Planning Database System (PDS) is an **On-premises software** (sometimes abbreviated as "onprem") installed and run on computers on the premises which serves as a **platform for data**

storage. It **holds the data** which is being **sent across various systems**.

Data is brought in planning database from GEHC systems like **GLPROD, ITCS, SBOM, IB and MWS**

as source and is sent over to **Servigistics** for planning post two layers of data massaging i.e.

Business Rule and SPM Rule.

Data is brought in planning database from Servigistics and is sent over to GEHC systems like

**GLPROD, GLPROD FTP Locations, Informatica FTP** Locations post data massaging

Briefly, data will be captured in PDS and several business logics will be applied and thereafter

the transformed data will be fed to the destination system.

**Execution Tables:**

1. **GE\_PLN\_TRANSFORMATION\_TABLES**: Contains the Table **Name**, Table **Alias** and Table **Code**

2. **GE\_SPM\_RULE\_HEADERS\_ALL**: Contains all the **Rules Headers Details** which is mentioned in the set up document.

3. **GE\_SPM\_RULE\_LINES\_ALL**: Contains all the **Rules Lines Details** which is mentioned in the set up document.

4. **GE\_PLN\_TRANSFORMATION\_BASE\_DTL**: Contains all the **base queries for each activity.**

The base query will be determined by **activity type** and **logical flow**.

**Activity type** depicts the type of data modification activity like **insert** or **update**.

**Logical flow** determines the layer of flow of data i.e. **IP** (Dataflow from Inbound table to Processed table in PDS) and **PS** (Dataflow from Processed table to SPM table in PDS). All the base queries are mentioned in the set up document.

5. **GE\_PLN\_TRANSFORMATION\_EXEC\_DTL**: Contains all the **final validated query** **ready** for **execution**.

6. **GEMS\_IFACE\_SPM\_TABLE**: Contains the **execution details** of the **activity** .

7. **GEMS\_IFACE\_SPM\_TABLE\_DETAILS**: Contains every **steps of the execution details** for any **activity** .

8. **GE\_IFACE\_SPM\_RESTART\_MW**: Contains all the **activity names** with **unique process\_id** and **restart\_step\_id**. This table is solely used by **Boomi** to **restart an activity** in case of any failure situation.

The activity restart identification denotes the mode of failure:

a. Restart id **1** signifies **end to end flow needs** tobe **performed**

1. Restart id **2** signifies the **failure** at **PDS Stored Procedure**
2. Restart id **3** signifies the **failure** at **file transfer** from **PDS to destination system**

The Restart Id is maintained from Boomi when process starts. Also Boomi controls the id after completion of each individual steps and maintains the progress of the flow depending on this id.

1. **GEMS\_MW\_IFACE\_LOG\_IT\_TBL**: Contains **every steps of the execution details** for any **activity** from **Middle Ware** end.

**Packages :**

**1.** GE\_PLN\_**TRANSFORMATION:**

This package is responsible to

**create a query**

(It will build the executable queryconsidering the **base query from GE\_PLN\_TRANSFORMATION\_BASE\_DTL**)as per

the **user created rules**

( the user provided **conditions** **from** **GE\_SPM\_RULE\_LINES\_ALL** for a particular activity\_name, activity\_type, data\_stream and process\_flow )which will

be **executed at the time of data flow.**

And

**After** the **executable** is **built**, **GE\_PLN\_TRANSFORMATION.VALIDATE\_QUERY**  will **validate the query**.

After successful validation , the **executable query** is **saved in** **GE\_PLN\_TRANSFORMATION\_EXEC\_DTL** table.

The **error** will be **stored into** the table **GE\_RULE\_COMPILATION\_ERROR\_DTL** in case any failure happened during validation time .

1. **build\_query**: Builds the query **from** the **rules details** and **base query** using the tables

**GE\_SPM\_RULE\_HEADERS\_ALL**,

**GE\_SPM\_RULE\_LINES\_ALL**,

**GE\_PLN\_TRANSFORMATION\_TABLES** and

**GE\_PLN\_TRANSFORMATION\_BASE\_DTL**.

1. **validate\_query**: Validates if the **query** is **correct or not**.

Essentially validates the where clause. **If** there is any **validation error**, the same error **message** will be **inserted** **into** the table - **GE\_RULE\_COMPILATION\_ERROR\_DTL**

1. **save\_query**: **Saves** the **query** **into** the table **GE\_PLN\_TRANSFORMATION\_EXEC\_DTL**.

1. **execute\_transformation**: **Executes** the **saved query from** the table **GE\_PLN\_TRANSFORMATION\_EXEC\_DTL**.
2. **create\_query**:  This procedure performs the following sequential actions:
3. **Builds the query by calling the procedure build\_query.**
4. **Validate the query by calling the procedure validate\_query .**
5. **Save the query by calling the procedure save\_query**

* 1. GE\_PLN\_**TRANSFORMATION\_CALL**

GE\_PLN\_TRANSFORMATION\_CALL package **has** a **single procedure** **GE\_PLN\_TRANSFORMATION\_FLOW**.

This is driving implementation of 2 layers of logic i.e **Business logic** and **SPM logic**.

1. This procedure  allows user to enter the following **parameters** : **Activity Name**, **Data Stream** , **Activity Type** and the **flow** between layers.
2. The program will **accept** the **activity name** and will **check** the **following validations** for each :
3. It will check if this **program is already running for** the **given activity name within last 5 hours**. **If so**, it will **not allow** **rerunning this stored procedure**.

The stored procedure will **only run if** **prior** invoked **stored procedures** having the **same activity name** are **completed with status success or error** depending on the process flag.

1. This program will **throw** an **error** if the **activity name** is **NULL**

.

1. **After** the **successful validation** **data** will **flow** **from** Inbound (**INBD**) to Processed (**PRSD**) layer i.e. **IP** flow and then Processed (PRSD) to Outbound (SPM) layer of tables i.e. **PS flow**.
2. A **procedure** **GE\_PLN\_REQUEST\_SET** is **introduced** which will be **invoked** to **call** **child transformations** within any Parent transformations at below levels
3. Before-IP -> Prior to executing IP
4. Before PS -> After executing IP, Before executing PS
5. After-PS -> After executing PS
6. Before-Archive -> After executing PS, before executing Archive

The **sequence** and **control** will be defined in ‘**GE\_PLN\_FUNCTIONAL\_MAPPING\_TBL**’ Table where **MAPPING\_TYPE** is '**REQUESTSET**'.

1. GE\_**INBD\_PRSD\_STUB**

This stub is for the **execution** **of** the **logic** needs to be executed **before** the **records inserted into PRSD** table from INBD table and also **checks** the **mutual exclusiveness** of the **activity\_name** through lookup value in **GE\_PLN\_TRANSLATION\_LOOKUP** table

1. GE\_**PRSD\_SPM\_STUB**

This stub is for the execution of the logic needs to be executed before the records inserted into SPM table from PRSD table

1. GE\_**SPM\_STUB\_LOGIC**

 This stub is for the execution of the logic needs to be executed after the data inserted into the SPM table.

1. **GE\_SPM\_STUB**

This STUB is designed for implementing the archive logic.

1. **Critical to Process Variables**
   * 1. PROCESSED\_FLAG should be populated as ‘N’ means New in Inbound tables for data to be able to flow between INBD à PRSD à SPM Layer of tables.
2. **Definitions**

|  |  |
| --- | --- |
| SPM | Service Parts Management - Global planning tool |
| PDS | Planning Data Source |

1. Process flag of the tracking table : GEMS\_IFACE\_SPM\_TABLE holds the **status of the procedure**.
   * 1. The process flag denotes:
2. **‘I’**:  When the Store procedure is **initiated** the status flag will be set as ‘I’.
3. **‘C’**: When the Store procedure is completed successfully the status flag will be    set as ‘C’.
4. **‘W’**: When the store procedure is **completed with warning** then the status flag will be set as ‘W’.
5. **‘E’**: When the Store procedure is **completed  with error** the status flag will be set as ‘E’

A diagram of data processing

Description automatically generated

Above is achieved by following procedure/packages :

1.3.1 **GE\_PLN\_TRANSFORMATION Package**

GE\_PLN\_TRANSFORMATION package is responsible for **creating an executable query**.

The executable query for a **particular activity\_name**,  activity\_type, data\_stream and process\_flow is built **considering** the **details from base query** i.e. GE\_PLN\_TRANSFORMATION\_**BASE**\_DTL table **and** the **user provided conditions** from GE\_SPM\_**RULE**\_LINES\_ALL table. 

This program will **first check** the **active rule condition** in GE\_SPM\_**RULE**\_LINES\_ALL table corresponding to the **active rule** in GE\_SPM\_RULE\_**HEADERS**\_ALL table for a **particular activity name**. 

After the executable is built, this program **validates the query**. 

After successful validation the executable query is **saved** in GE\_PLN\_TRANSFORMATION\_**EXEC**\_DTL for a particular activity\_name, activity\_type, data\_stream and process\_flow. 

1.3.2 **GE\_PLN\_TRANSFORMATION\_CALL Package**

GE\_PLN\_TRANSFORMATION\_CALL package is **responsible** for data flow **from** **INBD** table of PDS to outbound (**SPM**) table of PDS.

It will first move the data from INBD table to PRSD table which is identified as **IP logical flow**. Then the data will move from PRSD table to outbound (SPM) table which is identified as **PS logical flow**. 

This program will take **Activity Name** as an **input** and will **consider** all the

**active rules** (where ENABLE\_FLAG is ‘**Y**’ for records in GE\_SPM\_RULE\_**HEADERS**\_ALL table) defined on the GE\_SPM\_RULE\_HEADERS\_ALL **for** the **input** **activity name**. 

These **active rules** will be **executed** in **order of priority** defined in the DATASTREAM\_**WEIGHTAGE** field of GE\_SPM\_RULE\_**HEADERS**\_ALL table. 

**For every active rules** in logical flow **IP**, the execution query for activity type ‘**INSERT’** will be **executed in** order of the **priority** set in the ACTIVITY\_SEQUENCE.

**After** this is successfully **completed**, the program will execute the execution query for activity type ‘**UPDATE’** **in order** of the **priority** set in the ACTIVITY\_**SEQUENCE** of logical flow **IP**. In this manner execution will be **performed for all active rules**.

**After** the successful completion of **IP flow**, all the **rules** having active **SPM enabled flag**  (SPM\_ENABLE\_FLAG is ‘**Y**’ of GE\_SPM\_RULE\_**HEADERS**\_ALL table) will be **considered** subsequently **for** the **PS** logical **flow**. 

Then **again** the execution query for the activity type ‘**INSERT**’ will be **executed** in **order** of the **priority** set in the ACTIVITY\_**SEQUENCE**. **After** successfully **completion** of **PS ‘INSERT’** execution query, the execution query for activity type ‘**UPDATE’** will be executed in order of the priority set in the ACTIVITY\_SEQUENCE. **After** successful **completion** of **PS** flow, **data** will move **to** the outbound **SPM** table from **PRSD** table.

There are four stubs which are triggered by this package.

1. **GE\_INBD\_PRSD\_STUB –**

This stub is executed **before** the commencement of **IP** flow. It **executes** the **logic** **of** all the intended **ammendments** on the records **in** the **INBD** table which are **inserted** **into** **PRSD** table **after checking** the mutual **exclusiveness**.

If the process gets initiated when any corresponding mutually exclusive process is already running, then the respective process will wait untill the mutually exclusive process gets completed.

1. **GE\_PRSD\_SPM\_STUB**:- This stub is executed **after** the **IP** flow and **before** the **PS** flow. Through this stub, records in PRSD table is  ammended.
2. **GE\_SPM\_STUB\_LOGIC**:- This stub is executed **after** the **PS** flow. Through this stub **SPM** **table** is **ammended**.
3. **GE\_SPM\_STUB**:- This STUB is designed mainly for **implementing** the **archive** logic and **purge** **logic**

Archival of a particular table and Purging is at the **discretion of users** and can be handled through the control table i.e. GE\_PLN\_SYSTEM\_**CONTROLS**

A procedure GE\_PLN\_**REQUEST\_SET** is introduced which will be invoked to **call** **child** transformations **within** any **Parent** transformations at below levels

i) **Before-IP** -> Prior to executing IP

ii) **Before PS** -> After executing IP, Before executing PS

iii) **After-PS** -> After executing PS

iv) **Before-Archive** -> After executing PS, before executing Archive

The sequence and control will be defined in ‘GE\_PLN\_FUNCTIONAL\_MAPPING\_TBL’ Table where MAPPING\_TYPE is 'REQUESTSET'. Below is how the mapping will be determined :

|  |  |
| --- | --- |
| **MAPPING\_TYPE** | REQUESTSET to determine the mapping for parent-child transformation relationships |
| **INTERNAL\_USE** | Parent Transformation |
| **ACTIVITY\_NAME** | Child Transformation |
| **ACTIVITY\_TYPE** | Control for the level of run : BEFORE\_IP, BEFORE\_IP, BEFORE\_PS, AFTER\_PS |
| **MAP\_VALUE1** | Frequency of child transformation runs |
| **CHAR\_VALUE1** | Enabled/Disabled |
| **NUMERIC\_VALUE1** | Priority |
| **CREATED\_BY** | SSO of the creator of this parent-child transformation relationships |
| **CREATION\_DATE** | Date of creation |
| **MODIFIED\_BY** | SSO of the modifier of this parent-child transformation relationships |
| **MODIFICATION\_DATE** | Date of modification |

A diagram of a company

Description automatically generated

**1.3.3 GE\_MW\_INTF\_UTIL Package**

This program is designed for **Middleware** to be able to **truncate** and **load inbound** table in every run.

Parameters to be passed are

* ACTIVITY\_NAME
* Layer : INBOUND

**1.3.4  GE\_VALIDATION\_PROCESS  Package**

This is a validation package **used** for **Validation** purposes

Has following functions or procedures

1. **DATE\_VALIDATION:** This function is used for **Part Master date validation**
2. **RULE\_COMPILE\_ALL :** This procedure is used to **compile all rules** that are **created by users** and **create execution query** from **base queries**.
3. **RESET\_PROCESS :** Used for making the **process flag = ‘N’** in **inbound** table.
4. **UPDATE\_TABLE :** Used for updating process flag = ‘N’ in inbound table. This function is **used** in **RESET\_PROCESS** procedure.
5. **DELETE\_TABLE:** Used for **truncating PRSD** and **SPM tables** on basis of activity name and process id. This function is **used in RESET\_PROCESS** procedure.
6. **GEMS\_GPO\_DEMAND\_CALCULATION:** This procedure is responsible to **calculate** the **demand** for each single **item** **present in** item **master** for the three poles **(AS, US and EU)** for last two year. It calculates the demand from GE\_PRSD\_GLP\_**PART\_DEMAND** table.
7. **WASHRATE\_CALCULATION:** This procedure is utilized to **calculate** the **wash rate** of individual **items** present in Item **Master table** GE\_SPM\_GLP\_**PART\_MASTER**\_AR, **considering** the various **types** of **transactions** the item goes through in a particular date range. This calculation is done based on the record present in GE\_PRSD\_GLP\_**PART\_TXN**. Also the part hierarchy is honored **calculate** the **cumulative washrate** of topmost parts in the part chain as defined in GE\_SPM\_MWS\_**PARTCHANGEUP**.
8. **DMD\_CONV\_PROCESS :** This procedure **enables** the business to perform Conversion of Data for Demand for any functional changes that requires the same. Demand conversion could be **required** **due to** some **change** in business **processes** that requires some **functional changes** in the **ERP** and same needs to be transpired in PDS for further SPM planning operations. Also it can handle scenarios where **demand data** **from** the **legacy** system needs to be **moved** **to** **PDS** to appraise SPM on the historical demands.
9. **GEMS\_GPO\_Indicated\_Pool\_Size :** This procedure **calculates** the **Indicated Pool Size** of individual **items** present in GE\_SPM\_GLP\_**PART\_MASTER**\_AR. This is calculated **through** the **summation** of all available **onhand** for the part as derived from  GE\_PRSD\_GLP\_**PART\_ONHAND** table. Also the part hierarchy is honored to calculate the cumulative Indicated Pool size of topmost parts in the part chain as defined in GE\_SPM\_MWS\_PARTCHANGEUP.
10. **GEMS\_GPO\_Priority\_Score :** This procedure calculates the Priority Score, Number of Opportunities and Supply Health of Individual Parts.
11. Priority Score determines the priority of individual parts considering the backorders, minimum quantity present in the network, available onhand, and average order quantity based on whether the location can either Procure, **Repair** or is the source pole  location or **replenishes** from other location as fetched from GE\_INBD\_SPM\_**PLN\_LVL**. Also the part hierarchy is honored to calculate the cumulative Priority Score of the topmost parts in the part chain as defined in GE\_SPM\_MWS\_PARTCHANGEUP.
12. Number of Opportunities is calculated through the **ratio** of **minimum quantity present** in the network as defined in GE\_INBD\_SPM\_PLN\_LVL table with respect to **average order quantity** as seen in GE\_PRSD\_GLP\_PART\_DEMAND table.
13. Supply health determines the supply provisioned for individual parts calculated as the ratio of Sum of Positive Priority Score with respect to Sum of Opportunities subtracted from 1.
14. **TXN\_CONV\_PROCESS :** This procedure enables the business to perform **Conversion of Data for Transactions** **for** any **functional changes** that requires the same. Transaction conversion could be required due to some change in business processes that requires some functional changes in the ERP and same needs to be transpired in PDS for further SPM planning operations. Also it can handle scenarios where Transaction data from the legacy system needs to be moved to PDS to appraise SPM on the historical transaction.
15. **REPAIR\_OPTIONS :** Repair options is a **break up** of all **probable repair combination** **for** a **particular part** **including** with its **part chain** with respect to the repair orgs and associated repair vendors. The data is inserted into GE\_INBD\_REPAIR\_OPTIONS table.
16. The logic works as, it would consider the Repair CDC of the part itself as well as the Repair CDC of all the parts present in the part chain.
17. **SEND\_MAIL:** This procedure is introduced to add a mail functionality to the PDS environment. This is presently being called in the Health Check implemented on OAO file that is received from SPM as a reverse flow file. This could be used in other sections of PDS as well.

**1.3.5 GE\_IFACE\_SPM\_DETAILS Package**

GE\_IFACE\_SPM\_DETAILS package is **responsible** to **modify** GEMS\_**IFACE\_SPM\_TABLE**  table and GEMS\_IFACE\_SPM\_TABLE\_**DETAILS** table **with activity name**  which is running in **PDS**. This is to monitor the activity which is running by initiating GE\_PLN\_TRANSFORMATION\_CALL.GE\_PLN\_TRANSFORMATION\_FLOW procedure .

Below functions are used in this package.

* **GE\_IFACE\_SPM\_INSERT:** This procedure will be called to **insert record** into GEMS\_IFACE\_SPM\_TABLE  with PROGRAM\_NAME,PROCEDURE\_NAME,START\_DATE,CREATION\_DATE,MESSAGE,STATUS\_FLAG, ACTIVITY\_NAME, PROCESS\_ID,DEBUG\_MESSAGE.
* **GE\_IFACE\_SPM\_UPDATE:** This procedure will be called to **update**  GEMS\_IFACE\_SPM\_TABLE as per PROCESS\_ID.
* **GE\_IFACE\_SPM\_UPDATE\_END:** This procedure used to **track when a program got executed**.

**1.3.6 GE\_REIMAGING\_EXECUTION Package**

GE\_REIMAGING\_EXECUTION is a package which is utilized to **remap** the **Part Changeup** file coming in **from** **MWS** and **transforming** the hierarchy of parts and relevant flags **to make** it **compatible** with the **requirement in SPM**.

The function which performs this activity is:

* **MWS\_PARTCHANGEUP:** This procedure will be called to **perform** the **re-imaging** of the part hierarchies in the GE\_PRSD\_MWS\_PARTCHANGEUP table. The GE\_PRSD\_MWS\_PART**CHANGEUP\_REF** table is taken as an interim table to **store** the **re-imaged data** and finally it is inserted back to GE\_PRSD\_MWS\_PARTCHANGEUP table.

**1.3.7 DATA\_CLEANUP\_PRG Package**

DATA\_CLEANUP\_PRG package is used to list down the obsolete tables and drop the tables after review. This package has 2 procedures:

1. Table listing: This procedure will list down the tables that are not used by any other code objects, transformation rule or is not in the exception list of tables.
2. Data cleanup: This procedure will drop the tables that have been identified to be dropped.

**Module Functionality Logic**

**Program Names:**

* GE\_PLN\_TRANSFORMATION,
* GE\_PLN\_TRANSFORMATION\_CALL
* GE\_MW\_INTF\_UTIL
* GE\_VALIDATION\_PROCESS
* GE\_IFACE\_SPM\_DETAILS
* GE\_REIMAGING\_EXECUTION

1. **(A) Program Description**

**Package Name: GE\_PLN\_TRANSFORMATION**

The parameters are

* Activity Name
* Data Stream
* Activity Type
* Process Flow

This procedure will build the **executable query** considering the **base query** from GE\_PLN\_**TRANSFORMATION\_BASE\_DTL** and the **user provided conditions** from GE\_SPM\_**RULE\_LINES\_ALL** for a particular activity\_name, activity\_type, data\_stream and process\_flow. After the executable is built, GE\_PLN\_TRANSFORMATION.**VALIDATE\_QUERY**  will **validate the query**. Once successful validation is done, the executable query is saved in GE\_PLN\_TRANSFORMATION\_EXEC\_DTL table.

**Inputs / Outputs**

1. Inputs

* Base query is in-place in GE\_PLN\_TRANSFORMATION\_BASE\_DTL table with a proper activity sequence for a particular activity name and logical flow.
* Setup Rules for a specific activity name in  GE\_SPM\_RULE\_HEADERS\_ALL maintaining a proper sequence and conditions will be in GE\_SPM\_RULE\_LINES\_ALL for each rule stream.

1. **Outputs**

An executable query will be built and saved into the GE\_PLN\_TRANSFORMATION\_EXEC\_DTL table.

**Program Logic**

**Pseudo code for ‘**GE\_PLN\_TRANSFORMATION Package‘

1. GE\_PLN\_TRANSFORMATION. **CREATE\_QUERY()** is called to **create the executable query**.
2. Parameters will take activity name, data stream, activity type and logical flow.
3. At the time of creation of the query, it will consider all the active rule lines from GE\_SPM\_RULE\_LINES\_ALL table for the given data stream name.
4. The conditions fetched from the rule lines are appended with the base query.
5. The query is validated.
6. After successful validation the query is saved into the GE\_PLN\_TRANSFORMATION\_EXEC\_DTL table.
7. For any exception error is thrown.

1. **Error Conditions**

Error conditions as depicted in the pseudo code are captured.

1. **Warning Conditions**

There are no warning conditions.

1. **(A) Program Description**

**Package Name:** GE\_PLN\_TRANSFORMATION\_CALL

The parameter is –

* Activity Name

The package is responsible for the data flow from INBD table of PDS to Outbound (SPM) table of PDS. It will first move the data from INBD table to PRSD table depending on the active defined rules and processed\_fag in Inbound Tables = ‘N’.

Then the data will move from PRSD table to outbound (SPM) table depending on the spm enable defined rules.

**Inputs / Outputs**

1. **Inputs**

Inbound (INBD) table data needs to be populated with the process flag = ‘N’

1. **Outputs**

All data in INBD table will move to the outbound (SPM) table based on the spm enable rule stream

**Program Logic (pseudo code)**

**Pseudo code for ‘**GE\_PLN\_TRANSFORMATION\_CALL Package’

1. GE\_PLN\_TRANSFORMATION\_CALL.GE\_PLN\_TRANSFORMATION\_FLOW procedure is called to initiate the procedure for a particular activity name.
2. GE\_IFACE\_SPM\_DETAILS.GE\_IFACE\_SPM\_INSERT procedure is called to insert into GEMS\_IFACE\_SPM\_TABLE for this activity name with the status flag of ‘I’
3. Check if the procedure is already running for this activity name for the commenced within last 5 hours. If yes the procedure will throw an error.
4. Else, GE\_INBD\_PRSD\_STUB.GE\_INBD\_PRSD\_FLOW procedure is called to execute the first STUB.
5. After successful completion of this stub it is checked if there exist any active rule for the activity name from GE\_SPM\_RULE\_HEADERS\_ALL. If No, It will follow from STEP:7
6. If Yes, Then for every single rule stream
7. GE\_PLN\_TRANSFORMATION.EXECUTE\_TRANSFORMATION procedure is called to execute the INSERT query for IP flow.
8. It will throw an error if the INSERT query is not successfully completed.
9. GE\_PLN\_TRANSFORMATION.EXECUTE\_TRANSFORMATION procedure is called to execute the UPDATE query for IP flow.
10. It will throw an error if the UPDATE query is not successfully completed.

1. GE\_PRSD\_SPM\_STUB.GE\_PRSD\_SPM\_FLOW procedure is called to execute the next STUB.
2. After successful completion of this stub, it is checked if there exist any SPM enabled active rule for the activity name from GE\_SPM\_RULE\_HEADERS\_ALL.

If No, It will follow from STEP: 10

1. If Yes, Then for every single rule stream
2. GE\_PLN\_TRANSFORMATION.EXECUTE\_TRANSFORMATION procedure is called to execute the INSERT query for PS flow.
3. It will throw an error if the INSERT query is not successfully completed.
4. GE\_PLN\_TRANSFORMATION.EXECUTE\_TRANSFORMATION procedure is called to execute the UPDATE query for PS flow
5. It will throw an error if the UPDATE query is not successfully completed

1. GE\_SPM\_STUB\_LOGIC.GE\_SPM\_FLOW procedure is called to execute the next STUB.
2. After successful execution of this STUB, IFACE table (GEMS\_IFACE\_SPM\_TABLE) is UPDATED with the STATUS ‘C’.
3. The procedure GE\_PLN\_TRANSFORMATION\_CALL  then will check if the GE\_SPM\_STUB is called by Middleware or not from the lookup value against the activity name. If it is not called by the MW then the procedure itself will call the below STUB.
4. GE\_SPM\_STUB.GE\_SPM\_FLOW procedure is called to execute the final STUB.
5. After successful execution, IFACE table (GEMS\_IFACE\_SPM\_TABLE) is UPDATED with the STATUS ‘C’.
6. **GE\_PLN\_REQUEST\_SET** procedure is introduced which will be invoked to call child transformations within any Parent transformations at multiple levels.

The sequence and control will be defined in ‘GE\_PLN\_FUNCTIONAL\_MAPPING\_TBL’ Table where MAPPING\_TYPE is 'REQUESTSET'. Below is how the mapping will be determined:

|  |  |
| --- | --- |
| **MAPPING\_TYPE** | REQUESTSET determing the mapping for parent-child transformation relationships |
| **INTERNAL\_USE** | Parent Transformation |
| **ACTIVITY\_NAME** | Child Transformation |
| **ACTIVITY\_TYPE** | Control for the level of run :  BEFORE\_IP -> Prior to executing IP  BEFORE\_IP -> After executing IP, Before executing PS  BEFORE\_PS -> After executing PS  AFTER\_PS -> After executing PS, before executing Archive |
| **MAP\_VALUE1** | Frequency of child transformation runs |
| **CHAR\_VALUE1** | Enabled/Disabled |
| **NUMERIC\_VALUE1** | Priority |
| **CREATED\_BY** | SSO of the creator of this parent-child transformation relationships |
| **CREATION\_DATE** | Date of creation |
| **MODIFIED\_BY** | SSO of the modifier of this parent-child transformation relationships |
| **MODIFICATION\_DATE** | Date of modification |

**STUB:**

1. GE\_INBD\_PRSD\_STUB - This stub is executed before the commencement of IP flow. It executes the logic of all the intended ammendments on the records in the INBD table which are inserted into PRSD table after checking the mutual exclusiveness . If the process gets initiated when any corresponding mutually exclusive process is already running, then the respective process will wait untill the mutually exclusive process gets completed.

As a part of this stub the Inbound data for Collaborative Planning Item and SupplyDemand forecast details is loaded into the GE\_INBD\_ITEM\_CP  table and GE\_INBD\_SUP\_DMD\_CP respectively :

1. Item Forecast Inbound insertion – The data is picked for distinct items in the Order Plan Processed Layer table GE\_PRSD\_PLAN\_ORDER and inserted into CP Inbound Item table based on below values, mapping and conditions :

|  |  |  |
| --- | --- | --- |
| **Column in GE\_INBD\_ITEM\_CP** | **Data** | **Comment** |
| ITEM\_NAME | Part\_Number | Distinct Part\_Number from GE\_PRSD\_PLAN\_ORDER table |
| ORGANIZATION\_CODE | 'GPO', | Hard coded value |
| MRP\_PLANNING\_CODE | '3', | Hard coded value |
| UOM\_CODE | 'EA', | Hard coded value |
| BUILT\_IN\_WIP\_FLAG | '2', | Hard coded value |
| PURCHASING\_ENABLED\_FLAG | '1', | Hard coded value |
| PLANNING\_MAKE\_BUY\_CODE | '2', | Hard coded value |
| ENGINEERING\_ITEM\_FLAG | '2', | Hard coded value |
| EFFECTIVITY\_CONTROL | '1', | Hard coded value |
| INVENTORY\_PLANNING\_CODE | '6', | Hard coded value |
| SOURCE\_INSTANCE\_CODE | 'LEG', | Hard coded value |
| DESCRIPTION | Item\_Description | Corresponding item\_description extracted from GE\_PRSD\_GLP\_PART\_MASTER table. There should be no '~' sign in the description |
| PLANNER\_CODE | planner\_code | Corresponding planner\_code derived from GE\_PRSD\_GLP\_PART\_MASTER table |
| BOM\_ITEM\_TYPE | '4', | Hard coded value |
| INVENTORY\_ITEM\_FLAG | '1', | Hard coded value |
| INVENTORY\_ASSET\_FLAG | 'Y', | Hard coded value |
| COMPANY\_NAME | 'GE Healthcare', | Hard coded value |
| PROCESSED\_FLAG | N' | Hard coded value. Initial status of Inbound layer |
| INBD\_PROCESSED\_DATE | SYSDATE | System Date |

Conditions:

1. Order\_Type in GE\_PRSD\_PLAN\_ORDER table for the item is ‘Procure’