fortimize

Client Leasing Project Technical Design

v1.0 Oct 8 2019

Yardi Integration

Salesforce is integrated with Yardi as a one way integration from Yardi to Salesforce using the Yardi Commercial API. The Yardi API is a SOAP API and the credentials for authenticating to it are stored in Custom Settings (and the InterfaceLicense which is stored in Custom Metadata). The data is retrieved in daily batch jobs and therefore updated in Salesforce every 24 hours.

There is a set of Apex classes that call the Yardi Commercial API and Common Data API. The Commercial API gives back a large XML response (sometimes as large as 10MB) holding most all the data for any property. The Yardi API classes retrieve and parse out this data, storing the results in a set of objects designed to mirror exactly the data model represented in the XML. This is useful for understanding how to process the Yardi data, and to debug with records that show the data as received from Yardi. These classes hold the response values, and some post processing calculations. Because of the size of the XML and the logic needed to process, the classes to retrieve from Yardi and parse must be run asynchronously and in batches to avoid CPU limits and timeouts.

There is another set of objects that represent the ideal data model for the requirements of the Client. These objects cherry pick the data needed from the Yardi objects and contain values from other systems such as Market SF from Link and data imported from other spreadsheets to give a consolidated view. They also offer the capability to have parent child Suite relationships to combine data that may reside in two or more separate Units in Yardi to show a combined Suite. In addition, the values in the Suite record may override and ignore that values from Yardi to account for the situation where a Suite has just been leased but accounting has not yet updated the information in Yardi.

Data Model

The set of objects used by Client to view Suite, Lease, and Account data, as well as FER values are:
Suite__c - Any suite from a property, it can have child suites, which rolls up values to the parent suite
Property__c - A property is one building which contains multiple suites
Lease__c - A lease is a contract for a suite or a set of many suites
Account - A Tenant Account is the Tenant for any lease

FER and Fully Escalated Rent

Data from Yardi Billing records is processed to calculate the Fully Escalated Rent and the FER per Market Square Feet.

For each Billing record at the unit level, if the Start Date is prior to today, and the End Date is after today, then the Billing record is considered current and should be used in calculating Fully Escalated Rent. If the End Date is null, it is treated as if the End date is today and it still applies. The code also checks for invalid end dates that are before the Start Date, and again it treats it as if the End Date is today and it still applies.

If the IncomeCategoryDescription contains '(R)', that is an indication that this record is one of the 20 different Base Rent charge codes. All such values for this unit (that are valid for today's date) are added together, using the ChargeRateAmount, which is a monthly value. The total is then multiplied by 12 for the Base Rent value on the Suite, and is one of the values that gets added for Fully Escalated Rent.

If the IncomeCategoryDescription contains '(TAX)', that is an indication that this record is one of the 8 different Real Estate Tax charge codes. All such values for this unit (that are valid for today's date) are added together, using the ChargeRateAmount, which is a monthly value. The total is then multiplied by 12 for the RE Tax value on the Suite, and is one of the values that gets added for Fully Escalated Rent.

If the IncomeCategoryDescription contains '(ERIF)', that is an indication that this record is one of the ERIF charge codes. All such values for this unit (that are valid for today's date) are added together, using the ChargeRateAmount, which is a monthly value. The total is then multiplied by 12 for the ERIF value on the Suite, but is **NOT** one of the values that gets added for Fully Escalated Rent.

If the ChargeCode = 'esccpi', that is an indication that this record is a CPI value. All such values for this unit (that are valid for today's date) are added together, using the ChargeRateAmount, which is a monthly value. The total is then multiplied by 12 for the CPI value on the Suite, and is one of the values that gets added for Fully Escalated Rent.

If the ChargeCode = 'escopx', that is an indication that this record is a OpEx value. All such values for this unit (that are valid for today's date) are added together, using the ChargeRateAmount, which is a monthly value. The total is then multiplied by 12 for the OpEx value on the Suite, and is one of the values that gets added for Fully Escalated Rent.

These 4 values (Base Rent, RE Tax, CPI, and OpEx) are all added together for the Fully Escalated Rent on the Suite. This value is then divided by the Market SF value (imported from Link) as is shown in the FER (Market SF) field on the Suite. This value is also divided by the Lease SF value from Yardi and shown as the FER (Lease SF) on the Suite.

Base Rent = sum of all current monthly R values x 12
RE Tax = sum of all current monthly Tax values x 12
CPI = sum of all current monthly CPI values x 12
OpEx = sum of all current monthly OpEx values x 12

Fully Escalated Rent = Base Rent + RE Tax + CPI + OpEx
FER (Market SF) = Fully Escalated Rent / Market SF

In some cases, Billing records are not associated at the suite level, but instead associated to a lease with multiple suites. In these cases, a pro-rated value is calculated, using the Market SF of each suite to determine the percent of the lease. Then the Base Rent, RE Tax, CPI, OpEx, and Fully Escalated Rent on each suite are a

percentage of the values from the lease. The FER per square feet values will be exactly the same for each suite in a lease in this case, since it is a rate per square feet which is then the same rate for each suite in the lease.

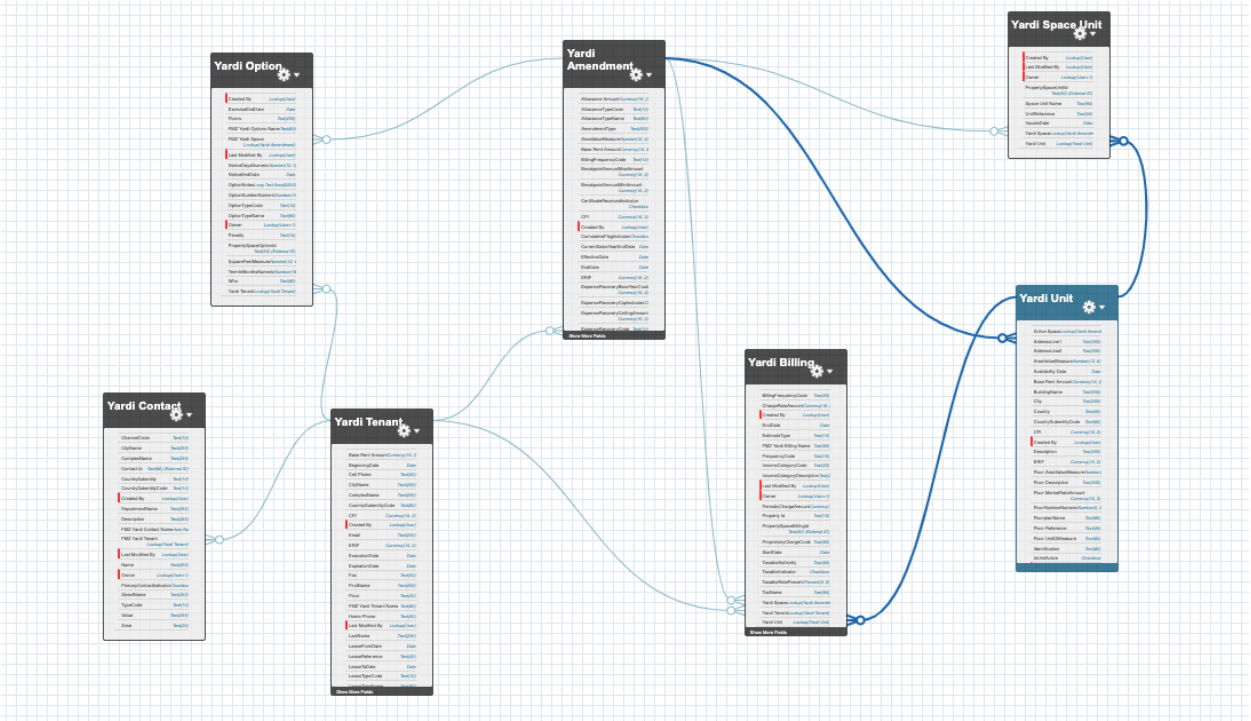
In some cases a suite may have child suites. In this case the Base Rent, RE Tax, CPI, OpEx, and Fully Escalated Rent is a total of all the child suites. The FER (Market SF) is then calculated divided by the sum of the Market SF of all the child suites.

Finally, the user may choose to enter a FER override on any Base Rent, RE tax, CPI, Op Ex, or ERIF on any suite. If this is entered, the calculated value from the Yardi billing data is overridden and instead it shows this value and uses it to calculate the Fully Escalated Rent and FER/sq ft. The user must also populate the FER Override Reason to explain why an override was needed, a validation rule will enforce this.

Yardi Data Model

The set of objects used in Salesforce to represent the Yardi API XML data are:
FMZ_Yardi_Tenant__c - a Yardi Tenant
FMZ_Yardi_Space__c - a Yardi Space/Amendment/Lease
FMZ_Yardi_Unit__c - a Yardi Unit
FMZ_Yardi_Space_Unit__c - Junction object to connect multiple Units to multiple Spaces
FMZ_Yardi_Contact__c - a Yardi contact for a Tenant
FMZ_Yardi_Option__c - a Yardi option to renew/expand/relocation/modification etc
FMZ_Yardi_Billing__c - a Yardi record used to calculate FER

These objects just store the results from Yardi as well as summaries from the Billing records. They should not typically be used by Client users. They can be used to see the raw data that was received from Yardi and the post processing results from the billing records, and understand the source data for FER values.



Yardi Batch Jobs

There are three sets of daily scheduled Apex jobs that need to be set up in order to refresh all the Yardi data in Salesforce daily.

These jobs are:

- **FMZ_Schedule_Yardi_PreProcessing** - Handles cleanup by deleting all existing billing records, and options records so that we will no longer process old billing and option values that were deleted in Yardi
- **FMZ_Schedule_Get_Yardi_Data** - Retrieves the data from the Yardi API calls including Export_CommercialLeaseData, GetUnitInformation, and GetTenants. The Export_CommercialLeaseData is split into two Yardi interfaces, one for original amendments, and one for other amendments. These interfaces are both queried for each property. Export_CommercialLeaseData gives very large responses for esb and ogcp, so splitting these interfaces allows us to process the responses in Salesforce without hitting CPU limits, heap memory limits, or other internal Salesforce errors.
- **FMZ_Schedule_Yardi_PostProcessing** - Processes the billing responses to calculate FER values, relate Link Suites to Yardi units, creates Lease records for all Yardi Amendments (if not already present), creates Accounts for all Yardi Tenants (if not already present), and rolls up values from child suites to their parent suites. These methods are all in the FMZ_Yardi_Utils class and documented in more detail below.

Scheduling

In order to deploy the Apex classes to an org, any existing Yardi jobs need to first be deleted. After deployment, these jobs need to be rescheduled. This can be done by pasting these commands into the Execute Anonymous window in the Developer Console:

```
FMZ_Schedule_Yardi_PreProcessing sypre = new
FMZ_Schedule_Yardi_PreProcessing(); // Schedule Apex Class Name
String sch = '0 0 0 1/1 * ? *'; //schedule interval time
System.schedule('Yardi_PreProcessing_Nightly', sch, sypre); // System Method To
Schedule Apex Class
FMZ_Schedule_Get_Yardi_Data sgyd = new FMZ_Schedule_Get_Yardi_Data();
String crontab = '0 0 1 1/1 * ? *';
String jobID = System.Schedule('Get_Yardi_Data_Nightly', crontab, sgyd);
```

FMZ_Schedule_Yardi_PostProcessing sypp = new
FMZ_Schedule_Yardi_PostProcessing();
*crontab = '0 0 2 1/1 * ? *';*
jobID = System.Schedule('Yardi_PostProcessing_Nightly', crontab, sypp);

These commands, set up the pre processing for midnight, the get Yardi data for 1AM, and the post processing for 2AM.

Ad Hoc Execution

Below are commands that can be used in the Execute Anonymous window in the Developer Console to run ad hoc instead of waiting for the next scheduled daily run

Preprocessing - Deleting old data:
FMZ_Schedule_Yardi_PreProcessing sypp = new
FMZ_Schedule_Yardi_PreProcessing(); sypp.execute(null);

Get Yardi Data (both interfaces, all properties):
FMZ_Schedule_Get_Yardi_Data sgyd = new FMZ_Schedule_Get_Yardi_Data();
sgyd.execute(null);

Post Processing:
FMZ_Schedule_Yardi_PostProcessing sypp = new
FMZ_Schedule_Yardi_PostProcessing(); sypp.execute(null);

Get Yardi Data Implementation

The following commands are part of the Get Yardi Data scheduled job, and can be executed individually if necessary.

Call getUnits for all properties

String cronID = System.scheduleBatch(new
FMZ_Get_Yardi_Data(FMZ_Yardi_API_Utils.callType.getUnits.name()),
'Batch-' + FMZ_Yardi_API_Utils.callType.getUnits.name(), 1, 1);

Call getTenants for all properties

```
String cronID = System.scheduleBatch(new
FMZ_Get_Yardi_Data(FMZ_Yardi_API_Utils.callType.getTenants.name()),
                        'Batch-' + FMZ_Yardi_API_Utils.callType.getTenants.name(), 1, 1);
```

Call Export_CommercialLeaseData for all properties

```
String cronID = System.scheduleBatch(new
FMZ_Get_Yardi_Data(FMZ_Yardi_API_Utils.callType.getCommData.name()),
                        'Batch-' + FMZ_Yardi_API_Utils.callType.getCommData.name(), 1,
1);
```

Call Export_CommercialLeaseData for one property

```
String cronID = System.scheduleBatch(new
FMZ_Get_Yardi_Data(FMZ_Yardi_API_Utils.callType.getCommData.name(), 'esb'),
                        'Batch-' + FMZ_Yardi_API_Utils.callType.getCommData.name(), 1,
1);
```

Post Processing Implementation

The following commands are part of the Yardi Post Processing scheduled job, and can be executed individually if necessary.

FMZ_ProcessUnitBillingOccupancyBatch:

Database.executeBatch(new FMZ_ProcessUnitBillingOccupancyBatch(), 200);

Processes All Units to update Unit Billing and Occupancy Data

Maps Suites To Units (constructor)
Processes Unit BillingOccupancy Data for each Unit (execute)
Detects Updates Occupancy For Units (execute)

Called in finish() method of FMZ_Get_Yardi_Data

Database.executeBatch(new FMZ_ProcessUnitBillingOccupancyBatch(), 200);

FMZ_ProcessSpaceBillingBatch:

Database.executeBatch(new FMZ_ProcessSpaceBillingBatch(), 200);

Processes All Spaces to Update Space Billing Data (execute)
Maps Spaces To Leases (finish)

Called in finish() method of FMZ_ProcessUnitBillingOccupancyBatch

Database.executeBatch(new FMZ_ProcessSpaceBillingBatch(), 200);

FMZ_MapTenantsToAccountsBatch:

Database.executeBatch(new FMZ_MapTenantsToAccountsBatch(), 200);

Maps Tenants To Accounts in Batch (execute)

Called in finish() method of FMZ_ProcessSpaceBillingBatch

Database.executeBatch(new FMZ_MapTenantsToAccountsBatch(), 200);

/* Deprecated ***/**

FMZ_ProcessPropertyBillingOccupancyBatch:

Database.executeBatch(new FMZ_ProcessPropertyBillingOccupancyBatch(), 1);

Processes All Properties where Yardi Property Id != null <blank> to update Properties - Unit/Space Billing and Occupancy Data.

Maps Suites To Units
Processes Property BillingOccupancyData for each Property
Maps Spaces To Leases

Called in finish() method of FMZ_Get_Yardi_Data

Database.executeBatch(new FMZ_ProcessPropertyBillingOccupancyBatch(), 1);
BatchSize of 1 - to process each Property individually / avoid CPU Timeout(s)

Custom Apex Code

FMZ_Yardi_API class

This class contains the base code to send queries to the Yardi SOAP API and parse the XML responses received. The data is saved in a data model that matches the format of the XML as closely as possible with very little processing to avoid limits and timeouts on these sometimes large XML responses.

FMZ_Yardi_Util class methods

These methods in FMZ_Yardi_Util are used by the post processing scheduled jobs. In some cases multiple wrappers for methods or parameters were implemented to better allow for testing of each method individually just by passing in an ID. For efficiency, the scheduled jobs will pass in objects already in memory to avoid unnecessary SOQL queries.

processLeaseCharges - DEPRECATED, uses the LeaseCharge values from the GetTenantData API to add up values for FER calculations. These values are at the Tenant level, so they are not good for calculating unit or space level FER. Instead we are now using the Billing values from the Export_CommercialLeaseData API per recommendation from Yardi.

processBilling - Called by the processTenantBilling, processSpaceBilling, and processUnitBilling methods, so they can all use the same logic. This is where new charge codes would be added, or any logic changes for ERIF__c, Real_Estate_Tax__c, CPI__c, Rent_Commencement_Date__c, Operating_Expenses__c, or Base_Rent_Amount__c should be made. Fully Escalated Rent and FER are formula fields in the Suite__c object, but uses these values from the linked Yardi Units as input.

processTenantBilling - Adds up Billing values at the Tenant level. Also finds the Rent Commencement Date. If a List<FMZ_Yardi_Billing__c> is passed in, it will use it without querying for values using the Tenant ID. If a FMZ_Yardi_Tenant__c is passed in, it will use it without querying and will return the updated Tenant without saving. Skipping the queries and database save operations is useful for bulk operations to avoid SOQL limits.

processSpaceBilling - Adds up Billing values at the Space (Amendment) level. Also finds the Rent Commencement Date. If a List<FMZ_Yardi_Billing__c> is passed in, it

will use it without querying for values using the Space ID. If a FMZ_Yardi_Space__c is passed in, it will use it without querying and will return the updated Space without saving. Skipping the queries and database save operations is useful for bulk operations to avoid SOQL limits.

processUnitBilling - Adds up Billing values at the Unit (Amendment) level. Also finds the Rent Commencement Date. If a List<FMZ_Yardi_Billing__c> is passed in, it will use it without querying for values using the Unit ID. If a FMZ_Yardi_Unit__c is passed in, it will use it without querying and will return the updated Unit without saving. Skipping the queries and database save operations is useful for bulk operations to avoid SOQL limits.

detectOccupancy - Uses the Space data to set the Leased__c, Occupied__c, Tenant_Name__c, Availability_Date__c, and Active_Space__c values at the Unit level. It looks at all the Spaces on for the Unit with the junction object, uses today's date to find the Active Space (StartDate is before today and End Date is after today) and sets these values on the Unit from this Space. This method is called by detectOccupancyForUnitId and detectOccupancyForUnits.

detectOccupancyForUnitId - Calls detectOccupancy for the given UnitId and updates it.

detectOccupancyForUnits - Calls detectOccupancy for the given List<FMZ_Yardi_Unit__c> and updates them.

processPropertyBillingAndOccupancy - Calls processUnitBilling, processSpaceBilling, and detectOccupancyForUnits for all Units and Spaces for the given PropertyCode. Needs to be run to update FER values on Suites after Yardi data has been updated.

mapSuitesToUnits - Should be called after importing Suite__c records from the Link spreadsheet and after each time the Yardi data has been queried. It maps the Suite__c records to the corresponding FMZ_Yardi_Unit__c records using the PropertyCode and SuiteNumber. It then sets the Suite Floor__c, Leasable_SF__c, and Deactivated__c values. It contains error correction to map Link Property 111 to Yardi Property 112, and pad Suite numbers with leading 0s or 00s or 000s or LLs to facilitate a match.

mapSpacesToLeases - Maps existing Lease__c records to FMZ_Yardi_Space__c records, also creates new Leases, if not already present, for all Yardi Spaces. Sets the Lease Yardi_Tenant__c, Name, and Market_Square_Feet__c (sum of all Suite Market Square Feet) and sets the Percent_of_Lease__c on each Suite. It also adds up values from the child Suites to set the parent Suite's values for Children_Base_Rent__c,

Children_CPI__c, Children_ERIF__c, Children_Op_Ex__c, and Children_RE_Tax__c if applicable. When complete, it updates all Leases and Suites. Must be called after processPropertyBillingAndOccupancy so that the values are already present from processBilling.

mapTenantsToAccounts - Maps existing Accounts to FMZ_Yardi_Tenant__c records, based on the Tenant ID value. It also creates new Accounts for any Tenant that does not have a corresponding Account. It then goes through all Leases to set the Tenant__c lookup to the appropriate Account ID.