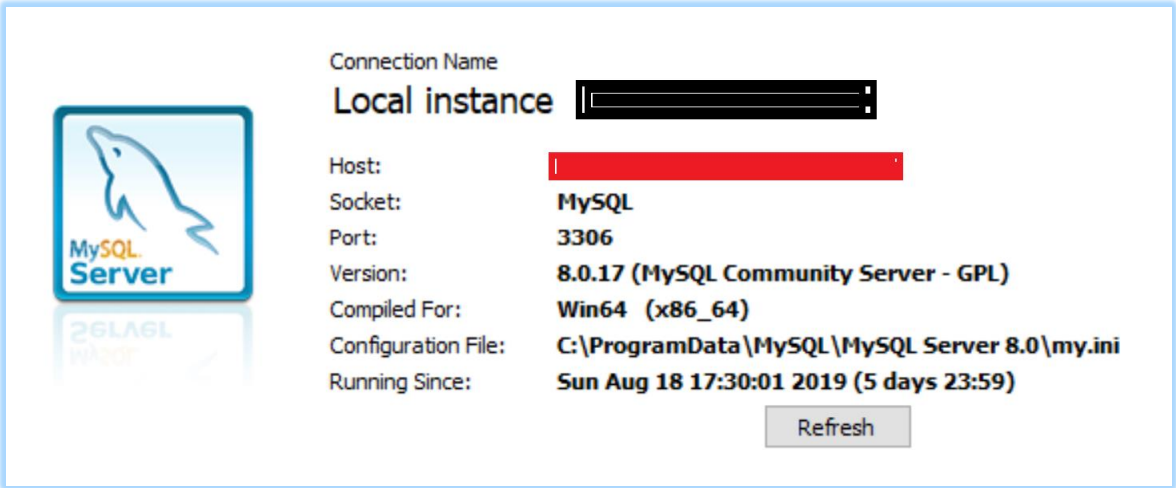

Create New Data Source
Select a driver for which you want to set up a data source.
Name
Microsoft Access dBASE Driver (*.dbf, *.ndx, *.mdx)
Microsoft Access Driver (*.mdb)
Microsoft Access Text Driver (*.txt, *.csv)
Microsoft Access-Treiber (*.mdb)
Microsoft dBase Driver (*.dbf)
Microsoft dBase-Treiber (*.dbf)
3.
4.
< Back
Finish
Cancel

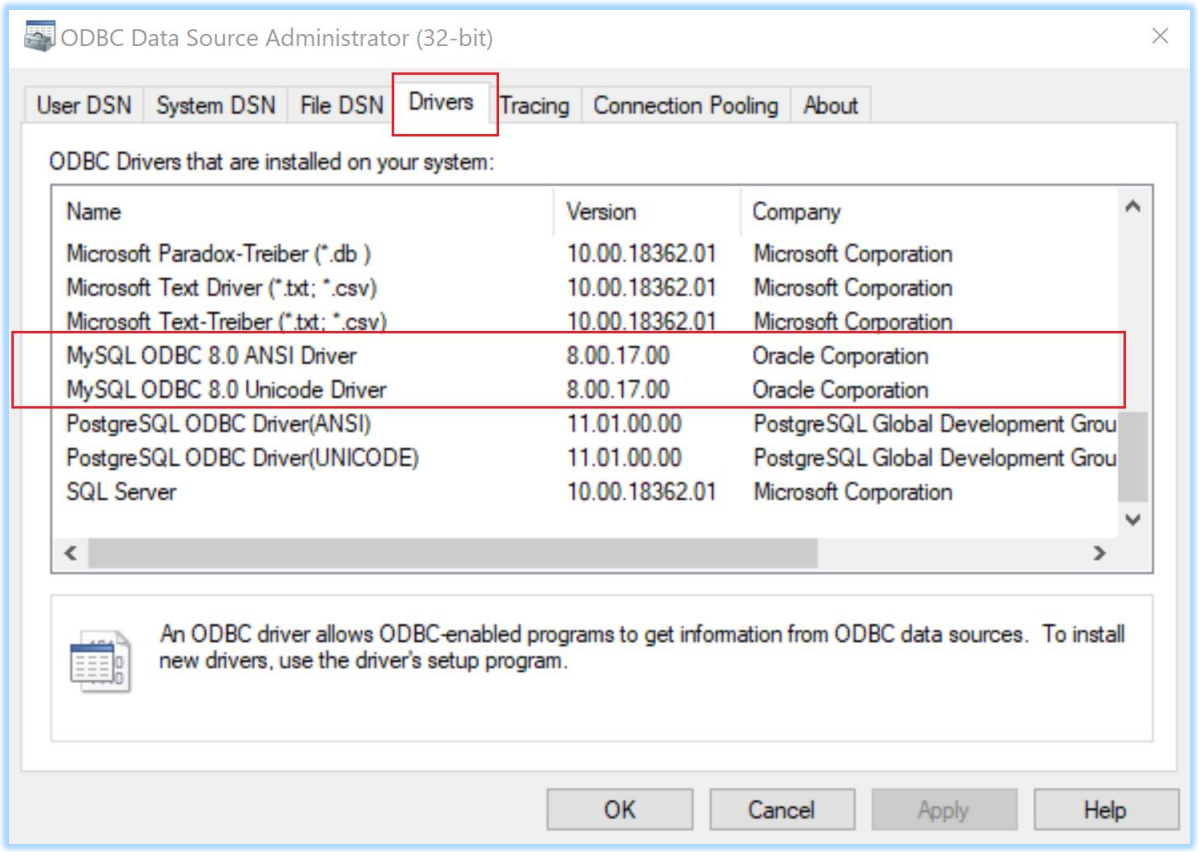

ODBC Microsoft Access Setup
5.
Data Source Name:
RPA_ConfigDB32_Local
Description:
ODBC DSN to Local MS Access RPA Configuration
OK
Cancel
Help
Advanced...
Database
Database: C:\...\config\ConfigDB.accdb
6.
7.
Select...
Create...
Repair...
Compact...
System Database
None
Database:
System Database...
Options>>



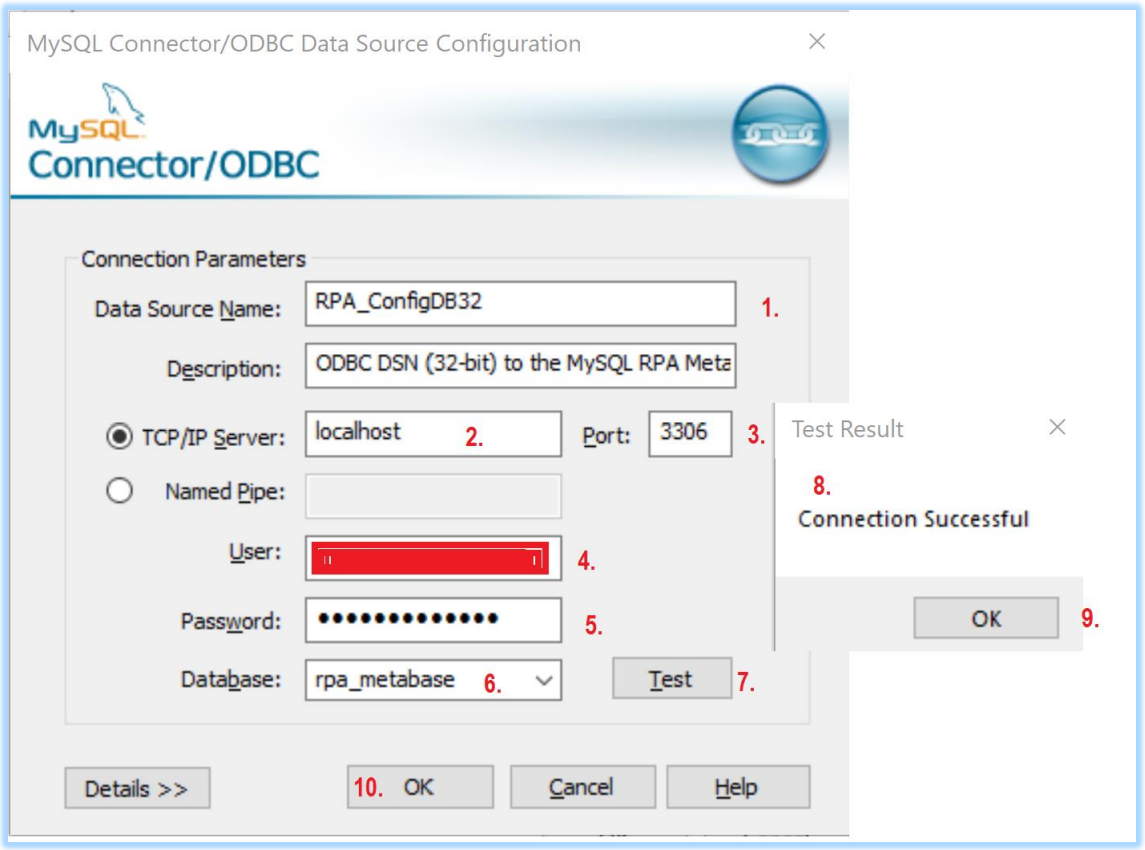
Setting up a Remote Database DSN on the Robot Machine

We refer to this as a Remote Database DSN in general because the database might be any commercial, enterprise grade SQL server such as MySQL or SQL Server, just to name a few. In this example, I will be using the MySQL v8.0.x Community Edition. Note that it's easier to install the ODBC drivers whilst installing the Server as it will be made available automatically in the ODBC Manager after the install.





Configuring the DSN for a remote DSN is similar to the previous series of steps, except that it has to be done in an ODBC connector box that's slightly different depending on which database you're connecting to. Make sure that you test the connection using the **Test** button. As usual is the case, note down the DSN name for use in the next steps.

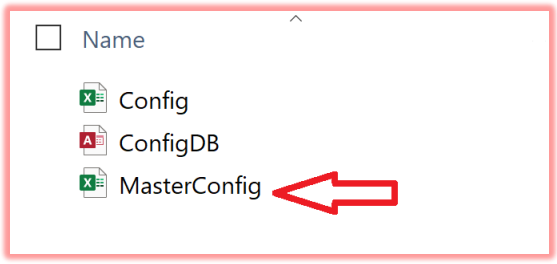


CAUTION

Do not use the default root user account for Production deployments!

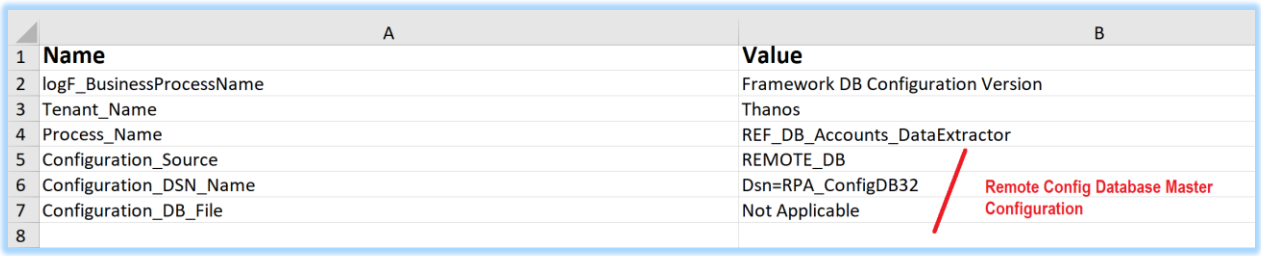
The MasterConfig.xlsx Framework Configuration File

A new addition to this Advanced Template is the Master Configuration file **MasterConfig.xlsx** file under the **Data** folder.



Note that, this file is a mandatory addition and not a replacement for Config.xlsx.

This file will need to be configured for each RPA Process, and the settings are few and simple. Here below is a typical master configuration set up if your Configuration database is a remote SQL database:



Master Configuration Key	Description
logF_BusinessProcessName	Legacy | Set to the value of Process_Name
Tenant_Name	The Tenant name from the *Tenant* database table
Process_Name	The RPA Process from the *Process* database table deployed to the above tenant
Configuration_Source	**REMOTE_DB** if configuration database is a remote SQL database **LOCAL_DB** if configuration is a local MS Access database **CONFIG_FILE** if using Config.xlsx configuration file
Configuration_DSN_Name	*Required only if you're configuring settings in a database. Otherwise, leave blank!* This is the name of the DSN configured in the previous section. The format of the connection string must be: **Dsn=Name_Of_Your_DSN** No quotes or blank spaces!
Configuration_DB_File	*Required if you're using a Local MS Access Database or the Config.xlsx file . Blank otherwise.*
Enable_DB_Configuration	Defaults to No (N). By default the configuration reads from the Config.xlsx file. You must explicitly enable the template to read from a database once the connection has been set up and tested.

Here's how the Master configuration is set up if you're using a configuration database:

Here's how the Master configuration is set up if you're using a Local MS Access Database:

Configuration_Source	LOCAL_DB
Configuration_DSN_Name	Dsn=RPA_ConfigDB32_Local
Configuration_DB_File	Data\ConfigDB.accdb

And this is how the Master configuration will change if you're using the traditional Config.xlsx file:

Enable_DB_Configuration	N

Configuration_Source	CONFIG_FILE
Configuration_DSN_Name	
Configuration_DB_File	Data\ConfigDB.xlsx

In this case you will need to add all your settings to the Config.xlsx just as in the traditional Framework, and not in the configuration database. Also, the **Enable_DB_Configuration** attribute must be set to N.

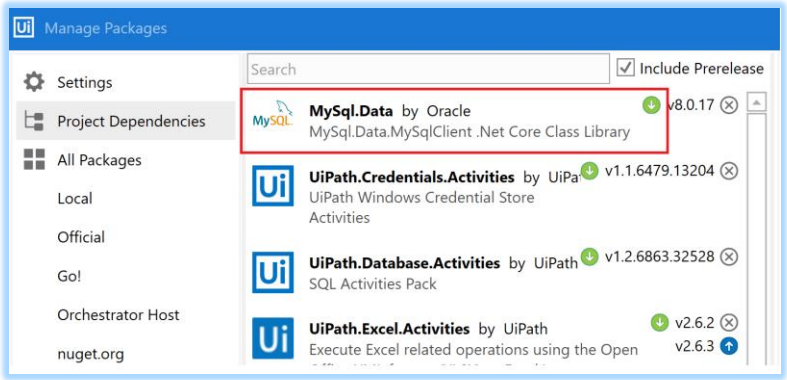
NOTE

If you are not using Config.xlsx, keep it empty but do not remove it from the Data folder!

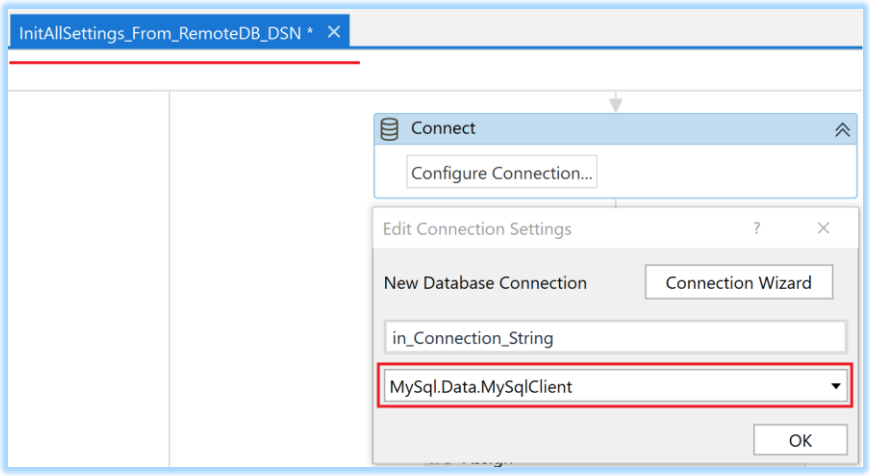
Importing Additional Database Libraries

In case you are not going to use a DSN to configure a connection to the database, then you may need to import additional dependencies into your RPA Process based on which database you plan to connect to.

Here's an example of importing the SQL Client libraries for the MySQL database into the RPA Process.



The approach here is to connect to the MySQL database by using the SQL Client library and a connection string. In this case, you will have to rewrite the **InitAllSettings_From_RemoteDB_DSN.xaml** component to connect to MySQL using the SQL Client library (*as shown below*). Or, replace this component with your own.



The *in_Connection_String* parameter in this case will change from a simple DSN name to a fully formed connection string.

Configuring a New RPA Process Flow in the Configuration Database

The following steps are applicable to the local MS Access database or the remote database. If you have the data tables configured identically, these steps should work without undue problems. The following example queries are modelled after the MySQL database. You will have to change them accordingly to suit your setup.

Create a Tenant Entry in the Tenant Table

```
/*-----------------------------------------------------------------*/
/* Make RPA Tenat Entry in the Tenant table if not already existing */
/* Set active_flg to True to activate Tenant
/*-----------------------------------------------------------------*/
insert into rpa_metabase.tenant
(tenant_nm,tenant_description, active_flg)
```

```
values
('Thanos', 'The Default Thanos Win10 SOHO Tenant',    1);
```

Create an RPA Process Entry in the Process Table

```
/*-----------------------------------------------------------------*/
/* Make RPA Process Entry for Application # 1 */
/* Activate Process by setting Active_Flg to True */
/*-----------------------------------------------------------------*/
insert into process(Process_Name, Process_Description, active_flg)
values
('REF_DB_Accounts_DataExtractor',
'Extracts Work Items of type WI1 and description "Verify Account Position"
from ACME System1 into an Excel File',
1);
```

Copy Standard Configuration Settings into the Config Table

```
/*-----------------------------------------------------------------------*/
/* Copy Standard System-defined settings to your process                 */
/* Update settings as applicable to the current RPA Process
/*-----------------------------------------------------------------------*/
insert into  config(
    Setting_Name, Setting_Value, Setting_Description,
    Tenant_ID, Setting_Type, Setting_Category_ID,
    active_flg, Process_ID
)
select
Setting_Name, Setting_Value, Setting_Description,
Tenant_ID, Setting_Type, Setting_Category_ID,
active_flg,
(select ID from Process where Process_Name = 'REF_DB_Accounts_DataExtractor')
Process_ID
```

```
from config where

/* settings from default/standard project  */

Process_ID = (select ID from Process where Process_Name =
'FRAMEWORK_DEFAULT_SETTINGS')

/* deployed to your tenant */

and

Tenant_ID = (select Tenant_ID from tenant where Tenant_Nm = 'Thanos')

/* All system defined standard settings */

and

Setting_Category_ID = (select Setting_Category_ID from setting_category where
Setting_Category='SYSTEM_DEFINED')

order by ID

;

/* Update Application Name */

update config

set Setting_Value = 'REF_DB_Accounts_DataExtractor'

where Setting_Name = 'logF_BusinessProcessName'

and Tenant_ID = (select Tenant_ID from tenant where Tenant_Nm = 'Thanos')

;

/* If Orchestrator is not applicable, update entry to indicate so */

update config

set Setting_Value = 'Orch_Not_Applicable'

where Setting_Name = 'OrchestratorQueueName'

and Tenant_ID = (select Tenant_ID from tenant where Tenant_Nm = 'Thanos')

and Setting_Value = 'KibanaDemoQueue'

and Process_Id = (select ID from Process where Process_Name =
'REF_DB_Accounts_DataExtractor')

;

/* If no Orchestrator, set MaxRetryNumber to a Non-zero value */

update config
```

```
set Setting_Value = '2'

where Setting_Name = 'MaxRetryNumber'

and Tenant_ID = (select Tenant_ID from tenant where Tenant_Nm = 'Thanos')

and Process_Id = (select ID from Process where Process_Name =
'REF_DB_Accounts_DataExtractor')

and Setting_Value = '0';
```

Insert Process-specific Settings applicable to the Current Process

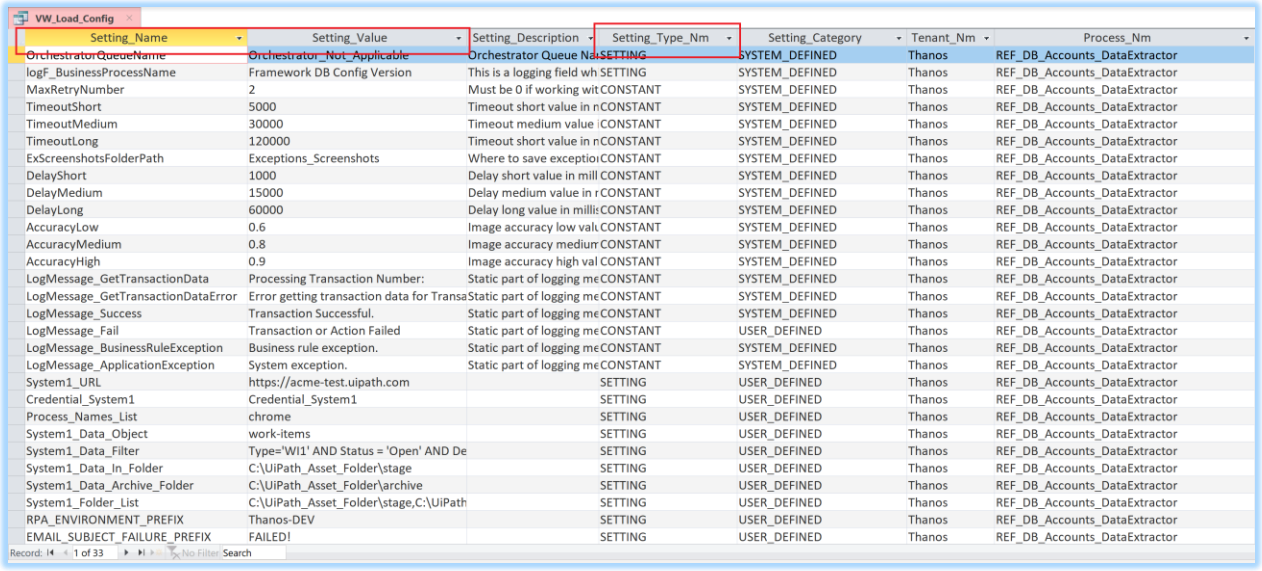
This section depends on the process you're trying to build. An example would be an application URL that the Robot must navigate to:

```
insert into  config(

    Setting_Name, Setting_Value, Setting_Description,

    Tenant_ID,  Setting_Type, Setting_Category_ID,

    active_flg,  Process_ID )

select

'System1_URL' Setting_Name,

'https://acme-test.uipath.com' Setting_Value,

'The System URL that is the target of the RPA Process' as
Setting_Description,

(select Tenant_ID from tenant where Tenant_Nm = 'Thanos') as Tenant_ID,

(select Setting_Type_ID from setting_type where Setting_Type_Nm = 'SETTING')
AS Setting_Type,

(select Setting_Category_ID from setting_category where
Setting_Category='USER_DEFINED')    Setting_Category_ID,

1 AS  active_flg,

(select ID from Process where Process_Name = 'REF_DB_Accounts_DataExtractor')
AS Process_ID;
```

Configure as many settings your RPA Process will need to complete the Database configuration step.

Finally, Test the Interface View

```
SELECT * FROM VW_Load_Config
WHERE Tenant_Nm = 'Thanos' AND Process_Nm  = 'REF_DB_Accounts_DataExtractor';
```



VW_Load_Config

Setting_Name	Setting_Value	Setting_Description	Setting_Type_Nm	Setting_Category	Tenant_Nm	Process_Nm
~~OrchestratorQueueName~~	~~Orchestrator_Not_Applicable~~	Orchestrator Queue Na	SETTING	SYSTEM_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
logF_BusinessProcessName	Framework DB Config Version	This is a logging field wh	SETTING	SYSTEM_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
MaxRetryNumber	2	Must be 0 if working wit	CONSTANT	SYSTEM_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
TimeoutShort	5000	Timeout short value in n	CONSTANT	SYSTEM_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
TimeoutMedium	30000	Timeout medium value i	CONSTANT	SYSTEM_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
TimeoutLong	120000	Timeout short value in n	CONSTANT	SYSTEM_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
ExScreenshotsFolderPath	Exceptions_Screenshots	Where to save exceptio	CONSTANT	SYSTEM_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
DelayShort	1000	Delay short value in mill	CONSTANT	SYSTEM_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
DelayMedium	15000	Delay medium value in r	CONSTANT	SYSTEM_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
DelayLong	60000	Delay long value in millis	CONSTANT	SYSTEM_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
AccuracyLow	0.6	Image accuracy low val	CONSTANT	SYSTEM_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
AccuracyMedium	0.8	Image accuracy mediur	CONSTANT	SYSTEM_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
AccuracyHigh	0.9	Image accuracy high val	CONSTANT	SYSTEM_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
LogMessage_GetTransactionData	Processing Transaction Number:	Static part of logging me	CONSTANT	SYSTEM_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
LogMessage_GetTransactionDataError	Error getting transaction data for Transa	Static part of logging me	CONSTANT	SYSTEM_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
LogMessage_Success	Transaction Successful.	Static part of logging me	CONSTANT	SYSTEM_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
LogMessage_Fail	Transaction or Action Failed	Static part of logging me	CONSTANT	USER_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
LogMessage_BusinessRuleException	Business rule exception.	Static part of logging me	CONSTANT	SYSTEM_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
LogMessage_ApplicationException	System exception.	Static part of logging me	CONSTANT	SYSTEM_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
System1_URL	https://acme-test.uipath.com		SETTING	USER_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
Credential_System1	Credential_System1		SETTING	USER_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
Process_Names_List	chrome		SETTING	USER_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
System1_Data_Object	work-items		SETTING	USER_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
System1_Data_Filter	Type='WI1' AND Status = 'Open' AND De		SETTING	USER_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
System1_Data_In_Folder	C:\UiPath_Asset_Folder\stage		SETTING	USER_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
System1_Data_Archive_Folder	C:\UiPath_Asset_Folder\archive		SETTING	USER_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
System1_Folder_List	C:\UiPath_Asset_Folder\stage,C:\UiPath		SETTING	USER_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
RPA_ENVIRONMENT_PREFIX	Thanos-DEV		SETTING	USER_DEFINED	Thanos	REF_DB_Accounts_DataExtractor
EMAIL_SUBJECT_FAILURE_PREFIX	FAILED!		SETTING	USER_DEFINED	Thanos	REF_DB_Accounts_DataExtractor

Record: 1 of 33 No Filter Search

All the system-defined settings and user-defined settings for the specified Tenant and Process names must be visible in the view. If you don't see all of them make sure that the active_flg is set to true in the Config, Process and the Tenant table. Also make sure that the Tenant and Process IDs are correctly associated with each of the entries in the Config table.

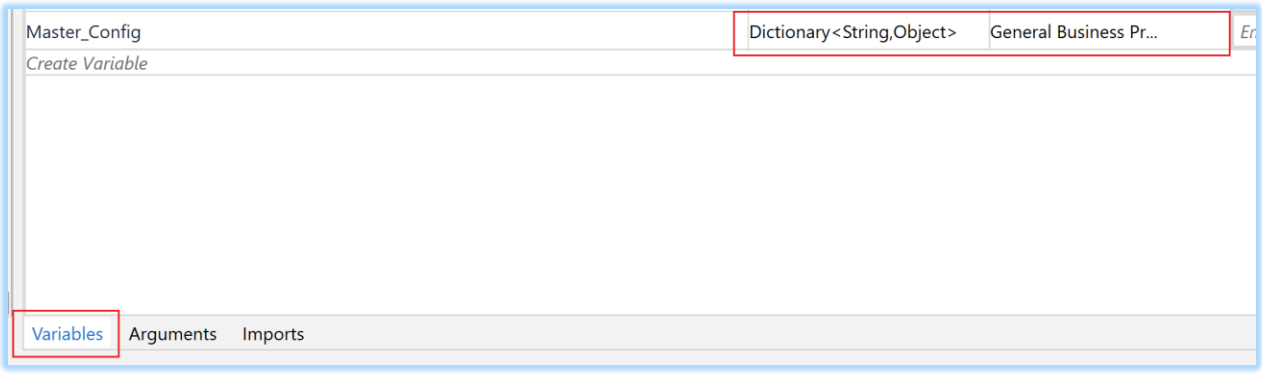
On a similar note, if you want to prevent a setting from loading up, set the active_flag to false in the config table for that Tenant and Process.

Configuring a New RPA Process Flow

Before you proceed, you will need to complete the Database set up and create all the entries for the process settings in the database as described in the previous sections. You will need to be familiar with the original UiPath component to continue. The UiPath training and documentation covers the configuration of the original template in detail. Therefore, this document covers only the key differences and changes in how this Advanced Template is different from the original. in order to proceed further.

Main.xaml

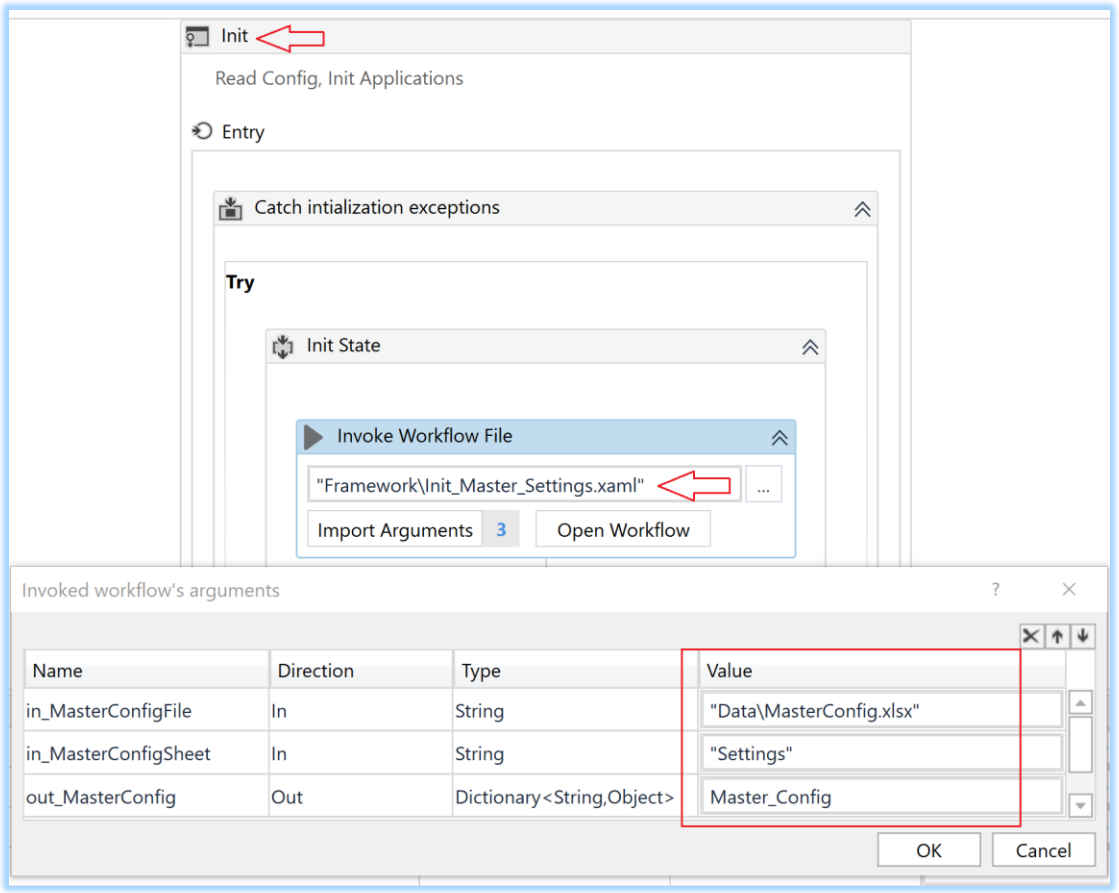
Visually, there isn't much change. However, there is one additional variable added to this component.



The configurations from the MasterConfig.xlsx will be loaded into the **Master_Config** dictionary collection in the **Init** state machine.

Init State Machine

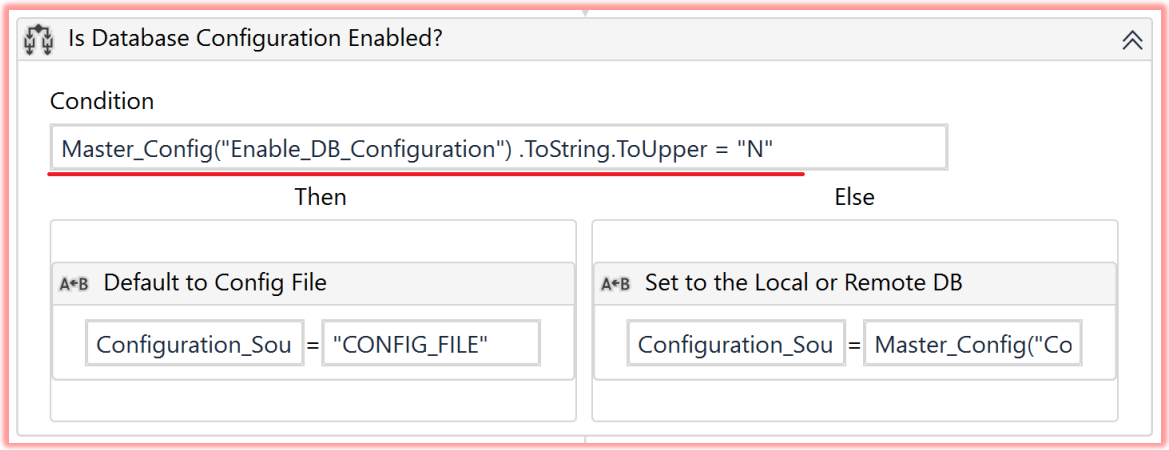
The Init State machine invokes the **Framework\Init_Master_Settings.xaml** sequence to load the configurations from the MasterConfig.xlsx and returns the Master_Config dictionary collection.



No changes required here other than to make sure that the arguments are being passed in correctly to this workflow.

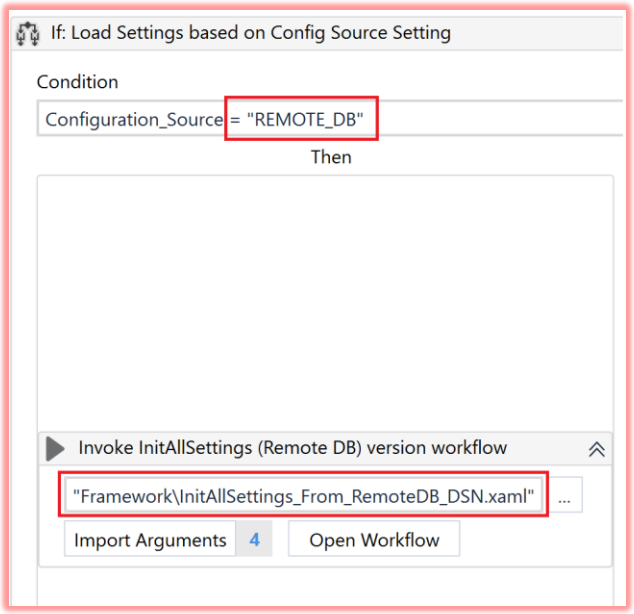
Loading Configuration Settings – Most Significant Change

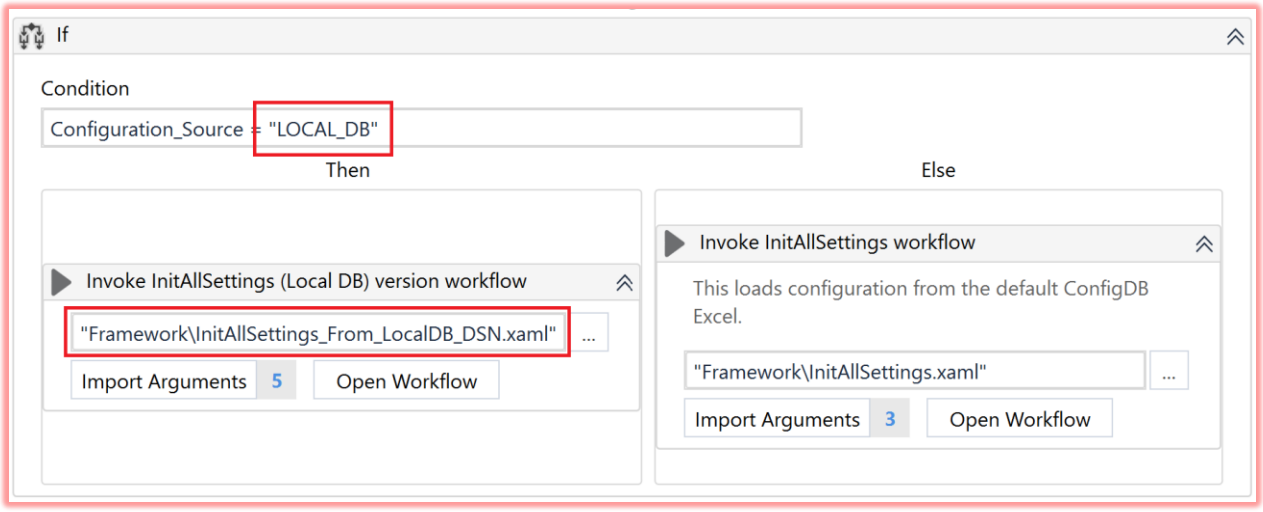
The sequence shown below has been extensively modified to load configuration settings based on how the MasterConfig.xlsx has been configured. The following screen shots show the extent of the modification.



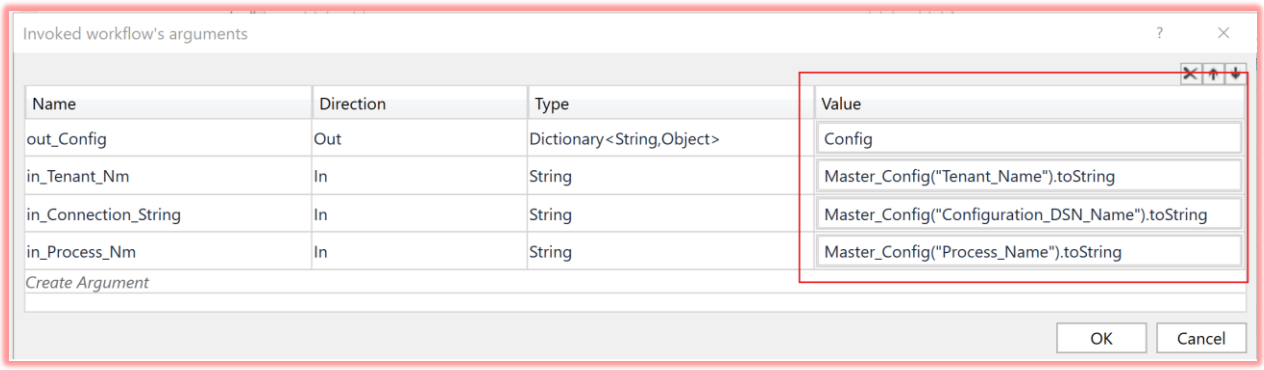
NOTE

The InitAllSettings_From_RemoteDB_DSN.xaml and InitAllSettings_From_LocalDB_DSN.xaml will be invoked only if the master setting Enable_DB_Configuration is enabled and if a DSN has been successfully configured to connect to a local or remote database from the Robot machine.





Note that the new components **InitAllSettings_From_RemoteDB_DSN.xaml** and **InitAllSettings_From_LocalDB_DSN.xaml** take the *DSN Name*, the *Tenant Name* and *Process Name* from the *Master_Config* Dictionary object as input arguments and return the standard Config dictionary as the output after loading their configurations settings from the database.



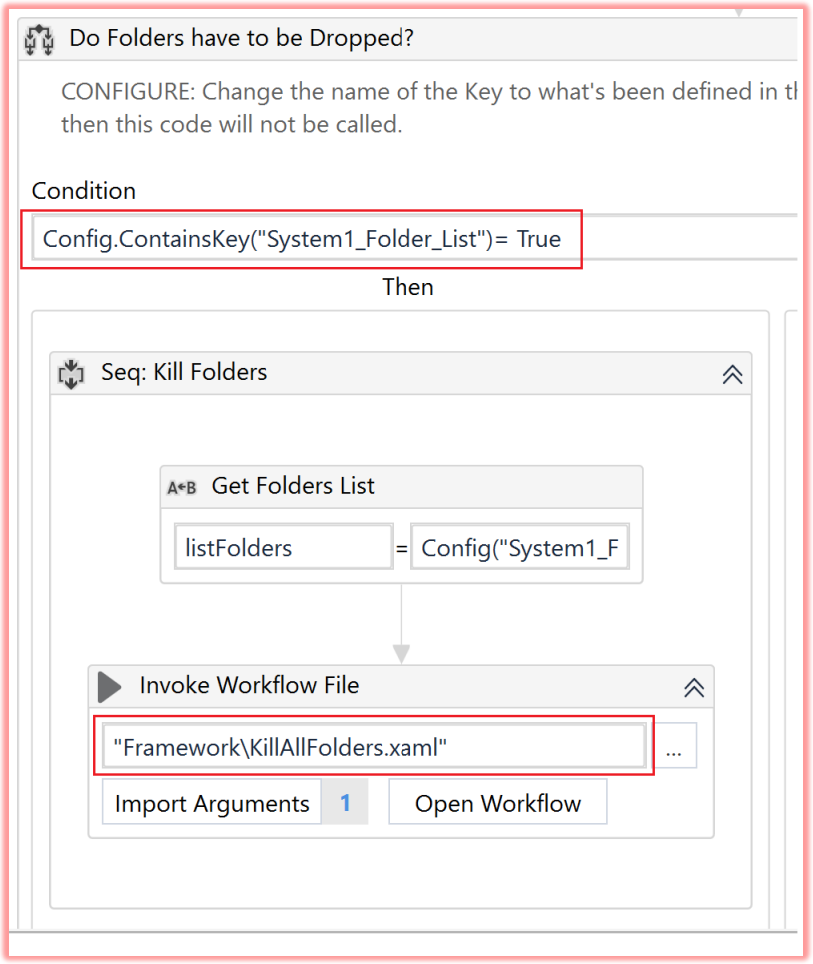
KillAllFolders.xaml - New Component to Drop Folders

Optionally, you can configure the framework to drop folders related to the process during the restart sequence.

For Example: In this process, a setting named *System1_Folder_List* has been configured as a User-defined setting. The value of this setting would be a comma-delimited list of folders that need to be dropped when the robot restarts like so:

```
C:\UiPath_Asset_Folder\stage,C:\UiPath_Asset_Folder\archive
```

You can configure this variable accordingly based on how you name it. It takes the value of the *System1_Folder_List* parameter and deletes all the folders. Note that this part of the code will run only if the setting is found in the Config dictionary!



Move_or_Delete_Directory.xaml

This new addition in the Common folder is invoked by *KillAllFolders.xaml* to move or drop the folders. If a folder contains files, all files are deleted before the folder can be dropped or moved.

The sequence contains documentation on how to use the component.

Move_or_Delete_File.xaml

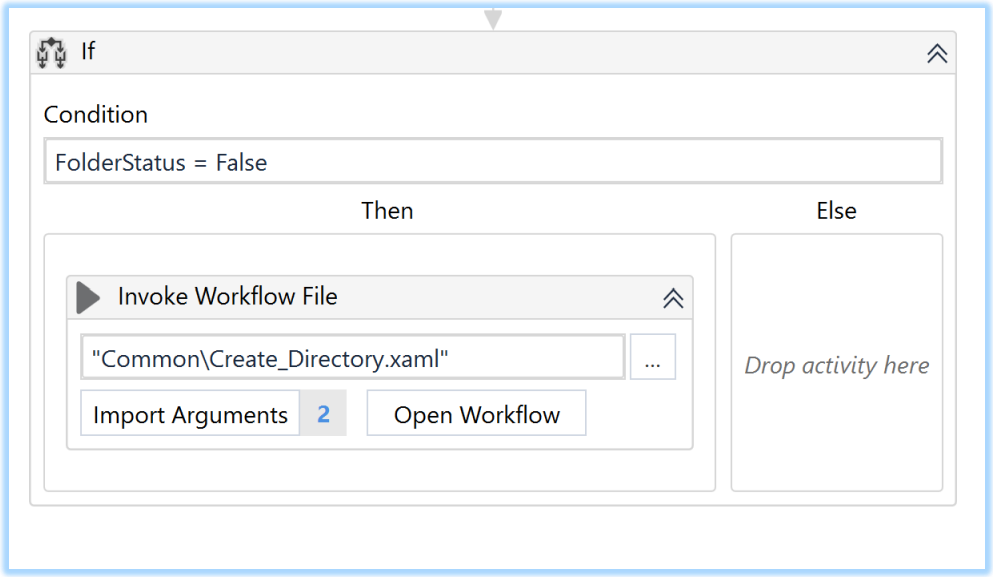
This is a component like the *Move_or_Delete_Directory.xaml* except that it moves or drops files. Documentation for this component is elaborated in the sequence.

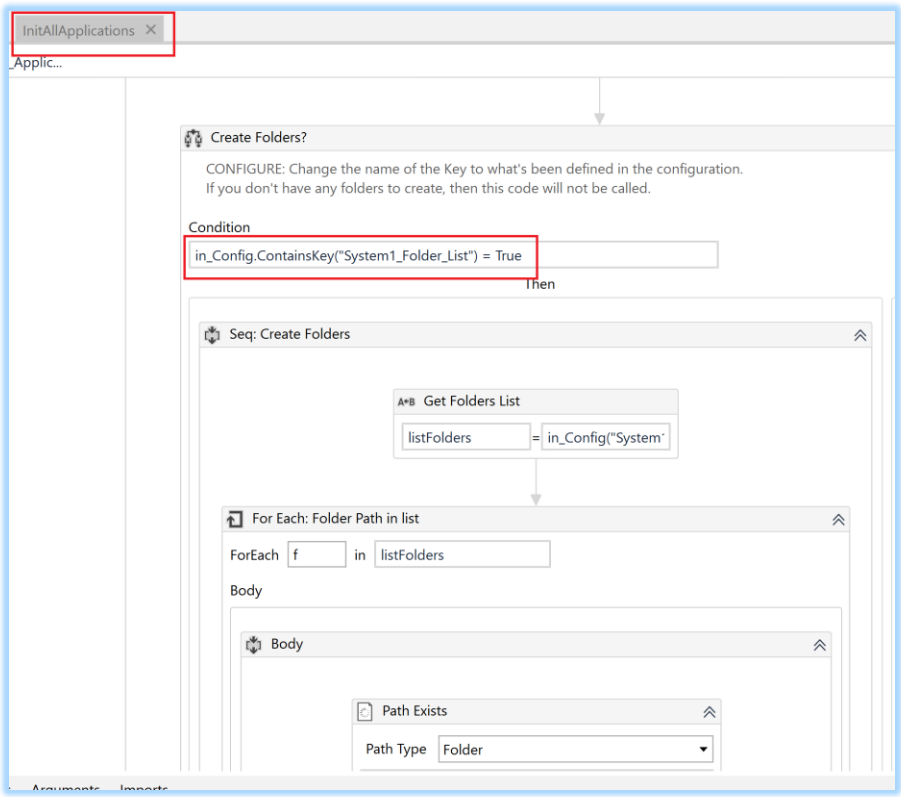
InitAllApplications.xaml

This standard component has been modified to create folders during the Robot startup. Again, this code runs only if a suitable parameter has been configured for the application.

Create_Directory.xaml - New Component to Create Folders

This component is integrated into the InitAllApplications.xaml to create folders during robot start up. Just like the KillAllFolders.xaml. this component takes the list of comma-delimited folder paths and creates the folders.





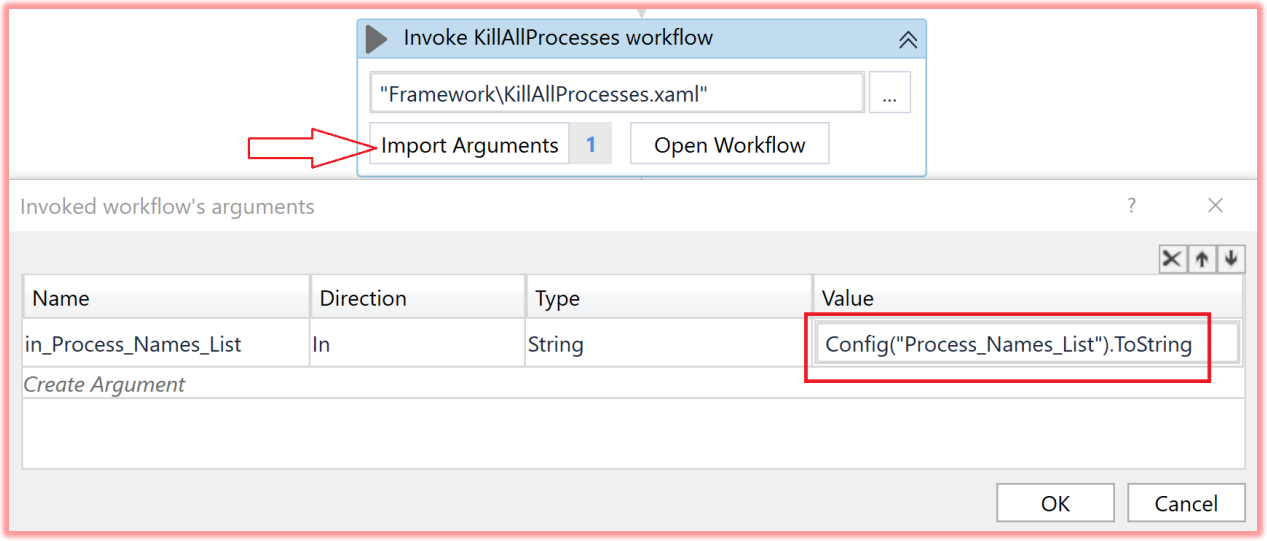
KillAllProcesses.xaml

This component has been modified to take one parameter. Just as the *KillAllFolders.xaml*, a parameter named *Process_Names_List* has been configured as a user-defined setting. It contains a comma-delimited list of all the processes that need to be killed during start up.

Example: If your RPA Process needs to kill Internet Explorer and Chrome, then the value of this argument would be

```
iexplore,chrome
```

If only one process needs to be killed, then only one process name can be configured without the delimiting comma.



Utility Component - Identify_Processes_By_Name.xaml

To help generate a list of process names to kill, a new utility component has been added to the *Common* folder.

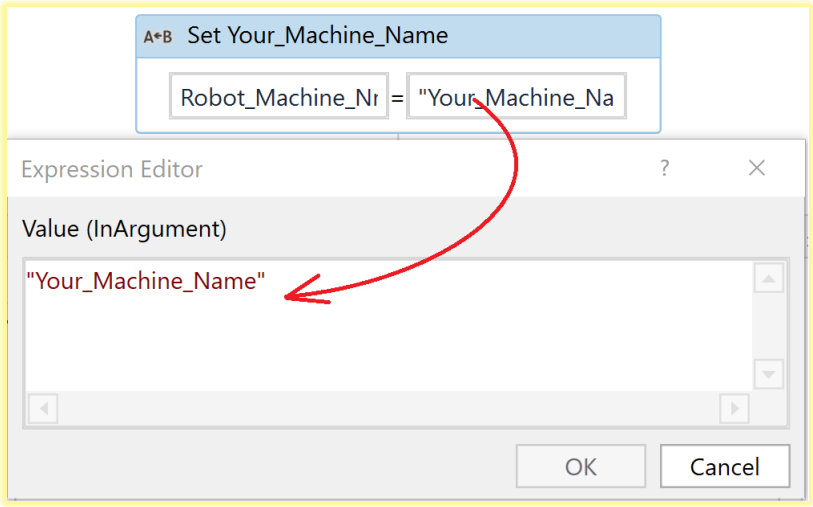
To generate the list of processes, follow these steps:

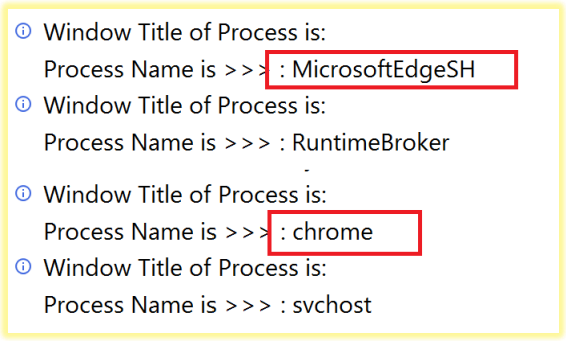
1. Open this component in Studio
2. Disable sequence named like *"Seq 2:"* if not already done
3. Enable sequence named like *"Seq 1:"* if not already enabled
4. Change the name of the machine to match your robot machine
5. Run the Sequence independently
6. Copy the names of the processes you want to kill from the Output tab

NOTE

The machine name can be obtained by going to *My Computer* > *Properties* on the windows machine.

The *System.Diagnostics* object does not support remote computer names- therefore do not use the "Full computer name", if the name of the computer is like *computername.domain.com*!





Configure the process names as a CSV list for the parameter *Process_Names_List* in the configuration database or the Config.xlsx file. From the above screen shot, this would be:

```
MicrosoftEdgeSH,chrome
```

NOTE

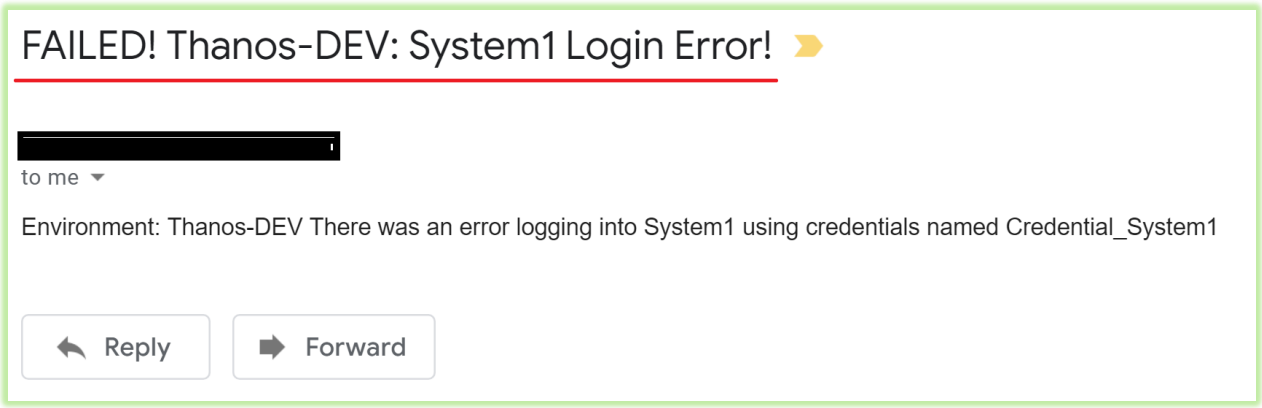
The processes you want to kill must be already running before running this utility - they will not show up on the process list otherwise.

Send_Email_Notification.xaml

This component has been modified to take several parameters for the purposes of sending emails notifications. These parameters are configured as user-defined settings in the configuration.

RPA_ENVIRONMENT_PREFIX	Thanos-DEV		USER_DEFINED
EMAIL_SUBJECT_FAILURE_PREFIX	FAILED!		USER_DEFINED
EMAIL_SUBJECT_SUCCESS_PREFIX	Success!		USER_DEFINED
Email_From	[redacted]		USER_DEFINED
Email_To	[redacted]		USER_DEFINED

The settings above will feature in an email such as the failure notification I receive each time there is a login failure:



The purpose of these settings is to provide quick hints in the email subject and body to help recipients to better respond to such notifications.

UiPath foundation training already covers the subject of configuring email activities in detail.

Validating the Project before Deployment

Before deploying the process to the Orchestrator, make sure to validate the project completely to make sure that the deployment does not fail.

You will most likely encounter this error.

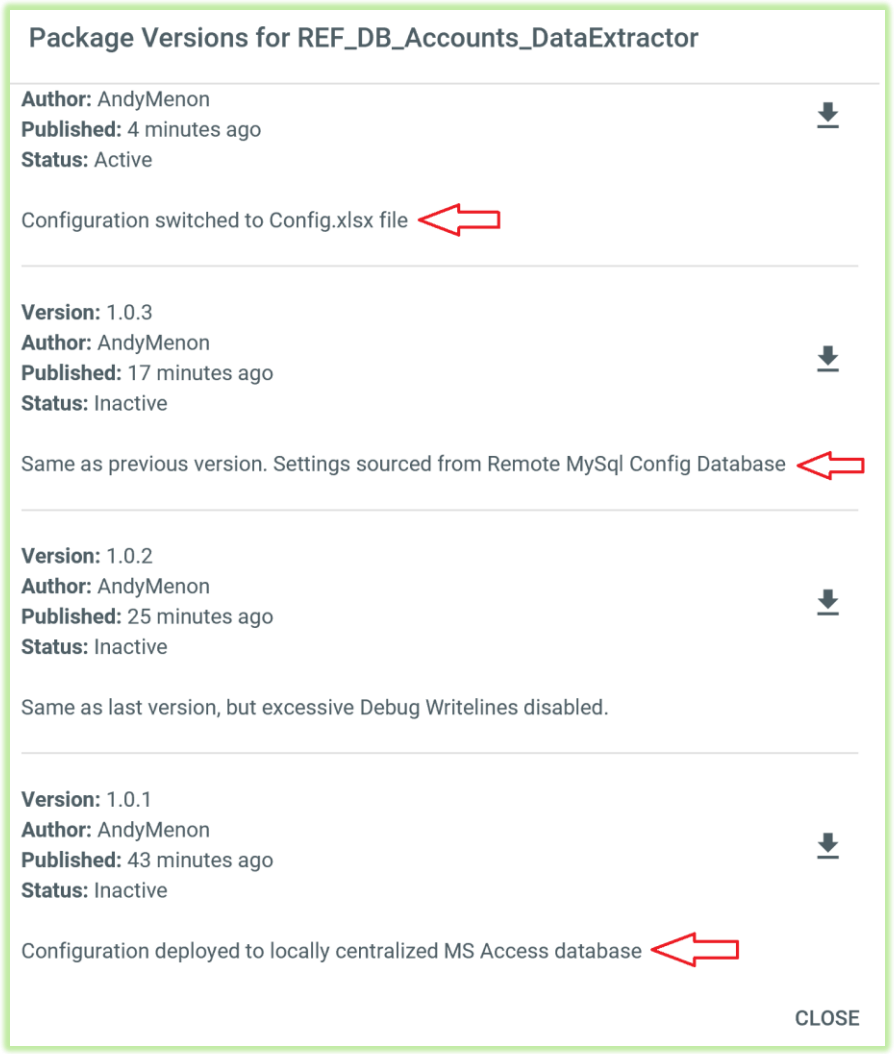
A variable named *Conn_MySqlDB* of type *MySqlConnection* will cause a validation failure in the **InitAllSettings_From_RemoteDB_DSN.xaml** component. That is because, it has reference to the MySql Connection library that was removed before this Template was published to the UiPath Go! Marketplace.

Solution: Open the XAML file **InitAllSettings_From_RemoteDB_DSN.xaml** and click the "Remove Unused variables" from the Menu bar. This will remove the orphaned variables.

Test Results and End Note

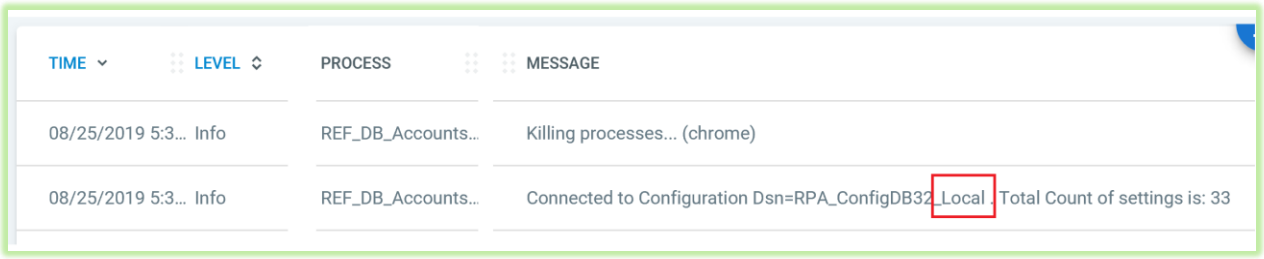
The Advanced Enterprise RPA Process Template has been tested by building a fully functional RPA Process. The configuration settings for this process were deployed to a locally centralized MS Access database, a Remote MySQL database and the traditional Config.xlsx database. The process was then tested by switching the options in the MasterConfig.xlsx file to consume settings from the local database, the remote database and the Excel config.

The screen shot below displays the version information for each of the tests:



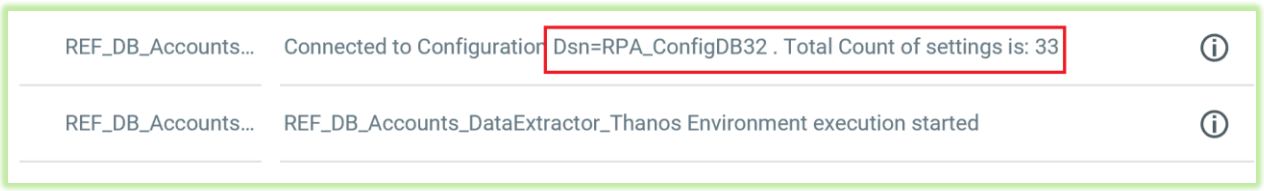
And the following screen shots indicate the fact that the settings were loaded from a different configuration source during each of the three tests.

Local MS Access Configuration:



TIME	LEVEL	PROCESS	MESSAGE
08/25/2019 5:3...	Info	REF_DB_Accounts...	Killing processes... (chrome)
08/25/2019 5:3...	Info	REF_DB_Accounts...	Connected to Configuration Dsn=RPA_ConfigDB32_Local . Total Count of settings is: 33

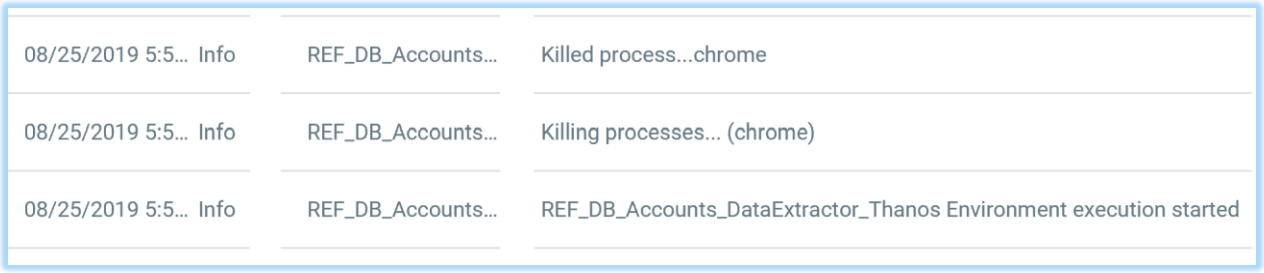
Remote MySQL Database Configuration:



REF_DB_Accounts...	Connected to Configuration Dsn=RPA_ConfigDB32 . Total Count of settings is: 33	ⓘ
REF_DB_Accounts...	REF_DB_Accounts_DataExtractor_Thanos Environment execution started	ⓘ

Traditional Config.xlsx Configuration:

Obviously, the log output for this configuration has no visible hints because the original component that loads the settings from the Excel has not been modified in any way.



08/25/2019 5:5...	Info	REF_DB_Accounts...	Killed process...chrome
08/25/2019 5:5...	Info	REF_DB_Accounts...	Killing processes... (chrome)
08/25/2019 5:5...	Info	REF_DB_Accounts...	REF_DB_Accounts_DataExtractor_Thanos Environment execution started

The process ran end-to-end without any incident when executed from the Orchestrator when switched between each of the three configurations .

End

Change Log

09.21.2019	Minor edits and typos corrected. New section titled *Validating the Project before* Deployment added	Andy Menon
08.30.2019	**Document renamed to:** Advanced Enterprise RPA Process Template - User Guide.docx **Major Content changes:** All references to the terms 'Robotic Enterprise Framework' and 'REFramework' removed! Any references to these terms are done via web links that will direct the user to the subject of this reference. New section titled *Importing Additional Database Libraries* added Instructions revised further for section titled: *Utility Component - Identify_Processes_By_Name.xaml* Section titled *The MasterConfig.xlsx Framework Configuration File* updated Section titled *Loading Configuration Settings – Most Significant Change* updated Table of Contents updated **This change log section added**	Andy Menon
08.26.2019	Minor typos corrected Instructions revised for section titled: *Utility Component - Identify_Processes_By_Name.xaml*	Andy Menon
08.25.2019	Original	Andy Menon

© Andy Menon