//------------------------------------------------------------------------------------------------------------------------------------

//Purpose: the JavaScript Override file allows the developer to enhance the CRE structure for run time

// execution features. The goal is to prevent needing to modify the the creCrud.js layer which

// is the primary engine

//------------------------------------------------------------------------------------------------------------------------------------

this.additionalFilters = function (profileline\_recordName)

{

creRecord = this;

var filters = [];

if (profileline\_recordName)

{

switch(profileline\_recordName) {

case "TRANSACTION":

/\*if (creRecord.RawData.inputGLAccount)

{

filters.push (new nlobjSearchFilter("internalid", null, "anyof", creRecord.RawData.inputGLAccount));

}\*/

if (creRecord.RawData.inclusionsList)

{

filters.push (new nlobjSearchFilter("internalid", null, "ANYOF", creRecord.RawData.inclusionsList));

}

break;

}

}

return filters;

};

if (creRecord.RawData)

{

creRecord.RawData["pad\_position"] = {

"left" : 1,

"right" : 2

};

}

//------------------------------------------------------------------------------------------------------------------------------------

//Example

//Function:

//Parameters:

//Description: Add Custom values to JSON for usage in Trimpath Template;

//------------------------------------------------------------------------------------------------------------------------------------

String.prototype.yymmdd = function (date\_to\_parse)

{

var x = null;

if (date\_to\_parse)

{

x = new Date(date\_to\_parse);

}

else

{

x = new Date();

}

var y = x.getFullYear().toString().substr(2);

var m = (x.getMonth() + 1).toString();

var d = x.getDate().toString();

var hours = x.getHours().toString();

var minutes = x.getMinutes().toString();

if (d.length === 1)

{

d = "0" + d;

}

if (m.length === 1)

{

m = "0" + m;

}

var \_yymmdd = y + m + d;

return \_yymmdd;

};

function areSameDate(d1, d2)

{

return d1.getFullYear() === d2.getFullYear()

&& d1.getMonth() === d2.getMonth()

&& d1.getDate() === d2.getDate();

}

//GetFileIDModifier is caluculated on a fly to provide most truthful information for every execution

//If first file transmission of current date, then will equal "1" .

//If additional files sent on same day, the increment value +1

String.prototype.GetNEXTFileIDModifier = function (lastFileControlDate, lastFileControlNumber)

{

var \_lastFileControlDate = null;

if (isNaN(parseInt(lastFileControlNumber,10)))

{

return 1;

}

else

{

\_lastFileControlDate = new Date(lastFileControlDate);

if (areSameDate(\_lastFileControlDate, new Date()))

{

//if they are the same, then increment LastFileControlNumber by one

return parseInt(lastFileControlNumber, 10) + 1;

}

else

{

//dates are different. Update date to today.

//set control number to 1

return 1;

//creRecord.RawData.NEWLastFileControlDate = "".mmddyyyyslash();

}

}

};

//Payment Bank. LAST TRACE NUMBER.

//This should have a starting number.

//Will start with 1000, if blank.

//This number will be unique to the transaction and will help identify the transaction in case of an inquiry.

//next trace number is calculated here to give Tester highest integrity of the output file

String.prototype.GetNextTraceNumber = function (lastTraceNumber)

{

var retValue = 0;

if (isNaN(parseInt(lastTraceNumber,10)))

{

retValue = 1000;

}

else

{

retValue = parseInt(lastTraceNumber,10)+1;

}

return retValue;

};

//optimally, this function would return next number back to

//just after Execute via object reference. This appears not to be the case

String.prototype.GetNextBatchNumber = function (lastBatchNumber)

{

var retvalue = 0;

if (isNaN(parseInt(lastBatchNumber,10)))

{

retvalue = 1;

}

else

{

retvalue = parseInt(lastBatchNumber,10)+1;

}

return retvalue;

};

function sum( obj )

{

var \_sum = 0;

var el = "";

for(el in obj )

{

if( obj.hasOwnProperty( el ) )

{

if (el !== "column3and4")

{

\_sum += parseInt( obj[el],10);

}

}

}

return \_sum;

}

String.prototype.getBlockCount = function (headercount, batchcount, ppdcount, companycount, filecount)

{

if (isNaN(parseInt(headercount,10)))

{

headercount = 1;

}

if (isNaN(parseInt(batchcount,10)))

{

batchcount = 1;

}

if (isNaN(parseInt(ppdcount,10)))

{

ppdcount = 1;

}

if (isNaN(parseInt(companycount,10)))

{

companycount = 1;

}

if (isNaN(parseInt(filecount,10)))

{

filecount = 1;

}

var retvalue = headercount\*sum(creRecord.RawData["header\_lengths"])

+ batchcount\*sum(creRecord.RawData["batch\_header\_lengths"])

+ ppdcount\*sum(creRecord.RawData["ppd\_entry\_lengths"])

+ companycount\*sum(creRecord.RawData["company\_batch\_lengths"])

+ filecount\*sum(creRecord.RawData["file\_control\_lengths"]);

return retvalue;

};

String.prototype.mmddyy = function (date\_to\_parse)

{

var x = null;

if (date\_to\_parse)

{

x = new Date(date\_to\_parse);

}

else

{

x = new Date();

}

var y = x.getFullYear().toString().substr(2);

var m = (x.getMonth() + 1).toString();

var d = x.getDate().toString();

var hours = x.getHours().toString();

var minutes = x.getMinutes().toString();

if (d.length === 1)

{

d = "0" + d;

}

if (m.length === 1)

{

m = "0" + m;

}

var \_mmddyy = m + d + y;

return \_mmddyy;

};

String.prototype.mmddyyyyslash = function (date\_to\_parse)

{

var x = null;

if (date\_to\_parse)

{

x = new Date(date\_to\_parse);

}

else

{

x = new Date();

}

var y = x.getFullYear().toString();

var m = (x.getMonth() + 1).toString();

var d = x.getDate().toString();

var hours = x.getHours().toString();

var minutes = x.getMinutes().toString();

if (d.length === 1)

{

d = "0" + d;

}

if (m.length === 1)

{

m = "0" + m;

}

var \_mmddyy = m + "/"+ d + "/" + y;

return \_mmddyy;

};

String.prototype.hhmm = function ()

{

var x = new Date();

var hours = x.getHours().toString();

var minutes = x.getMinutes().toString();

if (hours.length === 1)

{

hours = "0" + hours;

}

if (minutes.length === 1)

{

minutes = "0" + minutes;

}

var \_hhmm = hours + minutes;

return \_hhmm;

};

//produces empty string with length

//var debugmode = true;

//"".createbuffer(10) --> "##########"

//

String.prototype.createbuffer = function (len, character)

{

var retvalue = "";

var i = 0;

if (!character)

{

character = " "; //default spacer is single space

}

if (!len)

{

return;

}

for (i = 0; i < len; i+=1)

{

retvalue += character;

}

return retvalue;

};

//pads left or pads right

//"".formatted\_string("0000",123,pad\_position.left); --> 0123

//"".formatted\_string("00000000",123,pad\_position.right); --> 12300000

String.prototype.formatted\_string = function(pad, user\_str, pad\_pos)

{

//we may need to enhance this function to automatically pad left or right based on alphanumeric string

var padleft = 1;

var padright = 2;

if (typeof user\_str === "undefined")

{

return pad;

}

if (pad\_pos === padleft)

{

return (pad + user\_str).slice(-pad.length);

}

if (pad\_pos === padright)

{

return (user\_str + pad).substring(0, pad.length);

}

};

String.prototype.handle\_bankname\_hierachy = function(name)

{

var retvalue = "";

if (!name)

{

return "";

}

if (name.indexOf(":")<0)

{

return name;

}

return name.substring(name.lastIndexOf(":")+1).trim();

};

if (creRecord.RawData)

{

creRecord.RawData["header\_lengths"] = {

"column1" : 1,

"column2" : 2,

"column3" : 10,

"column4" : 10,

"column5" : 6,

"column6" : 4,

"column7" : 1,

"column8" : 3,

"column9" : 2,

"column10" : 1,

"column11" : 23,

"column12" : 23,

"column13" : 8

};

creRecord.RawData["batch\_header\_lengths"] = {

"column1" : 1,

"column2" : 3,

"column3" : 16,

"column4" : 20,

"column5" : 10,

"column6" : 3,

"column7" : 10,

"column8" : 6,

"column9" : 6,

"column10" : 3,

"column11" : 1,

"column12" : 8,

"column13" : 7

};

creRecord.RawData["ppd\_entry\_lengths"] = {

"column1" : 1,

"column2" : 2,

"column3" : 8,

"column4" : 1,

"column3and4" : 9,

"column5" : 17,

"column6" : 10,

"column7" : 15,

"column8" : 22,

"column9" : 2,

"column10" : 1,

"column11" : 15

};

creRecord.RawData["company\_batch\_lengths"] = {

"column1" : 1,

"column2" : 3,

"column3" : 6,

"column4" : 10,

"column5" : 12,

"column6" : 12,

"column7" : 10,

"column8" : 19,

"column9" : 6,

"column10" : 8,

"column11" : 7

};

creRecord.RawData["file\_control\_lengths"] = {

"column1" : 1,

"column2" : 6,

"column3" : 6,

"column4" : 8,

"column5" : 10,

"column6" : 12,

"column7" : 12,

"column8" : 39

};

creRecord.RawData["NewLastBatchNumber"] = "";

creRecord.RawData["NewLastTraceNumber"] = "";

creRecord.RawData["NACHA\_HEADER"] = {};

creRecord.RawData.YYMMDD = "".yymmdd();

creRecord.RawData.MMDDYY = "".mmddyy();

creRecord.RawData.HHMM = "".hhmm();

String.prototype.updateFileControlInformation = function (paymentbankid, nextControlDate, nextFileControlNumber)

{

var retvalue = "-1";

if (paymentbankid && nextControlDate && nextFileControlNumber)

{

var paymentbank = nlapiLoadRecord("customrecord\_payment\_bank", paymentbankid);

paymentbank.setFieldValue('custrecord\_pb\_last\_file\_cd', nextControlDate);

paymentbank.setFieldValue('custrecord\_pb\_last\_file\_cn', nextFileControlNumber);

retvalue = nlapiSubmitRecord(paymentbank, false, true);

}

return retvalue;

};

}

if (creRecord.RawData && (typeof creRecord.RawData.UpdateControlTable === "undefined"))

{

creRecord.RawData.UpdateControlTable = false;

}

if (creRecord.RawData && (typeof creRecord.RawData.data\_in === "undefined"))

{

creRecord.RawData.data\_in = {};

creRecord.RawData.data\_in["constants"] = {};

creRecord.RawData.data\_in["constants"]["PFT Account Options"] = {};

creRecord.RawData.data\_in["constants"]["PFT Account Options"]["Uses Omnibus Account"] = 1;

creRecord.RawData.data\_in["constants"]["PFT Account Options"]["GL Account Selection Required"] = 2;

}

/\*

if (creRecord.RawData)

{

//Answer Id: 10248

var conf = nlapiLoadConfiguration("companyinformation");

var company\_addressfields = [

"attention","addressee","phone","address1","address2","address3","city","state","city","country",

"returnattention","returnaddressee","returnphone","returnaddress1","returnaddress2","returnaddress3","returncity","returnstate","returncity","returncountry",

"shippingattention","shippingaddressee","shippingphone","shippingaddress1","shippingaddress2","shippingaddress3","shippingcity","shippingstate","shippingcity","shippingcountry"

]

for (field in company\_addressfields)

{

creRecord.RawData[company\_addressfields[field]] = conf.getFieldValue(company\_addressfields[field]);

}

}

\*/