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

Technical Specification Document



Project Name: SugarCRM Integration

Date: July 2022

# Prepared By

|  |  |
| --- | --- |
| **Document Owner(s)** | **Project/Organization Role** |
| Justin Pope | Software Developer |

# Modification Version Control

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Issue. Version** | **Date** | **Author** | **Tag ID** | **Change Description** |
| 1.0 | 05/23/2022 | Justin Pope |  | Initial Creation |

Table of Contents

[Prepared By 2](#_Toc108686355)

[Modification Version Control 2](#_Toc108686356)

[2 Design Overview 4](#_Toc108686357)

[1.2 Audience 4](#_Toc108686358)

[1.3 Reference 4](#_Toc108686359)

[1.4 Terms and Definitions 4](#_Toc108686360)

[1.5 Technical Design Diagram 4](#_Toc108686361)

[1.6 Assumptions/Dependencies 6](#_Toc108686362)

[1.7 Outstanding Questions /Open issues 7](#_Toc108686363)

[3 Data Design 7](#_Toc108686364)

[2.1 Database Management System Files 7](#_Toc108686365)

[2.1.3 Functions 8](#_Toc108686366)

[2.1.3.1 Scalar Valued Functions 8](#_Toc108686367)

[2.1.3.2 Table Valued Functions 9](#_Toc108686368)

[3.2 Non-Database Management System Files 9](#_Toc108686369)

[3.2.1 SSIS CSV Files 9](#_Toc108686370)

[3.2.2 Talend Complete CSV Files 10](#_Toc108686371)

[3.2.3 Talend Log Files 11](#_Toc108686372)

[4 Detailed Design 11](#_Toc108686373)

[4.1 Software Detailed Design 11](#_Toc108686374)

[4.2.1 SQL Job – Sugar CRM Export 12](#_Toc108686375)

[4.2.2 SSIS Package – Sugar CRM Export 13](#_Toc108686376)

[4.2.3 Talend Job – Sugar Master Integration 14](#_Toc108686377)

[4.3 Communications Detailed Design 15](#_Toc108686378)

[5 Appendices 16](#_Toc108686379)

[Appendix A: Analysis Documentation 16](#_Toc108686380)

[Appendix B: Issues 16](#_Toc108686381)

[Appendix C: Other supporting documentation 16](#_Toc108686382)

[Appendix D: Benchmark Design and Development Estimate 16](#_Toc108686383)

# Design Overview

This document contains the system process for the data that is transmitted to SugarCRM. This system uses SQL Database, a SQL SSIS package, and Talend jobs to transmit to SugarCRM. A SQL Job is set up to trigger the SSIS package to run and how to run. The SSIS package calls SQL procedures to update information to transmit to SugarCRM and create CSV Export files. Talend jobs can read the CSV files created, manipulate the date into the JSON format compatible with the SugarCRM API and finally call the API to send the information. Talend also creates files for logging based on what happens.

## 2.2 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

## 2.3 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 | SugarCRM Export System Diagram | [SugarCRM Export System Diagram.pdf](https://summerclassics.sharepoint.com/:b:/s/ITDepartment/EXymeOEYd61Cq0HGTDDgYf4BrGNa3Bz8i4dcovQWYbupCw?e=2FmAMq) |

## 2.4 Terms and Definitions

| Term or Acronym | Definition |
| --- | --- |
| CSV | Comma Separated Values |
| SQL | Structured Query Language |
| SSIS | SQL Server Integration Services |
| SugarCRM | Sugar Customer Relations Management Software |

## 1.5 Technical Design Diagram

The overarching system is designed with five exports which are as follows:

1. Accounts
2. Quote Header
3. Quote Details
4. Sales Orders
5. Sales Order Details

Each of the systems that make up the integration has a sub process per export that handles the data. The following diagram and descriptions of logic is intended to communicate what is happening within each system and how the data flows from SQL Server to SugarCRM.

Graphical user interface

Description automatically generated

Figure 1 : SugarCRM Export System Diagram: SQL and SSIS

Graphical user interface, website

Description automatically generated

Figure 2 - SugarCRM Export System Diagram: Talend and APIs

For a better view of this system diagram use the following link: [SugarCRM Export System Diagram.pdf](https://summerclassics.sharepoint.com/:b:/s/ITDepartment/EXymeOEYd61Cq0HGTDDgYf4BrGNa3Bz8i4dcovQWYbupCw?e=l8oQPz).

**SQL08 and SugarCRM SSIS Package:**

SQL08 is our server with PRODUCT\_INFO as the database that houses the tables, procedures and functions used in this integration. A SQL Agent Job (Sugar CRM) is used to initiate the SSIS Package and interate through the five processes that transmit data to Talend. The SSIS Package executes procedures, and select data from PRODUCT\_INFO and creates the Comma Seperated Values (CSV) files into the SugarCRM file directory that the Talend Jobs are expecting.

**SQL**

The above procedures named as Update[PROCESS]ReferenceTable are used to update reference tables within PRODUCT\_INFO to now if an update or addition is needed to be sent to SugarCRM. The reference tables are then used within the functions named as tvf\_Build[PROCESS]Dataset to build the set of data to transmit to SugarCRM. The procedures named Flag[Process]AsSubmitted are used to build the audit tables within PRODUCT\_INFO as a copy of what the system just sent.

**SSIS**

The SSIS Package SugarCRM is called by the SQL Server Agent Job per process that has been initiated. The SQL Job supplies the SSIS package parameters to initiate the the specific export. After start, the SSIS package follows the following order:

1. Execute sql update table reference procedures.
2. Query data set to send using sql build dataset functions.
3. Write .CSV file with dataset and save it in the SugarCRM/Active dirctory.
4. Execute sql flag as submitted procedures.

**Talend:**

The Talend jobs have five process to pick up the CSV files that were created. Each job reads in the file, validate the data against SugarCRM using API endpoints, and submits the updates or additions to SugarCRM. Talend creates some logging in the form of CSV files in the the SugarCRM file directory.

## 2.6 Assumptions/Dependencies

The following issues to the existing system.

1. The SSIS package is time consuming to manage when a change is needed to the data feed.
2. The current way the Reference tables are set up could be more efficient and send data only when a data point has changed.
3. The current Talend job contains the following inefficiencies:
   1. Querying entire tables from SugarCRM. This could be specified to the data that needs to be validated.
   2. Error handling and logging is inefficient to diagnosis if there is a problem.

## 2.7 Outstanding Questions /Open issues

The following are open items.

1. Currently in the works with a contractor UPSERT to fix issues with integrating with SugarCRM. This would address the above concerns.

# Data Design

## 3.1 Database Management System Files

**3.1.1 Tables**

Table definitions can be found in SugarCRM Table Structure excel document listed in Appendix C.

**3.1.2 Procedures**

Detailed definitions can be found in the SugarSQLObjects directory listed in Appendix C. Also, each object listed will have a link to a script that creates the object in SQL.

* [FlagCustomersAsSubmitted](https://summerclassics.sharepoint.com/:u:/s/ITDepartment/EcuMQd5M4-dJp_ZSv_qsu6kBhOixs21c9nYtuIVFpPYIgA)
  + This procedure inserts records into the CustomerExport\_Audit table and flags ArCustomer\_Ref and ArCustomer+\_Ref as submitted.
* [FlagQuoteDetailsAsSubmitted](https://summerclassics.sharepoint.com/:u:/s/ITDepartment/Ef6s2lNTYolJtR3fMsRbQ7gBNgPpWYVuhmZUNjOHlRsyUw)
  + This procedure inserts records into the QuoteDetailExport\_Audit table and flags the QuoteDetail\_Ref tables as submitted.
* [FlagQuoteHeadersAsSubmitted](https://summerclassics.sharepoint.com/:u:/s/ITDepartment/EWATrD5Mv3hBvCCSNGR2nnkBq09wRaHJdgDOMJ_MZ2GGcw)
  + This procedure inserts records into the QuoteHeaderExport\_Audit table and flags the QuoteHeader\_Ref tables as submitted.
* [FlagSalesOrderHeadersAsSubmitted](https://summerclassics.sharepoint.com/:u:/s/ITDepartment/EVYpMHhYHM9OjOj1dbCTWdMBYTOuXTiY9BZJestKWYCKEw)
  + This procedure inserts records into the SalesOrderHeader\_Audit table and flags the SalesORderHeader\_Ref tables as submitted.
* [FlagSalesOrderLinesAsSubmitted](https://summerclassics.sharepoint.com/:u:/s/ITDepartment/EUgbZLF-R-NFljgUFlhCkiEBCBi12VT700XmBcO5apvgcw)
  + This procedure inserts records into the SalesOrderLineExport\_Audit table and flags the SalesOrderLine\_Ref tables as submitted.
* [LogExportFileCreation](https://summerclassics.sharepoint.com/:u:/s/ITDepartment/EVJFsNXfBiFChkpW2dicdDgBBcKFaXVQ4U_MamFntXzzWA)
  + This procedure creates a log of the export files created.
* [UpdateCustomerReferenceTable](https://summerclassics.sharepoint.com/:u:/s/ITDepartment/EZC_qm_TqUZNvjp5RJDOaA8BiuJBcTKZOjqLcKg6IxrqpQ)
  + This procedure updates or inserts into the ArCustomer\_Ref or ArCustomer+\_Ref tables when new data exist from [SysproCompany100].[dbo].[ArCustomer] and [SysproCompany100].[dbo].[ArCustomer+] respectively.
* [UpdateQuoteDetailReferenceTable](https://summerclassics.sharepoint.com/:u:/s/ITDepartment/ERlc5QHySIJKuJE30S70BGsBD-NcxKaFKycNnaABgCb3qA)
  + This procedure inserts into the QuoteDetail\_Ref table new data from the table [Ecat].[dbo].[QuoteDetail].
* [UpdateQuoteHeaderReferenceTable](https://summerclassics.sharepoint.com/:u:/s/ITDepartment/ESY0G6p6Q09AjmgxnmQyJl8BM_Y7lqYs3xVBrkeStrp5hw)
  + This procedure inserts into the QuoteHeader\_Ref table new data from the table [Ecat].[dbo].[QuoteMaster].
* [UpdateSalesOrderHeaderReferenceTable](https://summerclassics.sharepoint.com/:u:/s/ITDepartment/EV0xlGn3-7lMlKlVKt3wfI8BeD97UqmL1fOwabcA4SZNmg)
  + This procedure inserts or updates into the SalSalesperson+\_Ref table with new data from the table SysproCompany100.dbo.[SalSalesperson+] and insert into SalesOrderHeader\_Ref new data from the table SysproCompany100.dbo.SorMaster.
* [UdpateSalesOrderLineReferenceTable](https://summerclassics.sharepoint.com/:u:/s/ITDepartment/EUe_jghM2_VNgCni0fxoS2cBaZzJHclgofDWis4bxBgN_Q)
  + This procedure inserts or updates the table SalesOrderLine\_Ref with new data from the table SysproCompany100.dbo.SorDetail.
* [usp\_GetSetting\_Upload](https://summerclassics.sharepoint.com/:u:/s/ITDepartment/EUagndBC2JJHptpJOpTYa4EB8y-Q4faMwtbDgOl5QNMl2w)
  + This procedure is used to get setting for the SugarCRM integration.
* [usp\_WriteLog\_Export](https://summerclassics.sharepoint.com/:u:/s/ITDepartment/ETxALokNGgxLmZWZ_S2_Tq0B0N6K-afL_yVC-JG_1eQ9kw)
  + This procedure is used to write a log upon error.

## 3.1.3 Functions

## 3.1.3.1 Scalar Valued Functions

* [ParseFileCategory](https://summerclassics.sharepoint.com/:u:/s/ITDepartment/EVv7ZxQU1GFAtRTn4FO0BNYBUML7WBFUzhcNZvKesrkHGA)
  + Parameters: @FileName as Varchar (1000)
  + Return Values: Varchar (300)
* [ParseFileDate](https://summerclassics.sharepoint.com/:u:/s/ITDepartment/EV7lk8g-v3ZPm1HBiDAPf7UB600fRusizTdnDoGwrdWicg)
  + Parameters: @FileName as Varchar (1000)
  + Return Values: DATETIME2

## 3.1.3.2 Table Valued Functions

* [tvf\_BuildCustomerDataset](https://summerclassics.sharepoint.com/:u:/s/ITDepartment/ETucVbAFfTJMg5nQm1gEUrUBdw7nJOfAkyJCaCg9qQRvtA)
  + Selects data for Customer Export.
* [tvf\_BuildQuoteDetailDataset](https://summerclassics.sharepoint.com/:u:/s/ITDepartment/Eceuuf5ZwR9MnbHtMMViT_0BVeK2U89Lg66Gj-gO99OEHw)
  + Selects data for QuoteDetail Export.
* [tvf\_BuildQuoteHeaderDataset](https://summerclassics.sharepoint.com/:u:/s/ITDepartment/EaYkGBcy17dKrszzJRB3GjwBTJUymbM4Wu7W0aJ61qsjhQ?e=drfeM1)
  + Selects data for QuoteHeader Export.
* [tvf\_BuildSalesOrderHeaderDataset](https://summerclassics.sharepoint.com/:u:/s/ITDepartment/EXGp7iydZHNHiwxUG_a0FoMBrjAi4XJfE7zFTAZzB_2xDg)
  + Selects data for SalesOrderHeader Export.
* [tvf\_BuildSalesOrderLineDataset](https://summerclassics.sharepoint.com/:u:/s/ITDepartment/EesckO0DrIBMkoah4F_KvmcBpWfoUdw4voSaeXa-qKkSJg?e=UjlumA)
  + Selects data for SalesOrderLine Export.

## Non-Database Management System Files

## SSIS CSV Files

* SugarCustomerExport\_YYYYMMDDThhmmss.csv
  + This csv file is created from tvf\_BuildCustomerDataset
  + Output of SSIS package SugarCRM to directory: \\sql08\p\SSIS\Data\Datafeed\SugarCRM\Active
  + Input of the Talend Job Accounts
  + Example: [SugarCustomerExport\_20220614T055302.txt](https://summerclassics.sharepoint.com/:t:/s/ITDepartment/ERipgVipcJlNg1t_NmDOI_0B-snjbf06EW4i0NM_Eq5-Bw?e=YeDYxe)
* SugarQuoteDetailExport\_YYYMMDDThhmmss.csv
  + This csv file is created from tvf\_BuildQuoteDetailDataset
  + Output of SSIS package SugarCRM to directory: \\sql08\p\SSIS\Data\Datafeed\SugarCRM\Active
  + Input of the Talend Job Quotes
  + Example: [SugarQuoteDetailExport\_20220614T055319.txt](https://summerclassics.sharepoint.com/:t:/s/ITDepartment/EUPyTO1uT05LqWPir2SUr1MBOfQ2ePVXDc9EftPv1qgu9Q?e=8fIzHL)
* SugarQuoteHeaderExport\_YYYYMMDDThhmmss.csv
  + This csv file is created from tvf\_BuildQuoteHeaderDataset
  + Output of SSIS package SugarCRM to directory: \\sql08\p\SSIS\Data\Datafeed\SugarCRM\Active
  + Input of the Talend Job Quote Line Items
  + Example: [SugarQuoteHeaderExport\_20220614T055317.txt](https://summerclassics.sharepoint.com/:t:/s/ITDepartment/EZtpLBDev6NKojUt7Rv-ltEB5mWonQqXOkJmAgUszCpBQw?e=opE639)
* SugarSalesOrderHeaderExport\_YYYYMMDDThhmmss.csv
  + This csv file is created from tvf\_BuildSalesOrderHeaderDataset
  + Output of SSIS package SugarCRM to directory: \\sql08\p\SSIS\Data\Datafeed\SugarCRM\Active
  + Input of the Talend Job Orders
  + Example: [SugarSalesOrderHeaderExport\_20220613T235305.txt](https://summerclassics.sharepoint.com/:t:/s/ITDepartment/EVVoH1ZAtnhOuI4HK_HowpABX-tw4lMZxkxKd4rVGxHyoA?e=Fu0clS)
* SugarSalesOrderLineExport\_YYYYMMDDThhmmss.csv
  + This csv file is created from tvf\_BuildSalesOrderLineDatset
  + Output of SSIS package SugarCRM to directory: \\sql08\p\SSIS\Data\Datafeed\SugarCRM\Active
  + Input of the Talend Job Order Line Items
  + Example: [SugarSalesOrderLineExport\_20220613T235309.txt](https://summerclassics.sharepoint.com/:t:/s/ITDepartment/EaotGHwiulVPnyNlhcFIeFoBGVQOxS-qXixm8noOMR326Q?e=AXrjMQ)

## Talend Complete CSV Files

* YYY\_MM\_DD hh\_mm\_ssSugarCustomerExport\_YYYYMMDDThhmmss.csv
  + This csv file is the SugarCustomerExport\_YYYYMMDDThhmmss.csv renamed and moved to the directory \\sql08\p\SSIS\Data\Datafeed\SugarCRM\Completed.
  + output of the Talend Job Accounts
  + Example: [SugarCustomerExport\_20220614T055302.txt](https://summerclassics.sharepoint.com/:t:/s/ITDepartment/ERipgVipcJlNg1t_NmDOI_0B-snjbf06EW4i0NM_Eq5-Bw?e=YeDYxe)
* YYY\_MM\_DD hh\_mm\_ssSugarQuoteDetailExport\_YYYMMDDThhmmss.csv
  + This csv file is the SugarQuoteDetailExport\_YYYMMDDThhmmss.csv renamed and moved to the directory \\sql08\p\SSIS\Data\Datafeed\SugarCRM\Completed.
  + Output of the Talend Job Quote Line Items
  + Example: [SugarQuoteDetailExport\_20220614T055319.txt](https://summerclassics.sharepoint.com/:t:/s/ITDepartment/EUPyTO1uT05LqWPir2SUr1MBOfQ2ePVXDc9EftPv1qgu9Q?e=8fIzHL)
* YYY\_MM\_DD hh\_mm\_ssSugarQuoteHeaderExport\_YYYYMMDDThhmmss.csv
  + This csv file is the SugarQuoteHeaderExport\_YYYYMMDDThhmmss.csv renamed and moved to the directory \\sql08\p\SSIS\Data\Datafeed\SugarCRM\Completed.
  + Output of the Talend Job Quotes
  + Example: [SugarQuoteHeaderExport\_20220614T055317.txt](https://summerclassics.sharepoint.com/:t:/s/ITDepartment/EZtpLBDev6NKojUt7Rv-ltEB5mWonQqXOkJmAgUszCpBQw?e=opE639)
* YYY\_MM\_DD hh\_mm\_ssSugarSalesOrderHeaderExport\_YYYYMMDDThhmmss.csv
  + This csv file is the SugarSalesOrderHeaderExport\_YYYYMMDDThhmmss.csv renamed and moved to the directory \\sql08\p\SSIS\Data\Datafeed\SugarCRM\Completed.
  + Output of the Talend Job Orders
  + Example: [SugarSalesOrderHeaderExport\_20220613T235305.txt](https://summerclassics.sharepoint.com/:t:/s/ITDepartment/EVVoH1ZAtnhOuI4HK_HowpABX-tw4lMZxkxKd4rVGxHyoA?e=Fu0clS)
* YYY\_MM\_DD hh\_mm\_ssSugarSalesOrderLineExport\_YYYYMMDDThhmmss.csv
  + This csv file is the SugarSalesOrderLineExport\_YYYYMMDDThhmmss.csv renamed and moved to the directory \\sql08\p\SSIS\Data\Datafeed\SugarCRM\Completed.
  + Output of the Talend Job Order Line Items
  + Example: [SugarSalesOrderLineExport\_20220613T235309.txt](https://summerclassics.sharepoint.com/:t:/s/ITDepartment/EaotGHwiulVPnyNlhcFIeFoBGVQOxS-qXixm8noOMR326Q?e=AXrjMQ)

## Talend Log Files

* YYYY\_MM\_DD\_hh\_mm\_ss\_nn\_logs\_Master.csv
  + Example: [2022\_07\_12\_12\_36\_00\_logs\_Master.csv](https://summerclassics.sharepoint.com/:x:/s/ITDepartment/EbChkh4jLMFAj9BKcFeUNJ4Bijv-X4aecQmHF4nrPZR1Mw?e=pLjO0Y)
  + Logs high level events:
    - Start
    - Stop
    - Error Occurred in Child Job
* YYYY\_MM\_DD\_hh\_mm\_ss\_nn\_logs\_Accounts.csv
  + Example: [2022\_07\_12\_13\_21\_08\_logs\_Accounts.csv](https://summerclassics.sharepoint.com/:x:/s/ITDepartment/EUHerpkH_rJAmiLkGs2iwr8BCI1_S6wMZ8xEESKJaaSrXw?e=kHroJu)
  + Logs events for the Account job.
* YYYY\_MM\_DD\_hh\_mm\_ss\_nn\_logs\_Quotes.csv
  + Example: [2022\_07\_12\_13\_21\_32\_logs\_Quotes.csv](https://summerclassics.sharepoint.com/:x:/s/ITDepartment/ESqaEhY4YfRMhCb7vZzW-jABcrmY1nDdzL3HmduOQey0Tg?e=QXd7GZ)
  + Logs events for the Quotes job.
* YYYY\_MM\_DD\_hh\_mm\_ss\_nn\_logs\_Quote\_Line\_Items.csv
  + Example: [2022\_07\_12\_13\_22\_09\_logs\_Quote\_Line\_Items.csv](https://summerclassics.sharepoint.com/:x:/s/ITDepartment/EZplzkCxMsZFjrpwm1qB0acBDVBZ6M3FRSx_nGXyw1rqvQ?e=LV5e6y)
  + Logs events for Quote Line Items
* YYYY\_MM\_DD\_hh\_mm\_ss\_nn\_logs\_Orders.csv
  + Example: [2022\_07\_12\_13\_55\_54\_logs\_Orders.csv](https://summerclassics.sharepoint.com/:x:/s/ITDepartment/EbW4LsdPYF9MmrTmD1iddQQBmiFt46EbeVF6ctUMESc0Qg?e=rGxrlW)
  + Logs events for the Orders job.
* YYYY\_MM\_DD\_hh\_mm\_ss\_nn\_logs\_Order\_Line\_Items.csv
  + Example: [2022\_07\_12\_13\_57\_58\_logs\_Order\_Line\_Items.csv](https://summerclassics.sharepoint.com/:x:/s/ITDepartment/EcZKCd7ZhAlAuREEDykuRm8BM-S9I_7kBscNaQLukXi-1w?e=079O5F)
  + Logs events for the Order Line Items job.

# Detailed Design

## 4.1 Software Detailed Design

### 4.2.1 SQL Job – Sugar CRM Export

The Sugar CRM Export SQL Job is stored on SQL08 and runs every forty-five minutes between 6:00 am and 8:00 pm. The job has the following steps:

1. SguarCRM Export Customers  
   Runs the SSIS Package Sugar CRM Export with the parameters:
   1. SiteName: SugarCRM
   2. DatasetType: Customers
2. SguarCRM Export Sales Order Header  
   Runs the SSIS Package Sugar CRM Export with the parameters:
   1. SiteName: SugarCRM
   2. DatasetType: SalesOrder\_Header
3. SguarCRM Export Sales Order Line  
   Runs the SSIS Package Sugar CRM Export with the parameters:
   1. SiteName: SugarCRM
   2. Dataset Type: SalesOrder\_Line
4. SguarCRM Export Quote Header  
   Runs the SSIS Package Sugar CRM Export with the parameters:
   1. SiteName: SugarCRM
   2. DatasetType: Quote\_Header
5. SguarCRM Export Quote Detail  
   Runs the SSIS Package Sugar CRM Export with the parameters:
   1. SiteName: SugarCRM
   2. DatasetType: Quote\_Detail
6. Log Export File Creation  
   Executes the procedure [PRODUCT\_INFO].[SugarCrm].[LogExportFileCreation]

### 4.2.2 SSIS Package – Sugar CRM Export

Graphical user interface

Description automatically generated

Figure 3 - SSIS Package Sugar CRM Export

The above picture is a look of the design view for the SSIS Package Sugar CRM Export. This package excepts parameters SiteName and DatasetType. Based on the DatasetType, the Package will select the corresponding export process type and call the corresponding update, query, csv creation and flag record logic boxes. If an error occurs, the SQL Log Success-Failure Reason block will execute [PRODUCT\_INFO].[SugarCrm].[usp\_WriteLog\_Export]. The CSV file structure can be seen at 3.2.1 SSIS CS3.2.1V File. These files are saved to the directory \\sql08\p\SSIS\Data\Datafeed\SugarCRM\Active.

### 4.2.3 Talend Job – Sugar Master Integration

4.2.3.1 Task Scheduler – Sugar\_Master\_Integeration

The Talend Job is set up as a Windows Task on the IIS server. The task executes the following batch file:

C:\Talend\Archived Jobs\Sugar\_Master\_0.1\_MR\_20220113\Sugar\_Master\_New\Sugar\_Master\_New\_run.bat

The task is scheduled to run every five minutes. This initiates the Talend processing.

4.2.3.2 Talend – Sugar\_Master\_New

The following is the Design view of the job:

Diagram

Description automatically generated

Figure 4 - Sugar\_Master\_New

The job handles all interactions with SugarCRM. It also handles the order in which the export process is arranged. The main export jobs are Accounts, Quotes, Quote\_Line\_Items, Orders, and Order\_Line\_Items. The following jobs are maintenance jobs: UpdateOrphanQuoteLinks, OrdersHeadersStagingTruncate, OrdersHeadersStagingUpdate, and OrdersHeadersRollupLineItemSales. Each of these jobs will be explained in further detail.

tLogCatcher\_1 catches the messages from tWarn\_1 and tWarn\_2. tFileOutputDelimited\_1 creates the main log file. Details of this log file can be found in 3.2.3 Talend Log Files.

4.2.3.3 Sugar\_SysPro\_Accounts\_V1\_Prod\_Master

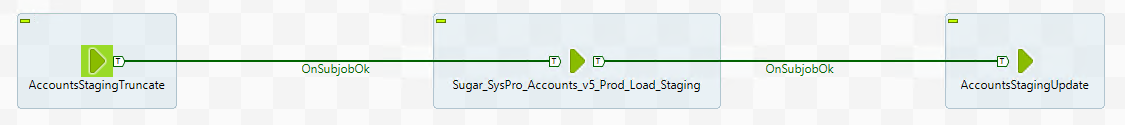


Figure 5 - Sugar\_SysPro\_Accounts\_V1\_Prod\_Master

This job is the Accounts job from Sugar\_Master\_New. It calls jobs associated with accounts in SugarCRM. The main export job is Sugar\_SysPro\_Accounts\_v5\_Prod\_Load\_Staging. The jobs AccountsStagingTruncate and AccountsStagingUpdate are maintenance routines.

4.2.3.3.1 AccountsStagingTruncate

A picture containing diagram

Description automatically generated

Figure 6 – AccountsStagingTruncate

This job’s main purpose is to execute the following SQL using the SugarCRM API endpoint ‘/queryToDb’.

SQL: Truncate table as\_accounts\_staging; Truncate table as\_accounts\_staging\_cstm;

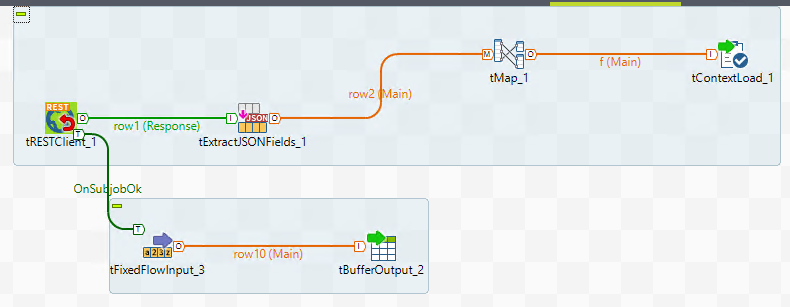


Figure 7 - tRunJob\_1

Figure 6 shows the logical block of tRunJob\_1. This shows one way in how the Sugar\_Master\_New Talend job requests a security token from SugarCRM. The routine uses the SugarCRM API endpoint ‘/oauth2/token’ with the following parameters: grant\_type, client\_id, client\_secret, username, password, and base. The routine stores the authority token within the global context and is used in proceeding API calls. The authority token is required for an API endpoints that are called throughout the Talend Job.

4.2.3.3.2 Sugar\_SysPro\_Accounts\_v5\_Prod\_Load\_Staging

Diagram

Description automatically generated with medium confidence

Figure 8 - Job Sugar\_SysPro\_Accounts\_v5\_Prod\_Load\_Staging

This job is the main work to import the SugarCustomerExports csv into SugarCRM. This link ([Job Sugar\_SysPro\_Accounts\_v5\_Prod\_Load\_Staging 0.1.png](https://summerclassics.sharepoint.com/:i:/s/ITDepartment/EU7bfDdMGtdElNhDekmga_4B9cdTsqxiG8k3s6htjs4hnA?e=NfZ08e)) can be used to get a better view of Figure7. The process follows the following steps that illustrate the main process:

1. Read in SugarCustomerExports csv
2. Pull down related SugarCRM Records
   1. Accounts
   2. Users
3. Compile the csv data with SugarCRM data
4. Import data into SugarCRM
5. Final files clean up and update SugarCRM Accounts records.

The following blocks can read in the SugarCustomerExports csv and store it into memory: tFileList\_2, tFileList\_1, tFileCopy\_1, tFixedFlowInput\_1, tNormalize\_1, tFileInputDelimited\_2, tMap\_7, file\_data. tFileList\_2 checks the /Processing directory to ensure that no other job is running. tFielList\_1 checks the /Active directory for a csv file to process while tFileCopy\_1 copies the file over to the /Processing directory. tFileInputDeliminited\_2 reads the csv, tMap\_7 maps the columns of the csv to the data structure that will be used within file\_data. tFileOutputDeliminited\_1 captures rows that were errored out on tFielInputDelimited\_2 and records the row within the Rejected\_Records csv. tFileOutputDelminited\_2 catches rows from tMap\_7 that were found to have a null or empty string for the Customer value and records the row within the Rejected\_Records csv. If tFileInputDeliminited\_2 completely errors out, the job ends by copying the process csv into the /Rejected directory by tFileCopy\_2, delete the process csv by tFileDelete\_3, and delete the active csv by tFileDelete\_4.

The next major block of logic retrieves information from Sugar using the end point ‘getQueryResultFromDb’. The block get\_accounts loop through and retrieves all accounts using the following sql statement:

"select accounts.id , accounts.name , accounts\_cstm.account\_number\_c from accounts left join accounts\_cstm on accounts.id = accounts\_cstm.id\_c where accounts.deleted = 0 order by accounts.id asc limit 20000 offset " + context.offset

The block get\_users execute the following sql statement to retrieve users' data:

select users.id, users.user\_name, users.first\_name, users.last\_name, ifnull(concat(users.first\_name, ' ', users.last\_name),users.last\_name) as full\_name, users\_cstm.syspro\_uid\_c, lower(ea.email\_address) as email from users left join users\_cstm on users.id = users\_cstm.id\_c left join email\_addr\_bean\_rel eabr on eabr.bean\_id = users.id left join email\_addresses ea on eabr.email\_address\_id = ea.id where users.deleted = 0 and eabr.deleted = 0 and ea.deleted = 0 and eabr.primary\_address = 1

With the Accounts and Users data we just retrieved from SugarCRM, we are going to update the csv data with SugarCRM specific data. tMap\_1 through tMap\_8 combines the data and then we save it into all\_rec.

Then the system uses the all\_rec information and forms it into a Json format for the BulkCRUD end point. The process loops through all records. Once completed, the process cleans up the files by copying the processing csv to the /Completed directory by tFileCopy\_3, deletes the processing csv by tFileDelete\_1, and deletes the active csv by tFileDelete\_5. Lastly, we execute the following sql by update\_accounts:

update accounts set team\_id =1, team\_set\_id =1 where created\_by = '3c67fe18-9716-4a20-9831-5ee4a9eb098c' and team\_id is null and team\_set\_id is null;

tLogCatcher\_1 catches all tWarn and tDie blocks throughout the job to generate the Accounts log file detailed in 3.2.3 Talend Log Files.

4.2.3.3.3 AccountsStagingUpdate

Timeline

Description automatically generated with medium confidence

Figure 9 - AccountsStagingUpdate

This job just executes the following SQL statements:

truncate table as\_accounts\_staging\_dup; insert into as\_accounts\_staging\_dup(recs,max\_id,account\_number\_c) select count(\*) recs, max(id) max\_id,account\_number\_c from as\_accounts\_staging an inner join as\_accounts\_staging\_cstm ac on a.id=ac.id\_c where a.deleted=0 group by ac.account\_number\_c having count(\*)>1; update as\_accounts\_staging\_cstm ac inner join as\_accounts\_staging a on ac.id\_c=a.id inner join as\_accounts\_staging\_dup ad on ac.account\_number\_c=ad.account\_number\_c and ad.max\_id<> a.id set a.deleted=1; insert into accounts(id,date\_entered,date\_modified,created\_by,modified\_user\_id,deleted,assigned\_user\_id,team\_id,team\_set\_id) select t.id,utc\_timestamp(),utc\_timestamp(),st.created\_by,st.modified\_user\_id,st.deleted,st.assigned\_user\_id,'1','1' from as\_accounts\_staging\_cstm stc inner join as\_accounts\_staging st on stc.id\_c=st.id left join accounts\_cstm ac on stc.account\_number\_c=ac.account\_number\_c left join accounts a on ac.id\_c=a.id where a.id is null and st.deleted=0 and not st.id in(select id from accounts); insert into accounts\_cstm(id\_c,account\_number\_c) select st.id,stc.account\_number\_c from as\_accounts\_staging\_cstm stc inner join as\_accounts\_staging st on stc.id\_c=st.id left join accounts\_cstm ac on ac.account\_number\_c=stc.account\_number\_c where ac.id\_c is null and st.deleted = 0 and not stc.id\_c in(select id\_c from accounts\_cstm); update accounts an inner join accounts\_cstm ac on a.id=ac.id\_c inner join as\_accounts\_staging\_cstm stc on ac.account\_number\_c=stc.account\_number\_c inner join as\_accounts\_staging st on stc.id\_c=st.id set a.account\_type=st.account\_type, ac.account\_type\_ms\_c=case when ifnull(stc.account\_type\_ms\_c,'')='' then '' else concat('^',stc.account\_type\_ms\_c,'^') end, a.billing\_address\_street=st.billing\_address\_street, a.billing\_address\_city=st.billing\_address\_city, a.billing\_address\_state=st.billing\_address\_state, a.billing\_address\_postalcode=st.billing\_address\_postalcode, a.billing\_address\_country=st.billing\_address\_country, a.shipping\_address\_street=st.shipping\_address\_street, a.shipping\_address\_city=st.shipping\_address\_city, a.shipping\_address\_state=st.shipping\_address\_state, a.shipping\_address\_postalcode=st.shipping\_address\_postalcode, a.shipping\_address\_country=st.shipping\_address\_country, ac.branch\_c=stc.branch\_c, ac.channel\_c=case when ifnull(stc.channel\_c,'')='' then '' else concat('^',stc.channel\_c,'^') end, ac.user\_id\_c=ifnull(csr\_c,''), a.date\_modified=st.date\_modified, a.description=st.description, ac.discount\_level\_c=stc.discount\_level\_c, ac.lead\_source\_c=stc.lead\_source\_c, a.name=st.name, a.phone\_office=st.phone\_office, ac.primary\_contact\_c=stc.primary\_contact\_c, ac.user\_id1\_c=ifnull(salesperson1\_c,''), ac.user\_id2\_c=ifnull(salesperson2\_c,''), ac.user\_id3\_c=ifnull(salesperson3\_c,''), ac.tax\_exempt\_id\_c=stc.tax\_exempt\_id\_c, a.deleted=0 where st.deleted=0;"  
"insert into email\_addresses(id,email\_address,email\_address\_caps,invalid\_email,opt\_out,date\_created,date\_modified,deleted) select distinct uuid(),email,upper(email),0,0,utc\_timestamp(),utc\_timestamp(),0 from as\_accounts\_staging an inner join as\_accounts\_staging\_cstm ac on a.id=ac.id\_c left join email\_addresses e on a.email=e.email\_address and e.deleted=0 where ifnull(a.email,'')<>'';  
insert into email\_addr\_bean\_rel(id,email\_address\_id,bean\_id,bean\_module,date\_modified,deleted) select distinct uuid(),e.id,a.id,'Accounts',utc\_timestamp(),0 from accounts aa inner join accounts\_cstm aac on aa.id=aac.id\_c inner join as\_accounts\_staging\_cstm ac on aac.account\_number\_c=ac.account\_number\_c inner join as\_accounts\_staging a on ac.id\_c=a.id and a.deleted=0 inner join email\_addresses e on a.email=e.email\_address and e.deleted=0 where ifnull(a.email,'')<>'';

4.2.3.3 Sugar\_eCat\_Quotes\_v6\_Prod

Graphical user interface

Description automatically generated with medium confidence

Figure 10 - Sugar\_eCat\_Quotes\_v6\_Prod

This job is the Quotes job from Sugar\_Master\_New. This link ([Sugar\_eCat\_Quote\_Line\_Items\_v6\_Prod\_QuoteDetail\_Fix.png](https://summerclassics.sharepoint.com/:i:/s/ITDepartment/EX2TsB6FQ4pIpIGHPSXjAZ4BTwcc7PtIIpcwVgN4pg8njQ?e=xgZmxq)) can be used to get a better view of Figure 9. This job follows a similar process with 4.2.3.3.2 Sugar\_SysPro\_Accounts\_v5\_Prod\_Load\_Staging:

1. Read in the Quotes CSV file
2. Pull down Quote related data from SugarCRM
   1. Quotes
   2. Users
   3. Accounts
   4. Leads
3. Compile CSV data and SugarCRM data
4. Import data into SugarCRM
5. Final files cleanup

The job reads in the SugarQuoteHeaderExport csv file from and stores into memory. This process runs as follows:

1. Confirm there is not a job running by checking the /Processing directory
2. Check if there is a SugarQuoteDetailExport csv file in the /Active directory
3. Copy SugarQuoteDetailExport csv file into the /Processing directory
4. Read csv file
   1. Line error on read creates a Rejected Record
   2. Null value for Quote Number creates a Rejected Record
   3. Completed error on read moves the entire file in /Processing into /Rejected directory

Pulling down SugarCRM data is just directly querying the SugarCRM database using the ‘getQueryResultFromDb’ endpoint. The following are the sql statements for pulling down the data:

1. Quotes  
   This query is used in conjunction with a loop after getting the count of all quotes records.  
   select id, name from wquo1\_quotes where deleted = 0 order by id asc limit 20000 offset context.offset
2. Users  
   This is the query that is used to get users data:  
   select users.id, users.user\_name, users.first\_name, users.last\_name, ifnull(concat(users.first\_name, ' ', users.last\_name),users.last\_name) as full\_name, users\_cstm.syspro\_uid\_c, lower(ea.email\_address) as email from users left join users\_cstm on users.id = users\_cstm.id\_c left join email\_addr\_bean\_rel eabr on eabr.bean\_id = users.id and eabr.deleted = 0 and eabr.primary\_address = 1 left join email\_addresses ea on eabr.email\_address\_id = ea.id and ea.deleted = 0 where users.deleted = 0;
3. Accounts  
   This query is used in conjunction with a loop after getting the count of all accounts records.  
   select accounts.id, accounts.name, accounts\_cstm.account\_number\_c, accounts.billing\_address\_street, accounts.billing\_address\_state, accounts.billing\_address\_postalcode, accounts.billing\_address\_country from accounts left join accounts\_cstm on accounts.id = accounts\_cstm.id\_c where accounts.deleted = 0 and accounts\_cstm.account\_number\_c is not null order by accounts.id asc limit 20000 offset context.offset
4. Leads  
   This query is used in conjunction with a loop after getting the count of all leads records.  
   select l.id, l.first\_name, l.last\_name,l.primary\_address\_city,l.primary\_address\_country,l.primary\_address\_postalcode,l.primary\_address\_state,l.primary\_address\_street,lower(ea.email\_address) as email from leads l left join email\_addr\_bean\_rel eabr on eabr.bean\_id = l.id and eabr.deleted = 0 and eabr.primary\_address = 1 left join email\_addresses ea on eabr.email\_address\_id = ea.id and ea.deleted = 0 where l.deleted = 0 order by l.id asc limit 20000 offset context.offset

The job then complies with the data together and imports the data into SugarCRM. Lastly the job moves the Processing csv into the /Completed directory and cleans up the /Active and /Processing directory.

Like Accounts there is logging to catch Warn and Die logic blocks.

4.2.3.4 Sugar\_eCat\_Quote\_Line\_Items\_v6\_Prod\_QuoteDetail\_Fix

A picture containing chart

Description automatically generated

Figure 11 - Sugar\_eCat\_Quote\_Line\_Items\_v6\_Prod\_QuoteDetail\_Fix

This job is the Quote\_Line\_Items job from Sugar\_Master\_New. This link ([Sugar\_eCat\_Quote\_Line\_Items\_v6\_Prod\_QuoteDetail\_Fix.png](https://summerclassics.sharepoint.com/:i:/s/ITDepartment/EX2TsB6FQ4pIpIGHPSXjAZ4BTwcc7PtIIpcwVgN4pg8njQ?e=ZkfabT)) can be used to get a better view of Figure 10. This job follows a similar process with 4.2.3.3.2 Sugar\_SysPro\_Accounts\_v5\_Prod\_Load\_Staging:

1. Read in the Quotes Line Items CSV file
2. Pull down Quote Line Items data from SugarCRM
3. Compile CSV data and SugarCRM data
4. Import data into SugarCRM
5. Final files cleanup

This job reads in the SugarQuoteDetailExport csv file like the previous two jobs. This job only pulls down Quote Line Items data using the following sql statement:

select distinct q.assigned\_user\_id , q.id as quote\_id, q.name as quote\_name, ifnull(qli.id, '-1') as quote\_line\_item\_id, ifnull(qli.name, '-1') as quote\_line\_item\_name from wquo1\_quotes q left join wquo1\_quotes\_wqli\_quotedlineitems\_1\_c qliq on q.id = qliq.wquo1\_quotes\_wqli\_quotedlineitems\_1wquo1\_quotes\_ida and qliq.deleted = 0 left join wqli\_quotedlineitems qli on qliq.wquo1\_quotes\_wqli\_quotedlineitems\_1wqli\_quotedlineitems\_idb = qli.id and qli.deleted = 0 where q.deleted = 0 order by q.id asc limit 20000 offset context.offset

This job has two calls to bulkCRUD with first deleted Quote\_Line\_Items and then updated and added Quote\_Line\_Items. This job does the same file cleaning as the other jobs. Lastly, the job executes the following sql on SugarCRM:

update wqli\_quotedlineitems set team\_id =1, team\_set\_id =1 where created\_by = '3c67fe18-9716-4a20-9831-5ee4a9eb098c' and team\_id is null and team\_set\_id is null;

4.2.3.5 Sugar\_SysPro\_Orders\_v6\_Prod

A picture containing chart

Description automatically generated

Figure 12 - Sugar\_SysPro\_Orders\_v6\_Prod

This job is the Orders job from Sugar\_Master\_New. This link ([Sugar\_SysPro\_Orders\_v6\_Prod.png](https://summerclassics.sharepoint.com/:i:/s/ITDepartment/EZOFjI2ti3tNj8wvQhqyyI8BbJvDeAP1o8pxCFJbyB0lVg?e=AlE3hE)) can be used to get a better view of Figure 11. This job follows a similar process to other jobs:

1. Read in the Orders CSV file
2. Pull down Order related data from SugarCRM
   1. Quotes
   2. Orders
   3. Users
   4. Accounts
3. Compile CSV data and SugarCRM data
4. Import data into SugarCRM
5. Final files cleanup

The job reads in the SguarSalesORderHeaderExport csv file and stores it into memory. Then pulls in down SugarCRM using the following sql statements:

Accounts:  
select accounts.id, accounts.name, accounts\_cstm.account\_number\_c, accounts.billing\_address\_street, accounts.billing\_address\_state, accounts.billing\_address\_postalcode, accounts.billing\_address\_country from accounts left join accounts\_cstm on accounts.id = accounts\_cstm.id\_c where accounts.deleted = 0 and accounts\_cstm.account\_number\_c is not null order by accounts.id asc limit 20000 offset context.offset

Orders:  
select id, name, assigned\_user\_id from wso1\_orders where deleted = 0 order by id asc limit 20000 offset context.offset

Quotes:  
select id, name from wquo1\_quotes where deleted = 0 order by id asc limit 20000 offset context.offset

Users:  
select users.id, users.user\_name, users.first\_name, users.last\_name, ifnull(concat(users.first\_name, ' ', users.last\_name),users.last\_name) as full\_name, users\_cstm.syspro\_uid\_c, lower(ea.email\_address) as email from users left join users\_cstm on users.id = users\_cstm.id\_c left join email\_addr\_bean\_rel eabr on eabr.bean\_id = users.id left join email\_addresses ea on eabr.email\_address\_id = ea.id where users.deleted = 0 and eabr.deleted = 0 and ea.deleted = 0 and eabr.primary\_address = 1

The job continues with compiling the SugarCRM data with the csv data and sending it in to SugarCRM via the BulkCRUD endpoint. Once done with transmitting the data, the job cleans up the files within the directory and then sends the following sql update:

update wso1\_orders set team\_id =1, team\_set\_id =1 where created\_by = '3c67fe18-9716-4a20-9831-5ee4a9eb098c' and team\_id is null and team\_set\_id is null;

4.2.3.6 Sugar\_SysPro\_Order\_Line\_Items\_v3\_Prod

Diagram

Description automatically generated

Figure 13 - Sugar\_SysPro\_Order\_Line\_Items\_v3\_Prod

This job is the Order\_Line\_Items job from Sugar\_Master\_New. This link ([Sugar\_eCat\_Quote\_Line\_Items\_v6\_Prod\_QuoteDetail\_Fix.png](https://summerclassics.sharepoint.com/:i:/s/ITDepartment/EX2TsB6FQ4pIpIGHPSXjAZ4BTwcc7PtIIpcwVgN4pg8njQ?e=ZCtXWy)) can be used to get a better view of Figure 12. This job follows a similar process with 4.2.3.4 Sugar\_eCat\_Quote\_Line\_Items\_v6\_Prod\_QuoteDetail\_Fix:

1. Read in the Order Line Items CSV file
2. Pull down Order Line Items data from SugarCRM
3. Compile CSV data and SugarCRM data
4. Import data into SugarCRM
5. Final files cleanup

The job reads in the SugarSalesOrderLineExport csv file and stores it into memory. The job pulls down SugarCRM data using the following sql:

select o.assigned\_user\_id, o.id as order\_id, o.name as order\_name, ifnull(oli.id, '-1') as order\_line\_item\_id, ifnull(oli.name, '-1') as order\_line\_item\_name, olic.initial\_line\_number\_c as initial\_line\_number from wso1\_orders o left join wso1\_orders\_woli1\_orderlineitems\_1\_c olio on o.id = olio.wso1\_orders\_woli1\_orderlineitems\_1wso1\_orders\_ida left join woli1\_orderlineitems oli on olio.wso1\_orders\_woli1\_orderlineitems\_1woli1\_orderlineitems\_idb = oli.id left join woli1\_orderlineitems\_cstm olic on oli.id = olic.id\_c where o.deleted = 0 and ifnull(olio.deleted, 0) = 0 and ifnull(oli.deleted, 0) = 0 order by o.id asc limit 20000 offset context.offset

The job then compiles the data and sends it to SugarCRM using the BulkCrud endpoint. Lastly, the job cleans up the files within the directories and executes the following sql on SugarCRM:

update woli1\_orderlineitems set team\_id =1, team\_set\_id =1 where created\_by = '3c67fe18-9716-4a20-9831-5ee4a9eb098c' and team\_id is null and team\_set\_id is null;

4.2.3.7 UpdateOrphanQuoteLinks

This job is set up like the job in 4.2.3.3.3 AccountsStagingUpdate. The flow follows get access token and then call sql endpoint to run a sql statement. This job executes the following statement:

insert into wquo1\_quotes\_wqli\_quotedlineitems\_1\_c (id,date\_modified,deleted,wquo1\_quotes\_wqli\_quotedlineitems\_1wquo1\_quotes\_ida ,wquo1\_quotes\_wqli\_quotedlineitems\_1wqli\_quotedlineitems\_idb) select distinct uuid(),utc\_timestamp(),0,q2.id, qli.id from wqli\_quotedlineitems qli inner join wqli\_quotedlineitems\_cstm qc on qli.id=qc.id\_c inner join wquo1\_quotes q2 on qc.quote\_header\_num\_c=q2.name and q2.deleted=0 left join wquo1\_quotes\_wqli\_quotedlineitems\_1\_c qliq on qli.id = qliq.wquo1\_quotes\_wqli\_quotedlineitems\_1wqli\_quotedlineitems\_idb and qliq.deleted = 0 left join wquo1\_quotes q on wquo1\_quotes\_wqli\_quotedlineitems\_1wquo1\_quotes\_ida =q.id and q.deleted=0 where qli.deleted = 0 and q.id is null;

4.2.3.8 ORdersHeadersStagingTruncate

This job is set up like the job in 4.2.3.3.3 AccountsStagingUpdate. The flow follows get access token and then call sql endpoint to run a sql statement. This job executes the following statement:

truncate table so002\_orders\_staging;  
truncate table so002\_orders\_staging\_cstm;

4.2.3.9 OrdersHeadersStagingUpdate

This job is set up like the job in 4.2.3.3.3 AccountsStagingUpdate. The flow follows get access token and then call sql endpoint to run a sql statement. This job executes the following statement:

insert into wso1\_orders(id,name,date\_entered,date\_modified,created\_by,modified\_user\_id,team\_id,team\_set\_id,assigned\_user\_id,deleted) select so.id,so.name,utc\_timestamp(),utc\_timestamp(),'1','1','1','1','1',0 from so002\_orders\_staging so left join wso1\_orders s on so.name=s.name and s.deleted=0 where s.id is null and so.deleted=0;  
insert into wso1\_orders\_cstm(id\_c) select id from wso1\_orders s left join wso1\_orders\_cstm c on s.id=c.id\_c where c.id\_c is null and s.deleted=0;  
update wso1\_orders o inner join wso1\_orders\_cstm c on o.id=c.id\_c inner join so002\_orders\_staging s on o.name=s.name inner join so002\_orders\_staging\_cstm sc on s.id=sc.id\_c set c.order\_date\_c=sc.orderdate\_c, c.account\_number\_c=sc.customer\_c, c.base\_rate=1.0, c.shipping\_address\_state\_c=right(sc.shipaddress3\_c,2), c.shipping\_address\_city\_c=left(sc.shipaddress3\_c,length(sc.shipaddress3\_c)-4), c.shipping\_address\_street\_c=trim(concat(ifnull(sc.shipaddress1\_c,''),' ',ifnull(sc.shipaddress2\_c,''))), c.shipping\_address\_postalcode\_c=sc.shippostalcode\_c, c.order\_status\_c=sc.orderstatus\_c, c.order\_date\_c=sc.orderdate\_c, c.branch\_c=sc.branch\_c, c.brand\_c=sc.brand\_c, c.market\_segment\_c=sc.marketsegment\_c, c.noearlierthandate\_c=sc.noearlierthandate\_c, c.nolaterthandate\_c=sc.nolaterthandate\_c, c.shipment\_request\_c=sc.shipmentrequest\_c, c.po\_number\_c=sc.customerponumber\_c, o.description=sc.customertag\_c where o.deleted=0 and s.deleted=0;  
insert into accounts\_wso1\_orders\_1\_c(id,date\_modified,deleted,accounts\_wso1\_orders\_1accounts\_ida,accounts\_wso1\_orders\_1wso1\_orders\_idb) select distinct uuid(),utc\_timestamp(),0,a.id,s.id from so002\_orders\_staging ss inner join so002\_orders\_staging\_cstm ssc on ss.id=ssc.id\_c inner join wso1\_orders s on ss.name=s.name and s.deleted=0 inner join accounts\_cstm ac on ssc.customer\_c=ac.account\_number\_c inner join accounts a on ac.id\_c=a.id left join accounts\_wso1\_orders\_1\_c aw on a.id=aw.accounts\_wso1\_orders\_1accounts\_ida and s.id=accounts\_wso1\_orders\_1wso1\_orders\_idb and aw.deleted=0 where aw.id is null;  
update wso1\_orders\_cstm cc inner join wso1\_orders o on cc.id\_c=o.id and o.deleted=0 inner join so002\_orders\_staging s on o.name=s.name and s.deleted=0 inner join accounts\_wso1\_orders\_1\_c aw on cc.id\_c=aw.accounts\_wso1\_orders\_1wso1\_orders\_idb and aw.deleted=0 set cc.account\_id\_c=aw.accounts\_wso1\_orders\_1accounts\_ida;  
update wso1\_orders\_cstm cc inner join wso1\_orders o on cc.id\_c=o.id and o.deleted=0 inner join so002\_orders\_staging s on o.name=s.name and s.deleted=0 inner join accounts\_wso1\_orders\_1\_c aw on cc.id\_c=aw.accounts\_wso1\_orders\_1wso1\_orders\_idb and aw.deleted=0 inner join accounts a on aw.accounts\_wso1\_orders\_1accounts\_ida=a.id and a.deleted=0 set cc.billing\_address\_street\_c=a.billing\_address\_street,cc.billing\_address\_city\_c=a.billing\_address\_city,cc.billing\_address\_state\_c=a.billing\_address\_state,cc.billing\_address\_postalcode\_c=a.billing\_address\_postalcode,cc.billing\_address\_country\_c=a.billing\_address\_country ;  
update wso1\_orders\_cstm cc inner join wso1\_orders o on cc.id\_c=o.id and o.deleted=0 inner join so002\_orders\_staging s on o.name=s.name and s.deleted=0 set cc.channel\_c=case when cc.branch\_c='240' then 'Ecommerce' when cc.branch\_c='200' then 'Wholesale' when cc.branch\_c='220' then 'Wholesale' when cc.branch\_c='210' then 'Contract' when cc.branch\_c='230' then 'Private Label' else 'Retail' end ;  
update wso1\_orders\_cstm cc inner join wso1\_orders o on cc.id\_c=o.id and o.deleted=0 inner join so002\_orders\_staging s on o.name=s.name and s.deleted=0 inner join so002\_orders\_staging\_cstm sc on s.id=sc.id\_c inner join email\_addresses e on sc.salesperson\_c=e.email\_address and e.deleted=0 inner join email\_addr\_bean\_rel eb on e.id=eb.email\_address\_id and eb.deleted=0 and bean\_module='Users' inner join users u on eb.bean\_id=u.id and u.deleted=0 set o.assigned\_user\_id=u.id ;  
insert into wquo1\_quotes\_wso1\_orders\_1\_c(id,date\_modified,deleted,wquo1\_quotes\_wso1\_orders\_1wquo1\_quotes\_ida,wquo1\_quotes\_wso1\_orders\_1wso1\_orders\_idb) select distinct uuid(),utc\_timestamp(), '0' as deleted,a.id,s.id from so002\_orders\_staging ss inner join so002\_orders\_staging\_cstm ssc on ss.id=ssc.id\_c inner join wso1\_orders s on ss.name=s.name and s.deleted=0 inner join wquo1\_quotes a on ssc.webordernumber\_c=a.id and a.deleted=0 left join wquo1\_quotes\_wso1\_orders\_1\_c aw on a.id=aw.wquo1\_quotes\_wso1\_orders\_1wquo1\_quotes\_ida and s.id=wquo1\_quotes\_wso1\_orders\_1wso1\_orders\_idb and aw.deleted=0 where aw.id is null;

4.2.3.10 OrdersHeadersRollupLineItemSales

This job is set up like the job in 4.2.3.3.3 AccountsStagingUpdate. The flow follows get access token and then call sql endpoint to run a sql statement. This job executes the following statement:

update wso1\_orders o inner join wso1\_orders\_cstm oc on o.id=oc.id\_c inner join (select sum(ext\_price\_c) total, wso1\_orders\_woli1\_orderlineitems\_1wso1\_orders\_ida order\_id from wso1\_orders\_woli1\_orderlineitems\_1\_c w inner join woli1\_orderlineitems ol on w.wso1\_orders\_woli1\_orderlineitems\_1woli1\_orderlineitems\_idb=ol.id and ol.deleted=0 inner join woli1\_orderlineitems\_cstm c on ol.id=c.id\_c group by wso1\_orders\_woli1\_orderlineitems\_1wso1\_orders\_ida) r on o.id=r.order\_id set subtotal\_price\_c=r.total where o.deleted=0;

## 4.3 Communications Detailed Design

The following endpoints are used to communicate with SugarCRM. The following is used as the base URL:

https://summerclassics.sugarondemand.com/rest/v11\_3

* /oauth2/token  
  Used to get the authority token to get credentials to access SugarCRM. The data uses the following data to POST:

Text, letter

Description automatically generated

Figure 14 - /oauth2/token Json

The following is the expected return Json:

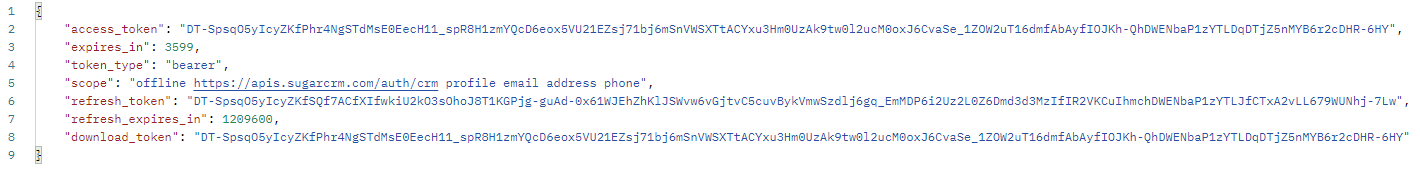


Figure 15 - /oauth2/token return Json

* /getQueryResultFromDb  
  Used to query data quicker from the SugarCRM DB. The following is the Json to POST:

Graphical user interface, text, application, email

Description automatically generated

Figure 16 - getQueryResultFromDb input Json

The return is dependent on what information that is returned but will be a Json object list of the query supplied.

* /bulkCrud  
  Used to make data related updates to SugarCRM. The following is the Json to POST:

Text

Description automatically generated

Figure 17 - bulkCrud Json

# Appendices

## Appendix A: Analysis Documentation

## Appendix B: Issues

## Appendix C: Other supporting documentation

1. [SugarCRM Table Structure.xlsx](https://summerclassics.sharepoint.com/:x:/s/ITDepartment/EdoJ94qVnixInH07Fl4BXx4BlkqZ7wMLg1lwsI5Y5whF-Q?e=phZeN3)  
   This document houses the structure of each table within SQL that is utilized by this integration.
2. [SugarSQLObjects](https://summerclassics.sharepoint.com/:f:/s/ITDepartment/EuCVqkI4J21OuEVEPWzrayYB5yXFZ9u33MjCLSKiYlYBdA?e=d45SJ3)  
   This directory contains scripts that define SQL objects.

## Appendix D: Benchmark Design and Development Estimate

Benchmark Design and Development estimate. Please list the task name, hours needed to complete, and a benchmark estimate date of completion.

| Task | Hours | Benchmark est. Date |
| --- | --- | --- |
|  |  |  |
|  |  |  |