|  |
| --- |
| Predictive Health Notification  Alinity ci RSM Pick and Load Errors |
| ALINITY ci Immunoassay Analyzer  July 1, 2019 |

**PHN - Alinity ci RSM Pick and Load Error Spec Sheet for Apollo**

**References**

D000059484 Alinity cd RSM Pick and Load Errors PHN Design Document

**Summary**

To implement a Predictive Health Notification (PHN) for Alinity-ci Analyzers that will detect RSM Pick and Load Errors.

|  |  |
| --- | --- |
| PHN Descriptor | Alinity CI RSM Pick and Load |
| PHN Experience Code / Name | A832 PHN\_Alinity\_ci SCM; PHN\_RSM Transport; Pick and Load |
| PHN KM Article Number/ Name | K91150585 PHN\_Alinity\_ci SCM: RSM Transport; Pick and Load |
| Service Level | 2- Advanced Service |
| Always On Package | Always On 01DP6- 01, 02, 03, 09, 70, 79, 80, 83, 84, 89 |
| IDA Table(s) | IDAQOWNER.ICQ\_INSTRUMENTACTIVITY |
| IDA Table Fields | LOGDATE\_LOCAL  SYSTEMSN  COMPONENT  ACTIVITY |
| Analysis Frequency | Daily |
| Data Required | Previous 7 days |
| Data Aggregation | None |
| Run Time Estimate | N/A |
| Flag Criteria | If Linear ≥ 3.97, Then Flag, Else No Flag |
| Probable Failure Modes | RSM pick and load issues could be associated with rack transfer failure at the load area, loading area pick error, rack or reagent cartridge damage, inability to load into the reagent carousel, reagent carousel load error, reagent transport move error, reagent cartridge damage, RSM being out of alignment, RSM transport arm being dirty, RSM transport sensor failure, or reagent carousel reagent pack detector failure. |
| Suppression Experience Codes | A4Z5, A4Z1, A4Z2, A4Z6, A4ZG, A4Z7, A4Z3, A4Z8, A4ZH, A4Z9, A4ZA, A4Z4 |
| Applicable Work Done Codes (WDC) | DA\*\* |

**Data Processing Steps**

|  |  |
| --- | --- |
| Data Processing Steps | |
| 1 | Query all data from the previous 7 days for each instrument (IDA Table: IDAQOWNER.ICQ\_INSTRUMENTACTIVITY) |
| 2 | Calculate Frac\_Recover, Frac\_Engage, PerDay\_Recover, PerDay\_Engage, Linear |
| 3 | If Linear ≥ 3.97, Then Flag, Else No Flag |

**APPENDIX 1:**

**Algorithm Code**

CREATE OR REPLACE PROCEDURE SVC\_PHM\_OWNER.PHM\_ICQ\_RSM\_Pick\_Load\_PROC (

V\_ALG\_NUM NUMBER,

V\_RUN\_DATE DATE,

V\_BATCH\_NUM VARCHAR2,

V\_UNIX\_ID VARCHAR2)

IS

-- AUDIT LOG VARIABLE

V\_PROCESS\_TYPE VARCHAR2 (25);

V\_PROCESS\_STATUS VARCHAR2 (25) := 'STARTED';

V\_PROCESS\_ID NUMBER (15);

V\_PROD\_FAMILY VARCHAR2 (25);

V\_RUN\_MODE VARCHAR2 (10);

V\_ROUTINE\_NAME VARCHAR (35);

V\_ROUTINE\_TYPE VARCHAR (35);

V\_ERROR\_MESSAGE VARCHAR (4000);

-- ALGORITHM LOCAL VARAIBLES TO HANDLE THE PROCESS FLOW

-- < CHANGE >

V\_FLAG VARCHAR (5);

V\_REC\_COUNT NUMBER := 0;

V\_REC\_INS\_COUNT NUMBER := 0;

V\_FLAG\_DATE\_TIME DATE;

V\_FLAG\_COUNT NUMBER;

V\_RES\_COUNT NUMBER;

V\_IHN\_LEVEL3\_DESC\_VAL VARCHAR (500);

VALGNAME VARCHAR (200);

V\_ALG\_DFN\_SK NUMBER;

-- ALGORITHM PARAMETER VARAIBLES TO HANDLE THE PROCESS FLOW

-- < CHANGE >

V\_IHN\_LEVEL3\_DESC VARCHAR2(200);

V\_FLAGGED\_PL VARCHAR2 (10);

V\_FLAGGED\_EXP\_CODE VARCHAR2 (10);

-- </ CHANGE >

vcFlag INTEGER := 0;

--Create a table to store FLAG\_LIST in -- use BULK COLLECT so operation only has to be carried out once (instead of cursor loop)

TYPE TAB IS RECORD

(

DEVICEID NUMBER,

MODULESNDRM VARCHAR (10)

);

TYPE TBL IS TABLE OF TAB;

FLG\_TBL TBL;

-- Cursor to identify all instruments available in IDA during batch (taken from PHM\_ODS\_RESULTS\_CC)

CURSOR DEVICE\_SN\_LIST

IS

SELECT IA.DEVICEID,

UPPER (IA.SYSTEMSN) SERIAL\_NUM,

MAX (IL.PL) PL,

MAX (IL.CUSTOMER\_NUM) CUSTOMER\_NUMBER,

MAX (IL.CUSTOMER) CUSTOMER\_NAME,

MAX (PC.COUNTRY) COUNTRY\_NAME,

MAX (PC.AREAREGION) AREA,

MAX (IL.CITY) CITY,

COUNT (\*) DEVICE\_SN\_CNT

FROM SVC\_PHM\_ODS.PHM\_ODS\_CI\_SCM\_INSTACTIVITY IA,

INSTRUMENTLISTING IL,

PHM\_COUNTRY PC

WHERE IA.BATCH\_NUM = V\_BATCH\_NUM

AND IA.RUN\_DATE = V\_RUN\_DATE

AND UPPER (IA.SYSTEMSN) = UPPER (IL.SN)

AND PC.COUNTRY\_CODE = IL.COUNTRY\_CODE

AND IL.PL = '214' -- PL for Alinity i

GROUP BY IA.DEVICEID, IA.SYSTEMSN;

V\_EXISTING\_REC\_CNT NUMBER;

BEGIN

-- STEP 1 :PURPOSE TO GET PROCESSID OF CURRENT EXECUTION

V\_PROCESS\_ID := PHM\_ALGORITHM\_UTILITIES\_1.PHM\_GET\_PROCESS\_ID ();

--V\_PROCESS\_ID := -1;

V\_PROCESS\_STATUS := 'STARTED';

/\*

DBMS\_OUTPUT.PUT\_LINE (

'PHM\_CC\_CUVETTE\_PROC EXECUTION STARTED FOR : V\_BATCH\_NUM: '

|| V\_BATCH\_NUM

|| ', V\_RUN\_DATE: '

|| V\_RUN\_DATE);

\*/

-- STEP 2 : PURPOSE TO GET THE REQUIRED ALGORITHM INFORMATION FROM CONFIGURATION TABLES

SELECT AR.ROUTINE\_NAME,

AR.ROUTINE\_TYPE,

AR.RUN\_MODE,

AR.ROUTINE\_INVOKE\_COMMAND,

PF.PRODUCT\_FAMILY\_NAME

INTO VALGNAME,

V\_PROCESS\_TYPE,

V\_RUN\_MODE,

V\_ROUTINE\_NAME,

V\_PROD\_FAMILY

FROM PHM\_ALGORITHM\_ROUTINES AR, PHM\_PATTERNS PP, PHM\_PRODUCT\_FAMILY PF

WHERE AR.PHM\_PATTERNS\_SK = V\_ALG\_NUM

AND PP.PHM\_PATTERNS\_SK = AR.PHM\_PATTERNS\_SK

AND PP.PHM\_PROD\_FAMILY\_SK = PF.PHM\_PROD\_FAMILY\_SK;

/\*

DBMS\_OUTPUT.PUT\_LINE (

'VALGNAME: '

|| VALGNAME

|| ', V\_PROCESS\_TYPE: '

|| V\_PROCESS\_TYPE

|| ', V\_RUN\_MODE: '

|| V\_RUN\_MODE

|| ', V\_ROUTINE\_NAME: '

|| V\_ROUTINE\_NAME

|| ', V\_PROD\_FAMILY: '

|| V\_PROD\_FAMILY);

\*/

-- GET ALGORITHM\_DEFINITION\_SK

SELECT PP.PHM\_ALGORITHM\_DEFINITIONS\_SK

INTO V\_ALG\_DFN\_SK

FROM PHM\_PATTERNS PP, PHM\_ALGORITHM\_DEFINITIONS PAD

WHERE PP.PHM\_ALGORITHM\_DEFINITIONS\_SK =

PAD.PHM\_ALGORITHM\_DEFINITIONS\_SK

AND PP.PHM\_PATTERNS\_SK = V\_ALG\_NUM

AND ALGORITHM\_NAME IN (SELECT ROUTINE\_NAME

FROM PHM\_ALGORITHM\_ROUTINES

WHERE PHM\_PATTERNS\_SK = V\_ALG\_NUM);

--DBMS\_OUTPUT.PUT\_LINE ('V\_ALG\_DFN\_SK: ' || V\_ALG\_DFN\_SK);

-- Ex: 12941 ARCHITECT IA ALG Oracle Procedure Oracle Procedure FEP PHM\_FE\_PRESSURE Batch 9/8/2016 10:20:36 PM STARTED 9/8/2016 9/8/2016 10:20:36.000000 PM BTH2200 NULL 1003

PHM\_ALGORITHM\_UTILITIES\_1.PHM\_PROCESS\_AUDIT\_LOG (V\_PROCESS\_ID,

V\_PROD\_FAMILY,

V\_PROCESS\_TYPE,

V\_ROUTINE\_TYPE,

VALGNAME,

V\_ROUTINE\_NAME,

V\_RUN\_MODE,

V\_PROCESS\_STATUS,

V\_ERROR\_MESSAGE,

V\_RUN\_DATE,

SYSDATE,

V\_BATCH\_NUM,

V\_UNIX\_ID,

V\_ALG\_NUM);

--DBMS\_OUTPUT.PUT\_LINE ( 'V\_PROCESS\_ID: '|| V\_PROCESS\_ID || ', VALGNAME: ' || VALGNAME || ', V\_ROUTINE\_NAME: '|| V\_ROUTINE\_NAME);

-- STEP 3 : PURPOSE - TO GET THE ALL THE PARAMETERS THAT WERE DEFINED IN THE ALGORITHM SCREEN

FOR I IN (SELECT PARAMETER\_VALUES, PARAMETER\_NAME, PHM\_PATTERNS\_SK

FROM PHM\_THRESHOLD\_PARAMETER

WHERE PHM\_PATTERNS\_SK = V\_ALG\_NUM ) --AND DELETE\_FLAG IS NULL)

LOOP

-- PURPOSE - IN CASE OF NEW PERAMETER DEFEINED IN ALGORITHM DEFINITION - WRITE CODE WITH A NEW IF CONDITION TO GET NEW PARAMETER VALUE

-- <CHANGE >

IF I.PARAMETER\_NAME = 'IHN\_LEVEL3\_DESC'

THEN

V\_IHN\_LEVEL3\_DESC := I.PARAMETER\_VALUES;

END IF;

/\*

IF I.PARAMETER\_NAME = 'MIN\_VALUE'

THEN

V\_ONEDAYGROUP := TO\_NUMBER (I.PARAMETER\_VALUES);

END IF;

IF I.PARAMETER\_NAME = 'MAX\_VALUE'

THEN

V\_SIXDAYGROUP := TO\_NUMBER (I.PARAMETER\_VALUES);

END IF;

\*/

-- < CHANGE>

END LOOP;

--DBMS\_OUTPUT.PUT\_LINE ( 'V\_IHN\_LEVEL3\_DESC= ' || V\_IHN\_LEVEL3\_DESC || ', V\_INTEGRATEDVacuum\_Sensor\_MAX= '|| V\_INTEGRATEDVacuum\_Sensor\_MAX || ', V\_INTEGRATEDVacuum\_Sensor\_SD= ' || V\_INTEGRATEDVacuum\_Sensor\_SD);

SELECT \*

BULK COLLECT INTO FLG\_TBL

FROM (SELECT DEVICEID, Instrument

FROM

(SELECT DEVICEID, Instrument,

(CASE WHEN Num\_Recover > 0 THEN Num\_Recover / Num\_RSM\_Move

ELSE 0 END) AS Frac\_Recover,

(CASE WHEN Num\_Engage > 0 THEN Num\_Engage / Num\_RSM\_Move

ELSE 0 END) AS Frac\_Engage,

(CASE WHEN Num\_Recover > 0 THEN Num\_Recover / Num\_Days

ELSE 0 END) AS PerDay\_Recover,

(CASE WHEN Num\_Engage > 0 THEN Num\_Engage / Num\_Days

ELSE 0 END) AS PerDay\_Engage

FROM

(SELECT DEVICEID, Instrument, COUNT(Day) AS Num\_Days,

SUM(Num\_Retry - 2\*Num\_Exceed) AS Num\_Recover,

SUM(Num\_Engage) AS Num\_Engage,

SUM(Num\_Scans + Num\_Retry - Num\_Exceed) AS Num\_RSM\_Move

FROM

(SELECT TRUNC(LOGDATE\_LOCAL) AS Day, DEVICEID, SYSTEMSN AS Instrument,

SUM(CASE WHEN COMPONENT = 'CarrierScheduler: CarrierScanned'

THEN 1 ELSE 0 END) AS Num\_Scans,

SUM(CASE WHEN COMPONENT LIKE '%Load%Pick%' AND ACTIVITY LIKE 'Retry%'

THEN 1 ELSE 0 END) AS Num\_Retry,

SUM(CASE WHEN COMPONENT LIKE '%Load%Pick%' AND ACTIVITY LIKE 'Exceed%'

THEN 1 ELSE 0 END) AS Num\_Exceed,

SUM(CASE WHEN COMPONENT LIKE '%Load%Pick%' AND ACTIVITY LIKE '%engagement%'

THEN 1 ELSE 0 END) AS Num\_Engage

FROM SVC\_PHM\_ODS.PHM\_ODS\_CI\_SCM\_INSTACTIVITY --IDAQOWNER.ICQ\_INSTRUMENTACTIVITY

WHERE TRUNC(LOGDATE\_LOCAL) >= TRUNC(SYSDATE) - 7 AND

TRUNC(LOGDATE\_LOCAL) <= TRUNC(SYSDATE) - 1 AND

SYSTEMSN LIKE 'SCM%'

GROUP BY TRUNC(LOGDATE\_LOCAL), DEVICEID, SYSTEMSN

)

GROUP BY DEVICEID, Instrument

)

)

WHERE 2.3 \* Frac\_Recover + 2.6 \* Frac\_Engage + 0.68 \* PerDay\_Recover + 0.85 \* PerDay\_Engage >= 3.97

ORDER BY DEVICEID, Instrument

);

-- PURPOSE : TO CONFIRM THE AVALIABILITY OF ODS BASIC DETAILS

IF VALGNAME IS NOT NULL

THEN

-- STEP 5a : CHECK DATA EXISTS FOR BATCH AND RUN DATE IN THE ALGORITHM OUTPUT TABLE , IF DATA EXISTS DELETE THE DATA FROM OUTPUT TABLE

SELECT COUNT (\*)

INTO V\_EXISTING\_REC\_CNT

FROM PHM\_ALG\_OUTPUT

WHERE BATCH\_NUM = V\_BATCH\_NUM

AND RUN\_DATE = V\_RUN\_DATE

AND PHM\_PATTERNS\_SK = V\_ALG\_NUM; --AND ROWNUM < 5;

--DBMS\_OUTPUT.PUT\_LINE('EXISTING RECORD COUNT IN PHM\_ALG\_OUTPUT: ' || V\_EXISTING\_REC\_CNT);

IF V\_EXISTING\_REC\_CNT > 0

THEN

DELETE FROM PHM\_ALG\_OUTPUT

WHERE BATCH\_NUM = V\_BATCH\_NUM

AND RUN\_DATE = V\_RUN\_DATE

AND PHM\_PATTERNS\_SK = V\_ALG\_NUM;

COMMIT;

END IF;

--DBMS\_OUTPUT.PUT\_LINE('TOTAL RECORDS DELETED FROM PHM\_ALG\_OUTPUT: ' || V\_EXISTING\_REC\_CNT);

-- STEP 5b : CHECK DATA EXISTS FOR BATCH AND RUN DATE IN THE ALGORITHM CHART OUTPUT TABLE , IF DATA EXISTS DELETE THE DATA FROM OUTPUT TABLE

SELECT COUNT (\*)

INTO V\_EXISTING\_REC\_CNT

FROM PHM\_ALG\_CHART\_OUTPUT

WHERE BATCH\_NUM = V\_BATCH\_NUM

AND RUN\_DATE = V\_RUN\_DATE

AND PHM\_PATTERN\_SK = V\_ALG\_NUM; --AND ROWNUM < 5;

--DBMS\_OUTPUT.PUT\_LINE('EXISTING RECORD COUNT IN PHM\_ALG\_CHART\_OUTPUT: ' || V\_EXISTING\_REC\_CNT);

IF V\_EXISTING\_REC\_CNT > 0

THEN

DELETE FROM PHM\_ALG\_CHART\_OUTPUT

WHERE BATCH\_NUM = V\_BATCH\_NUM

AND RUN\_DATE = V\_RUN\_DATE

AND PHM\_PATTERN\_SK = V\_ALG\_NUM;

COMMIT;

END IF;

--DBMS\_OUTPUT.PUT\_LINE('TOTAL RECORDS DELETED FROM PHM\_ALG\_CHART\_OUTPUT: ' || V\_EXISTING\_REC\_CNT);

-- STEP 6 : ALGORIOTHM SPECIFIC CODE - TO WRITE INTO COMMON OUTPUT TABLE>

-- FOR EACH OF THE INSTRUMENTS COMING IN THE BATCH, IDENTIFY IF IT IS IN THE FLAGGED LIST,

-- IF YES, it is flagged, otherwise it is unflagged

V\_FLAG\_COUNT := 0;

FOR DL IN DEVICE\_SN\_LIST

LOOP

BEGIN

V\_FLAG := 'NO';

V\_IHN\_LEVEL3\_DESC\_VAL := NULL;

V\_FLAG\_DATE\_TIME := V\_RUN\_DATE;

V\_RES\_COUNT := 0;

V\_FLAGGED\_PL := NULL;

V\_FLAGGED\_EXP\_CODE := NULL;

FOR indx IN 1 .. FLG\_TBL.COUNT

LOOP

IF FLG\_TBL (indx).MODULESNDRM = DL.SERIAL\_NUM

THEN

V\_FLAG := 'YES';

V\_IHN\_LEVEL3\_DESC\_VAL := V\_IHN\_LEVEL3\_DESC;

V\_RES\_COUNT := 1;

V\_FLAG\_COUNT := V\_FLAG\_COUNT + 1;

-- Get the PL and experience code for the flagged instrument

PHM\_ALGORITHM\_UTILITIES\_1.PHM\_GET\_PL\_EXP\_CODE (

V\_ALG\_NUM,

DL.PL,

NULL,

V\_FLAGGED\_PL,

V\_FLAGGED\_EXP\_CODE);

END IF;

END LOOP;

-- INSERT THE DATA INTO COMMON RESULT OUTPUT TABLE

PHM\_ALGORITHM\_UTILITIES\_1.PHM\_ALGORITHM\_OUTPUT\_IN\_EXP\_PL (

DL.CUSTOMER\_NAME,

DL.CUSTOMER\_NUMBER,

DL.DEVICEID,

DL.SERIAL\_NUM,

DL.COUNTRY\_NAME,

DL.AREA,

V\_ALG\_DFN\_SK,

-1,

V\_FLAG\_DATE\_TIME,

V\_RES\_COUNT,

V\_FLAG,

V\_IHN\_LEVEL3\_DESC\_VAL,

NULL,

VALGNAME,

NULL,

V\_BATCH\_NUM,

V\_ALG\_NUM,

V\_RUN\_DATE,

V\_PROCESS\_ID,

V\_FLAGGED\_PL,

V\_FLAGGED\_EXP\_CODE);

-- INSERT THE DATA INTO COMMON CHART OUTPUT TABLE

PHM\_ALGORITHM\_UTILITIES\_1.PHM\_ALG\_CHART\_INSERT (

DL.DEVICEID,

DL.PL,

DL.SERIAL\_NUM,

DL.COUNTRY\_NAME,

DL.AREA,

V\_ALG\_NUM,

NULL,

NULL,

V\_FLAG\_DATE\_TIME,

V\_RES\_COUNT,

TO\_CHAR (GET\_MS\_FROM\_DATE (V\_FLAG\_DATE\_TIME)),

VALGNAME,

SYSDATE,

V\_BATCH\_NUM,

V\_RUN\_DATE,

V\_ALG\_DFN\_SK);

V\_REC\_COUNT := V\_REC\_COUNT + 1;

IF V\_REC\_COUNT > 5000

THEN

V\_REC\_COUNT := 0;

COMMIT;

END IF;

V\_REC\_INS\_COUNT := V\_REC\_INS\_COUNT + 1; -- DL%ROWCOUNT

EXCEPTION

-- PURPOSE - TO CATCH ALL THE RUN TIME EXCEPTIONS AND TO UPDATE THE AUDIT TABLES WITH ERROR STATUS

WHEN OTHERS

THEN

V\_ERROR\_MESSAGE :=

' PHM\_ICQ\_RSM\_Pick\_Load\_PROC EXECUTION HAS FAILED FOR '

|| V\_ALG\_NUM

|| ' FOR '

|| DL.SERIAL\_NUM

|| ' FOR DATE '

|| V\_FLAG\_DATE\_TIME

|| ', ERROR :'

|| SQLERRM;

V\_PROCESS\_STATUS := 'ERRORED';

PHM\_ALGORITHM\_UTILITIES\_1.PHM\_PROCESS\_AUDIT\_LOG (

V\_PROCESS\_ID,

V\_PROD\_FAMILY,

V\_PROCESS\_TYPE,

V\_ROUTINE\_TYPE,

VALGNAME,

V\_ROUTINE\_NAME,

V\_RUN\_MODE,

V\_PROCESS\_STATUS,

V\_ERROR\_MESSAGE,

V\_RUN\_DATE,

SYSDATE,

V\_BATCH\_NUM,

V\_UNIX\_ID,

V\_ALG\_NUM);

EXIT;

END;

END LOOP;

-- </ CHANGE >

--DBMS\_OUTPUT.PUT\_LINE('PHM\_ICQ\_RSM\_Pick\_Load\_PROC Execution COMPLETED Successfully. Total records inserted: ' || V\_REC\_INS\_COUNT || ', Flagged Count: ' || V\_FLAG\_COUNT);

-- STEP 7 PURPOSE - TO UPDATED THE PROCESS WITH COMPLETED STATUS IN THE AUDIT TABLES

V\_PROCESS\_STATUS := 'COMPLETED';

V\_ERROR\_MESSAGE := '';

PHM\_ALGORITHM\_UTILITIES\_1.PHM\_PROCESS\_AUDIT\_LOG (V\_PROCESS\_ID,

V\_PROD\_FAMILY,

V\_PROCESS\_TYPE,

V\_ROUTINE\_TYPE,

VALGNAME,

V\_ROUTINE\_NAME,

V\_RUN\_MODE,

V\_PROCESS\_STATUS,

V\_ERROR\_MESSAGE,

V\_RUN\_DATE,

SYSDATE,

V\_BATCH\_NUM,

V\_UNIX\_ID,

V\_ALG\_NUM);

COMMIT;

ELSE

V\_ERROR\_MESSAGE :=

' NOT ABLE FIND BASIC INFORMATION OF ALGORITHM '

|| V\_ALG\_NUM

|| ' WITH ERROR '

|| SQLERRM;

V\_PROCESS\_STATUS := 'ERRORED';

PHM\_ALGORITHM\_UTILITIES\_1.PHM\_PROCESS\_AUDIT\_LOG (V\_PROCESS\_ID,

V\_PROD\_FAMILY,

V\_PROCESS\_TYPE,

V\_ROUTINE\_TYPE,

VALGNAME,

V\_ROUTINE\_NAME,

V\_RUN\_MODE,

V\_PROCESS\_STATUS,

V\_ERROR\_MESSAGE,

V\_RUN\_DATE,

SYSDATE,

V\_BATCH\_NUM,

V\_UNIX\_ID,

V\_ALG\_NUM);

END IF;

EXCEPTION

-- PURPOSE - TO CATCH ALL THE RUN TIME EXCEPTIONS AND TO UPDATE THE AUDIT TABLES WITH ERROR STATUS

WHEN OTHERS

THEN

V\_PROCESS\_STATUS := 'ERRORED';

V\_ERROR\_MESSAGE :=

'ALGORITHM EXECUTION FAILED FOR PHM\_ICQ\_RSM\_Pick\_Load\_PROC, DUE TO: '

|| SQLERRM;

PHM\_ALGORITHM\_UTILITIES\_1.PHM\_PROCESS\_AUDIT\_LOG (V\_PROCESS\_ID,

V\_PROD\_FAMILY,

V\_PROCESS\_TYPE,

V\_ROUTINE\_TYPE,

VALGNAME,

V\_ROUTINE\_NAME,

V\_RUN\_MODE,

V\_PROCESS\_STATUS,

V\_ERROR\_MESSAGE,

V\_RUN\_DATE,

SYSDATE,

V\_BATCH\_NUM,

V\_UNIX\_ID,

V\_ALG\_NUM);

COMMIT;

END PHM\_ICQ\_RSM\_Pick\_Load\_PROC;

Note: In Apollo, the codes have been added to the algorithm in order to populate the QlikView Dashboard for the following:

* Country
* Customer Name
* Healthy Instruments
* No Data

**Appendix 2**

**Apollo Algorithm Details**

(\* is Mandatory)

|  |  |
| --- | --- |
| **Apollo Details** |  |
| Algorithm ID \* | Alinity CI RSM Pick and Load |
| Algorithm Name \* | Alinity CI RSM Pick and Load |
| Algorithm Description \* | To implement a Predictive Health Notification (PHN) for ALINITY ci Analyzers that will detect Reagent and Sample Manager (RSM) Pick and Load issues before the customer experiences transport issues. |
| Product Family \* | Alinity CI SCM |
| Algorithm Group \* | Alinity RSM |
| Functional Area | N/A |
| Algorithm Category 1 | N/A |
| Algorithm Category 2 | N/A |
| Algorithm Category 3 | N/A |
| Remaining Useful Life Value | 7 |
| Remaining Useful Life Unit | Day |
| Keep Results Num Days | 14 |
| **Routine Details** |  |
| Routine Source | Define New Routine |
| Routine Type | Oracle Procedure |
| Run Mode | Batch |
| Routine Invoke Command | PHM\_ICQ\_RSM\_Pick\_Load\_PROC |
| Status | Enable |
| **ODS Routine Details** |  |
| ODS Routine Name | PHM\_ODS\_SCM\_ACTIVITY\_PROC |
| **Predictive Health Notification Details** |  |
| PHN Code | PHN\_Alinity\_ci SCM\_A832 |
| Issue Description (Use Algorithm Name) |  |
| Experience Code | A832-214- PHN\_Alinity\_ci SCM:PHN\_RSM Transport; Pick and Load |
| **Knowledge Management DB Articles** |  |
| KM Article ID | K91150585 |
| KM Article | PHN\_Alinity\_ci SCM: RSM Transport; Pick and Load |
| **Parameters** |  |
| Parameter Group Name | RSM |
| **Parameter Name** | **Parameter Values** |
| Alinity CI RSM Pick and Load | IHN\_LEVEL3\_DESC, Alinity CI RSM Pick and Load |
| Alinity CI RSM Pick and Load | THRESHOLDS\_COUNT |
| **Chart Details** |  |
| Chart Title | Alinity ci RSM Transport |
| Chart Type | BAR |
| Chart Threshold Parameter | Alinity ci RSM Transport - Threshold Count |
| Group ID | Group7 |
| Chart X Axis Name | Instrument Date/Time |
| Chart Y Axis Name | Error Count |

**APPENDIX 2:** Algorithm Verification in Apollo

During the time of the algorithm verification, Apollo Dev was unavailable due to BDAA transition activities. What follows is testing that is possible via manual execution on Apollo Prod. The tests ran in Apollo Prod with a given run date / batch number. Verification will ensure only that all flags that exist in Apollo are also generated via manual execution on IDA. We expect more flags in the manual execution on IDA than in the Apollo Prod execution due to Apollo Prod batching covering two hours and manual execution on IDA covering the entire day.

**Apollo Prod execution**

The serial numbers generated were based on a run date of “13-JUL-19” and batch number of “BTH1000” during a manual execution of ‘RSM Pick and Load’ algorithm in Apollo Prod. Apollo Prod stores the data generated by the run in the “svc\_phm\_owner.phm\_alg\_output” table. Below is the query and results from this run.

select o.sn

from PHM\_ALG\_OUTPUT O

where O.PHM\_PATTERNS\_SK in (20531)

and flag\_yn= 'YES'

and trunc(DATE\_CREATED) = to\_date('07/13/2019', 'mm/dd/yyyy')

and batch\_num = 'BTH1000'

order by O.DATE\_CREATED desc

SN

SCM01895

SCM01818

SCM02405

SCM02616

SCM02208

SCM03133

SCM02549

SCM03429

SCM02547

**Local execution from IDAQOWNER**

The serial numbers generated were using PABBTO IDAQOWNER tables instead of PABBTO SVC\_PHM\_ODS tables created for Apollo. The algorithm was run on the PHM developer’s local machine using R and a run date of “2019-07-13”.

SN

SCM01207

SCM01381

SCM01560

SCM01738

SCM01447

SCM02018

SCM01904

SCM01895

SCM02220

SCM01818

SCM01854

SCM01987

SCM01970

SCM01116

SCM01977

SCM01731

SCM02547

SCM02633

SCM02405

SCM03133

SCM02549

SCM02608

SCM02208

SCM02616

SCM02682

SCM02897

SCM03429

**Apollo Prod execution**

The serial numbers generated were based on a run date of “14-JUL-19” and batch number of “BTH0600” during a manual execution of ‘RSM Pick and Load’ algorithm in Apollo Prod. Apollo Prod stores the data generated by the run in the “svc\_phm\_owner.phm\_alg\_output” table. Below is the query and results from this run.

select o.sn

from PHM\_ALG\_OUTPUT O

where O.PHM\_PATTERNS\_SK in (20531)

and flag\_yn= 'YES'

and trunc(DATE\_CREATED) = to\_date('07/14/2019', 'mm/dd/yyyy')

and batch\_num = 'BTH1000'

order by O.DATE\_CREATED desc

SN

SCM02682

SCM02897

SCM02208

SCM02608

**Local execution from IDAQOWNER**

The serial numbers generated were using PABBTO IDAQOWNER tables instead of PABBTO SVC\_PHM\_ODS tables created for Apollo. The algorithm was run on the PHM developer’s local machine using R and a run date of “2019-07-14”.

SN

SCM01207

SCM01213

SCM01381

SCM01560

SCM01738

SCM01447

SCM01666

SCM02018

SCM01904

SCM01895

SCM01818

SCM01854

SCM01987

SCM01970

SCM01116

SCM01977

SCM01731

SCM02547

SCM02633

SCM02405

SCM03133

SCM02549

SCM02608

SCM02208

SCM02616

SCM02682

SCM02897

SCM02382

SCM03429

**Algorithm Transition Summary**

Based on the outputs from both the Apollo Prod run and local execution from IDAQOWNER, and the understanding of the batched Apollo Prod output, the delivered algorithm is confirmed. Both the Apollo Prod run and PHM developer analyzed the same data set and got the same results (with the exception of the instruments not in the Apollo batch as mentioned previously). This means that both the Apollo Prod run and PHM developer flagged the same algorithm violations within the given data set.