|  |  |
| --- | --- |
|  |  |

Technical Specification Document



Project Name: WIP Kit Issues (Labor and Materials) and PO Receipt

Issue Title: Re-write the process. The current process has frequent errors and does not accommodate outsourced operations

Date: 03/05/2022

# Prepared By

|  |  |
| --- | --- |
| **Document Owner(s)** | **Project/Organization Role** |
| Dondi Colgrove | Software Developer |

# Modification Version Control

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Issue. Version** | **Date** | **Author** | **Tag ID** | **Change Description** |
| 1 | 04/15/2022 |  | SDM: 28864 |  |
| 1.1 | 04/27/2022 |  | SDM: 28864 | Added additional details from dev outline call |

# Table of Contents

[Prepared By 2](#_Toc96335885)

[Modification Version Control 2](#_Toc96335886)

[Table of Contents 3](#_Toc96335887)

[1 Design Overview 5](#_Toc96335888)

[1.1 Business Case/Scope 6](#_Toc96335889)

[1.2 Audience 6](#_Toc96335890)

[1.3 Reference 6](#_Toc96335891)

[1.4 Terms and Definitions 7](#_Toc96335892)

[1.5 Technical Design Diagram 7](#_Toc96335893)

[1.5.1 Application Architecture Diagram 8](#_Toc96335894)

[1.5.2 Integration Architecture Diagram 9](#_Toc96335895)

[1.6 Assumptions/Dependencies 14](#_Toc96335896)

[1.7 Outstanding Questions /Open issues 14](#_Toc96335897)

[2 Data Design 14](#_Toc96335898)

[2.1 Database Management System Files 14](#_Toc96335899)

[2.2 Non-Database Management System Files 16](#_Toc96335900)

[3 Detailed Design 17](#_Toc96335901)

[3.1 Hardware Detailed Design 17](#_Toc96335902)

[3.2 Software Detailed Design 17](#_Toc96335903)

[3.2.1 Module [X] 17](#_Toc96335904)

[3.2.2 Module [X] 22](#_Toc96335905)

[3.3 Communications Detailed Design 25](#_Toc96335906)

[4 External Interface Design 25](#_Toc96335907)

[4.1 Interface Architecture 25](#_Toc96335908)

[4.2 Interface detailed design 25](#_Toc96335909)

[5 Human-Machine Interface 25](#_Toc96335910)

[5.1 Interface Design Rules 25](#_Toc96335911)

[5.1.1 Inputs 25](#_Toc96335912)

[5.1.2 Outputs 25](#_Toc96335913)

[5.2 Navigation Hierarchy 25](#_Toc96335914)

[5.2.1 Screen [x.1] 26](#_Toc96335915)

[5.2.3 Screen [x.2] 26](#_Toc96335916)

[6 Appendices 26](#_Toc96335917)

[Appendix A: Analysis Documentation 26](#_Toc96335918)

[Appendix B: Issues 26](#_Toc96335919)

[Appendix C: Other supporting documentation 26](#_Toc96335920)

[Appendix D: Benchmark Design and Development Estimate 26](#_Toc96335921)

# Design Overview

Create a windows service that will be started and continuously run on the 7Syspro. This service will be set to start automatically (with delay) upon reboots. Upon starting, the service will query for staged records to be processed. Once records are processed, the service will enable a timer that will wait 5 minutes until it checks for records again. When starting the service or the timer has elapsed, it will check for new “staged” records within the SysproDocument.WKI.Stage\_Job table via a SQL procedure “SysproDocument.WKI.Stage\_Job\_Get”. This will pull in the records that are flagged and needing to be processed. If records are found needing to be processed, use the data from the staging table to first get the list of materials that need to be posted to Syspro for the specific job and operation. This information is pulled from Syspro via their business object (WIPRMI) for that Job and Operation. The data will need to be stored and utilized to post the transaction then to Syspro. First post the labor using the Syspro business object (WIPTLP). Once that is successful, post the material using the Syspro Business Object (WIPTMI). Once that is successful, if there is a PO tied to the job and operation, then post a PO receipt utilizing the Syspro Business Objects (PORTOR). Once all related postings are successful, update the staged record with a status of “Processed”, “ToBeProcessed = 0” and LastStatusChangeDate = getdate(). If there is an error within the posting of the labor, the application should log the error, send email notification and the staged record should be updated with a status of as “Failed” and LastStatusChangeDate = getdate(). If there is an error during the posting of the material, the labor postings should be reversed using the Syspro Business Object (WIPTLP) with a negative labor hours and the error should be logged, an email notification should be sent and the staged record should be updated with status of “Failed” and LastStatusChangeDate = getdate(). Once all the staged records are processed (or failed) that were initially loaded, the service will then check to see if there are any new records that are staged. If so, it will loop through and process the next records. If there are no records found, then it will start the timer to check for records again in 5 minutes. If the service hits the max time of the day (10:00 pm) (or is between the max and min times) to actively check for records, it will set the timer to not check for records again until next morning (5:00 am).

The service will be built on .net framework and visual studio. The service will utilize the “SYSPROWCFServicesClientLibrary40” and the “GWC” dll libraries as references. The “SYSPROWCFServicesClientLibrary40.dll” will be the library used to connect to the Syspro web services utilizing XML. The “GWC.dll” library file will be referenced and used for logging throughout the different stages the application is running through. When the service first loads, it will pull in the general settings information from the tables located in SysproDocument.dbo.Setting. This information may include the start and stop time the application would actively be checking the tables for records (5:00 am – 10:00 pm), the length of the timer that the service pauses until it checks for records again (5 minutes), email notification settings, etc. Authentication information for the Syspro Business Objects will be pulled from Global.Settings.SysproAccess. The base template structure of the XML for the Syspro Business Objects will be stored located in SysproDocument.dbo.Parameter and SysproDocument.dbo.XmlInTemplate database tables. These will be pulled in and populated with the data needed for the specific transaction that is being posted for that job and operation. Individual Job failures during postings, will be recorded in SysproDocument.WKI.JobFailures.

## 1.1 Business Case/Scope

Process to post kit issues (labor and materials) and Purchase Order receipts to Syspro based off scanning data. The current process has frequent errors and does not accommodate outsourced operations. It has been determined that revisiting from the ground up would be the best method for resolution. Utilizes business objects WIP Build Material Issue (WIPRMI), WIP Specific Issues (WIPTMI), WIP Labor Postings (WIPTLP) and Purchase Order Receipts (PORTOR). Data for postings will be written to a SQL table where the WKI service will pick it up and post to Syspro. Should be structured similarly to the WIP Job Receipt service (WJR). Appropriate error handling to walk back postings if related posting fails.

## 1.2 Audience

This is a technical document targeted for technical readers that will maintain the designed application. It is assumed the reader has some technical background related to the technologies mentioned in the above description

## 1.3 Reference

List all references that are included in the document. Also include applicable policies and procedures associated with the document and its contents.

| Ref# | Document Name | URL |
| --- | --- | --- |
| 1 | Application diagram | [sample-application-architecture.drawio - diagrams.net](https://app.diagrams.net/) |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## 1.4 Terms and Definitions

| Term or Acronym | Definition |
| --- | --- |
| WIP | Work In Progress |
| WKI | WIP Kit Issue |
| PO | Purchase Order |
|  |  |
|  |  |
|  |  |
|  |  |

## 1.5 Technical Design Diagram

### 1.5.1 Application Architecture Diagram

*Diagram, engineering drawing

Description automatically generated*

### 1.5.2 Integration Architecture Diagram

1. Syspro Business Objects
   1. WIPRMI – WIP Material Retrieve
      1. XML Sent
         1. Instance defaults located in SysproDocument.dbo.Parameter

Text

Description automatically generated

* + 1. XML Returned

Graphical user interface, text

Description automatically generated with medium confidence

* 1. WIPTLP – WIP Labor Posting
     1. XML Parameter Sent
        1. Parameter defaults located in SysproDocument.dbo.Parameter

Text, letter

Description automatically generated

* + 1. XML Instance Sent
       1. Parameter defaults located in SysproDocument.dbo.XmlInTemplate

Graphical user interface, text, application

Description automatically generated

* + 1. XML Received

Text

Description automatically generated

* 1. WIPTMI – WIP Material Posting
     1. XML Parameter Sent
        1. Parameter defaults located in SysproDocument.dbo.Parameter

Graphical user interface, text, application, letter

Description automatically generated

* + 1. XML Instance Sent
       1. Parameter defaults located in SysproDocument.dbo.XmlInTemplate

Text, letter

Description automatically generated

* + 1. XML Received

Graphical user interface, text, application, email

Description automatically generated

* 1. PORTOR – PO Receipt Posting
     1. XML Parameters Sent
        1. Parameter defaults located in SysproDocument.dbo.Parameter

Graphical user interface, text, application

Description automatically generated

* + 1. XML Instance Sent
       1. Parameter defaults located in SysproDocument.dbo.XmlInTemplate

Text

Description automatically generated with low confidence

* + 1. XML Received

Graphical user interface, text, application, email

Description automatically generated

## 1.6 Assumptions/Dependencies

1. 7Syspro server is available with network connectivity
2. SQL08 server is available with network connectivity

## 1.7 Outstanding Questions /Open issues

1. ~~Old method of posting Labor took the Run Time \* Qty from WIPJobAllLab table. Is that correct or just post run time put in staging table?~~
   1. The Staging table will store the complete Run Time that needs to be posted.
2. Need to create records for the staging table and a guide on how things can be tested to validate working correctly.
3. Need to get login credentials created with access to appropriate business objects.

# Data Design

## 2.1 Database Management System Files

* DataConnectionString
  + SQL08
  + InitialCatalog: SysproDocument
  + IntegratedSecurity: SSPI
  + PersistSecurityInfo: true
* Tables / Adding Records
  + See Appendices C for Excel document with Table Structures and Examples
* Stored Procedures to be created
  + WKI.usp\_Stage\_Job\_Get
    - Pulls in staged records
    - No Parameters
    - SQL statement

select staged.\*, MatToPost.MaterialsToPost

from WKI.Stage\_Job staged

LEFT OUTER JOIN (

SELECT staged2.JobId, staged2.Operation, COUNT(\*) as MaterialsToPost

FROM WKI.Stage\_Job staged2

LEFT OUTER JOIN SysproCompany100.dbo.WipJobAllMat mat

on staged2.JobNumber = mat.Job

and staged2.Operation = mat.OperationOffset

WHERE staged2.[Status] = 'Staged' and staged2.ToBeProcessed = 1

GROUP BY staged2.JobId, staged2.Operation

) MatToPost on staged.JobId = MatToPost.JobId and staged.Operation = MatToPost.JobId

WHERE staged.[Status] = 'Staged' and staged.ToBeProcessed = 1

* + WKI.usp\_Stage\_Job\_Set
    - Updates the staged record to the appropriate status
    - Parameters
      * @JobId
      * @Operation
      * @NewStatus
      * @ToBeProcessed
    - Statement

Update WKI.Stage\_Job

SET [Status] = @NewStatus, ToBeProcessed = @ToBeProcessed

WHERE JobId = @JobId and Operation = @Operation

* + WKI.usp\_Job\_CheckForPO
    - Returns the PO information needed to perform PO receipt if needed
    - Parameters
      * @JobId
      * @Operation
    - Statement

select staged.JobId, staged.Operation, staged.JobNumber po.PurchaseOrder, po.Line as POLine, po.SubQtyPer as POQty

FROM WKI.Stage\_Job staged

LEFT OUTER JOIN SysproCompany100.dbo.WipJobAllLab lab

on staged.JobNumber = lab.Job

and staged.Operation = lab.Operation

and SubcontractOp = 'Y'

LEFT OUTER JOIN SysproCompany100.dbo.PorMasterDetail po

on lab.Job = po.MJob

and po.MSubcontractOp = lab.Operation

and po.MStockCode = lab.SupPoStkCode

WHERE staged.JobId = @JobId and staged.Operation = @Operation

## Non-Database Management System Files

# Detailed Design

## 3.1 Hardware Detailed Design

## 3.2 Software Detailed Design

### 3.2.1 Application Code Organization

### Data Context Class

Create a new class within the Visual Studio project that will be a place to store all the SQL calls to the database. This will make it easier to track down procedures that are used when troubleshooting in the future. Here are some of the functions that should be created and stored in this class to return the values needed for the service operation.

1. GetApplicationSettings
   1. Call to Sysprodocument.GEN.usp\_Settings\_Get using the application ID as a parameter
   2. Returns the XML document with the setting information stored
2. GetBusinessObjectParameters
   1. Execute stored procedure SysproDocument.dbo.usp\_GetParametersByNameAndBranch
      1. BusObjId
      2. BranchId = 0
      3. ApplicationId
   2. Returns default parameters stored in the SysproDocument.dbo.Parameter table for the specific Application ID and Business Object ID to be used on the Syspro Business Objects
3. GetTemplateData
   1. Executes stored procedure SysproDocument.GEN.usp\_XmlInTemplate\_Get with xml parameters for the ApplicationId and TemplateName
      1. <GetRequest><ApplicationId>18</ApplicationId><TemplateName>ARSSCS</TemplateName></GetRequest>
   2. Returns an XML document with the template information stored in SysproDocument.dbo.XmlInTemplate table
4. GetStagedJobs
   1. WKI.usp\_Stage\_Job\_Get
      1. Status – Create
         1. Will query the data from the WKI.Stage\_Job table to pull in records that are marked as “Staged” and ToBeProcessed = 1
      2. Parameters – None
      3. Purpose – Pull in the records and the data needed to do the postings
5. SetStagedJobs
   1. WKI.usp\_Stage\_Job\_Set
      1. Status - Create
      2. Parameters – JobID, Operation
      3. Purpose - Update the WKI.Stage\_Job staging record that was just processed and for the appropriate JobID and Operation in WKI.Stage\_Job
6. SetJobFailure
   1. WKI.usp\_Stage\_Job\_Failure\_Set
      1. Status - Create
      2. Parameters – JobID, Operation, JobNumber, ErrorMessage
      3. Purpose – Insert record into WKI.Job\_Failures table
7. GetGroupingID
   1. Graphical user interface, text, application

      Description automatically generated

### WKI\_Service Class

This will be the primary class that will store the processing functions and procedures:

1. Variables / Objects
   1. tmrPrimary as new timers.timer
   2. DT\_StagedJobs as datatable
   3. CurrentMaterialToPost as xml
2. OnStart procedure
   1. Load application settings needed on ModMain
   2. Write to log that service started
   3. Set timer interval for 1 minute and enable timer.
3. OnStop procedure
   1. Make sure any records that are in process finish prior stopping service.
   2. Write to log that service stopped
4. TmrPrimary\_Elapsed
   1. Disable Timer
   2. Call ProcessRecords procedure
   3. Enable timer.
5. Function to logon to Syspro / Check Syspro locked out with byref parameter Error\_Message for any Error Message
   1. If not locked out
      1. Return the GUID in of the Syspro Login ID
      2. Set “” as the error message
   2. If locked out
      1. Return nothing
      2. Error\_Message set to any error description outlining the message returned from Syspro
6. Functions to process each of the business objects listed below in Process\_Record procedure
   1. Each function will be set up to handle the process for 1 business object and return true / false for pass fail.
7. Procedure Record\_Job\_Failure to record job failure and send notification
   1. Log record in WKI.Job\_Failures
   2. Send failure / error notifications
8. Procedure Check\_Staged\_Jobs to query for Staged Jobs and populate the DT\_StagedJobs datatable with any records
9. ProcessRecords Procedure
   1. Call Check\_Staged\_Jobs procedure to populate any staged records needing processed.
   2. If DT\_StagedJobs record count > 0 then
      1. Call Function to Logon Syspro / Check Syspro Locked
         1. If Valid Syspro ID return, set modMain.SysproGUID to the id
         2. If not valid
            1. Clear StagedJobs datatable
            2. Log Error\_Message
      2. Set GoTo Point for ProcessDatatable
      3. If SysproID is valid
         1. Query to get a new GroupingId
         2. Log Record Count
         3. Loop through each record of the data table to call the Process\_Record procedure below for the data row that is being referenced.
         4. Call Check\_Staged\_Jobs procedure to re-populate any staged records needing processed.
      4. Check DT\_StagedJobs record count
         1. If > 0 then go to the “ProcessDatatable” go to point set above.
         2. If 0 then
            1. Log off Syspro user
            2. Set modmain.SysproGuid to nothing
   3. Call SetTimerInterval procedure
10. Process\_Record that will make the calls for each of the steps of the process and track any failures and be ready to reverse transactions as needed.
    1. If Materials to Post
       1. Call Syspro Business Object WIPRMI to get transaction information needed to post material and store in MaterialsToPostXml
    2. Call function to Syspro Business Object WIPTLP to post labor and check for errors in xml.
       1. If errors
          1. Run procedure Record\_Job\_Failure
          2. Skip any future posting to business object in process cycle move to next record
          3. Set Record status as “Failed”
    3. Call function to Syspro Business Object WIPTMI to post labor and check for errors in xml
       1. If errors
          1. Call function to Syspro Business Object WIPTLP with qty \* -1 to reverse the labor posting
          2. Run procedure Record\_Job\_Failure
          3. Skip any future posting to business object in process cycle move to next record
          4. Set Record status as “Failed”
    4. If PO Receipt needed, Call function to Syspro Business Object PORTOR and check for errors in xml
       1. If errors
          1. Run procedure Record\_Job\_Failure
          2. Set Record status as “Failed”
    5. Update the Staged Record status to either “Failed” if errors occurred during processing or “Processed” if successful.
11. Procedure SetTimerInterval to set the timer interval until the next run
    1. Check the time
       1. If the time is outside the execution window time frame that is set within the settings, set the timer to run once again at the beginning of the time window.
       2. If the time is within the execution window, reset the time for the interval passed within the settings file.

### ModMain Module

Used to store the generalized variables used within the application. This would include items like

1. Connection String
   1. "Data Source=SQL08; Initial Catalog =SysproDocument; Integrated Security = SSPI;Persist Security Info =true
   2. Public
      1. ApplicationID
      2. ApplicationName
      3. Settings
         1. Settings\_Timer\_Interval\_Seconds
         2. Start\_Time
         3. End\_Time
         4. Success\_Email\_Send
         5. Success\_Email\_Message as GWC.Mail.Message
         6. Failure\_Email\_Send
         7. Failure\_Email\_Message as GWC.Mail.Message
         8. Error\_Email\_Send
         9. Error\_Email\_Message as GWC.Mail.Message
      4. Syspro Login Information
         1. Syspro\_WCF\_BaseAddress
         2. Syspro\_UserName
         3. Syspro\_Password
         4. Syspro\_Company
         5. Syspro\_CompanyPW
         6. Syspro\_GUID
      5. Business Object ID information (from SysproDocument.GEN.Ref\_BusObj)
         1. WIPRMI\_ID = 1527
         2. WIPTLP\_ID = 1539
         3. WIPTMI\_ID = 1541
         4. PORTOR\_ID = 1360
   3. Any Structures that might be used and feel are needed
   4. Any other variables / objects or generalized procedures / functions
      1. Load Application Settings
      2. Notification message procedure

### 3.2.2 Using Syspro Library

* 1. Library dll available in [*Libraries or References - Repos (azure.com)*](https://dev.azure.com/GWC/GWC-Apps/_git/Development%20Resources?path=%2FLibraries%20or%20References&version=GBmain)
  2. Declaring wcfService

WebServer\_URL will be the WebServer\_URL field value from the Global.Settings.SysproAccess database table

SYSPROWCFServicesClientLibrary40.SYSPROWCFServicesPrimitiveClient swcf = new SYSPROWCFServicesClientLibrary40.SYSPROWCFPrimitiveClient(WebServer\_URL,SYSPROWCFBinding.NetTcp);

* 1. Example of Logging on to Syspro Web Services to POST

Graphical user interface, text, application, email

Description automatically generated

* 1. Checking Logon for Lock

Graphical user interface, text, application

Description automatically generated

* 1. Checking for Errors in Returned XML

Graphical user interface, text

Description automatically generated

### 3.2.3 Using GWC Library

1. GWC Library
   1. Library dll found in [Libraries or References - Repos (azure.com)](https://dev.azure.com/GWC/GWC-Apps/_git/Development%20Resources?path=%2FLibraries%20or%20References&version=GBmain)
   2. Email Object
      1. Creating Email Object Example

Graphical user interface, text, application

Description automatically generated

* + 1. Send Notification Email Example

Text

Description automatically generated

* + 1. Testing emails sent through GWC in Dev
       1. Emails that are sent through DEV are all trapped and sent to a consolidated account.
       2. When using the GWC email object to send, it will post a record to the following Notifications tables:
          1. SysproDocument.NFY.Notification\_Message
          2. SysproDocument.NFY.Notification\_Recipient
       3. There is another service that runs on 7Syspro that will send out any staged records in this table.
       4. So as long as the email message shows up in this table, the email notification should get picked up and sent within the production environment.
  1. DynamicLogger object

This will add log entries into the SysproDocument.dbo.Log\_Application\_Master, Header and Detail tables for the details that are specified in the calls.

* + 1. Creating Object





Graphical user interface, text

Description automatically generated

* + 1. Logging Status



Using the data example from WIP Kit Issues Table Structures appendix file, this will write log status regarding using the parameter structure:

1. the eventID associated to ServiceError (264)
2. the specific GroupingId (stored in the application used to group log records together to know transactions are associated to same execution loop)
3. Add no custom information to the Header table
4. Add Detail info. Will be separated by comma’s by default into breakdown listed in the Log\_Settings record
   * MethodName = “SendFailureEmail()”
   * Message = ex.message

## 3.3 Communications Detailed Design

# External Interface Design

## 4.1 Interface Architecture

## 4.2 Interface detailed design

# Human-Machine Interface

## 5.1 Interface Design Rules

### 5.1.1 Inputs

### 5.1.2 Outputs

## 5.2 Navigation Hierarchy

### 5.2.1 Screen [x.1]

### 5.2.3 Screen [x.2]

# Appendicess

## Appendix A: Analysis Documentation

## Appendix B: Issues

## Appendix C: Other supporting documentation

1. WIP Kit Issues Table Structures

This will outline all the individual custom table structures that will either need to be created, records inserted / updated or used for staging.

## Appendix D: Benchmark Design and Development Estimate

1. Coding
   1. Creating a new Visual Studio project creating a windows service that can be installed on 7Syspro. Service will be set to continuously run on the server and utilize a timer in order to know when to check for staged jobs. Service will utilize SQL Stored Procedures in order to pull the data into the application and to update records in the database. Service will make calls to the Syspro Business Objects via XML.
   2. Creating a SQL script that will:
      1. Creating any new SQL tables that are utilized by the service.
      2. Insert any settings, templates, records that are needed for the application to execute properly.
      3. Create any new Stored Procedures that are referenced by the service.
   3. SQL Script will also be needed and used to assign permissions to the login that is created that the service will execute under.
2. Testing
   1. Complete testing for staged records that will be created to ensure all aspects of service will be working. Specific situations specifically needing tested (but not limited to):
      1. Post Successful through the process
         1. Records that would have Materials to Post, has Labor to Post with no PO Receipt to post.
         2. Records that do not have Materials to Post, has Labor to Post with no PO Receipt to post.
         3. Records that would have Materials to Post, Labor to Post with PO Receipt to post.
      2. Posting Errors to test (Failure record, notifications, staging record updated)
         1. Records that fail during call to WIPRMI to get Materials to post
         2. Records that fail during Labor Posting (WIPTLP)
         3. Records that fail during Material Posting (WIPTMI)
            1. Labor Postings should be reversed
         4. Records that fail during PO Receipt (PORTOR)
      3. Service runs between specified times and interval is based on the setting file
3. Documentation
   1. Document any additional details on how service that may have been missed in this document.
   2. Complete any adjustments to documentation listed here once service is created.
4. Go Live Support
   1. Available for any issues once service is in place and records are being staged automatically.

Benchmark Design and Development estimate. Please list the task name, hours needed to complete and a benchmark estimate date of completion. **Remember to add task and hours for testing, documentation and go live support**

| Task | Hours | Benchmark est. Date |
| --- | --- | --- |
| Coding | <hours> | <MM/DD/YYYY> |
| Testing |  |  |
| Documentation |  |  |
| Go Live Support |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |