**Posiflex Project Guide**

The Posiflex solution has been vastly condensed and optimized. It now consists of the following.

1. 6 Custom Tables
2. 1 PowerShell Script
3. 1 LabTech Script
4. 1 XML file
5. 1 XSD file

**Custom Tables**

1. Hpf\_cashregister
   1. Columns
      1. PosiflexDataID
      2. CR\_Number
      3. CR\_Config1
      4. CR\_DrawerFailedOpenCount
      5. CR\_ModelName
      6. CR\_PID
      7. CR\_RecordNumber
      8. EntryDate
   2. All of these values are pulled directly from the XML with the exception of the Entry Date, CR\_RecordNumber, and the PosiflexDataID.
      1. The CR\_Record Number is a running count of how many times this CR has been put in the DB
      2. The Entry Date will be the current datetime of the entry,
2. Hpf\_receiptprinter
   1. Columns
      1. PP\_RecordNumber
      2. PP\_PrinterInstallDate
      3. PP\_PrinterAge
      4. PP\_POSConfig
      5. PP\_Number
      6. PP\_ModelName
      7. PP\_MechanicalRevision
      8. PP\_FailedPaperCutCount
      9. PP\_CharacterPrintedCount
      10. PosiFlexDataID
      11. EntryDate
3. HPF\_PoleDisplay
   1. Columns
      1. PosiflexDataID
      2. PD\_Number
      3. PD\_CommunicationErrorCount
      4. PD\_ModelName
      5. PD\_SerialNumber
      6. PD\_RecordNumber
      7. EntryDate
4. Hpf\_Posiflex
   1. Columns
      1. PosiflexDataID
      2. ComputerID
      3. ClientID
      4. LocationID
      5. RecordNumber
      6. EntryDate
5. Hpf\_EpsonPrinter
   1. Columns
      1. PosiflexDataID
      2. EpsonPP
      3. EPP\_RecordNumber
      4. EntryDate
6. HPF\_StripeReader
   1. Columns
      1. PosiflexDataID
      2. MSR\_Number
      3. MSR\_HoursPoweredCount
      4. MSR\_FailedRead
      5. MSR\_ModelName
      6. MSR\_UnreadbaleCard
      7. MSR\_RecordNumber
      8. EntryDate

**Powershell Script**

Below is a rough outline of how the Powershell script functions.

1. Some Variable Declarations are made for variables used throughout the script.
2. We validate that both the XML and XSD file are found in the location they are supposed to be in.
3. We do the schema validation to make sure the XML file valid and conforms to the XSD.
4. If any of the first 3 steps fail, we exit the script and passed a failed message back to LabTech.
5. We make some further variable declarations.
6. We attempt to read the log date and log time from the XML file. If that fails we error out the script.
7. We loop through any and all Cashregisters that are in the XML and gather data assigning them to their respective variables.
8. We create a value portion of the SQL insert statement. Once all have been looped through we go to step 9.
9. We complete the insert statement in the $CR Variable.
10. We then do steps 7-9 again for each device type.
11. Once all device information has been gathered we check to see if any of the device variables are null.
12. If all are null we return an error.
13. We then right back our output to LabTech.

**Labtech Script**

1. We verify that the Powershell Script present on the device matches against the MD5 hash of the current file. If not we download the new one.
2. We run the Powershell and get a return value.
3. We check for the error values and if any exist we log an error and exit the script.
4. We split the return value we got form Powershell into many separate variables.
5. We do all of the required SQL inserts.
6. We check to make sure that after doing those we don’t have outdated data in the table.