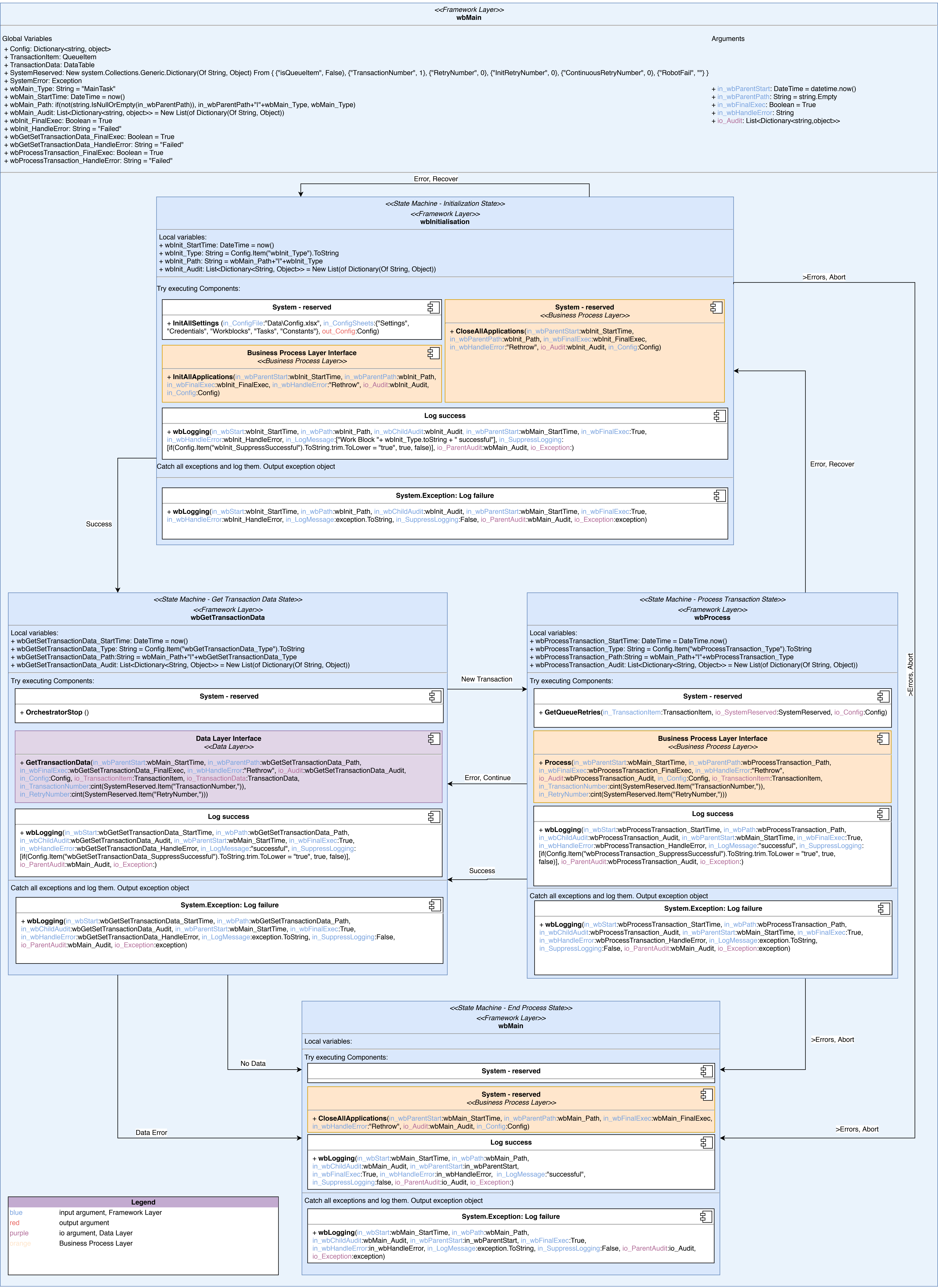
Flat view (UML)



<<Framework Layer>>
wbMain

Global Variables

+ Config: Dictionary<string, object>
+ TransactionItem: QueueItem
+ TransactionData: DataTable
+ SystemReserved: New system.Collections.Generic.Dictionary(Of String, Object) From { {"isQueueItem", False}, {"TransactionNumber", 1}, {"RetryNumber", 0}, {"InitRetryNumber", 0}, {"ContinuousRetryNumber", 0}, {"RobotFail", ""} }
+ SystemError: Exception
+ wbMain_Type: String = "MainTask"
+ wbMain_StartTime: DateTime = now()
+ wbMain_Path: if(not(string.IsNullOrEmpty(in_wbParentPath)), in_wbParentPath+"I"+wbMain_Type, wbMain_Type)
+ wbMain_Audit: List<Dictionary<string, object>> = New List(of Dictionary(Of String, Object))
+ wbInit_FinalExec: Boolean = True
+ wbInit_HandleError: String = "Failed"
+ wbGetSetTransactionData_FinalExec: Boolean = True
+ wbGetSetTransactionData_HandleError: String = "Failed"
+ wbProcessTransaction_FinalExec: Boolean = True
+ wbProcessTransaction_HandleError: String = "Failed"

Arguments

+ in_wbParentStart: DateTime = datetime.now()
+ in_wbParentPath: String = string.Empty
+ in_wbFinalExec: Boolean = True
+ in_wbHandleError: String
+ io_Audit: List<Dictionary<string,object>>

Error, Recover

<<State Machine - Initialization State>>
<<Framework Layer>>
wbInitialisation

Local variables:
+ wbInit_StartTime: DateTime = now()
+ wbInit_Type: String = Config.Item("wbInit_Type").ToString
+ wbInit_Path: String = wbMain_Path+"I"+wbInit_Type
+ wbInit_Audit: List<Dictionary<String, Object>> = New List(of Dictionary(Of String, Object))

Try executing Components:

System - reserved

+ **InitAllSettings** (in_ConfigFile:"Data\Config.xlsx", in_ConfigSheets:{"Settings", "Credentials", "Workblocks", "Tasks", "Constants"}, out_Config:Config)

System - reserved
<<Business Process Layer>>

+ **CloseAllApplications**(in_wbParentStart:wbInit_StartTime, in_wbParentPath:wbInit_Path, in_wbFinalExec:wbInit_FinalExec, in_wbHandleError:"Rethrow", io_Audit:wbInit_Audit, in_Config:Config)

Business Process Layer Interface
<<Business Process Layer>>

+ **InitAllApplications**(in_wbParentStart:wbInit_StartTime, in_wbParentPath:wbInit_Path, in_wbFinalExec:wbInit_FinalExec, in_wbHandleError:"Rethrow", io_Audit:wbInit_Audit, in_Config:Config)

Log success

+ **wbLogging**(in_wbStart:wbInit_StartTime, in_wbPath:wbInit_Path, in_wbChildAudit:wbInit_Audit, in_wbParentStart:wbMain_StartTime, in_wbFinalExec:True, in_wbHandleError:wbInit_HandleError, in_LogMessage:["Work Block "+ wbInit_Type.toString + " successful"], in_SuppressLogging:[if(Config.Item("wbInit_SuppressSuccessful").ToString.trim.ToLower = "true", true, false)], io_ParentAudit:wbMain_Audit, io_Exception:)

Catch all exceptions and log them. Output exception object

System.Exception: Log failure

+ **wbLogging**(in_wbStart:wbInit_StartTime, in_wbPath:wbInit_Path, in_wbChildAudit:wbInit_Audit, in_wbParentStart:wbMain_StartTime, in_wbFinalExec:True, in_wbHandleError:wbInit_HandleError, in_LogMessage:exception.ToString, in_SuppressLogging:False, io_ParentAudit:wbMain_Audit, io_Exception:exception)

>Errors, Abort

Error, Recover

Success

<<State Machine - Get Transaction Data State>>
<<Framework Layer>>
wbGetTransactionData

Local variables:
+ wbGetSetTransactionData_StartTime: DateTime = now()
+ wbGetSetTransactionData_Type: String = Config.Item("wbGetTransactionData_Type").ToString
+ wbGetSetTransactionData_Path:String = wbMain_Path+"I"+wbGetSetTransactionData_Type
+ wbGetSetTransactionData_Audit: List<Dictionary<String, Object>> = New List(of Dictionary(Of String, Object))

Try executing Components:

System - reserved

+ **OrchestratorStop** ()

Data Layer Interface
<<Data Layer>>

+ **GetTransactionData**(in_wbParentStart:wbMain_StartTime, in_wbParentPath:wbGetSetTransactionData_Path, in_wbFinalExec:wbGetSetTransactionData_FinalExec, in_wbHandleError:"Rethrow", io_Audit:wbGetSetTransactionData_Audit, in_Config:Config, io_TransactionItem:TransactionItem, io_TransactionData:TransactionData, in_TransactionNumber:cint(SystemReserved.Item("TransactionNumber,")), in_RetryNumber:cint(SystemReserved.Item("RetryNumber,")))

Log success

+ **wbLogging**(in_wbStart:wbGetSetTransactionData_StartTime, in_wbPath:wbGetSetTransactionData_Path, in_wbChildAudit:wbGetSetTransactionData_Audit, in_wbParentStart:wbMain_StartTime, in_wbFinalExec:True, in_wbHandleError:wbGetSetTransactionData_HandleError, in_LogMessage:"successful", in_SuppressLogging:[if(Config.Item("wbGetSetTransactionData_SuppressSuccessful").ToString.trim.ToLower = "true", true, false)], io_ParentAudit:wbMain_Audit, io_Exception:)

Catch all exceptions and log them. Output exception object

System.Exception: Log failure

+ **wbLogging**(in_wbStart:wbGetSetTransactionData_StartTime, in_wbPath:wbGetSetTransactionData_Path, in_wbChildAudit:wbGetSetTransactionData_Audit, in_wbParentStart:wbMain_StartTime, in_wbFinalExec:True, in_wbHandleError:wbGetSetTransactionData_HandleError, in_LogMessage:exception.ToString, in_SuppressLogging:False, io_ParentAudit:wbMain_Audit, io_Exception:exception)

New Transaction

Error, Continue

Success

<<State Machine - Process Transaction State>>
<<Framework Layer>>
wbProcess

Local variables:
+ wbProcessTransaction_StartTime: DateTime = DateTime.now()
+ wbProcessTransaction_Type: String = Config.Item("wbProcessTransaction_Type").ToString
+ wbProcessTransaction_Path:String = wbMain_Path+"I"+wbProcessTransaction_Type
+ wbProcessTransaction_Audit: List<Dictionary<String, Object>> = New List(of Dictionary(Of String, Object))

Try executing Components:

System - reserved

+ **GetQueueRetries**(in_TransactionItem:TransactionItem, io_SystemReserved:SystemReserved, io_Config:Config)

Business Process Layer Interface
<<Business Process Layer>>

+ **Process**(in_wbParentStart:wbMain_StartTime, in_wbParentPath:wbProcessTransaction_Path, in_wbFinalExec:wbProcessTransaction_FinalExec, in_wbHandleError:"Rethrow", io_Audit:wbProcessTransaction_Audit, in_Config:Config, io_TransactionItem:TransactionItem, in_TransactionNumber:cint(SystemReserved.Item("TransactionNumber,")), in_RetryNumber:cint(SystemReserved.Item("RetryNumber,")))

Log success

+ **wbLogging**(in_wbStart:wbProcessTransaction_StartTime, in_wbPath:wbProcessTransaction_Path, in_wbChildAudit:wbProcessTransaction_Audit, in_wbParentStart:wbMain_StartTime, in_wbFinalExec:True, in_wbHandleError:wbProcessTransaction_HandleError, in_LogMessage:"successful", in_SuppressLogging:[if(Config.Item("wbProcessTransaction_SuppressSuccessful").ToString.trim.ToLower = "true", true, false)], io_ParentAudit:wbProcessTransaction_Audit, io_Exception:)

Catch all exceptions and log them. Output exception object

System.Exception: Log failure

+ **wbLogging**(in_wbStart:wbProcessTransaction_StartTime, in_wbPath:wbProcessTransaction_Path, in_wbChildAudit:wbProcessTransaction_Audit, in_wbParentStart:wbMain_StartTime, in_wbFinalExec:True, in_wbHandleError:wbProcessTransaction_HandleError, in_LogMessage:exception.ToString, in_SuppressLogging:False, io_ParentAudit:wbMain_Audit, io_Exception:exception)

>Errors, Abort

No Data

Data Error

>Errors, Abort

<<State Machine - End Process State>>
<<Framework Layer>>
wbMain

Local variables:

Try executing Components:

System - reserved

System - reserved
<<Business Process Layer>>

+ **CloseAllApplications**(in_wbParentStart:wbMain_StartTime, in_wbParentPath:wbMain_Path, in_wbFinalExec:wbMain_FinalExec, in_wbHandleError:"Rethrow", io_Audit:wbMain_Audit, in_Config:Config)

Log success

+ **wbLogging**(in_wbStart:wbMain_StartTime, in_wbPath:wbMain_Path, in_wbChildAudit:wbMain_Audit, in_wbParentStart:in_wbParentStart, in_wbFinalExec:True, in_wbHandleError:in_wbHandleError, in_LogMessage:"successful", in_SuppressLogging:false, io_ParentAudit:io_Audit, io_Exception:)

Catch all exceptions and log them. Output exception object

System.Exception: Log failure

+ **wbLogging**(in_wbStart:wbMain_StartTime, in_wbPath:wbMain_Path, in_wbChildAudit:wbMain_Audit, in_wbParentStart:in_wbParentStart, in_wbFinalExec:True, in_wbHandleError:in_wbHandleError, in_LogMessage:exception.ToString, in_SuppressLogging:False, io_ParentAudit:io_Audit, io_Exception:exception)

Legend	
blue	input argument, Framework Layer
red	output argument
purple	io argument, Data Layer
orange	Business Process Layer