OAK RIDGE
NATIONAL LABORATORY
MANAGED BY UT-BATTELLE
FOR THE DEPARTMENT OF ENERGY

ORNL/TM-2012/239

Methodology for Determining Radiation Portal Monitor Availability from Daily Files

December 2012

Prepared by

Tyler Guzzardo
Alex Enders
Scott Alcala

UT-BATTELLE
ORNL-27 (4-00)

ORNL/TM-2012/239

Global Nuclear Security Technology Division

METHODOLOGY FOR DETERMINING RADIATION PORTAL MONITOR AVAILABILITY FROM DAILY FILES

Tyler Guzzardo
Alex Enders
Scott Alcala

Date Published: December 2012

Prepared by
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37831-6283
managed by
UT-BATTELLE, LLC
for the
U.S. DEPARTMENT OF ENERGY
under contract DE-AC05-00OR22725

CONTENTS

Page

1. OVERVIEW

The Second Line of Defense (SLD) Program works with partner countries to enhance their capabilities in deterring, detecting, and interdicting the illicit trafficking of nuclear and radiological materials through international borders by providing radiation detection equipment and related training. With agreement from SLD partner countries, the program collects daily file data from installed TSA Systems pedestrian, conveyor, vehicle, and rail radiation portal monitors (RPMs).[1] Analysis of the data can provide information related to the state of health and configuration of the radiation monitoring equipment. By analyzing this data from the field over long periods of time, the program is able to determine valuable information such as the general reliability of the deployed equipment. In order to characterize the program's overall effectiveness, data from daily files is used to determine the availability of RPMs that submit daily files to Oak Ridge National Laboratory (ORNL). Availability in this context is the proportion of time the RPM is in a functioning condition. The RPM is available when all the components of the system are functioning properly, thereby allowing the proper functioning of the RPM as a whole. This document explains the methodology and assumptions used to arrive at an availability metric for the RPMs from which ORNL receives data.

2. DAILY FILES

Data from the RPMs is stored in the form of daily files that contain a log of operations performed by the RPM for 24 hours. Each line in the daily file contains a two-character code that defines the contents of the line, followed by several data elements that pertain to the two-character code. The definitions of each type of two-character code are listed in Fig. 1. Some of the basic information contained in the daily files includes gamma and neutron background count rates, gamma and neutron signal count rates, alarm conditions, current RPM settings, and fault conditions. Although numerous countries send data to ORNL,

```
NB,000001,000001,000001,000001,05-45-39.921
GB,000301,000311,000306,000304,05-45-40.015
NB,000001,000001,000001,000001,05-45-44.921
GB,000303,000309,000306,000304,05-45-45.015
SP,0.1122,06.077,009.78,000000,05-45-46.640
NS,000001,000001,000000,000002,05-45-46.765
GS,000061,000049,000072,000055,05-45-46.765
GS,000059,000064,000048,000056,05-45-46.765
GS,000055,000056,000066,000043,05-45-46.765
GS,000057,000066,000054,000053,05-45-46.765
GS,000069,000040,000062,000064,05-45-46.765
GS,000059,000045,000053,000047,05-45-46.765
GS,000060,000038,000050,000047,05-45-46.937
GS,000062,000047,000050,000037,05-45-47.125
GS,000060,000059,000043,000040,05-45-47.343
GS,000053,000059,000053,000050,05-45-47.531
NS,000001,000002,000001,000000,05-45-47.765
GS,000053,000061,000052,000058,05-45-47.765
```

Daily File Line Types	
GB	Gamma Background
NB	Neutron Background
GS	Gamma Scan
NS	Neutron Scan
SP	Speed Message
GX	End of Occupancy
GA	Gamma Alarm
NA	Neutron Alarm
SG	Gamma Setting
SN	Neutron Setting
TT	Tamper Fault
TC	Tamper Cleared
GH	Gamma High Fault
GL	Gamma Low Fault
NH	Neutron High Fault

Fig. 1. Sample of daily file *(left)* and line types *(right)*.

[1] *Train Monitor TM-850 Operations and Service Manual,* Doc 5022 Rev B, TSA Systems, Ltd., January 2011. *Vehicle Monitor VM-250 Operations and Service Manual,* Doc 5039 Rev A, TSA Systems, Ltd., January 2011. *Pedestrian Monitor PM-700 Operations and Service Manual,* Doc 5038 Rev A, TSA Systems, Ltd., January 2011.

the files received are only a small subset of all possible daily files. Therefore, RPM availability is only representative of the sites that send data to ORNL.

3. ORNL SLD RPM DATABASE

ORNL receives approximately 40 gigabytes of data per month in the form of daily files from partner countries. Data is extracted from the daily files and uploaded to the RPM database to organize the data and increase access speeds. It is because of the database that ORNL data analysts can quickly query daily file data and troubleshoot potential issues with the RPMs. The large dataset allows statistically significant conclusions to be drawn about the deployed RPMs. Instead of reading directly from daily files, scripts written at ORNL query the database and extract the data required for calculating availability.

Since ORNL receives a large quantity of data, measures were taken to reduce the amount of data uploaded to the database. As a result, all the data in the daily files is not present in the database. Gamma Background lines are written to daily files every 5 seconds but are only recorded in the database every 10 minutes. Similarly, faults are recorded in the database every 10 minutes, and the precise time of occurrence is not stored. For non-alarming occupancies, only eight entries (four gamma and four neutron) are stored for each occupancy as compared to higher frequency data for alarming occupancies. These differences between the daily files and the database were considered when determining the methodology for calculating availability.

4. METHODOLOGY

When calculating RPM availability, ORNL adopted a philosophy to restrict "unavailable" to those times where the RPM was demonstrably inoperable, per the data available. There are many instances where the data received by ORNL is not continuous, but the absence of data is insufficient evidence to consider the RPM as unavailable: RPMs may be turned off when not in use, or daily files may be overlooked in the data collection process. Thus, no credit or penalty is applied for missing data.

For the data that ORNL does receive, daily files are examined against five criteria that provide evidence of a broken RPM component. Only one component of the RPM needs to be broken to make the system unavailable. Availability is calculated by analyzing all the daily files received and determining what fraction of the daily files shows evidence of RPM failure:

$$Availability = \frac{Daily\ File\ Data\ Received - Daily\ File\ Data\ with\ Evidence\ of\ RPM\ Failure}{Daily\ File\ Data\ Received}$$

The criteria used in the availability algorithm were chosen specifically for their ability to differentiate between when an RPM is operable and inoperable and should not be confused with metrics to determine system effectiveness. It is possible an RPM deemed available through the methodology used here may not be capable of effectively detecting special nuclear material (SNM). To accurately define system effectiveness, an algorithm needs to consider RPM alarm rate, staffing requirements for the site, secondary inspection ability, minimum detectable quantity of SNM, accuracy of settings, and numerous other factors that contribute to a site's ability to interdict SNM. Therefore, the availability metric should not be used to determine when a system is effective, but only to calculate when the system is available.

The criteria that provide evidence of an RPM failure are empty daily files, hours of missing data, hours without a background update, fault conditions, and days with two neutron signals constantly measuring zero counts. No single criterion can measure system availability, and these criteria can only approximate

when an RPM component is broken. However, the methodology uses the data available in the RPM database to form a justifiable and reproducible approximation of availability. Each criterion is described in detail below.

4.1 EMPTY DAILY FILES

Criterion	Condition	Decrement
Empty	Daily file <1 kB in size?	24 hours of availability

An empty daily file is defined as a daily file with a file size of less than 1 kilobyte. A file this small is evidence of a communication failure between the RPM and the CAS (Central Alarm Station), which can prevent alarm indicators from being sent to the CAS. As a result, the CAS operator may only be notified of an alarm by the RPM relay (alarm lights and audible annunciator) if the CAS operator is stationed near the RPM. However, the RPM relay is often disabled at Megaports sites, making the CAS the sole provider of alarm indicators. Overall, an empty daily file is evidence of an RPM system whose alarms may not be received by the operator. An empty daily file removes 24 hours from RPM availability.

Empty files are evidence of a communication failure and not a failure of the software that writes the daily file. If the daily file writer was broken, all the daily files for the site would be empty. This has not been observed in the data; usually empty files are seen for particular lanes while the other lanes at the site produce regular daily files.

An empty daily file is treated differently from not receiving a daily file because the creation of the daily file shows there was a failed opportunity to record data. When a daily file is not received by ORNL, nothing is known about the contents of the file. For instance, the RPM may have operated normally, but a poor internet connection may have prevented the file from being transferred. A lack of daily files cannot remove time from system availability since the system may be functioning properly whether or not ORNL receives the file. However, an empty daily file contains information that demonstrates the CAS was operating but was unable to communicate with the RPM.

4.2 HOURS OF MISSING DATA

Criterion	Condition	Decrement
Hours Missing	Data present each hour?	1 hour of availability

Daily files that are not empty can have periods of time where data is missing within the file. Just as with empty files, this missing data is evidence of a communication failure between the RPM and the CAS. To quantify the missing data, ORNL examines daily files to determine if the timestamp at the end of each line is present for every hour of the day. Every missing hour of timestamps removes 1 hour from RPM availability.[2] The system is only counted as unavailable when all the lines are missing from the hour. Even if one line is present, the whole hour will be counted as available. This metric ensures the system is

[2] Partial days of missing data are measured in terms of whole hours, but evaluating every minute of data was considered. However, it was determined that using a time resolution of 1 minute would likely not increase the accuracy of the availability calculation. This is especially true when the monitor is recording background. Since Gamma Background lines are only stored in the database every 10 minutes, there is no knowledge of whether the system continued to record background between the 10 minute entries. As a result, using a time resolution of 1 minute could imply a detailed precision that is not present in the calculation.

truly unavailable by checking for the presence of multiple lines that should have been uploaded to the database within the hour.

4.3 HOURS WITHOUT A BACKGROUND UPDATE

Criterion	Condition	Decrement
Hours no Bkgd	GB or NB line present each hour?	1 hour of availability

When the background radiation level is not updated regularly, the alarm threshold can no longer be considered valid for current occupancies. This can result in false alarms when the recorded background is less than the current background and in false negatives (missed sources) when the recorded background is greater than the current background. Broken occupancy sensors, vehicles parking between the RPM pillars, and extreme tailgating are possible pathways to preventing the RPM from updating background appropriately. If an hour in the database does not contain a Gamma Background or Neutron Background line, 1 hour is removed from RPM availability.

4.4 GAMMA AND NEUTRON FAULTS

Criterion	Condition	Decrement
Faults	GH, GL, or NH present?	5 seconds of availability / fault

When a background line is written to the daily file, a boundary check is performed that can produce Gamma High, Gamma Low, and Neutron High faults. If a gamma or neutron background measurement is out of range, the associated fault line is written to the daily file instead of the regular Gamma Background or Neutron Background line. The purpose of the faults is to notify the operator of unusual background data. Due to the likely invalid background values, when the RPM is in a fault condition it is unable to make a reliable alarm decision based on the detection system that caused the fault. Incorrect background data can lead to false alarms, missed sources, and operator frustration. To warn the operator of this state, the RPM annunciator is triggered every time the RPM is occupied regardless of the count rate during occupancy. Since the RPM cannot perform the task it was designed to do, the RPM is considered unavailable for every 5 second interval labeled Gamma High, Gamma Low, or Neutron High.[3,4,5] The cause of the faults is irrelevant with regard to availability since anytime a RPM is in a fault condition, no reliable alarm decisions can be made by the associated detection system.

[3] Tamper faults produced from power failure, power restoration, and opening and closing a cabinet are not considered evidence of RPM failure. Routine checks often involve opening the RPM cabinet, and the loss of RPM power is accounted for by checking for missing data.

[4] When the monitor is in occupied mode, fault lines are not written to the daily file, so this time is not removed from availability. The average down time due to faults could be calculated and then extrapolated over the time the RPM is occupied, but assumptions about absent data were avoided when possible. Furthermore, hours of constant occupancy without a background update are already removed from availability, so the daily files that would contribute the most to this positive bias are already accounted for.

[5] Due to limitations of the database, it is possible a pair of Gamma High and Neutron High faults occurring at the same time could decrement availability for 10 seconds when only occurring over a 5 second interval. To minimize this bias, a threshold was established to limit the maximum availability decrement of a 10 minute time period to 10 minutes. In addition, simultaneously triggering Gamma High and Neutron High faults is not a common occurrence.

4.5 TWO NEUTRON SIGNALS MEASURING ZERO

Criterion	Condition	Decrement
Neutron Failure	Neutron Signal = 0 for detector pair?	24 hours of availability

A Gamma Low fault is built into the RPM fault logic to detect an abnormally low signal from the gamma detectors, which is often indicative of failing detector electronics. Similar logic does not exist for the neutron detectors, since a reading of zero counts is a valid and common background. Consequently, an alternative failure condition is required for quantifying neutron detector failure.

The derivation of the failure condition is complex because the values stored in the database for non-alarming occupancies are not the exact values found in the daily files. As an effort to conserve disk space, only four neutron entries are stored for each non-alarming occupancy. These include the signal values from the last Neutron Background line prior to occupancy, the lowest signal values from the Neutron Scan lines during the occupancy, the highest signal values from the Neutron Scan lines during the occupancy, and the signal values from the last Neutron Background line prior to occupancy (again). As a result, non-zero values are only expected from the highest signal values of non-alarming occupancies and the Neutron Scan lines from alarming occupancies. All of the lines stored in the database used to determine neutron detector failure will be referred to as neutron signal lines.

Vehicle and rail RPMs normally output four neutron signal values on a neutron signal line during occupancy.[6] When two of the four signal values are repeatedly zero, this is evidence that half the neutron detectors are not functioning correctly. If all the neutron signal lines in the database from one day (with a minimum of 10 neutron signal lines) contain a pair of signal values equaling zero, then the RPM is considered unavailable for the day.[7,8] If less than 10 neutron signal lines are present in the database during the day, by default the RPM is considered available since the availability of the system is not statistically certain. All pedestrian and conveyor RPMs are not analyzed and considered available by default since certain configurations of these RPMs will only output two non-zero neutron signal values.

For an RPM to be deemed unavailable by means of the neutron failure condition, the same two signal values must be zero for the entire day. If different signal values are zero throughout the day, the system is deemed available:

[6] Neutron Scan lines are recorded in the daily file once per second *during an occupancy.*

[7] Using only Neutron Background lines for calculating availability was considered. Neutron Background lines are written to the daily file every 5 seconds and contain a 20 or 120 second (depending on firmware) average of the measured background. When calculating the average, the result is truncated. This often produces Neutron Background lines that contain all zeros. For instance, if 119 neutron counts were recorded within the last 120 seconds, the neutron background would equate to 0 counts/second. However, since Neutron Scan lines are written to the daily file every second, no division or truncation is required and non-zero numbers are often recorded. For this reason, Neutron Scan lines are considered when determining if a neutron detector is functioning correctly.

[8] Conducting the neutron analysis with a time resolution of 1 hour instead of 1 day was also considered. However, unresponsive neutron detectors do not frequently change between available and unavailable. Typically, the detectors will stay unresponsive until a maintenance team is able to fix them. As a result, analyzing the neutron signals every hour may artificially increase availability; when no neutron signal lines are present in the hour, the system would be deemed available by default, when in actuality the detectors may very well still be unresponsive during the hour without any occupancies. Conducting the neutron analysis on a daily basis provides more opportunity for neutron signal lines to be present in the dataset and for an accurate determination of availability.

Unavailable:

NS,000000,000000,000001,000002,04-30-42.653
NS,000000,000000,000001,000001,04-30-43.653
NS,000000,000000,000001,000004,04-50-14.778
NS,000000,000000,000002,000003,04-50-15.762

Available:

NS,000000,000000,000001,000002,04-30-42.653
NS,000001,000000,000000,000003,04-30-43.653
NS,000001,000002,000000,000000,04-50-14.778
NS,000000,000000,000002,000003,04-50-15.762

A minimum threshold of 10 neutron signal lines is required to ensure a statistical occurrence does not deem the system unavailable. For example, with no threshold, a daily file with a single neutron signal line containing two signal values of zero would deem the system unavailable. The threshold value was determined by sampling all the neutron signal values in the database from 2011 and calculating the fraction of neutron signals with a value of zero. This fraction was used to determine that 10 sequential pairs of zeros are required to ensure the readings of zero are not caused by random statistical events.[9]

5. EXCLUDED METRICS

Numerous metrics were considered when defining the RPM availability algorithm, and some were deemed to not accurately represent whether part of the detection system was broken. The justifications for excluding metrics from the algorithm are described in detail below.

5.1 BAD FORMAT

A set of rules govern the content of daily files so every file can be processed the same way. Daily files are defined as bad format when more than 1% of the lines in the daily file do not follow the daily file line convention. Although a daily file not formatted correctly is difficult to post-process, it is not necessarily evidence of RPM failure. Software on the CAS is responsible for writing the data transmitted from the RPM to the daily file. The RPM may be operating correctly, but if the daily file writer is set up incorrectly, the daily file will be in the wrong format. If the majority of the daily file is in the correct format, it is not well understood what component of the system causes single lines to be printed incorrectly. Therefore, badly formatted daily files are not included in the RPM availability algorithm.

5.2 OVERSIZE FILES

Daily files are flagged as oversize when their file size is greater than a set limit. Over time, this limit has been adjusted for various reasons but is normally in the range of 5 to 10 megabytes. Using this file size as a metric of system availability was considered but ultimately not incorporated into the algorithm. Daily files often become oversize when the RPM is constantly occupied and many signal lines are written to the daily file. During an occupancy, one Gamma Signal line is written to the daily file every 200 milliseconds as compared to background mode where one Gamma Background line is written to the file every 5 seconds. As a result, extremely large daily files are less likely to contain background lines at appropriate intervals. Large daily files can be caused by a variety of situations including broken occupancy sensors,

[9] See Appendix for detailed calculation.

vehicles parking between the RPM pillars, extreme tailgating, and daily files having two timestamps per line. Although some of these situations are evidence of RPM failure, the root cause of the failure is due to a lack of background lines. By spot checking large daily files, files as large as 28.7 MB (double timestamps) and 22.2 MB (single timestamp) were found that included at least one background line. In addition, a file with no background lines could have a small file size due to partial days of missing data. Due to these inconsistencies, it was deemed inappropriate to use file size as a metric for system availability. Instead of file size, every hour of the daily file is checked to determine if the RPM background was updated within the hour. Utilizing this method provides the closest link between the daily file and system availability.

6. CONCLUSION

The methodology used to determine RPM availability from daily files is thoroughly defined, and the assumptions made to create the algorithm are described in detail. Since a set of well-defined metrics define the availability algorithm, RPM availability is a reproducible value that does not require the subjective view of data analysts. The algorithm for calculating RPM availability from SLD daily files can be utilized to justifiably track the operation of RPMs across the program.

APPENDIX A

A.1 CALCULATION OF MINIMUM REQUIRED NEUTRON SIGNAL LINES

The probability of a neutron signal value of zero occurring by means of random statistical events (i.e., chance alone) was estimated by sampling all the available data from 2011. The fraction of all the neutron signal values in the database with a value of zero was 0.45. However, this method assumes all RPM configurations have an equal likelihood of measuring neutron signal values of zero. This is unlikely since many factors such as the number of He-3 tubes and the altitude of the installation influence the neutron signal value. Consequently, a histogram of daily average neutron signal values recorded during occupancies (Fig. A.1) was created to determine the extent of variation between RPM configurations. Since daily averages are presented, each data point is representative of the average neutron signal value from a particular lane. Averaging the data in this manner allows neutron signal value trends to be exposed.

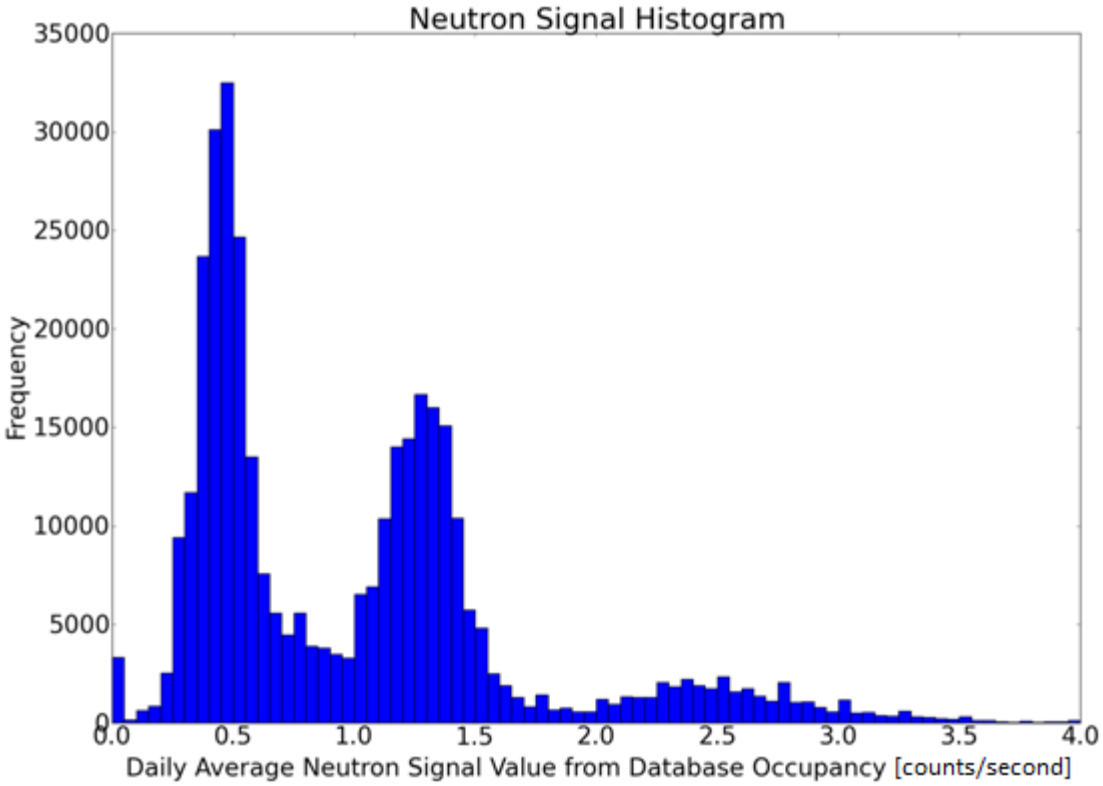

Fig. A.1. Histogram of daily average neutron signal values in 2011.

The distinctive peaks are evidence the average neutron signal value is dependent on the RPM configuration and that all RPM configurations will not experience the same probability of a neutron signal value measuring zero.[10] However, the peaks in the distributon do not vary so greatly that it is unreasonable to sample the entire database to estimate the probability of measuring a neutron signal value of zero. The histogram illustrates the inherent uncertainty in this calculation and provides an explanation to why the probability of 0.45 is an estimation.

Given the probability, p, of one neutron signal value measuring zero, the probability, P, of a sequence of L trials also measuring zero can be determined:

[10] Determining the RPM configurations responsible for these trends is outside the scope of this work.

$$p^L = P$$

However, the probability of a sequence of zeros in at least two of the four neutron signal values is in question. A binomial distribution is used to represent a combination of possible successful outcomes and is set equal to the acceptable probability of occurrence, one in one million. The successful outcomes include two of four neutron signal values measuring zero, three of four measuring zero, and four of four measuring zero.

$$\frac{n!}{k!\,(n-k)!}[P^k(1-P)^{n-k}] = x$$

where

k = number of successes (number of neutron signal values measuring zero),
n = number of events (number of neutron signal values on a neutron signal line),
P = probability of success for any single event (p^L),
x = probability of at least two of four neutron signal values measuring zero in a sequence of *L* trials.

$$6[P^2(1-P)^2] + 4[P^3(1-P)^1] + P^4 = 1 * 10^{-6}$$
$$P = 4 * 10^{-4} = p^L$$

This probability is then used to determine the required number of trails in the sequence.

$$p^L = P$$
$$0.45^L = 4 * 10^{-4}$$
$$L = 10$$

As a result, at least 10 neutron signal lines must be present in the database to deem the system unavailable by means of the neutron failure criterion. To make the failure condition more conservative, more neutron signal lines may be analyzed.

A.2 PROGRAMMING SOURCE CODE FOR ALGORITHM

```
Imports System.Data.SqlClient
Imports System.IO
Imports System.Reflection
Imports System.Resources

Public Class rptAnnualDsa

   REM Data Access variables
   Dim appConn As SqlConnection
   Dim sqlDA As SqlDataAdapter
   Dim command As SqlCommand
   Dim appConnStr, sqlStr As String

   REM Output File Stuff
   Dim outputFile As System.IO.File
   Dim dataArray(0) As String

   REM General Use variables
   Dim reportFrom, reportThru As Date
   Dim fileId, countryId, siteId, laneId As Integer
   Dim fileName, cname, sname, laneNum, reportFile As String
   Dim cntDays, cntExpected, cntReceived, cntEmpty, faults, cntFault, cntNS, cntGHF, cntGLF, cntNHF As Integer
   Dim totReceived, totEmpty, totFault, totNS, upHours As Integer
   Dim downTime, totDownTime, cntGood, totGood, utPct, totPct As Double
   Dim passedExclusion, exceedsLimit As Boolean
   Dim faultThreshold As Integer = 100
   Dim tb As String = "    "

   REM Config Variables hard-coded for this version
   Dim cfg_basePath As String = "c:\DART Files"
   Dim cfg_dataSource As String = "NNPRDDB-SLD"
   Dim cfg_srcCatalog As String = "RpmFy2012"

   Private Sub rptAnnualDsa_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
         MyBase.Load
      Call initialize()
   End Sub

   Private Function initialize() As Boolean

      Dim initialized As Boolean = False

      reportFrom = New Date(2011, 10, 1) 'fileDate as zeros for time, so this is sufficient
      reportThru = New Date(2012, 9, 30) 'fileDate as zeros for time, so this is sufficient
      cntExpected = DateDiff(DateInterval.Day, reportFrom, reportThru) + 1
      reportFile = cfg_basePath + "\Reports\Detection System Availability\DSA_" + Format(reportThru,
            "yyyy_MM_dd") + ".csv"
```

```
    REM Derive source connection info
    REM ==============================
    appConnStr = "Data Source=" + cfg_dataSource + ";Initial Catalog=" + cfg_srcCatalog + ";User
          ID=dartUser;Password=dartUserPassword;Connect Timeout=30;MultipleActiveResultSets=True"

    Try
      appConn = New SqlConnection(appConnStr)
      appConn.Open()
      initialized = True
    Catch ex As Exception
      MsgBox(tb + "rptDSA: unable to connect to source database: " + ex.Message)
      MsgBox(tb + "rptDSA: process aborted.")
    Finally
      appConn.Close()
    End Try

    REM Setup Heading Row
    REM ==================
    dataArray(0) = "Country,Site,Lane#,Expected,Received,Empty,Faults,Failed Neutron,downTime,Uptime (%)"

    Return initialized

End Function

Private Sub btnRun_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnRun.Click
    Call processDriver()
End Sub

Private Sub processDriver()

    If fileExists(reportFile) Then
      MsgBox(tb + "rptDSA: .csv file for this month already exists.")
    Else
      'MsgBox(tb + "rptDSA: generating .csv file...")
      Try
        sqlStr = "SELECT um.id, um.countryId, um.siteId, um.laneId, um.oversize, um.noData, um.badFormat,
              um.fileName, "
        sqlStr += "c.name, s.name, l.laneNumber FROM "
        sqlStr += "uploadMaster um "
        sqlStr += "JOIN RpmCurrent.dbo.country c ON c.id=um.countryId "
        sqlStr += "JOIN RpmCurrent.dbo.site s ON s.countryId=um.countryId AND s.id=um.siteId "
        sqlStr += "JOIN RpmCurrent.dbo.lane l ON l.countryId=um.countryId AND l.siteId=um.siteId AND
              l.id=um.laneId "
        sqlStr += "WHERE um.fileDate BETWEEN @fromDate AND @thruDate "
        sqlStr += "ORDER BY c.name, s.name, l.laneNumber, um.fileDate "
        command = New SqlCommand(sqlStr, appConn)
        command.Parameters.AddWithValue("@fromDate", reportFrom)
        command.Parameters.AddWithValue("@thruDate", reportThru)
        appConn.Open()
        Using dataReader As SqlDataReader = command.ExecuteReader
          If dataReader.HasRows Then
            Do While dataReader.Read
```

```
            REM Lane Change
            REM ===========
            If laneId = 0 Then
               countryId = dataReader.GetValue(1)
               siteId = dataReader.GetValue(2)
               laneId = dataReader.GetValue(3)
               cname = dataReader.GetValue(8)
               sname = dataReader.GetValue(9)
               laneNum = dataReader.GetValue(10)
            ElseIf laneId <> dataReader.GetValue(3) Then
               Call updateValues()
               countryId = dataReader.GetValue(1)
               siteId = dataReader.GetValue(2)
               laneId = dataReader.GetValue(3)
               cname = dataReader.GetValue(8)
               sname = dataReader.GetValue(9)
               laneNum = dataReader.GetValue(10)
            End If

            cntReceived += 1
            fileId = dataReader.GetValue(0)
            fileName = dataReader.GetValue(7)

            REM Main Processing Driver
            REM ======================
            If dataReader.GetValue(5) = 1 Then 'Empty
               cntEmpty += 1
            ElseIf passedNsExclusions() Then
               If exceedsNsLimit() Then
                  cntNS += 1
               Else
                  downTime += getDowntime()
                  cntFault += faults
               End If
            End If

         Loop
      End If
   End Using
Catch ex As Exception
   MsgBox(tb + ex.Message)
   MsgBox(tb + "rptDSA: process aborted.")
Finally
   appConn.Close()
End Try

REM Final Calcs and output
REM ======================
Call finalCalcs()

REM Output to file
REM ==============
Try
```

```
            outputFile.WriteAllLines(reportFile, dataArray)
            'MsgBox(tb + "rptDSA: .csv file created")
        Catch ex As Exception
            MsgBox(tb + ex.Message)
        End Try

    End If

End Sub

Private Sub updateValues()

    cntGood = cntReceived - (cntEmpty + cntNS + downTime)
    utPct = Math.Round((cntGood * 100) / cntReceived, 2)

    ReDim Preserve dataArray(dataArray.Length)
    dataArray(UBound(dataArray)) = cname + "," + sname + "," + laneNum + "," + cntExpected.ToString + "," + _
        cntReceived.ToString + "," + cntEmpty.ToString + "," + cntFault.ToString + "," + cntNS.ToString + "," + _
        Format(downTime, "#0.00") + "," + Format(utPct, "#0.00")

    totReceived += cntReceived : totEmpty += cntEmpty : totFault += cntFault : totNS += cntNS : totDownTime +=
            downTime
    cntReceived = 0 : cntEmpty = 0 : cntFault = 0 : cntNS = 0 : downTime = 0

End Sub

Private Sub finalCalcs()

    REM Update for last lane
    REM ====================
    Call updateValues()

    REM Total Calcs
    REM ===========
    totGood = totReceived - (totEmpty + totNS + totDownTime)
    totPct = Math.Round((totGood * 100) / totReceived, 2)

    REM Add Totals to Array
    REM ====================
    ReDim Preserve dataArray(dataArray.Length)
    dataArray(UBound(dataArray)) = "Totals,,,," + totReceived.ToString + "," + totEmpty.ToString + "," +
            totFault.ToString + "," + _
    totNS.ToString + "," + Format(totDownTime, "#0.00") + "," + Format(totPct, "#0.00")

End Sub

Private Function passedNsExclusions() As Boolean

    passedExclusion = True

    REM Mobile Detection Sensor is excluded
    REM ===================================
    If InStr(fileName, "MDS") > 0 Then
```

```
    passedExclusion = False
  End If

  REM If still in the running...
  REM ==========================
  If passedExclusion Then
    Try
      sqlStr = "SELECT r.monitorType FROM "
      sqlStr += "RpmCurrent.dbo.lane l "
      sqlStr += "JOIN RpmCurrent.dbo.rpm r ON r.id=l.modelId "
      sqlStr += "WHERE l.id=@laneId "
      command = New SqlCommand(sqlStr, appConn)
      command.Parameters.AddWithValue("@laneId", laneId)
      Using dataReader As SqlDataReader = command.ExecuteReader
        If dataReader.HasRows Then
          dataReader.Read()
          If dataReader.GetValue(0) = "Pedestrian" Then
            passedExclusion = False
          ElseIf dataReader.GetValue(0) = "Conveyor" Then
            passedExclusion = False
          End If
        End If
      End Using
    Catch ex As Exception
      MsgBox(tb + ex.Message)
      MsgBox(tb + "rptDSA: nsExclusionCheck aborted.")
    End Try
  End If
  Return passedExclusion

End Function

Private Function exceedsNsLimit() As Boolean

  exceedsLimit = False
  Try
    sqlStr = "SELECT sum(det1), sum(det2), sum(det3), sum(det4), count(*) FROM "
    sqlStr += "(SELECT det1, det2, det3, det4 FROM "
    sqlStr += "detectorData "
    sqlStr += "WHERE fileId=@fileId AND particle='n' AND occupFlag=1) work "
    command = New SqlCommand(sqlStr, appConn)
    command.Parameters.AddWithValue("@fileId", fileId)
    Using dataReader As SqlDataReader = command.ExecuteReader
      If dataReader.HasRows Then
        dataReader.Read()
        If dataReader.GetValue(4) >= 22 Then
          If (dataReader.GetValue(0) = 0 And dataReader.GetValue(1) = 0) Or _
            (dataReader.GetValue(0) = 0 And dataReader.GetValue(2) = 0) Or _
            (dataReader.GetValue(0) = 0 And dataReader.GetValue(3) = 0) Or _
            (dataReader.GetValue(1) = 0 And dataReader.GetValue(2) = 0) Or _
            (dataReader.GetValue(1) = 0 And dataReader.GetValue(3) = 0) Or _
            (dataReader.GetValue(2) = 0 And dataReader.GetValue(3) = 0) Then
             exceedsLimit = True
```

```
            End If
          End If
        End If
      End Using
    Catch ex As Exception
      MsgBox(tb + ex.Message)
      MsgBox(tb + "rptDSA: nsLimitCheck aborted.")
    End Try
    Return exceedsLimit

End Function

Private Function getDowntime() As Double

    Dim downDays As Double = 0
    Try
      sqlStr = "SELECT COALESCE(NULLIF(COUNT(HH),0),24), COALESCE(SUM(faultCount),0) FROM "
      sqlStr += "(SELECT LEFT(CONVERT(varchar(8),timestamp,108),2) HH, COALESCE(SUM(faultCount),0)
            faultCount FROM "
      sqlStr += "(SELECT g.timestamp timestamp, "
      sqlStr += "     CASE WHEN g.highFaultCount+g.lowFaultCount+n.highFaultCount > 120 THEN 120 ELSE "
      sqlStr += "        g.highFaultCount+g.lowFaultCount+n.highFaultCount END faultCount "
      sqlStr += "FROM detectorData g "
      sqlStr += "LEFT JOIN detectorData n ON g.fileId=n.fileId AND g.timestamp=n.timestamp AND n.particle='n' "
      sqlStr += "WHERE g.fileId=@fileId AND g.particle='g' AND g.occupFlag=0) work1 "
      sqlStr += "GROUP BY LEFT(CONVERT(varchar(8),timestamp,108),2)) work2 "
      command = New SqlCommand(sqlStr, appConn)
      command.Parameters.AddWithValue("@fileId", fileId)
      Using dataReader As SqlDataReader = command.ExecuteReader
        If dataReader.HasRows Then
          dataReader.Read()
          upHours = dataReader.GetValue(0)
          faults = dataReader.GetValue(1)
          downDays = (24 - ((upHours - (faults * 5) / 3600))) / 24
        Else
          downDays = 0
        End If
      End Using
    Catch ex As Exception
      MsgBox(tb + ex.Message)
      MsgBox(tb + "rptDSA: downTime calculation aborted for file: " + fileId.ToString)
    End Try
    Return downDays

End Function

Private Function fileExists(ByVal FileFullPath As String) As Boolean

    Dim f As IO.FileInfo
    Try
      f = New IO.FileInfo(FileFullPath)
      Return f.Exists()
    Catch ex As Exception
```

```
            MsgBox(FileFullPath + "-" + ex.Message)
            Return False
        End Try

    End Function

    Private Sub btnClose_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnClose.Click
        Me.Close()
    End Sub

End Class
```