**[SLCCORIC].[Parameter].[SP\_StartProcess]**

**Manual**

1. SP\_RegisterPeriod: Insert into the Period table from CORIC DB a new period to be created.
2. SP\_RegisterEntities: First we update the current Entities existing into the Entity table, to have the latest value for the fields EntityName, IsActive, IsTerminated from the table **RLAPPS..Accounts** by his EntityId (**RLAPPS..Account.AccoutId**). Then we insert the accounts missing in the table Entity from **RLAPPS..Accounts**.
3. SP\_RegisterProducts: First we get the periodDate value using the PeriodId as parameter (exec Parameter.SP\_GetPeriodDate).

Then we select from **rlapps..Accounts** and **rlapps..Portfolios** all the portoflios where there is a relationship with **rlapps..AccountBenchmarks** table (is this table we have the relationship of an Account against Benchmarks). By this way we get all the portfolios that are Benhmarks, This data is stored into a temporary table called **Benchmarks\_CTE** where we have the products that are Benchmark category.

Then we select again from **rlapps..Accounts** and **rlapps..Portfolios** all the portfolios that are Assets and we insert this data into a temporary table called **Assets\_CTE.**

Next, by a union of both tables (**Benchmarks\_CTE, Assets\_CTE**) we will insert into a temporary table called @Products\_Temp all the portfolios, benchmarks and assets. Having this temporary table with all the products we will use it to refresh all the current product information into the Coric product table, by updating theirs Names, descriptions, Category, and PortfolioType.

Finally, we sill insert into our Product Table (Coric) the new products missing by comparing the temporary table @Products\_Temp against Product table.

1. SP\_RegisterEntityProducts: In this step we will insert into Coric DB **Parameter.EntityProduct** table the relationship between a client (Entity) and the products, this insert is made through the previous tables filled Parameter.Entity, Parameter.Produt and RLAPPS.Accounts where client’s data is stored.
2. SP\_RegisterProductPeriod: In this step we will insert the productPeriod data, that is the relationship between each product and a period, this data is obtained from **development..cellcal** (to filter by portfolio name and the calculating date for that portfolio by the fileld RecalcDate), **parameter.Product** (to get the product information), and **rlapps..Accounts** and **rlapps.AccountBenchmarks** to be able to get all the data for product Benchmarks and Assets, this data is also filtered by the max Startdate from **rlapps.AccountBenchmarks** where the this StartDate is less than the @PerdioDate requested .
3. SP\_RegisterCusipPeriod: First the process save into a temporary table called **Positions\_CTE** all the positions (CUSIPS) with their Issuer, then the next step is to insert into a temporary table called **Cusips\_CTE** selecting from the first table **Positions\_CTE** and joining into **Development..MaturityDates** table to get the Cusips with their ratings (Moody, SP, SP2) also QRate and Rate.

Finally the process will insert into the Coric table called **Parameter.CusipPeriod** joining into the second temporary table (**Cusips\_CTE**) and **Development..RateMap** to get the Rate name for that rating assigned to each CUSIP, for example A1, for both Moody and S&P, also is saved the RLRAVG value that is obtained from the table **Development..CELLDEF** where the CellName not equal to ‘CASH’ or ‘TOTAL’.

1. SP\_RegisterPortfolios: In this step of the whole process, the purpose is to insert the portfolios with and initial value for each of them, there is a special action that is calculate the RiskControls of the Portfolio, this data is obtained from **RLAPPS..RiskControls** table by joining into the tables **RLAPPs.Accounts** and  **RLAPPS..RiskControlAccounts** by the AccountId, and all these value are concatenated by the SQL function STRING\_AGG. There is a final value called Quality into the table **Parameter.Portfolio** tahta is calculated by a case function depending of the value Name from the table RLApps..Mandates that contains all the names for each Mandate code.
2. SP\_RegisterCusipPortfolio: the idea in this step is to save every Cusip by Portfolio, calculating the total SUM for Paramount, MarketValue, and MarketWeight; the MAX value for YTW, Convexity, MarketPrice, Mduration, Term and the AVG for TsySpread.

In this StoreProcedute we take the mainly data from the Cusips from the table **Development..ISSUEDATA** like PortfolioName, Cusip, Paramount, MarketValue, YTW, Convexity, MarketPrice, MarketWeight, MDuration, IssueDataDate, Duration, Term; then with that data we calculate the TsySpread by cusip using the function Development.dbo.GetTsySpread; and then having al this information the process execute the SUM, Max and AVG function for the aggrupation of data by portfolio, cusip, and cellgroup mainly.

1. SP\_RegisterPortfolioStats: This process will create a cursor for all the portfolios for the specific PerdiodId that we are creating the data into the DB, after have the cursor created the process will iterate through this cursor to calculate MarketValue, Coupon, FeatureMarketWeight, YTW, MDuration, Convexity, DailyReturn, MTD, QTD, YTD, SIDReturn, CurrentYield, NumIssues, Paramount, Duration, Term, Price, APR, PerformanceMarketWeight using the StoreProcedure **Parameter.SP\_GetCellData** by CellGroup. Once we have all these data calculated for each period, the process will insert this data into the table **Parameter.PortfolioStats.**

And finally, the process will update these values calculated for each portfolio in the **Parameter.Portfolio** table, like MTD, QTD, YTD, SID by portfolio, so in this way the latest values are refreshed by portfolio.

1. SP\_RegisterPortfolioStats: same process as before but with a different CellGroup.
2. SP\_RegisterPASSInformation: In this process is created a Cursor again for iterate through the portfolios and their periods, one we have this data created, the process will call the StoreProcedure **Paramter.SP\_GetPASSData** with each portfolio and his period to fill the table **Parameter.AttributionSumary,** in this last process will be calculated the information per year and by portfolio according to the table **Parameter.AttributionYears**, that contain all the years to be taken into account in the calculation of the values for the **Parameter.AttributionSumary** table.
3. SP\_RegisterPortfolioStats\_TwoYears:This process is quite similar to the previous one but this is taking into account the months. So this process start by creating a cursor in the same way that the previous one of all the portfolios and their periods, iterating through these portfolios, but the next step is to create another cursor for all the months actives into the **Parameter.PortfolioStartsMonth** table, right now they are 24 months (2 years), and for each month we will calculate the values using the SP\_GetCellData Store Procedure for that specific portfolio being iterated in the first cursor. Once we have this data calculated for a portfolio in a specific month, we save that data into the Parameter.PortfolioStatsHistorical table. So in that way we will have in this table all the data by portfolio and by month to be able generate the report requested