#### 

**Planner Housekeeping Program –**

**Design & Approach Document**

19 July 2017

#### Document information

|  |  |  |
| --- | --- | --- |
| Document owner | Tom Parkin | |
| PowerSteering code | Not Applicable | |
| Prepared by | Ranjith Gopalankutty – UST Global | |
| Contact details | Ranjith.gopalankutty@boots.co.uk | |
| Document classification  (indicate with a X in the right hand column) | Key Project Document |  |
| Supporting Project Document | X |
| Process Document |  |
| Operating Procedure |  |
| Quality Standard |  |
| Document location and filename |  | |
| Version number | 1.2 | |
| Document status | Final | |

Document history

| **Version no.** | **Date** | **Change** |
| --- | --- | --- |
| 1.0 |  | Final version |
| 1.1 |  | Addressed with Charlie’s Comments |
| 1.2 |  | Updated the design with SRMAP reading as well |

Reviewer/Authorisation

| **Name** | **Title** | **Responsibilities** | **Authorisation Signature** |
| --- | --- | --- | --- |
| Tom Parkin | **Service Manager, Retail Application Management** | Reviewer/Authoriser |  |
| Ranjith Gopalankutty | **System Analyst** | Reviewer/Authoriser |  |
| Paul Bowers | **Manager Solution Design** | Reviewer/Authoriser |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Distribution

| **Name** | **Title** |
| --- | --- |
| Anne Greener | **Release Manager** |
|  | **Test Manager** |

Related documents (if any)

| **Document name** | **Location** |
| --- | --- |
| Nil | Nil |

### Planner Housekeeping Program Design and Approach

1. Issue

Planner housekeeping program SRP10 had a defect for years. As part of the housekeeping, SRP10 house keeps records from SRPOG file leaving behind SRSXF, SRMOD and SRPDF. These files depend on master file SRPOG to build the key. Over the time planner files piles up on 3 files and resulting in file full issue and causing POGOK failure.

Also its been found that SRPOG has unnecessary records as well due to mismatch in housekeeping between intactix and controller. So its decided to use the SRMAP as master file, so any planner after reading will be checked against SRMAP, if no match found that particular planner record will be ignored.

1. Analysis

The root cause has been traced to SRP10 and fixed the same. This program rolled out to all stores in July 2016. This program will now housekeep the records fine in all 4 files without issues. But the old records present in the files will not get house kept. The only way to remove the old records from the file was to do a planner initial load.

By that approach, planner files will get created fresh live data will be sent down to stores from mainframe. By doing so all expired data will be removed from the files. Over the time from July 2016 to march 2017, over 600 plus stores had planner initial load and got the files created fresh with live data. But there are close to 2000 stores still need to get this housekeeping activity.

As there are close to 2000 stores to get the activity, it is impossible to do POGIL on those stores. There is a limit of number of stores we can planner initial load per day, and it is 50. Even if it is 50, the amount of data created on mainframe will be huge and will be bottleneck for NFM. So that approach is not viable.

After the discussions with concerned stakeholders it is agreed that a new program will be written, the program will be responsible for housekeeping the expired data in the files. That will help us in executing the program locally on each store controller. By this approach, data traffic can be avoided and new files will be created afresh in the stores with only live data.

1. Design approach

No

Ignore the record

Planner present in SRMAP

Create 3 new keyed files for SRMOD, SRSXF and SRPDF and Error file

Yes

Read SRPOG in direct mode and move the records to an array. If its matching in SRMAP

Set SRPOG counter

Set SRMOD counter based on planner key

Read SRMOD in direct mode and move the records to array

Set SRPDF counter for based on planner key

Read SRPDF in direct mode and move the records to array

Set SRSXF counter key based on planner and module

Read SRSXF and move the records to array

Ignore the records which are not present in the master planner file

SRPDF Array

SRSXF Array

SRMOD Array

Key Exist in SRPDF?

Key exist in SRMOD?

No

Key exist in SRSXF?

No

Set the write counter against planner key

No

YesS

YesS

Set the write counter against planner

YesS

Set the write counter against planner

Write to new to SRSXF keyed file

Write to new SRPDF keyed file

Write To new SRMOD keyed file

Rename the .NEW file as okay file and rename the original files as .BKP .

Set the distribution type correctly for the new files

1. Above Flow chart explained:

Program will start reading the SRPOG file sequentially also it will check the planer presence in SRMAP file. If its present in SRMAP , it will start building the Planner number and DBKEY in to an array ,non matching records will be ignored. Once read is completed. Program will start reading the other 3 files sequentially and start building the array for the records it has found. Each array will be of capacity 10000 (As array can hold up to 64k pointers, which roughly translates to 16000 plus records based on 4 byte for array pointer. Should give us enough capacity to store all records).

After sorting all the records in to an array comparison operation will start between arrays. Planner keys in SRPOG will be checked against SRMOD, SRPDF and SRSXF array. If match found then the array records will be written in to the new keyed files SRMOD.NEW, SRSXF.NEW and SRPDF.NEW. As part of the write, a counter will be set as well.

Once the write operation is completed the counter will be compared against the initial sequential counter we found while reading all files, if there are differences it will be reported in to log file. Ok file will be set with flags as below

OK file will have 4 flags as like ONS suite status with below flag status.

First byte = SRPOG read status

Second byte = SRMOD file status

Third byte = SRSXF file status

Fourth byte = SRPDF status

X = Failure

S = Started

Y = No file

E = All success

Log file as part of the program will track the progress of execution. Towards the end of the file program will put up a status of number of records it has read during sequential read and how many records have been added back to new file