Script Name: **Atstratus - HAYS - MapReduce.js**

**PROCESS FLOW:**

* HAYS Data comes in (HAYS Integration > HAYS Event)  
  Text

  Description automatically generated with medium confidence  
    
  NOTE: There is a formatted payload tab to better view the payload data  
  Text

  Description automatically generated  
  All the data would be the data that would be created for the records
* Map/Reduce Executes (Atstratus – HAYS – MapReduce)  
  Graphical user interface, application

  Description automatically generated  
  There are various deployment records available with the three highlighted for manual processing of data  
    
  Graphical user interface, text, application

  Description automatically generated  
  The script deployment parameters are more on ‘filters’ as to what data should be processed depending on the parameters.  
    
  Record Type (what type would be created?)  
  **customer  
  invoice  
  payment  
  creditmemo  
  customerrefund**  
    
  Status (HAYS Event Status)  
  **Pending  
  Success  
  Processed – With Errors  
  Ignored  
  Manually Created**  
    
  HAYS Event – Internal ID of the HAYS Event Custom Record to be processed  
  Saved Search – The Saved Search of the HAYS Event(s) to be processed depending on search criteria

**HAYS Events**

Graphical user interface, application, Word

Description automatically generated

Payload – JSON Format data from integration

Record Type – Type of NS record to be created

Error Code – NS-generated errors (**note:** will be helpful when customer requests what the problem is for failed imports from HAYS)

Subsidiary – Different subsidiaries have different requirements. Usually important

Status – Determines the status of the HAYS Record. Some Map/Reduce script deployments depend on the status of the HAYS Events for processing (ie. Processed with Errors, Failed, etc.)

**The Script**

    if(eventSubsidiary != SUBSIDIARY.DOCTORALIA\_INTERNET\_SL && eventSubsidiary != SUBSIDIARY.DOCPLANNER\_ITALY\_SRL\_CON\_SOCIO\_UNICO && eventSubsidiary != SUBSIDIARY.DOCTORALIA\_MEXICO\_SA\_DE\_CV && eventSubsidiary != SUBSIDIARY.DOCPLANNER\_TEKNOLOJI\_AAZ && eventSubsidiary != SUBSIDIARY.ZNANYLEKARZ\_SP\_Z\_O\_O && eventSubsidiary != SUBSIDIARY.DOCTORALIA\_BRASIL\_SERVIASOS\_ONLINE\_E\_SOFTWARE\_LTDA) return;

The line above is where you’ll edit stuff in the event that the customer wishes to go-live. Simply add the subsidiary here for additional subsidiary “go-lives”

The subsidiaries can be found here,

  var SUBSIDIARY = {

    DOCPLANNER\_HOLDINGS\_BV: 1,

    DOCPLANNER\_SUBHOLDINGS\_BV: 2,

    DOCPLANNER\_ITALY\_SRL\_CON\_SOCIO\_UNICO: 6,

    DOCPLANNER\_TEKNOLOJI\_AAZ: 11,

    DOCTORALIA\_BRASIL\_SERVIASOS\_ONLINE\_E\_SOFTWARE\_LTDA: 5,

    DOCTORALIA\_INTERNET\_SL: 10,

    DOCTORALIA\_MEXICO\_SA\_DE\_CV: 8,

    IANIRI\_INFORMATICA\_SRL: 13,

    TUOTEMPO\_SRL: 7,

    ZNANYLEKARZ\_SP\_Z\_O\_O: 9

  }

With regards to the flow of the script it depends. From current understanding,

**Customer** – Created as is, no other process follows

**Invoice** – Created as is, no other process follows

**Payment** – Usually accompanied with Invoice record

* Invoice created 1st
* Payment created 2nd

**Credit Memo** – Usually accompanied with Invoice and/or Payment record

* Invoice created 1st
* Payment Created 2nd (depends if there’s a match)
  + Applied to invoice
* Credit Memo is created 3rd
  + Depends its application to invoice

**Customer Refund** – Usually accompanied by Invoice, Payment, and Credit Memo

* Invoice created 1st
* Payment created 2nd (depends if there’s a match)
  + Applied to Invoice
* Credit Memo is created 3rd
  + Not applied to invoice
* Customer Refund
  + Unapply payment from invoice
  + Created customer refund
  + Apply payment to customer refund
  + Apply credit memo to Invoice