|  |
| --- |
| Predictive Health Notification  Alinity i Trigger/Pre-Trigger Dispense |
| ALINITY i Immunoassay Analyzer  July 1, 2019 |

**PHN - Alinity i Trigger/Pre-Trigger Dispense PHN Spec Sheet for Apollo**

**References**

D000072998/B Predictive Health Notification for ALINITY i Trigger/Pre-Trigger Straw

**Summary**

To implement a Predictive Health Notification (PHN) for ALINITY i Analyzers that will monitor depressed optic reads to detect degrading trigger/pre-trigger dispense operations before the customer begins to experience an increase in optics-related errors.

|  |  |
| --- | --- |
| PHN Descriptor | Alinity IA Trigger/Pre-Trigger Dispense |
| PHN Experience Code / Name | CCN1 PHN\_Alinity\_IA; PHN\_Bulk Solutions; Trigger/Pre-Trigger Dispense |
| PHN KM Article Number/ Name | K66203158 PHN\_Alinity\_IA: Trigger/Pretrigger Dispense |
| Service Level | 1. Basic Service/Ambassador 2. NOTE: Apollo to send ticket to Call Center |
| Always On Package | Always On 01DP5- 01, 02, 03, 09, 70, 79, 80, 83, 84, 89 |
| IDA Table(s) | IDAQOWNER.ICQ\_RESULTS  IDAQOWNER.ICQ\_RESULTS\_READS |
| IDA Table Fields | LOGDATE\_LOCAL  MODULESN  ICQ\_RESULTS\_ID  INTEGRATEDSIGNALCOUNT  DARKAVERAGE  CORRECTEDCOUNT |
| Analysis Frequency | Daily |
| Data Required | 1 Day |
| Data Aggregation | None |
| Run Time Estimate | N/A |
| Flag Criteria | For each MODULESN (external instruments with no active service event):  If ShapeFlagRate > 1% Then Flag, Else No Flag |
| Probable Failure Modes | Cracks in the Trigger or Pre-Trigger straws, loose connections with the fittings and tubing. |
| Suppression Experience Codes | C9F1, C9F2, C9L1, C9L2, B8FA, B8LA, CBF2, CBF1, CBF3, CBF4, CBF5, CBL2, CBL1, CBL3, CBL4, CBL5 |
| Applicable Work Done Codes (WDC) | CE\*, CD\*, CS\* |
| Part Replaced | LN 04S68 |

**Data Processing**

See Design Document D000072998/B Predictive Health Notification for ALINITY i Trigger/Pre-Trigger Straw for additional data processing information.

|  |
| --- |
| By MODULESN  For Each Rep: calculate the average dark count adjusted signal count and peak shape ratio:    Determine ShapeFlag for each rep:  IF (15 < CORRECTEDCOUNT <= 30 AND PeakShapeRatio ≤ 0.44)  Then ShapeFlag = TRUE,  ELSE IF (31 < CORRECTEDCOUNT <= 50 AND PeakShapeRatio ≤ 0.4)  Then ShapeFlag = TRUE,  ELSE IF (51 < CORRECTEDCOUNT <= 70 AND PeakShapeRatio ≤ 0.35)  Then ShapeFlag = TRUE,  ELSE ShapeFlag = FALSE  Calculate ShapeFlagRate For Each Calendar Day, for each MODULESN:  ShapeFlagRate = Number of ShapeFlags/Number of Total Reps |

**APPENDIX 1:**

**Algorithm Code**

CREATE OR REPLACE PROCEDURE SVC\_PHM\_OWNER.PHM\_ICQ\_Trigger\_Straw\_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 PARAMETER VARAIBLES TO HANDLE THE PROCESS FLOW

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

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

V\_FLAGGED\_PL VARCHAR2 (10);

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

-- ALGORITHM LOCAL VARAIBLES TO HANDLE THE PROCESS FLOW

V\_EXISTING\_REC\_CNT NUMBER;

V\_FLAG VARCHAR (5);

V\_REC\_COUNT NUMBER := 0;

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

V\_FLAG\_DATE\_TIME DATE;

V\_RES\_COUNT NUMBER;

V\_FLAG\_COUNT NUMBER;

VALGNAME VARCHAR (25);

V\_ALG\_DFN\_SK NUMBER;

-- Cursor to identify flagged instruments. Written as a function

CURSOR FLAG\_LIST (

V\_MODULESNDRM VARCHAR2,

V\_DEVICEID NUMBER)

IS

SELECT MODULESN,DeviceId

FROM

(SELECT

DeviceId, MODULESN, TO\_CHAR(LOGDATE\_LOCAL, 'YYYY-MM-DD') AS date\_only,

(CASE

WHEN CORRECTEDCOUNT <= 30 AND peak\_adj\_signal / dark\_adj\_signal <= 0.44 THEN 1

WHEN CORRECTEDCOUNT <= 50 AND peak\_adj\_signal / dark\_adj\_signal <= 0.40 THEN 1

WHEN CORRECTEDCOUNT <= 70 AND peak\_adj\_signal / dark\_adj\_signal <= 0.35 THEN 1

ELSE 0

END) AS shapeflag

FROM

(SELECT

i.DeviceId, i.MODULESN, i.LOGDATE\_LOCAL, i.CORRECTEDCOUNT,

SUM(CASE WHEN r.signal > i.DARKAVERAGE AND r.time IN (4, 5, 6, 7)

THEN r.signal - i.DARKAVERAGE ELSE 0 END) AS peak\_adj\_signal,

SUM(CASE WHEN r.signal > i.DARKAVERAGE

THEN r.signal - i.DARKAVERAGE ELSE 0 END) AS dark\_adj\_signal

FROM SVC\_PHM\_ODS.PHM\_ODS\_ICQ\_RESULTS i

--IDAQOWNER.ICQ\_RESULTS i

INNER JOIN

(SELECT \*

FROM SVC\_PHM\_ODS.PHM\_ODS\_ICQ\_RESULTS\_READS --IDAQOWNER.ICQ\_RESULTS\_READS

UNPIVOT (signal FOR time IN

(S01 AS 1, S02 AS 2, S03 AS 3, S04 AS 4, S05 AS 5, S06 AS 6,

S07 AS 7, S08 AS 8, S09 AS 9, S10 AS 10, S11 AS 11, S12 AS 12,

S13 AS 13, S14 AS 14, S15 AS 15, S16 AS 16, S17 AS 17, S18 AS 18,

S19 AS 19, S20 AS 20, S21 AS 21, S22 AS 22, S23 AS 23, S24 AS 24,

S25 AS 25, S26 AS 26, S27 AS 27, S28 AS 28, S29 AS 29, S30 AS 30))

) r

ON

i.ID = r.ICQ\_RESULTS\_ID

WHERE i.MODULESN = V\_MODULESNDRM

AND i.DEVICEID = V\_DEVICEID AND

TRUNC(i.LOGDATE\_LOCAL) = TRUNC(SYSDATE) - 1 AND

TRUNC(r.LOGDATE\_LOCAL) = TRUNC(SYSDATE) - 1 AND

LOWER(i.SAMPLEID) NOT LIKE '%saline%' AND

LOWER(i.SAMPLEID) NOT LIKE '%buf%' AND

LOWER(i.OPERATORID) NOT LIKE 'fse' AND

ASSAYNUMBER NOT LIKE '%213%' AND

ASSAYNUMBER NOT LIKE '%216%'

GROUP BY

i.DeviceId, i.MODULESN, i.LOGDATE\_LOCAL, i.CORRECTEDCOUNT

)

)

GROUP BY

DeviceId, MODULESN, date\_only

HAVING

COUNT(\*) >= 50 AND

AVG(shapeflag) > 0.01;

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

CURSOR DEVICE\_SN\_LIST

IS

SELECT CC.DEVICEID,

UPPER (CC.MODULESN) 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\_RESULTS\_CC CC,

SVC\_PHM\_ODS.PHM\_ODS\_ICQ\_RESULTS CC,

INSTRUMENTLISTING IL,

PHM\_COUNTRY PC

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

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

AND UPPER (CC.MODULESN) = UPPER (IL.SN)

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

AND IL.INST\_STATUS = 'Active'

GROUP BY CC.DEVICEID, CC.MODULESN;

BEGIN

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

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

V\_PROCESS\_STATUS := 'STARTED';

-- 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;

-- 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);

-- 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);

-- 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 NVL (DELETE\_FLAG, 'N') <> 'Y')

LOOP

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

THEN

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

END IF;

END LOOP;

-- 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;

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;

-- 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;

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;

-- 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 x IN FLAG\_LIST (DL.SERIAL\_NUM, DL.DEVICEID)

LOOP

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 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\_Trigger\_Straw\_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 >

-- 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\_Trigger\_Straw, 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\_Trigger\_Straw\_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 IA Trigger/Pre-Trigger Dispense |
| Algorithm Name \* | Alinity IA Trigger/Pre-Trigger Dispense |
| Algorithm Description \* | To detect degrading trigger/pre-trigger dispense operations caused by depressed optic reads. |
| Product Family \* | Alinity IA |
| Algorithm Group \* | Alinity Bulk Solutions |
| 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\_Trigger\_Straw\_PROC |
| Status | Enable |
| **ODS Routine Details** |  |
| ODS Routine Name | PHM\_ODS\_ICQ\_RESULTS\_PROC |
| **Prognostic Health Notification Details** |  |
| PHN Code | PHN\_Alinity IA\_CCN1 |
| Issue Description (Use Algorithm Name) | Alinity IA Trigger/Pre-Trigger Dispense |
| Experience Code | CCN1-205- PHN\_Alinity\_IA: PHN\_Bulk Solutions: Trigger/Pretrigger Dispense |
| **Knowledge Management DB Articles** |  |
| KM Article | PHN\_Alinity\_IA: Trigger/Pretrigger Dispense |
| KM Article ID | K66203158 |
| **Parameters** |  |
| Parameter Group Name | Alinity IA Trigger/Pre-Trigger Dispense |
| **Parameter Name** | **Parameter Values** |
| Alinity IA Trigger/Pre-Trigger Dispense | IHN\_LEVEL3\_DESC, Alinity IA Trigger/Pre-Trigger Dispense |
| Alinity IA Trigger/Pre-Trigger Dispense | THRESHOLDS\_COUNT, 1 |
| **Chart Details** |  |
| Chart Title | Alinity IA Trigger/Pre-Trigger Dispense |
| Chart Type | Bar |
| Chart Threshold Parameter | Alinity IA Trigger/Pre-Trigger Dispense - THRESHOLDS\_COUNT |
| Group ID | Group 7 |
| Chart X Axis Name | Instrument Date/Time |
| Chart Y Axis Name | Error Count |

**APPENDIX 3:** 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.

In order to get flagged instrument, the Flag Criteria had changed to for each MODULESN (external instruments with no active service event):If ShapeFlagRate > 0.1% Then Flag, Else No Flag.

**Apollo Prod execution**

The serial numbers generated were based on a run date of “12-JUL-19” and batch number of “BTH0200” during a manual execution of the Trigger/Pre-Trigger 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 (20571)

and flag\_yn= 'YES'

and trunc(DATE\_CREATED) = to\_date('07/12/2019', 'mm/dd/yyyy')

and batch\_num = 'BTH0200'

order by O.DATE\_CREATED desc

SN

AI01268

AI02800

**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 “2019-07-12”.

SN

AI01228

AI01268

AI02800

**Apollo Prod execution**

The serial numbers generated were based on a run date of “13-JUL-19” and batch number of “BTH0600” during a manual execution of the Trigger/Pre-Trigger 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 (20571)

and flag\_yn= 'YES'

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

and batch\_num = 'BTH0600'

order by O.DATE\_CREATED desc

SN

AI03209

**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

AI03209

**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.