Documentation File for IBSGATEWAY840.dll

Updated : 23/11/2021

A new naming convention for the Infusion Gateway has been introduced. It is IBSGATEWAYnnn.dll, where the nnn is the version number. (Formally IBSGW.DLL)

This version number DOES NOT relate to the Infusion version number.

IBSGATEWAY815 and higher are currently able to work with Infusion version 8.100 and higher.

Within the IBSGATEWAYnnn.dll there is a class name “Gateway”.

Add the DLL (IBSGATEWAY8nn.DLL) to your applications references and then create an instance of the class contained within (IBSGATEWAY8nn.Gateway) as an object.

Example VFP :

```
oGateway=CREATEOBJECT("ibsgateway835.gateway")
nSuccess=oGateway.setpath("C:\Infusion\--DEMO--")
nSuccess=oGateway.login('My Application Name')

if nSuccess <> 1    (check properties)
        oGateway.LastError
        oGateway.LastErrorMessage
        ...


        Eg
        oGateway.LastError = 900
        oGateway.LastErrorMessage =
              'The system is in use, Possible Backup, Reindex or other in progess.
        Contact Administrator.'

        Get some other properties to inform User

        oGateway.LastError = 901
        oGateway.LastErrorMessage =
              'There is a Backup being done Log back on after ' +
        dtoc(oGateway.AllowOn)

        Properties set when login() and Other Methods called.

        oGateway.AllowOn      Date Time
        oGateway.LogOffBy     Date Time
        oGateway.AdminMessage Character
        oGateway.SentBy         Character
        oGateway.ForceOff     Character '.T.' or '.F.'
```

Alternatively, you can programme to interrogate the Windows Registry for the latest version of the Gateway file. An example of how to do this in .Net is:

```
public static class GatewayHelper
    {

        /// <summary>
        /// Gets the registered gateways from the registry.
        /// </summary>
        /// <returns>gateways in the format of [gatewayname][version].Gateway ie ibsg
ateway712.Gate            way</returns>
        private static List<String> GetRegisteredGateways()
        {
            var keys = Registry.ClassesRoot.GetSubKeyNames().Where(x => x.StartsWith(
"ibsgateway",             StringComparison.InvariantCultureIgnoreCase));

            return keys.ToList();
        }

        public static string GetLatestRegisteredGateway()
        {
            string result = string.Empty;

             List<String> registeredGateways = GetRegisteredGateways();

            // Create the dictionary for holding the registered gateway versions
            Dictionary<int, string> versions = new Dictionary<int, string>();

            // Loop each of the registered entries
            foreach (string registeredGateway in registeredGateways)
            {
                string[] gwArr = registeredGateway.Split('.');

                if (gwArr.Count() > 0)
                {
                    string[] versionNumArr = gwArr[0].ToUpper().Split(new string[] {
"IBSGATEWAY" }                  , StringSplitOptions.RemoveEmptyEntries);

                    if (versionNumArr.Count() > 0)
                    {
                        int versionNum = 0;
                        if (Int32.TryParse(versionNumArr[0], out versionNum))
                        {
                            if (!versions.ContainsKey(versionNum))
                            {
                                versions.Add(versionNum, registeredGateway);
                            }
                        }
```

```
                }
            }
        }

        if (versions.Count > 0)
        {
           result = versions.OrderByDescending(x => x.Key).FirstOrDefault().Value
  ;
        }

         return result;
    }
}

// Attempt to get the latest gateway version registered on the server
string latestGateway = GatewayHelper.GetLatestRegisteredGateway();
if (string.IsNullOrWhiteSpace(latestGateway))
{
    throw new InfusionException("A registered gateway could not be found");
}

// Create the gateway object
Type gatewayType = Type.GetTypeFromProgID(latestGateway);
dynamic ibsGw = Activator.CreateInstance(gatewayType);

// Set the Infusion data path
ibsGw.setpath(appContext.FixedAsset_JournalExportPath);
```

Along with this new naming convention we have introduced the need to 'Logon' to Infusion before calling any of the DLLs functions.

LOGIN – Connects the DLL to Infusion

This function is required to be called at the start of your application (or as required by your application). It must be called before any other functions will operate. Once called it is not necessary to call it again, until you have either called LOGOUT, DESTROY or removed the reference to the object which created the Login Call in your application.

login("My Application Name")

The parameter "My Application Name" is a string which informs Infusion the name of the application creating the connection. This is used by Infusion to display information about 'Who's Logged On'. While this string can be any value, it is expected that you will provide a useful value to the client.

LOGOUT – Disconnects the DLL from Infusion

Once you have finished interacting with Infusion use this call to disconnect the DLL from Infusion. There are no parameters required with this function. If you neglect to call this function before destroying the object within your application, the DLL will automatically logout.

logout()

Return Values

All Procedures and Functions have been updated to return standard values.

- Where the request is unsuccessful due to an error the DLL will return a value of 0 (Zero)
- Where the DLL is NOT logged in the DLL will return a value of -1 (negative 1)
- Where the request was successful and no other value needs to be returned, 1 will be returned
- Where the request was successful and a return value is expected, that value will be returned.

Example:

oGateway.Get("name") could return two types of values, 0 (failure) or "Some String" the required result value
oGateway.Append() could return either a 1 (success) or 0 (failure)

Where your application is expecting a 0 to be returned as a valid response, you will need to ensure the login is performed or the 0 being returned may be misinterpreted.

When 0 or -1 are returned you should always check the LastError and LastErrorMessage properties for the reason.

The following procedures and properties can be called / set from the base class.

Standard DBF Read / Write Functions

APPEND – Adds a blank record to the currently selected DBF table

append()

CLOSE - Closes a DBF table

close("AliasName")

DELETERECORD – Deletes the currently selected record in a DBF table

deleterecord("AliasName")

GET - Returns the value of a field in a DBF table

get("FieldName") - Note the field must be in the currently selected DBF table
get("AliasName.FieldName")

GetAuditFromCustInvNumber – Returns the Audit number of a customer Invoice

Returns the Audit Number for success and 0 for failure. Note: If the invoice does not exist or is Pending, then 0 will be returned.

GetAuditFromCustInvNumber(“12345”)

GOBOTT - Moves the record pointer to the bottom of the current DBF table.

gobott()

GOTOP - Moves the record pointer to the top of the current DBF table.

gotop()

LOCATE – Positions the record pointer at the first record which matches the supplied criteria.

Returns 1 for success and 0 for failure. Note: When locating for a character value enclose it in a set of single quotes or [].

locate(“account=123”)
locate(“grp=’ABC’”)
locate(“grp=[ABC]”)

OPEN - Opens a DBF table
Note: This also selects the DBF table as the current work area

open("DBFName","AliasName")

RECCOUNT - Returns the number of records in the current DBF table

reccount()

RECLOCK – Attemps to lock the current record in the currently selected DBF table

Returns 1 if the record lock was successful or 0 if it failed

reclock()
RECUNLOCK – Unlocks the current record in the currently selected DBF table

recunlock()

REPLACE – Writes a single value to a field in the currently selected DBF table
REPLACESTR, REPLACENUM, REPLACEDATE can also be used for specific values.
Note: You may optionally specify a DBF name preceding the field name.

Returns 1 if the replace command succeeded or 0 if it failed.

replace(“myStringField”,”my stringvalue”)
replace(“myNumberField”,123.45)
replace(“myLogicalField”,.t.)
replace(“myFile.mField”,”my stringvalue”)
replace(“myField”,”myOtherField”)

SELECT - Selects an open DBF table as the current work area
Note: This function also returns the work area number

select("AliasName")

SETFILTER - Sets a filter on the currently selected DBF table

setfilter("account=123")
setfilter("account>10 and account < 100")

SETORDER - Sets the controlling Index of the currently selected DBF table

setorder("account”)

SETPATH - Sets the path to the Infusion Data folder.
Note: This must be set or the files are expected to be in the same directory as your application is being run from. You can use either Mapped paths or UNC path names. It is optional to have a final \ in the path name.

Alternatively you can set this value in the property called DataPath (Note: It is not cleared by the VarReset function.) but be sure to add the \ on the end or your application will fail.

setpath("\\server\share name\directory\data")
setpath("\\server\share name\directory\data\")
setpath("c:\my application\data")
setpath("c:\my application\data\")
setpath("..\infusion\data") - the infusion directory is located in the parent folder of the current application folder.
setpath("..\infusion\data\")

SETPDFPATH - Sets the path to the folder containing the additional files required for the EmailCustomerInvoice and EmailSupplierInvoice functions. This function is in the main Gateway and PDFGateway DLLs

Note: This must be set or the files are expected to be in the same directory as your application is being run from. You can use either Mapped paths or UNC path names. It is optional to have a final \ in the path name. Call this function AFTER you have called SETPATH. This path value is appended to any other values set by SETPATH and SETROOTPATH

SETROOTPATH - Sets the root path to Infusion. This is where the IBSMAIN.EXE file is located. This function is in the main Gateway and PDFGateway DLLs. It is used to Identify the main Infusion folder.

Note: This must be set or the files are expected to be in the same directory as your application is being run from. You can use either Mapped paths or UNC path names. It is optional to have a final \ in the path name. Call this function AFTER you have called SETPATH. This path value is appended to any other values set by SETPATH and SETPDFPATH

SKIP - Moves to the next record in the currently selected DBF table.
Note: This function returns 1 if the skip was successful or 0 if it failed (EOF encountered)

skip(“AliasName”)

Next System Generated Number Functions

Passing a parameter of 0 will return the next number and then increment it. Passing a parameter of 1 will only enquire the next number, it will not be incremented.

NEXTAUDIT - Returns the next System Audit Number.
nextaudit(0)
nextaudit(1)

NEXTCUST - Returns the next Customer Account number.
Note: If automatic account numbering is turned off then 0 is returned, otherwise the account number is returned.

nextcust(0)
nextcust(1)

NEXTINV - Returns the next Customer Invoice number.

nextinv(0)
nextinv(1)

NEXTJOB - Returns the next Job Management Job ID.
Note: If automatic job numbering is turned off then 0 is returned, otherwise the job id number is returned.

nextjob(0)
nextjob(1)

NEXTLINK - Returns the next System Link Number.

nextlink(0)
nextlink(1)

NEXTPO - Returns the next Supplier Purchase Order number.

nextpo(0)
nextpo(1)

NEXTQUOTE - Returns the next Customer Quote number.

nextquote(0)
nextquote(1)

NEXTSUPP - Returns the next Supplier Account number.
Note: If automatic account numbering is turned off then 0 is returned, otherwise the account number is returned.

nextsupp(0)
nextsupp(1)

Customer / Supplier / Product Creation Functions

CREATECUST – Creates a new Customer account

The only parameter to be passed is the account number to be created. Once the account is created, use the replace commands to fill in other fields.

After all the other fields have been populated you need to call the Cust_UpdateSearchPlus function.

You can use the NEXTCUST function to obtain the next account number.

createcust(10)

CUST_UPDATESEARCHPLUS – Updates the Searchplus field of Customer.DBF with its calculated value

The only parameter to be passed is the account number to be updated. Call this function after updating any fields in the Customer.dbf

Cust_UpdateSearchPlus(10)

CREATESUPP – Creates a new Supplier account

The only parameter to be passed is the account number to be created. Once the account is created, use the replace commands to fill in other fields.

You can use the NEXTSUPP function to obtain the next account number.

createsupp(10)

CREATEPRODUCT(cProductCode, cProductDescription) – Creates a new Product Code

Two parameters are passed to this function. The Product Code to create and its Description.

The Product Code in Infusion is uppercase. The function automatically converts the supplied value to uppercase before the code is created.

The function populates the same defaults which are set when a product is created using the Infusion Interface:

GST Rates
Pricing Method
Allow Discounts
Default GL IDs
Use Discount Matrix
Customer Item Type
Searchplus Field
Product Creation Date

createProduct("PCODE123","Product Description")

The function returns 1 on success and 0 on failure. Check the lastErrorMessage value for the reason of

a failure.

All other fields of a product should be populated using the Replace command after placing a record lock on the newly created product shell.

ADDPRODUCTIMAGE – Adds an Image to a Product

This function will only work with version 8.200 or Higher.

Ensure you have copied the Image to be added to the Product Images Folder (Set in System Defaults) before calling this function. If the image does not exist then the function will return 0 (Failure).

A combination of the Product Code and the Image Filename is used to identify the image in the database.

Four parameters are passed to the function:

- Product Code
- Image Display Description
- Image Filename
- Is this Image the Default Image (Boolean)

If the Image is already associated with the Product the function will return 0
If the Image does not exist in the Product Images folder the function will return 0

Once the Default Product Image flag has been applied as supplied, the function checks there is still a Default Image associated with the product. If no image is set as default then the first image added to the product will be set as the Default Image.

addProductImage("CODE123","My Product Image","ImageName.jpg",.t.)
addProductImage("CODE123","My Product Image","ImageName.jpg",.f.)

UPDATEPRODUCTIMAGE – Updates the details of an existing Image associated with a Product

This function will only work with version 8.200 or Higher.

Ensure you have copied the Image to be added to the Product Images Folder (Set in System Defaults) before calling this function. If the image does not exist then the function will return 0 (Failure).

A combination of the Product Code and the Image Filename is used to identify the image in the database.

The function will set the Description and Default Status of the Image.

Four parameters are passed to the function:

- Product Code
- Image Display Description
- Image Filename
- Is this Image the Default Image (Boolean)

If the Image is NOT already associated with the Product the function will return 0
If the Image does not exist in the Product Images folder the function will return 0

Once the Default Product Image flag has been applied as supplied, the function checks there is still a Default Image associated with the product. If no image is set as default then the first image added to the product will be set as the Default Image.

updateProductImage("CODE123","My Product Image","ImageName.jpg",.t.)
updateProductImage("CODE123","My Product Image","ImageName.jpg",.f.)

DELETEPRODUCTIMAGE – Deletes an existing Image associated with a Product

This function will only work with version 8.200 or Higher.

This function only deletes the association from the Product. It DOES NOT physically delete the Image from the computer.

A combination of the Product Code and Image File Name are used to locate the record in the database.

Two parameters are passed to the function:

- Product Code
- Image Filename

If the Image is NOT already associated with the Product the function will return 0

Once the Image has been removed, the default Image is checked to ensure that there is still one set. (assuming there is still an Image associated). The first image in the list will be used as the Default Image if the Default Image has been deleted.

deleteProductImage(“CODE123”,”ImageName.jpg”)

CREATECONTACT – Creates a Contact for a Customer or Supplier

A single parameter is passed to the function to indicate if a Customer or Supplier Contact is to be created. Pass “S” for a Supplier or “C” for a Customer. Any value other than “S” will result in a Customer Contact being created.

The parameters for the Contact are detailed in the Parameters Explained section of this document.

createContact(“C”)
createContact(“S”)

UPDATECONTACT – Updates a field of a Contact for a Customer or Supplier

This function allows you to update a given field of a Contact.

Set the value in the pXxxxxx parameter of the Gateway Object and then call this function with the following three parameters:

- Link Number of the Contact
- Type of Contact “C” – Customer or “S” – Supplier
- Field to update

A list of the fields and their parameters is detailed in the Parameters Explained section of this document.

updateContact(123,”C”,”NAME”)
updateContact(456,”S”,”EMAIL”)

DELETECONTACT – Deletes a Contact for a Customer or Supplier

This function will delete a Contact from a Customer or a Supplier. If the contact has been setup as an Email Link, then this link will also be deleted.

Two parameters are passed to this function:

The Link Number of the Contact
The Type of Contact – “C” Customer, or “S” Supplier

deleteContact(123,”C”)
deleteContact(456,”S”)

CREATEJOB(nCustomer, cJobID, cOpenedBy, cJobType) – Creates a new Product Code

Four parameters are passed to this function.

Customer Account the Job is for
Job ID to Create (use NextJob to get the value from INFDATA. Remember to convert the Int to a String)
Staff ID of the person creating the Job
Job Type (*STD* is the Standard Job) - If this is not supplied, *STD* is used.

The function populates the following values:

Price Level
Sort Key of the Customer
Customer Account
Job ID
Name
Address Details
Site Address Details
Phone Numbers
Job Type
Opened By

createJob(1000, nNextJob.ToString(), "STAFF", "*STD*")

The function returns a positive number on success and 0 on failure. Check the lastErrorMessage value for the reason of a failure.

The number returned is the Job Link Number.

All other fields of a product should be populated using the Replace command after placing a record lock on the newly created job shell.

OPENCUSTRATECALCS (lOpen)

This function is used to open (or close) the files required for the Customer Rate Calculation Function.

The parameter passed as True will OPEN the tables. When passed as False will CLOSE the tables.:

openCustRateCalcs(.t.)
openCustRateCalcs(.f.)

GETCUSTRATE

Calculates the price of a Product based on the Customer, Qty and Date
Uses the Infusion Pricing Logic
Currently only available to Infusion Developers.

Requires the parameters pAccount, pPCode, pDate and pQty to be set

Result values set to pRate and pDiscount

Return value of 1 indicates success
Return value of 0 indicates an error occurred.

Transaction Functions

CLOSEGLJNLFILES
Closes the files required for posting Financial Journals

closegljnlfiles()

OPENGLJNLFILES
Opens the required files for posting Financial Journals

opengljnlfiles()

POSTGLTXN - Creates a Financial Journal.
A full detail of the required parameters for this function can be found in the Transaction Requirements section of this document.

Note: You must call OPENGLJNLFILES before posting transactions, this only need be done once, and the files will remain open until you issue the CLOSEGLJNLFILES command.

postgltxn()

POSTJOBTXN - Creates a Job Transactions based on the properties set.
A full detail of the required parameters for this function can be found in the Transaction Requirements section of this document.

postjobtxn()

CREATEINVOICE – Creates a new Invoice Header Record.

Note: A return value greater than zero indicates success. This number is the LINK number of the invoice. Use this value as the pLink parameter when calling INVOICEADDLINE or TOTALINVOICE. A value of zero indicates an error. The meaning of the error code is returned in the LastErrorMessage property.

A full detail of the required parameters for this function can be found in the Transaction Requirements section of this document.

After you have created an invoice header you can add lines to it using INVOICEADDLINE. Also remember to re total the invoice using the TOTALINVOICE function.

createinvoice()

INVOICEADDLINE – Adds a line to the specified invoice.

A full detail of the required parameters for this function can be found in the Transaction Requirements section of this document.

You must call the TOTALINVOICE function after you have added all the lines you want to a given invoice.

invoiceaddline()

INVOICEADDLINE2 – Adds a line to the specified invoice and checks available stock quantity.

A full detail of the required parameters for this function can be found in the Transaction Requirements section of this document. (See INVOICEADDLINE)

InvoiceAddLine2 ***DOES NOT*** *support adding lines for GL ID's (Use InvoiceAddLine instead)*

Supply the total required quantity in the pQTY parameter.
The function will check the available stock holding and automatically populate the pBACK parameter.

NOTE: This function only populates the Pending Invoices column for Backorders. It DOES NOT process the pending invoice to create the actual backorder.

You must call the TOTALINVOICE function after you have added all the lines you want to a given invoice.

Invoiceaddline2()

INVOICEDELETELINE – Deletes a line to the specified invoice.

This method deletes a line from a PENDING Customer Invoice.

You must open the INVLN table and locate to the record you wish to delete. Use the OPEN and LOCATE commands to do this. Afterwards you are also responsible for closing the INVLN table. Use the CLOSE command for this.

Pass the Alias of the INVLN table you have opened to the function.

This function deletes the line from INVLN and also updates the Allocated quantity of PRODBALS and PRODUCTS when required.

invoicedeleteline('a_invln')

TOTALINVOICE – Retotals the specified Invoice

A full detail of the required parameters for this function can be found in the Transaction Requirements section of this document.

This function should be called after all lines are added to an invoice.

totalinvoice()

UPDATEINVHEADER – Updates a given field of an existing pending invoice.

There is a list of the valid values for the parameter and the matching variables which should be set for each parameter in the Transaction Requirements section of this document.

Updates can only be performed on PENDING invoices.

updateinvheader("NAME")
updateinvheader("INVNUMBER")

CUSTOMERINVOICE_POST – Posts a pending customer invoice.

If the invoice contains the any of the following, it cannot be posted using the gateway.

- Trade In
- Back Orders
- Deferred Payment
- Recurring Invoice
- Recurring Job
- Deposit Paid

Pass the link number of the invoice to be posted as a parameter to the function.

The function returns a value of 1 if the invoice was successfully posted. It returns 0 if the invoice was not posted.

The lastError and lastErrorMessage parameters contain details of why an invoice was not posted.

- 10100 – Invoice Link Not Supplied
- 10101 – Invoice Link Not Found
- 10102 – Invoice Already Posted
- 10103 – Invoice is flagged as on hold
- 10104 – The invoice date is empty / earlier than your lockout or system start date.
- 10105 – The invoice date is after your lockout date.
- 10106 – Customer requires an order number on all invoices.
- 10107 – Customer Account not found.
- 10108 – Sales Rep Required.
- 10109 – Customer is on Stop Credit.
- 10110 – Customer is Inactive.
- 10111 – Customer is over their Credit Limit.
- 10112 – Invoice contains a deferred payment.
- 10113 – Invoice is flagged as recurring.
- 10114 – Invoice is linked to a recurring Job.
- 10115 – Invoice is linked to a Job with a Deposit.
- 10116 – Invoice is in use by another user.
- 10117 – Customer is being accessed by another user.
- 10118 – Not all serial numbers have been entered.
- 10119 – Invoice has GL code(s) with a zero rate.
- 10120 – Invoice contains a Trade In.
- 10121 – Invoice contains Back Ordered items.
- 10999 – An unknown invoice posting error occurred.

customerInvoice_post(123456)

CREATECUSTINVOICEPDF – Creates a PDF (Original) of a Customer Invoice with a specified filename and location.

This method uses the XFRX functions to create a PDF of an Invoice (Pending or Posted).
The setPDFPath and setRootPath methods of the Gateway should be called prior to using this method.
Infusion version 8.410 or higher is required.

The output filename (including path), Invoice Layout filename and Invoice Link Number are passed as parameters.

createCustInvoicePDF(nInvLink,cInvLayoutFile,cOutputFolderAndFilename)

createCustInvoicePDF(45678,'INV01','c:\output\Invoice1234.pdf')

CREATESUPPINVOICEPDF – Creates a PDF (Original) of a Supplier Invoice with a specified filename and location.

This method uses the XFRX functions to create a PDF of an Invoice (Pending or Posted).
The setPDFPath and setRootPath methods of the Gateway should be called prior to using this method.
Infusion version 8.410 or higher is required.

The Invoice can be for a Supplier Invoice (without products) or a Purchase Order Invoice.
The layout file used in the method call should be able to output the type of Invoice being requested.

The output filename (including path), Invoice Layout filename and Invoice Link Number are passed as parameters.

createSuppInvoicePDF(nInvLink,cInvLayoutFile,cOutputFolderAndFilename)

createSuppInvoicePDF(45678,'SUPPINV','c:\output\Invoice1234.pdf')

CUSTOMERBACKORDER_CREATE – Creates a Backorder header

This function along with CustomerBackOrder_AddLine are used to add a Customer Backorder as if it had been created by posting a Customer Invoice with quantities in the Backorder column.

A return value greater than zero indicates success. This number is the LINK number of the backorder. Use this value as the pLink parameter when calling CUSTOMERBACKORDER_ADDLINE. A value of zero indicates an error. The meaning of the error code is returned in the LastErrorMessage property.

A full detail of the required parameters for this function can be found in the Transaction Requirements section of this document.

*** Note There is no totaling function requited for Backorders, as there is in Invoices ***

customerBackorder_Create()

CUSTOMERBACKORDER_ADDLINE – Adds a line to a Customer Backorder

This function along with CustomerBackOrder_Create are used to add a Customer Backorder as if it had been created by posting a Customer Invoice with quantities in the Backorder column.

A return value of 1 indicates success. Any other value indicates an error. The meaning of the error code is returned in the LastErrorMessage property.

A full detail of the required parameters for this function can be found in the Transaction Requirements section of this document.

customerBackorder_AddLine()

CREATEQUOTE – Creates a new Customer Quote Header Record.

Note: A return value greater than zero indicates success. This number is the LINK number of the quote. Use this value as the pLink parameter when calling QUOTEADDLINE or TOTALQUOTE. A value of zero indicates an error. The meaning of the error code is returned in the LastErrorMessage property.

A full detail of the required parameters for this function can be found in the Transaction Requirements section of this document.

After you have created a quote header you can add lines to it using QUOTEADDLINE. Also remember to re total the quote using the TOTALQUOTE function.

createquote()

QUOTEADDLINE – Adds a line to the specified quote.

A full detail of the required parameters for this function can be found in the Transaction Requirements section of this document.

You must call the TOTALQUOTE function after you have added all the lines you want to a given quote.

quoteaddline()

QUOTEDELETELINE – Deletes a line to the specified quote.

This method deletes a line from a PENDING Customer Quote.

You must open the QUOTELN table and locate to the record you wish to delete. Use the OPEN and LOCATE commands to do this. Afterwards you are also responsible for closing the QUOTELN table. Use the CLOSE command for this.

Pass the Alias of the QUOTELN table you have opened to the function.

This function deletes the line from QUOTELN

quotedeleteline('a_quoteln')

TOTALQUOTE– Retotals the specified Quote

A full detail of the required parameters for this function can be found in the Transaction Requirements section of this document.

This function should be called after all lines are added to a quote.

totalquote()

CustomerPayment_StartTransaction – Initiates a Customer Payment Transaction

This function initiates a Customer Payment Transaction. It will return a value of 1 if the Payment has commenced.

It validates that the Customer account exists and that you have provided the required parameters.

Parameters**:**

ldDate – Date of the Payment (This is passed as a string in the format DD/MM/YYYY)

lnAccount – Customer Account Number as Integer

lnPaymentMethodID – This is the Type value from BANKMETH.DBF This is an Integer value

lcBankGLID - The GLID for the Bank Account the payment is for. This is a String Value

lcRefRef – A Reference for the Payment. This is a String Value. Pass an empty string if not required

lcRefBank – The Bank Detail for a Cheque Payment. This is a String Value. Pass an empty string if not required

lcRefBranch - The Branch Detail for a Cheque Payment. This is a String Value. Pass an empty string if not required

Example:

customerPayment_StartTransaction(“31/08/2014”,1000,1002,”9170”,”P00001”,””,””)

CustomerPayment_AddLine – Adds an Invoice/Credit Note etc to the Payment

This function adds a component to a Customer Payment that is in Progress.

First call the CustomerPayment_StartTransaction or this function will return 0 (Failure)

No validation is made on the Audit number nor amounts you enter as parameters at this level. Validation is made when the CustomerPayment_FinishTransaction function is called.

If the details are successfully stored, the function will return a value of 1

Parameters:

lnAudit – The Audit Number of the Item being Paid. If this is an unallocated payment, pass a zero here. – Integer

lnAmountPaid – The amount being paid on this line. (excluding Discount). – Double

lnDiscountAmount – The dollar value of discount being applied on this line. – Double

Example:

customerPayment_AddLine(12345,500.00,25.00)

CustomerPayment_FinishTransaction – Completes a Customer Transaction

This function finalizes a customer payment transaction. This should be called after the customerPayment_StartTransaction and all associated customerPayment_AddLine calls have been completed.

This function will return 1 on success.

When the function process, the lines added are validated.

The audit number is valid and belongs to the customer the payment is for. If this check returns false, then this portion of the payment is recorded as unallocated.

The balance is not being over paid (by amount and or discount). Any overpayment will be moved to unallocated.

The date of the payment is not before the date of the line being paid. If this check returns false, then this portion of the payment is recorded as unallocated.

There are no parameters to pass to this function.

Example:

customerPayment_FinishTransaction()

GETCUSTOMERATB - Calculates a Customer ATB Dataset as at a specified date.

This function prepares a JSON dataset containing the Customer Aged Trial Balance data.

The function accepts 6 parameters

tdDate	Date to caclulate balances at.
tnFromAcc	From Account
tnToAcc	To Account (if 0 is passed, then the system maximum of 9999999999 is used)
tcFromGrp	From Group
tcToGrp	To Group (if an empty string is passed then the system maximum of ZZZZZZZZZZ is used)
tcType	Customer Type (Only a single type can be used, pass an empty string to include ALL Types

This function may take a while to run and does not return any notification to the calling application until it is complete.

Example:

getCustomerATB(date(2021,4,20),0,9999999999,"","ZZZZZZZZZZ","")

The return value is a formatted JSon string:


```
[{
   "account": 1000,
```

```
   "balance": 1150.00,
   "future": 0.00,
   "current": 595.75,
   "30": 0.00,
   "60": 525.99,
   "90": 28.26
},
{
   "account": 1010,
   "balance": 50.00,
   "future": 0.00,
   "current": 0.00,
   "30": 45.00,
   "60": 5.00,
   "90": 0.00
}]
```


VARRESET - Reset All the Transaction Properties to their default values.
Note: This function should be called before each Transaction is begun.

varreset()

POSTGSTTXN – Creates a GSTDET record for the GST Report

A full detail of the required parameters for this function can be found in the Transaction Requirements section of this document.

postgsttxn()

POSTBANKRECTXN – Creates a BANKREC record for the Bank Reconciliation

A full detail of the required parameters for this function can be found in the Transaction Requirements section of this document.

postbankrectxn()

GETGSTTYPE - Gets the GSTTYPE value for use with the POSTGSTTXN function

A full detail of the required parameters for this function can be found in the Transaction Requirements section of this document.

The return value (1-6) is the value that should be passed to the PGSTTYPE parameter of the POSTGSTTXN function.

getgsttype()

POSTPRODTXN – Creates a Product Movement Transaction

A full detail of the required parameters for this function can be found in the Transaction Requirements section of this document.

postprodtxn()

SUPP_CREATEINVOICE – Creates a new Supplier Invoice Header Record. (Non Product)

Note: A return value greater than zero indicates success. This number is the LINK number of the invoice. Use this value as the pLink parameter when calling SUPP_INVOICEADDLINE or SUPP_INVOICETOTAL. A value of zero indicates an error. The meaning of the error code is returned in the LastErrorMessage property.

A full detail of the required parameters for this function can be found in the Parameters Explained section of this document.

After you have created an invoice header you can add lines to it using SUPP_INVOICEADDLINE. Also remember to re total the invoice using the SUPP_INVOICETOTAL function.

supp_createinvoice()

SUPP_INVOICEADDLINE – Adds a line to the specified Supplier Invoice.

A full detail of the required parameters for this function can be found in the Parameters Explained section of this document.

You must call the SUPP_INVOICETOTAL function after you have added all the lines you want to a given invoice.

supp_invoiceaddline()

SUPP_INVOICETOTAL – Retotals the specified Supplier Invoice

A full detail of the required parameters for this function can be found in the Parameters Explained section of this document.

This function should be called after all lines are added to an invoice.

supp_invoicetotal()

SUPP_UPDATEINVHEADER – Updates a given field of an existing Pending Supplier Invoice.

There is a list of the valid values for the parameter and the matching variables which should be set for each parameter in the Parameters Explained section of this document.

Updates can only be performed on PENDING invoices.

supp_updateinvheader("NAME")
supp_updateinvheader("INVNUMBER")

CreatePurchaseOrder – Creates a new Supplier Purchase Order Record.

Note: A return value greater than zero indicates success. This number is the LINK number of the invoice. Use this value as the pLink parameter when calling PURCHASEORDERADDLINE or TOTALPURCHASEORDER. A value of zero indicates an error. The meaning of the error code is returned in the LastErrorMessage property.

The Purchase Order is created in a Saved Status.

A full detail of the required parameters for this function can be found in the Parameters Explained section of this document.

After you have created a Purchase Order Header you can add lines to it using PURCHASEORDERADDLINE . Also remember to re total the invoice using the TOTALPURCHASEORDER function.

createPurchaseOrder ()

PurchaseOrderAddLine – Adds a line to the specified Purchase Order.

A full detail of the required parameters for this function can be found in the Parameters Explained section of this document.

You can only add lines to a Purchase Order which is in a Saved Status. Once it has been Ordered (via the Infusion Interface) you cannot add lines to it using the Gateway.

You must call the SUPP_ TOTALPURCHASEORDER function after you have added all the lines you want to a given Purchase Order.

purchaseOrderAddLine ()

TotalPurchaseOrder – Retotals the specified Purchase Order

A full detail of the required parameters for this function can be found in the Parameters Explained section of this document.

This function should be called after all lines are added to a Purchase Order.

totalPurchaseOrder ()

OrderPurchaseOrder – Processes a Saved Purchase Order into an Ordered Purchase Order

Note: This method DOES NOT produce a Printed or Emailed Purchase Order Document

The parameter pPOOrdNum must be populated with the Order Number to be processed.
The order must be Saved.
Does NOT support Job Management Orders set to Deliver to Job Sites.

orderPurchaseOrder()

UpdatePOHeader – Updates a given field of an existing pending Purchase Order.

There is a list of the valid values for the parameter and the matching variables which should be set for each parameter in the Parameters Explained section of this document.

Updates can only be performed on PENDING Purchase Orders (Ordered or Saved).

updatepoheader(“NAME”)
updatepoheader(“DATE”)

Loyalty System Functions

GetLoyaltyBalance – Gets the CURRENT Loyalty Account Balance for a Customer (External Loyalty System)

The returned value is a numeric value. It is the balance of the Account.

If the returned value is -1 please enquire the LastError value. If it is any value other than 0 then an error occurred. In this case please check the LastErrorMessage property. If the LastError value is 0, then -1 is the Loyalty Account Balance.

getLoyaltyBalance(123)

GetExpiringLoyaltyBalance – Gets the Expiring Loyalty Account Balances for a Customer (External Loyalty System)

The returned value is a numeric value. Either 1 (Success) or -1 (Error/Failure)

If the return value is 1, then enquire the rExpiringRewards1 (Current) Property and rExpiringRewards2 (Next Month) Property to obtain the values.

getExpiringLoyaltyBalance(123)

GetPendingLoyaltyBalance – Gets the Pending Loyalty Account Balances for a Customer (External Loyalty System)

The returned value is a numeric value. It is the pending balance of the Account.

If the returned value is -1 please enquire the LastError value. If it is any value other than 0 then an error occurred. In this case please check the LastErrorMessage property. If the LastError value is 0, then -1 is the Pending Balance.

getPendingLoyaltyBalance(123)

GetLoyaltyTransactionList – Creates a cursor containing the Loyalty Transactions for a given Customer and Date Range. (External Loyalty System)

There are 3 parameters require for this function.

- Account Number – Integer
- From Date – String in the format dd/mm/yyyy
- To Date – String in the format dd/mm/yyyy

The returned value is a numeric value. It is the number of records in the cursor. If the value is -1 then an error occurred. If the value is 0 then there are no matching transactions.

The resultant cursor is named c_LoyList and contains the following fields:

- Audit
- Date
- PostDate

- TrType
- Account
- Trans
- Ref
- Credit
- Debit

This cursor can be controlled using the standard table functions.

- Select('c_LoyList')
- GoTop()
- Get('Account')
- Skip('c_LoyList')

getLoyaltyTransactionList(123,"01/04/2015","20/04/2015")

PostLoyaltyTransaction – Creates a Loyalty Transactions for a given Customer. (External Loyalty System)

There are 6 parameters require for this function.

- Transaction Date – String in the format dd/mm/yyyy
- Account Number – Integer
- Transaction Description – String
- Transaction Reference – String
- Amount Including GST – Number
- GST Amount – Number

The returned value is a numeric value. Either 1 (Success) or -1 (Error/Failure)

This function posts GL Journals to the Expense GL ID and Holding GL ID configured within Infusion as well as the required transactions to the Loyalty System.

postLoyaltyTransaction("20/04/2015",123,"Extra Rewards","Extra",11.50,1.50)

Emailing Functions

EMAILCUSTOMERINVOICE – Creates a PDF for a specified Customer Invoice and adds it to the Pending Email queue.

This Function is available in the IBSEmailCustInvoice.DLL. You need logon and setup that DLL in the same as the main DLL.

In order to use this function additional files need to be stored in a folder accessible to your application. These files are:

Custinv.frt
Custinv.frx
Gdiplus.dll
Hndlib.dll
Xfrx.fxp
Xfrxlib.fll
Zlib.dll

The location of this folder should be supplied to the DLL using the method setPDFPath

The Customer Invoice can be pending or posted. This function does not use the Infusion Email Links. You need to provide the email address you wish to send to in the function call.

There are six parameters required in this function call:

nDocumentLink – This numeric value is the LINK number of the Invoice from the INVIDX record
cEmailAddress – The email address of the recipient
cEmailName – The display name of the recipient
cUserID – A valid Infusion UserID which will be used as the documents sender.
cSubject – Subject Line
cBody – Email Body

The function returns an audit number if successful. It returns 0 on failure. Also if the gateway is not logged on to the data file -1 is returned.

The invoice layout used is contained in the file CUSTINV.FRX/FRT and should be placed in the folder supplied in the SETPDFDATA call.

CUSTINV is the same as the standard Infusion Customer Invoice Layouts with the exception that the following code is placed in the BEFOREOPENTABLES method. It replaces the line DO DEFENV which is in the standard layout.

```
Set Status Bar Off
Set Resource Off
Set Century On
Set Decimals To 2 && was 4
Set Memowidth To 1000
Set Date BRITISH
Set Reprocess To 1
Set Excl Off
Set Deleted On
Set Talk Off
Set Echo Off
Set Bell Off
Set Null Off
Set Safety Off
Set ENGINEBEHAVIOR 70
SET COLLATE TO "MACHINE"
Clear Macros
```

```
Set TABLEVALIDATE To 13
Set Notify Off
```

Example:
emailCustomerInvoice(12345,email@company.co.nz,”Receiver Name”,”Subject Info”,”Body Info”)

EMAILSUPPLIERINVOICE – Creates a PDF for a specified Supplier Invoice and adds it to the Pending Email queue.

This Function is available in the IBSEmailSuppInvoice.DLL. You need logon and setup that DLL in the same as the main DLL.

In order to use this function additional files need to be stored in a folder accessible to your application. These files are:

Suppinv.frt
Suppinv.frx
Gdiplus.dll
Hndlib.dll
Xfrx.fxp
Xfrxlib.fll
Zlib.dll

The location of this folder should be supplied to the DLL using the method setPDFPath

Note: Since Infusion does not support Emailing of Supplier Invoices, it is treated as a Purchase Order for emailing purposes.

The Supplier Invoice can be pending or posted. This function does not use the Infusion Email Links. You need to provide the email address you wish to send to in the function call.

There are six parameters required in this function call:
nDocumentLink – This numeric value is the LINK number of the Invoice from the PORDIDX record
cEmailAddress – The email address of the recipient
cEmailName – The display name of the recipient
cUserID – A valid Infusion UserID which will be used as the documents sender.
cSubject – Subject Line
cBody – Email Body

The function returns an audit number if successful. It returns 0 on failure. Also if the gateway is not logged on to the data file -1 is returned.

The invoice layout used is contained in the file SUPPINV.FRX/FRT and should be placed in the folder supplied in the SETPDFDATA call.

SUPPINV is the same as the standard Infusion version with the exception that the following code is placed in the BEFOREOPENTABLES method. It replaces the line DO DEFENV which is in the standard layout.

```
Set Status Bar Off
Set Resource Off
Set Century On
Set Decimals To 2 && was 4
Set Memowidth To 1000
Set Date BRITISH
Set Reprocess To 1
Set Excl Off
Set Deleted On
Set Talk Off
Set Notify Off
Set Echo Off
Set Bell Off
Set Null Off
```

```
Set Safety Off
Set ENGINEBEHAVIOR 70
SET COLLATE TO "MACHINE"
Clear Macros
Set TABLEVALIDATE To 13
```

Example:
emailSupplierInvoice(12345,email@company.co.nz,”Receiver Name”,”Subject Info”,”Body Info”)

SENDEMAIL – Sends a single email from the Pending Email Queue

Use this function to send an email which exists in the Infusion Pending Email Queue.

Three parameters are required:

nEmailAudit – Audit number of the entry in Emails.DBF
cUserID – A valid Infusion UserID
cDocType – Selected from the following list. - An empty string can also be provided in place of this parameter

PORD – Purchase Order or Supplier Invoice
CRREMIT – Supplier Remittance
STATE – Customer Statement
INV – Customer Invoice
DRREC – Customer Receipt
QUOTE – Customer Quote
PACK – Customer Packing Slip

The senders email address is obtained by the following hierarchy

Email Address from the Staff ID of the Document Creator (If flagged to do this within Infusion)
Email Address from the document type under Email Defaults in Infusion.
Email Address from Email Defaults – Default Email Address

sendEmail(12345,”STAFF”,””)
sendEmail(12345,”STAFF”,”INV”)
sendEmail(12345,”STAFF”,”PORD”)

Contact Management Functions

clearCMLogParameters – Clears/Resets ALL pCMxxxxxx Parameters on the Gateway Object

This function should be called to reset all the Contact Management Parameters of the Gateway Object to their default values.

It is advisable to call this function before using any of the following functions:

- createCMLogEntry
- getCMLogEntryByID
- updateCMLogEntryByID

clearCMLogParameters()

createCMLogEntry

This function will add an entry to the Contact Management System.

First Populate the Gateway Objects pCM Parameters and then call this function. No parameters are required.

createCMLogEntry()

The function will return the Log ID on success, 0 or -1 on failure. Check the LastErrorMessage parameter for the reason for a failure.

At the very least you must supply:

- pCMModule C – Customers, S – Suppliers, Z – Staff
- pCMAccountID Not Applicable in the case of Staff Entry
- pCMStaffCode Not Applicable in the case of Customer and Supplier Entries
- pCMDateTime Date and Time of Entry
- pCMCreatedByID Staff ID who created the Entry
- pCMTypeID Contact Management Type ID from getCMTypes function

TypeIDs (CM Type,Group and Follow Up) are validated during creation of the Entry. If they are invalid then they are ignored (Remember CM Type is a required parameter)

getCMTypes – Creates a temporary table (cursor) which contains a list of valid Contact Management Contact Types.

Pass a single parameter indicating the Module you want the valid types for.

Valid Values are:

- C – Customers
- S – Suppliers
- Z – Staff
- Or Blank (Empty String) for ALL values (C,S and Z)

This function will return -1 if there is an error or an integer representing the number of records in the temporary table. This value may be zero.

The temporary table has the alias **c_CMTypes**. It contains 5 Fields :

- Code
- Description
- Customer - Logical Value identifying this value as valid for Customers
- Supplier - Logical Value identifying this value as valid for Suppliers
- Staff - Logical Value identifying this value as valid for Staff

nRecCount = getCMTypes()

You can iterate through the table using code similar to:

```
listBox1.Items.Clear();
Int32 r = ibsGW.Select("c_CMTypes");
do
{
    listBox1.Items.Add(ibsGW.Get("Desc"));
    ibsGW.Skip("c_cmTypes");
} while (!ibsGW.isEOF());
```

getCMGroups– Creates a temporary table (cursor) which contains a list of valid Contact Management Groups.

Pass a single parameter indicating the Module you want the valid groups for.

Valid Values are:

- C – Customers
- S – Suppliers
- Z – Staff
- Or Blank (Empty String) for ALL values (C,S and Z)

This function will return -1 if there is an error or an integer representing the number of records in the temporary table. This value may be zero.

The temporary table has the alias **c_CMGroups**. It contains 5 Fields :

- Code

- Description
- Customer - Logical Value identifying this value as valid for Customers
- Supplier - Logical Value identifying this value as valid for Suppliers
- Staff - Logical Value identifying this value as valid for Staff

nRecCount = getCMGroups()

You can iterate through the table using code similar to:

```
listBox1.Items.Clear();
Int32 r = ibsGW.Select("c_CMGroups");
do
{
    listBox1.Items.Add(ibsGW.Get("Desc"));
    ibsGW.Skip("c_cmGroups");
} while (!ibsGW.isEOF());
```

getCMFUTypes– Creates a temporary table (cursor) which contains a list of valid Contact Management Follow Up Types

Pass a single parameter indicating the Module you want the valid Follow Up Types for.

Valid Values are:

- C – Customers
- S – Suppliers
- Z – Staff
- Or Blank (Empty String) for ALL values (C,S and Z)

This function will return -1 if there is an error or an integer representing the number of records in the temporary table. This value may be zero.

The temporary table has the alias **c_CMFUTypes**. It contains 5 Fields :

- Code
- Description
- Customer - Logical Value identifying this value as valid for Customers
- Supplier - Logical Value identifying this value as valid for Suppliers
- Staff - Logical Value identifying this value as valid for Staff

nRecCount = getCMFUTypes()

You can iterate through the table using code similar to:

```
listBox1.Items.Clear();
Int32 r = ibsGW.Select("c_CMFUTypes");
do
{
```

```
    listBox1.Items.Add(ibsGW.Get("Desc"));
    ibsGW.Skip("c_CMFUTypes");
} while (!ibsGW.isEOF());
```

getCMLogEntryList– Creates a temporary table (cursor) which contains a list of Contact Management Log Entries

7 Parameters are accepted for this function, depending on the type of log entry you require some values can be blank or 0

- nFromAccount
- nToAccount
- cFromStaff
- cToStaff
- dFromDate
- dToDate
- cCMModule

The cCMModule parameter dictates which type of log entries will be returned and thus which of these parameters must be set.

Valid Values are:

- C – Customers
- S – Suppliers
- Z – Staff
- Y – Entered By (All Modules)
- X – Assigned To / Follow Up (All Modules)

Customers – Requires nFromAccount, nToAccount, dFromDate, dToDate and cCMModule
Suppliers – Requires nFromAccount, nToAccount, dFromDate, dToDate and cCMModule
Staff – Requires cFromStaff,cnToStaff, dFromDate, dToDate and cCMModule
Entered By – Requires cFromStaff, cToStaff, dFromDate, dToDate and cCMModule
Assigned To – Requires cFromStaff, cToStaff, dFromDate, dToDate and cCMModule

When cFromStaff and cToStaff are not required set them to an empty string.
When nFromAccount and nToAccount are not required set them to 0
This function will return -1 if there is an error or an integer representing the number of records in the temporary table. This value may be zero.

The following command will return a list of entries for the Customer Account 1 between 1/1/2015 and 31/3/2015
Note: Dates must be passed as valid dates in your programming language. (Not Strings)

nRecCount = getCMLogEntryList(1,1,"","",01/01/2015,31/03/2015,"C")

The resultant temporary table has the alias **c_CMLogEntryList.** It contains only 2 Fields :

- ID
- Module

You can iterate through the table using and retrieve the logs with code similar to:

```
ibsGW.Select("c_CMLogEntryList");
Int32 id = 0;
do
{
    id = ibsGW.get(“c_CMLogEntryList.id”)

    ibsGW.getCMLogEntryByID(id);
    .
    .
    .
    .

    ibsGW.Select("c_CMLogEntryList");
    ibsGW.Skip("c_CMLogEntryList");
} while (!ibsGW.isEOF());
```

getCMLogEntryByID – Populates the Gateway Object pCM parameters with the values from a given log entry.

A single integer parameter is required. This is the log entry ID returned in the getCMLogEntryList function.

You should call the clearCMParameters function before calling this function.

The return value will be the same Log ID if the parameters have been populated. 0 indicates there was an error.

getCMLogEntryByID(100)

updateCMLogEntryByID – Updates an existing Log Entry’s Field

Two parameters are required for this function:

- nID – Log Entry ID
- cVarCode – Key Value to Update (From the list below)

Populate the relevant pCM parameter of the Gateway Object and then call this function. You can set all the parameters first and then make sequential calls to update each field.

```
ibsGW.pCMAccountID = 100;
ibsGW.pCMCreatedByStaffID = “STAFF”;
ibsGW.updateCMLogEntryByID(1234,”ACCOUNTID”);
ibsGW.updateCMLogEntryByID(1234,”CREATEDBY”);
```

The valid cVarCode values and their matching pCM Parameters are:

Parameter	cVarCode	Data Value

pCMAccountID	ACCOUNTID	Numeric Account Number
pCMStaffCode	STAFFCODE	Character Staff ID from STAFF.DBF
pCMDateTime	DATETIME	DateTime representing Date and Time of Entry
pCMCreatedByStaffID	CREATEDBY	Character Staff ID from STAFF.DBF who created the Entry
pCMContactID	CONTACTID	Numeric ID of the Contact from SUPPCONT or CUSTCONT
pCMStatus	STATUS	Either “Completed” or “Active” are acceptable values
pCMTypeID	TYPEID	Character ID value from getCMTypes function
pCMTitle	TITLE	String
pCMNotes	NOTES	String
pCMFollowUpID	FUID	Character ID value from getCMFUTypes function
pCMFollowUpDate	FUDATE	Date
pCMFollowUpStaffID	FUSTAFFID	Character Staff ID from STAFF.DBF for the Follow Up
pCMPriority	PRIORITY	1 (Urgent), 2 (High), 3 (Normal), 4 (Low), 5 (None)
pCMGroupID	GROUPID	Character ID value from getCMGroups function
pCMReference	REFERENCE	Character

deleteCMLogEntry – Deletes a Contact Management Log Entry

A single parameter representing the ID of the Log Entry to delete is required.

deleteCMLogEntry(100)

The return value will be 1 if the deletion succeeded.

CREATECONTACT – Parameters Explained

The following parameters are used in the procedure CREATECONTACT. Required Parameters are underlined.

- **pAccount**

  Customer or Supplier Account Number the Contact is associated with
  Data type is Numeric

- pAdd1,2,3,4

  Contact Address Lines (1 – 4)
  Data type is AlphaNumeric

- pEmail

  Contact Email Address
  Data type is AlphaNumeric

- pFax

  Contact Phone Number
  Data type is AlphaNumeric

- pFirstName

  Contact First Name
  Data type is AlphaNumeric

- pMobile

  Contact Mobile Phone Number
  Data type is AlphaNumeric

- pPhone

  Contact Phone Number
  Data type is AlphaNumeric

- pPosi

  Contact Position
  Data type is AlphaNumeric

- pPostcode

  Contact Address Postcode
  Data type is AlphaNumeric

- pSurname

  Contact Surname
  Data type is AlphaNumeric

- pTitle

  Contact Title. (Mr, Mrs, Ms, Miss, Mstr, Dr, Rev, Sir, Dame)
  Data type is AlphaNumeric

- pVOIP

  Contact VOIP Number
  Data type is AlphaNumeric

UPDATECONTACT - Parameters Explained

The following parameters are used in the procedure UPDATECONTACT

This list contains the parameter to pass and the matching variable which should be populated with the value you wish to replace.

Parameter	Variable to be populated.	Field Replaced
ADD1	pAdd1	Address Line 1
ADD2	pAdd2	Address Line 2
ADD3	pAdd3	Address Line 3
ADD4	pAdd4	Address Line 4
EMAIL	pEmail	Email Address
FAX	pFax	Fax Number
FIRSTNAME	pFirstName	First Name
MOBILE	pMobile	Mobile Number
PHONE	pPhone	Phone Number
POSI	pPosition	Contact Position
POSTCODE	pPostCode	Post Code
SURNAME	pSurname	Surname
TITLE	pTitle	Contacts Title (Mr, Mrs etc)
VOIP	pVoip	VOIP Number

POSTJOBTXN – Parameters Explained

The following parameters are used in the procedure POSTJOBTXN. Required Parameters are underlined.

- pAudit

  The transaction audit number, if not supplied then it is automatically generated
  Data type is Numeric

- pAutoPrice

  Pass a value of 1 to automatically determine the selling rate based on customer settings in Infusion.
  Data type is Numeric

- pCostRate

  The cost rate of the item excluding GST. Note if pAutoPrice is set to 1 this will be calculated also
  Data type is Numeric

- **pDate**

  Date of the transaction
  Data type is Date

- pDepartment

  Department code, this value is case sensitive and must match that in Infusion.
  Data type is AlphaNumeric

- pDesc

  Line Description for the transaction
  This value is also inserted into the NOTES field if the pNotes parameter empty.
  Data type is AlphaNumeric

- pDiscount

  Discount Percentage to apply to the selling price. (12.50 = 12.5%)
  Data type is Numeric

- pFrom

  Start time for Staff transactions (24 hour preferred) 09:00, 13:30 etc
  Data type is AlphaNumeric

- **pJobID**

  Job ID, this must exist in Infusion. Alternatively you can ignore this and provide the link number (pLink). This value is case sensitive. If pLink and pJobID are passed then pJobID takes precedence if they refer to different jobs. Either one or both of pLink and pJobID must be provided.
  Data type is AlphaNumeric

- **pLink**

  Link number of the Job. If you do not know this then leave it as default. If pLink and pJobID are passed then pJobID takes precedence if they refer to different jobs. Either one or both of pLink and pJobID must be provided.
  Data type is Numeric

- pLocation

  Location number for this transaction, if not provided then the value assigned to the job in Infusion is used
  Data type is Numeric

- pNotes

  Notes Information. This value is inserted into the NOTES field of the JOBLN record.
  Data type is AlphaNumeric

- **pPCode**

  Product Code from Infusion.
  Data type is AlphaNumeric

- **pQty**

  Quantity for the transaction, any value other than 0 is valid.
  Data type is Numeric

- pRate

  Selling Rate (Excluding GST)
  Data type is Numeric

- **pSerial**

  Product Serial Number. Only used/required if the product is set to track sold serial number in Infusion.
  Data type AlphaNumeric

- pTo

  End time for Staff transactions (24 hour preferred) 09:00, 13:30 etc
  Data type is AlphaNumeric

POSTGLTXN – Parameters Explained

The following parameters are used in the procedure POSTGLTXN. Required Parameters are underlined.

- pAudit

  Audit Number for the transaction
  Data type is Numeric

- pBatch

  Batch ID for the transaction
  Data type is AlphaNumeric

- pCredit

  Credit amount for the Transaction (pass as a positive figure)
  Data type is Numeric

- **pDate**

  Date of the Transaction
  Data Type is Date

- pDebit

  Debit amount for the Transaction (pass as a positive figure)
  Data type is Numeric

- **pGLID**

  GLID to post the transaction to
  Data type is AlphaNumeric

- pRef

  A reference for the Transaction
  Data type is Alpha Numeric

- **pTrans**

  Transaction Description – Note this field must not be blank
  Data type is AlphaNumeric

- pTrans2

  Additional Transaction Description
  Data type is AlphaNumeric

UPDATEINVHEADER – Parameters Explained

The following parameters are used in the procedure UPDATEINVHEADER

You must populate the parameter **pLink** with the link number of the invoice you wish to update. Remember this can only be a pending invoice.

This list contains the parameter to pass and the matching variable which should be populated with the value you wish to replace.

Parameter	Variable to be populated.	Field Replaced
ACCOUNT	pAccount	Customer Number
ADD1	pAdd1	Address Line 1
ADD2	pAdd2	Address Line 2
ADD3	pAdd3	Address Line 3
ADD4	pAdd4	Address Line 4
ATTNLINK	pAttnLink	Charge To Attention Link
DADD1	pDAdd1	Delivery Address Line 1
DADD2	pDAdd2	Delivery Address Line 2
DADD3	pDAdd3	Delivery Address Line 3
DADD4	pDAdd4	Delivery Address Line 4
DATE	pDate	Invoice Date
DDATE	pDDate	Delivery Date
DELATTNLINK	pDelAttnLink	Deliver To Attention Link
DELNOTES	pNotes	Delivery Notes
DELSTATUS	pDelStatus	Delivery Status
DPOSTCODE	pDPostCode	Delivery Post Code
DNAME	pDName	Delivery Name
DUEDATE	pDueDate	Invoice Due Date
ENTEREDBY	pEnteredBy	Entered by
FREIGHT	pFreight	Freight (Invoice footer)
FAX	pFax	Delivery Fax Number
HDRDATE1	pHdrDate1	Custom Fields – Date 1
HDRDATE2	pHdrDate2	Custom Fields – Date 2
HDRUSER1	pHdrUser1	Custom Fields – User 1
HDRUSER2	pHdrUser2	Custom Fields – User 2
HDRUSER3	pHdrUser3	Custom Fields – User 3
HDRUSER4	pHdrUser4	Custom Fields – User 4
HDRUSER5	pHdrUser5	Custom Fields – User 5
HDRUSER6	pHdrUser6	Custom Fields – User 6
INVORDNUM	pInvOrdnum	Order Number
NAME	pName	Customer Name
NOTES	pNotes	Notes
PHONE	pPhone	Delivery Phone Number
POSTCODE	pPostCode	Post Code
REF	pRef	Reference
REP	pRep	Sales Rep
ROUNDING	pRounding	Rounding (Invoice footer)
TITLE	pTitle	Invoice Title
INVNUMBER	pInvNumber	Invoice Number

TOTALINVOICE – Parameters Explained

You must populate the parameter **pLink** with the link number of the invoice you wish to update.

Remember this can only be a pending invoice.

CREATEINVOICE – Parameters Explained

The following parameters are used in the procedure CREATEINVOICE. Required Parameters are underlined.

- **pAccount**

  Customer Account the invoice is to be charged to.
  Data Type is Numeric

- *pAdd1, pAdd2, pAdd3, pAdd4, pPostCode*

  Customers Charge to Address Details
  Data Type is AlphaNumeric

- *pAttnLink*

  Link number of the contact for the Charge To Attention Field.
  0 = Accounts Payable, 1 = None
  Any other value should be the Link Number from CUSTCONT

  Data Type is Numeric

- *pDAdd1, pDAdd2, pDAdd3, pDAdd4, pDPostCode*

  Customers Deliver to Address Details
  Data Type is AlphaNumeric

- **pDate**

  Date of the Transaction
  Data Type is Date

- *pDDate*

  Invoice Delivery Date
  Data Type is Date

- *pDelAttnLink*

  Link number of the contact for the Deliver To Attention Field.
  0 = Inwards Goods, 1 = None
  Any other value should be the Link Number from CUSTCONT

  Data Type is Numeric

- *pEnteredBy*

  Staff code of the person who entered the Invoice
  Data Type is AlphaNumeric

- *pFreight*

  Freight amount for the invoice footer. (Inclusive or exclusive of GST depending on your Product Price Settings)

Data Type is Numeric

- *pHdrUser1, pHdrUser2, pHdrUser3, pHdrUser4, pHdrUser5, pHdrUser6*

  Customer Fields User 1 to User 6
  Data Type is AlphaNumeric

- *pHdrDate1, pHdrDate2*

  Customer Fields User Dates 1 and 2
  Data Type is Date

- **pInvNumber**

  Invoice Number
  Data Type is AlphaNumeric

- *pInvOrdnum*

  Customers order number
  Data Type is AlphaNumeric

- **pLocation**

  Number of the location the invoice is to be generated from.
  Data Type is Numeric

- *pName*

  Customers Charge to Name
  Data Type is AlphaNumeric

- *pRep*

  Staff code of the Sales Rep
  Data Type is AlphaNumeric

- *pRounding*

  Rounding amount for the invoice footer.
  Data Type is Numeric

- *pTitle*

  Invoice Title
  Data Type is AlphaNumeric

Note:

The pDelStatus parameter does not populate the Invoice Delivery Status on CreateInvoice(). Please call the UpdateInvHeader() function after CreateInvoice() to do this.

INVOICEADDLINE – Parameters Explained

The following parameters are used in the procedure INVOICEADDLINE. Required Parameters are underlined.

- ***pLink***

  Invoice Link Number – This must be supplied and must relate to a Pending Invoice.
  Data Type is Numeric

- ***pPCode***

  (Only Manditory when Adding a Line for a Product Code. The parameter is not used when adding a line for a GL ID. It should be left as an empty string when adding a GL ID line)

  Product Code from the Products file.

  From IBSGateway837 forward you can pass the Primary Barcode in this field instead. It will first look for a matching Product Code, but then move to search the Barcode field. If a match is found as a barcode, the pPCode parameter is updated by the Gateway to the Product Code value and that is the value which will be stored in the Invoice line's code field.

  You can pass "/N" as the product code and a Notes Line will be created. The text for this should be placed in the pNotes parameter. Any dollar or qty values passed when using "/N" will be ignored.

  Data Type is AlphaNumeric

- ***pGLID***

  (Only Manditory when Adding a Line for a GL ID. The parameter is optional when adding a line for a Product Code)

  Supply the GL ID value only (no leading / as you would in the Customer Invoice Entry Screen)

  Data Type is AlphaNumeric

- *pCostRate*

  Unit Cost Rate (Excluding GST)

  Data Type is Numeric

- *pDesc*

  Description Text
  Data Type is AlphaNumeric

- *pDiscount*

  Discount Percentage

  If pAutoPrice is set then this value will be overwritten with the determined value.

Data Type is Numeric

- *pNotes*

  Notes Text
  Data Type is AlphaNumeric

- *pSerial*

  Serial Number of a Serial Tracked item. If the item is not Serial Tracked this is ignored.
  Data Type is AlphaNumeric

- *pQty*

  Quantity Supplied
  Data Type is Numeric

- *pRate*

  Selling Rate (Including or Excluding GST based on your Product Defaults setting)

  If pAutoPrice is set then this value will be overwritten with the calculated value.

  Data Type is Numeric

- *pAutoPrice*

  Auto Price Flag. Populate with any value other than empty string to have the DLL calculate the selling rate based on Infusion's best price rules.

  Note: This function is not run for a GL ID line. In this case use the pRate, pDiscount, and pCostRate parameters.

  Data Type is AlphaNumeric

- *pUSER1 (pUSER2)*

  Custom Fields for the Purchase Order Line.

  Data Type is AlphaNumeric

CUSTOMERBACKORDER_CREATE – Parameters Explained

The following parameters are used in the procedure CUSTOMERBACKORDER_CREATE. Required Parameters are underlined.

- **pAccount**

  Customer Account the invoice is to be charged to.
  Data Type is Numeric

- *pAdd1, pAdd2, pAdd3, pAdd4, pPostCode*

  (Recommended as this is passed to the Invoice when created)

  Customers Charge to Address Details
  Data Type is AlphaNumeric

- *pDAdd1, pDAdd2, pDAdd3, pDAdd4, pDPostCode*

  (Recommended as this is passed to the Invoice when created)

  Customers Deliver to Address Details
  Data Type is AlphaNumeric

- **pDate**

  Date of the Transaction
  Data Type is Date

- *pHdrUser1, pHdrUser2, pHdrUser3, pHdrUser4, pHdrUser5, pHdrUser6*

Customer Fields User 1 to User 6
Data Type is AlphaNumeric

- *pInvOrdnum*

  Customers order number
  Data Type is AlphaNumeric

- **pLocation**

  Number of the location the invoice is to be generated from.
  Data Type is Numeric

- *pName*

  (Recommended as this is passed to the Invoice when created)

  Customers Charge to Name
  Data Type is AlphaNumeric

- *pNotes*

  Notes from the Invoice Header
  Data Type is AlphaNumeric

- *pRep*

  Staff code of the Sales Rep
  Data Type is AlphaNumeric

- *pRef*

  Customer Reference
  Data Type is AlphaNumeric

- *pTitle*

  Invoice Title
  Data Type is AlphaNumeric

INVOICEADDLINE – Parameters Explained

The following parameters are used in the procedure INVOICEADDLINE. Required Parameters are underlined.

- ***pLink***

  Invoice Link Number – This must be supplied and must relate to a Pending Invoice.
  Data Type is Numeric

- ***pPCode***

  Product Code from the Products file.

  Note this property does not check the Barcode field values.

Data Type is AlphaNumeric

- *pDesc*

  Description Text
  Data Type is AlphaNumeric

- *pDiscount*

  Discount Percentage

  If pAutoPrice is set then this value will be overwritten with the determined value.

  Data Type is Numeric

- *pNotes*

  Notes Text
  Data Type is AlphaNumeric

- *pBack*

  Quantity placed on Backorder
  Data Type is Numeric

- *pRate*

  Selling Rate (Including or Excluding GST based on your Product Defaults setting)

  If pAutoPrice is set then this value will be overwritten with the calculated value.

  Data Type is Numeric

- *pAutoPrice*

  Auto Price Flag. Populate with any value other than empty string to have the DLL calculate the selling rate based on Infusion's best price rules.

  Data Type is AlphaNumeric

- *pUSER1 (pUSER2)*

  Custom Fields for the Purchase Order Line.

  Data Type is AlphaNumeric

CREATEQUOTE – Parameters Explained

The following parameters are used in the procedure CREATEQUOTE. Required Parameters are underlined.

- ***pAccount***

  Customer Account the invoice is to be charged to.
  Data Type is Numeric

- *pAdd1, pAdd2, pAdd3, pAdd4, pPostCode*

  Customers Charge to Address Details
  If these are not supplied, then the values from CUSTOMER.DBF are used

  Data Type is AlphaNumeric

- *pDAdd1, pDAdd2, pDAdd3, pDAdd4, pDPostCode*

  Customers Deliver to Address Details
  If these are not supplied, then the values from CUSTOMER.DBF are used

  Data Type is AlphaNumeric

- ***pDate***

  Date of the Transaction
  Data Type is Date

- *pEnteredBy*

  Staff code of the person who entered the Invoice
  Data Type is AlphaNumeric

- *pFreight*

  Freight amount for the quote footer. (Inclusive or exclusive of GST depending on your Product Price Settings)
  Data Type is Numeric

- *pJobID*

  Job Management JOBID value.
  Data Type is AlphaNumeric

- *pHdrUser1, pHdrUser2, pHdrUser3, pHdrUser4, pHdrUser5, pHdrUser6*

  Customer Fields User 1 to User 6
  Data Type is AlphaNumeric

- *pHdrDate1, pHdrDate2*

  Customer Fields User Dates 1 and 2
  Data Type is Date

- ***pQuoteNumber***

  Quote Number
  Data Type is AlphaNumeric

- *pInvOrdnum*

  Customers order number
  Data Type is AlphaNumeric

- ***pLocation***

  Number of the location the quote is to be generated from.
  Data Type is Numeric

- *pName*

  Customers Charge to Name
  Data Type is AlphaNumeric

- *pNotes*

  Notes for the header of the Quote
  Max length is 250 characters, Line breaks may not be supported.
  Data Type is AlphaNumeric

- *pRef*

  Reference Field
  Data Type is AlphaNumeric

- *pRep*

  Staff code of the Sales Rep
  Data Type is AlphaNumeric

- *pRounding*

  Rounding amount for the invoice footer.
  Data Type is Numeric

- *pSatus*

  Status CODE for the Quote.
  If not set, then *ACTIVE* is used.
  Data Type is AlphaNumeric

- *pTitle*

  Quote Title
  Data Type is AlphaNumeric

- pType

Type CODE for the Quote.
If not set, then *STD* is used.
Data Type is AlphaNumeric

QUOTEADDLINE – Parameters Explained

The following parameters are used in the procedure QUOTEADDLINE. Required Parameters are underlined.

- ***pLink***

  Quote Link Number – This must be supplied and must relate to a Pending Quote.
  Data Type is Numeric

- ***pPCode***

  Product Code from the Products file.

  Note this property does not check the Barcode field values.

  You can pass "/N" as the product code and a Notes Line will be created. The text for this should be placed in the pNotes parameter. Any dollar or qty values passed when using "/N" will be ignored.

  Data Type is AlphaNumeric

- *pCostRate*

  Unit Cost Rate (Excluding GST)

  Data Type is Numeric

- *pDesc*

  Description Text
  Data Type is AlphaNumeric

- *pDiscount*

  Discount Percentage

  If pAutoPrice is set then this value will be overwritten with the determined value.

  Data Type is Numeric

- *pNotes*

  Notes Text
  Data Type is AlphaNumeric

- *pQty*

  Quantity Quoted
  Data Type is Numeric

- *pRate*

  Selling Rate (Including or Excluding GST based on your Product Defaults setting)

If pAutoPrice is set then this value will be overwritten with the calculated value.

Data Type is Numeric

- *pAutoPrice*

  Auto Price Flag. Populate with any value other than empty string to have the DLL calculate the selling rate based on Infusion's best price rules.

  Data Type is AlphaNumeric

- *pUSER1 (pUSER2)*

  Custom Fields for the Quote Line.

  Data Type is AlphaNumeric

GETGSTTYPE – Parameters Explained

The following parameters are used in the procedure GETGSTTYPE. Required Parameters are underlined.

- ***pGSTType***

  GST Type as a character. Valid values are I, E, P, Q, R, S, X, Z, and C
  Data Type is AlphaNumeric

POSTGSTTXN – Parameters Explained

The following parameters are used in the procedure POSTGSTTXN. Required Parameters are underlined.

- ***pAudit***

  Audit Number for the transaction
  Data type is Numeric

- ***pGSTType***

  Value returned from the function GETGSTTYPE
  Data type is Numeric

- ***pDate***

  Date of the Transaction
  Data Type is Date

- ***pPayment***

  Indicator that this is part of the GST Input Credits (Your Payments).
  Pass a value of "Y" to indicate a payment, otherwise it will be treated as Income
  Data Type AlphaNumeric

- ***pTrans***

  Transaction Description
  Data type is AlphaNumeric

- pTrans2

  Transaction Description Line 2
  Data type is AlphaNumeric

- ***pRate***

  GST Inclusive Amount
  This is always passed as a positive (unless its negative income or negative expense)
  Data type is Numeric

- ***pGST***

  GST Content Amount
  This is always passed as a positive (unless its negative income or negative expense)
  Data type is Numeric

- pRef

  Transaction Reference
  Data type is AlphaNumeric

- pFlag

Indicator that this transaction is at the old GST Rate (12.5%) but dated after 1/10/2010
Pass a value of “Y” to indicate condition otherwise leave it blank
Data Type AlphaNumeric

POSTBANKRECTXN – Parameters Explained

The following parameters are used in the procedure POSTBANKRECTXN. Required Parameters are underlined.

- ***pAudit***

  Audit Number for the transaction
  Data type is Numeric

- ***pDate***

  Date of the Transaction
  Data Type is Date

- *pTrans*

  Transaction Description
  Data type is AlphaNumeric

- ***pRate***

  GST Inclusive Amount
  Pass this as a Positive for Deposit or Negative for Withdrawl
  Data type is Numeric

- pRef

  Transaction Reference
  Data type is AlphaNumeric

- **pGLID**

  GLID of the Bank Account
  Data type is AlphaNumeric

POSTPRODTXN – Parameters Explained

The following parameters are used in the procedure POSTPRODTXN. Required Parameters are underlined.

- *pAudit*

  Audit Number for the transaction. If this is not provided then one will be assigned. Use the NEXTAUDIT function to obtain this if required.
  Data type is Numeric

- *pCostRate*

  Used for Manual Sale Transaction only. GST Exclusive.
  Data type is Numeric

- ***pDate***

  Date of the Transaction
  Data Type is Date

- *pDiscount*

  Discount Percentage. Only used on Manual Sale Transactions
  Data type is Numeric

- *pGSTRate*

  GST Rate. Only required if Manual Sale Transaction and the Selling Rate INCLUDES GST.
  Data type is Numeric

- ***pLocation***

  Location ID (eg 1)
  Data type is Numeric

- *pRef*

  Transaction Reference
  Data type is AlphaNumeric

- ***pPCode***

  Product Code
  Data type is AlphaNumeric

- **pQTY**

  Quantity
  Data type is Numeric

- ***pRate***

In the case of Write Off and Receipt this amount should be GST Exclusive. In the case of Manual Sale this amount should be either GST Inclusive or GST Exclusive depending on the Infusion Product Settings.
Data type is Numeric

- pRef

  Transaction Reference
  Data type is AlphaNumeric

- *pSerial*

  Serial Number. Required if the Product code is set to track serial numbers.
  Data type is AlphaNumeric

- **pTrType**

  Transaction Type (case sensitive)
  R – Receipt
  M – Manual Sale
  W – Write Off
  Data type is AlphaNumeric

SUPP_**UPDATEINVHEADER – Parameters Explained**

The following parameters are used in the procedure SUPP_UPDATEINVHEADER

You must populate the parameter **pLink** with the link number of the Supplier Invoice you wish to update. Remember this can only be a pending invoice.

This list contains the parameter to pass and the matching variable which should be populated with the value you wish to replace.

Parameter	Variable to be populated.	Field Replaced
ACCOUNT	pAccount	Supplier Account Number
ADD1	pAdd1	Address Line 1
ADD2	pAdd2	Address Line 2
ADD3	pAdd3	Address Line 3
ADD4	pAdd4	Address Line 4
DATE	pDate	Invoice Date
DUEDATE	pDueDate	Invoice Due Date
ENTEREDBY	pEnteredBy	Entered by
FREIGHT	pFreight	Freight (Invoice footer)
HDRDATE1	pHdrDate1	Custom Fields – Date 1
HDRDATE2	pHdrDate2	Custom Fields – Date 2
HDRUSER1	pHdrUser1	Custom Fields – User 1
HDRUSER2	pHdrUser2	Custom Fields – User 2
HDRUSER3	pHdrUser3	Custom Fields – User 3
HDRUSER4	pHdrUser4	Custom Fields – User 4
HDRUSER5	pHdrUser5	Custom Fields – User 5
HDRUSER6	pHdrUser6	Custom Fields – User 6
INVORNUM	pInvOrdnum	Order Number
NAME	pName	Supplier Name
NOTES	pNotes	Notes
POSTCODE	pPostCode	Post Code
REF	pRef	Reference
ROUNDING	pRounding	Rounding (Invoice footer)
INVNUMBER	pInvNumber	Invoice Number

SUPP_CREATEINVOICE – Parameters Explained

The following parameters are used in the procedure SUPP_CREATEINVOICE. Required Parameters are underlined.

- **pAccount**

  Supplier Account the invoice is to be charged to.
  Data Type is Numeric

- *pAdd1, pAdd2, pAdd3, pAdd4, pPostCode*

  Supplier Address Details
  If pAdd1 is left blank then the all address values will be obtained from the Suppliers Account within Infusion

  Data Type is AlphaNumeric

- **pDate**

  Date of the Transaction
  Data Type is Date

- *pEnteredBy*

  Staff code of the person who entered the Invoice
  Data Type is AlphaNumeric

- *pFreight*

  Freight amount for the invoice footer. (Inclusive or exclusive of GST depending on your Product Price Settings)
  Data Type is Numeric

- *pHdrUser1, pHdrUser2, pHdrUser3, pHdrUser4, pHdrUser5, pHdrUser6*

  Supplier Invoice User Fields User 1 to User 6
  Data Type is AlphaNumeric

- *pHdrDate1, pHdrDate2*

  Supplier Invoice User Fields User Dates 1 and 2
  Data Type is Date

- pGSTInclFlag

  Pass Y if the Invoice line items are GST Inclusive, pass any other string value for GST Exclusive
  Data Type is AlphaNumeric

- **pInvNumber**

  Invoice Number
  Data Type is AlphaNumeric

- *pInvOrdnum*

  Purchase Order Number
  Data Type is AlphaNumeric

- *pName*

  Supplier Name
  If left blank then the Name will be obtained from the Suppliers Account within Infusion

  Data Type is AlphaNumeric

- *pRounding*

  Rounding amount for the invoice footer.
  Data Type is Numeric

SUPP_INVOICEADDLINE – Parameters Explained

The following parameters are used in the procedure SUPP_INVOICEADDLINE. Required Parameters are underlined.

- ***pLink***

  Invoice Link Number – This must be supplied and must relate to a Pending Supplier Invoice.
  Data Type is Numeric

- ***pGLID***

  GL ID from the General Ledger.

  You can pass "/N" as the value and a Notes Line will be created. The text for this should be placed in the pNotes parameter. Any dollar or qty values passed when using "/N" will be ignored.

  Data Type is AlphaNumeric

- *pDesc*

  Description Text
  If left blank the GL Name value will be used

  Data Type is AlphaNumeric

- *pDiscount*

  Discount Percentage

  Data Type is Numeric

- *pNotes*

  Notes Text
  Data Type is AlphaNumeric

- *pQty*

  Quantity

  Data Type is Numeric

- *pRate*

  Unit Rate (Including or Excluding GST based on the pGSTInclFlag value which was set when the Invoice Header was created)

  Data Type is Numeric

- *pUSER1 (pUSER2)*

  Custom Fields for the Purchase Order Line.

Data Type is AlphaNumeric

SUPP_INVOICETOTAL – Parameters Explained

You must populate the parameter **pLink** with the link number of the invoice you wish to update.

Remember this can only be a pending invoice.

CREATEPURCHASEORDER – Parameters Explained

The following parameters are used in the procedure CREATEPURCHASEORDER. Required Parameters are underlined.

- **pAccount**

  Supplier Account the Purchase Order is for.
  Data Type is Numeric

- *pAdd1, pAdd2, pAdd3, pAdd4, pPostCode*

  Supplier Address Details
  If pAdd1 is left blank then the all address values will be obtained from the Suppliers Account within Infusion

  Data Type is AlphaNumeric

- *pDAdd1, pDAdd2, pDAdd3, pDAdd4, pDPostCode*

  Delivery Address Details
  If pDAdd1 is left blank then the all address values will be obtained from the Delivery Locations Details within Infusion

  Data Type is AlphaNumeric

- *pDName*

  Delivery Location Name
  If left blank then the Name will be obtained from the Delivery Locations Details within Infusion

  Data Type is AlphaNumeric

- **pDate**

  Date of the Transaction
  Data Type is Date

- **pDDate**

  Required Delivery Date
  Data Type is Date

- *pEnteredBy*

  Staff code of the person who entered the Invoice
  Data Type is AlphaNumeric

- *pFreight*

  Freight amount for the Purchase Order footer. (Inclusive or exclusive of GST depending on your Product Price Settings)
  Data Type is Numeric

- *pHdrUser1, pHdrUser2, pHdrUser3, pHdrUser4, pHdrUser5, pHdrUser6*

  Purchase Order User Fields User 1 to User 6
  Data Type is AlphaNumeric

- *pHdrDate1, pHdrDate2*

  Purchase Order User Fields User Dates 1 and 2
  Data Type is Date

- ***pLocation***

  Purchase Order Location
  Data Type is Numeric

- **pPOOrdnum**

  Purchase Order Number
  Data Type is AlphaNumeric

- **pPOOrdType**

  Purchase Order Type

  - 1 = Standard Order
  - 2 = Multi Location Order
  - 3 = Customer Special Order
  - 4 = Job Management Order

  If you are creating a Customer Special Order, then the additional fields for the Customer Details etc need to be added to the Purchase Order either manually (in the Infusion Interface) or by using the Replace commands after creating the Purchase Order shell. Details of what is required for this can be obtained from Infusion Support.

- *pName*

  Supplier Name
  If left blank then the Name will be obtained from the Suppliers Account within Infusion

  Data Type is AlphaNumeric

- *pRounding*

  Rounding amount for the invoice footer.
  Data Type is Numeric

PURCHASEORDERADDLINE – Parameters Explained

The following parameters are used in the procedure PURCHASEORDERADDLINE. Required Parameters are underlined.

- ***pLink***

  Purchase Order Link Number – This must be supplied and must relate to a Saved Purchase Order.

  Data Type is Numeric

- ***pPCode***

  Product Code from the Products table.

  Data Type is AlphaNumeric

- *pAutoPrice*

  If this value is set to anything other than a blank string, the DLL will obtain the cost price rate from Infusion using the Infusion Pricing Logic.

  Data Type is AlphaNumeric

- *pDesc*

  Description Text
  If left blank the Products Description value will be used

  Data Type is AlphaNumeric

- *pDiscount*

  Discount Percentage

  Data Type is Numeric

- *pQty*

  Quantity Ordered

  Data Type is Numeric

- *pCostRate*

  Unit Rate (Excluding GST)

  Data Type is Numeric

- *pDiscount*

  Discount Percentage

  Data Type is Numeric

- *pLocation*

  Location Number for Multi Location Orders Only

  Data Type is Numeric

- *pPOJobID*

  Job ID for Job Management Orders Only

  Data Type is AlphaNumeric

- *pUSER1 (pUSER2)*

  Custom Fields for the Purchase Order Line.

  Data Type is AlphaNumeric

TOTALPURCHASEORDER – Parameters Explained

Populate the parameter **pLink** with the link number of the Purchase Order to be totaled.

This can only be a Pending Purchase Order (Saved or Ordered).

UPDATEPOHEADER – Parameters Explained

The following parameters are used in the procedure UPDATEINVHEADER

Populate the parameter **pLink** with the link number of the Purchase Order to be updated. Only Pending Purchase Orders (Ordered or Saved) can be updated.

This list contains the parameter to pass and the matching variable which should be populated with the value you wish to replace.

Parameter	Variable to be populated.	Field Replaced
ADD1	pAdd1	Address Line 1
ADD2	pAdd2	Address Line 2
ADD3	pAdd3	Address Line 3
ADD4	pAdd4	Address Line 4
DADD1	pDAdd1	Delivery Address Line 1
DADD2	pDAdd2	Delivery Address Line 2
DADD3	pDAdd3	Delivery Address Line 3
DADD4	pDAdd4	Delivery Address Line 4
DATE	pDate	Invoice Date
DPOSTCODE	pDPostCode	Delivery Post Code
DNAME	pDName	Delivery Name
ENTEREDBY	pEnteredBy	Entered by Staff Member
FREIGHT	pFreight	Freight (Invoice footer)
HDRDATE1	pHdrDate1	Custom Fields – Date 1
HDRDATE2	pHdrDate2	Custom Fields – Date 2
HDRUSER1	pHdrUser1	Custom Fields – User 1

HDRUSER2	pHdrUser2	Custom Fields – User 2
HDRUSER3	pHdrUser3	Custom Fields – User 3
HDRUSER4	pHdrUser4	Custom Fields – User 4
HDRUSER5	pHdrUser5	Custom Fields – User 5
HDRUSER6	pHdrUser6	Custom Fields – User 6
NAME	pName	Customer Name
NOTES	pNotes	Notes
ORDEREDBY	pRep	Ordered by Staff Member
POSTCODE	pPostCode	Post Code
REQDATE	pDDate	Required Date

Sample code based on VB.NET 2010

Creating a DLL Instance & Logging Into Infusion

Add theDLL as a Reference (COM)
Add the necessary IMPORTS line to the start of the form or method

Create an object reference as follows:

```
Dim gw As New ibsGateway.7.0.14_Gateway
```

Set the path to the Infusion Data and Login as follows:

```
gw.setpath("c:\infusion\data")
gw.login("Your Program Name")
```

Logging Out of Infusion

```
gw.Logout()
```

Destroying a DLL Instance (Release)

```
gw.Destroy()
```

Obtaining a list of all Customers

The DLL does not return a single list of all customers (or other list). You must open the relevant database, iterate through it retrieving the fields you desire, store them locally and then close the database when finished. An example is below.

```
Dim lnRecCount As Integer
  Dim lnAccount As Integer
  Dim lcName As String
  Dim lcAdd1 As String

  gw.open("Customer", "a_customer_get")
  gw.select("a_customer_get")
  lnRecCount = gw.reccount()
  gw.GoTop()
  For nCount = 1 To lnRecCount

      lnAccount = gw.get("Account")
      lcName = gw.get("Name")
      lcAdd1 = gw.get("Add1")
      gw.Skip("a_customer_get")
```

```
Next

gw.close("a_customer_get")
```

The same logic should be used for all lists, Customers, Products, Suppliers, GL Codes etc.

Product Prices

The product prices in Infusion are stored either GST Inclusive or Exclusive based on a user setting. This setting can be retrieved as follows:

```
gw.open("infdata", "a_infdata_get")
gw.select("a_infdata_get")
llGSTSetting = gw.get("stkincl")
gw.close("a_infdata_get")
```

The value llGSTSetting with be a logical value (true or false). Depending on your system, this may be returned as a 1 or a 0 also.

There are 8 standard price levels in Infusion, these are stored in the products master table. (PRODUCTS.DBF). The fields are SELL1, SELL2 etc.

A customer will be assigned to one of these price levels, but is not limited to using that level. The field which contains this setting is CUSTOMER.PRICELVL. This is a string field, and as long as the client is not using additional price levels, the values will be 1,2,3 etc in the first position. These simply map then to the SELL1, SELL2 fields.

Infusion does allow for Quantity Price Breaks. This is where when a customer purchases over a given quantity the price changes. There can be three breaks per product (per price level). You will know these are active by querying the QB1, QB2 etc field. One for each price level. The return value is Logical (same as the GST Inclusive Setting above)

If there are quantity breaks, then this will return True.

There are a set of 6 fields for each price level.

QB1Q1 QB1P1 The quantity required to get the price is stored in Q1 and its UNIT price in P1
QB1Q2 QB1P2
QB1Q3 QB1P3

Infusion also allows for a PROMO Price to be set. If there are Quantity Breaks active, then Promo Prices are NOT available for the given product / price level.

PROMO1 The per unit price
FROM1 The start date of the promo price. (blank, no start date)
TO1 The end date of the promo price. (blank, no end date)

Infusion also allows for Customer specific pricing (Contract Rates) per product. This takes priority over ALL other pricing structures.

There is also a discount matrix and default discount value which may be applied to the customers price.

The full pricing structure is quite complex and is not fully explained above. Currently the DLL has a Rate Calculation function built into the Invoice creation functions, but none to just give you the price for the customer on the fly.

GST Codes

The GST Code for the selling Rate is stored on each product in the GSTCAT field. It relates to a record in the GSTCODES.DBF. The RATE field is the rate which you should apply.

If a customer is International, then the flag in CUSTOMER.GSTZERO will be True, in which case the GSTCODES.CODE of "Z" should always be used.

GL Codes

GL IDs are stored on a three tier basis. PRODUCT By LOCATION, LOCATION, PRODUCT.

Checking PRODBALS.DBF (CODE = xxxxx and LOCATION = nn) for a value in SALES. If this is empty then step to lower level
Checking LOCATIONS.DBF (LOCNUM = nn) for a value in SALES. If this is empty then step to lower level
Checking PRODUCTS.DBF (CODE = xxxxx) for a value in SELLGL. (This should not be empty)

Updating a Customer

```
Dim lnRetVal As Integer
Dim lnAccount As Integer
Dim lcAdd1 As String

lnAccount = 2
lcAdd1 = "My New Address"

gw.open("Customer", "a_customer_update")
gw.select("a_customer_update")
lnRetVal = gw.locate("account=" + lnAccount.ToString)

If lnRetVal = 1 Then
    ' Found
    If gw.reclock() = 1 Then

        ' Yes it is locked (and yes the zero return value is correct)

        gw.replace("Name", "My New Name")
        gw.replace("Add1", lcAdd1)

        gw.recunlock() ' Dont forget to unlock the record.

    End If

Else
    ' Not Found
End If

gw.close("a_customer_update")
```

Adding a Customer

There is a function for adding a Customer. You can either pass it an account number or obtain one from the system.

```
' Pass Account Value (User assigned)

Dim lnRetVal As Integer

lnRetVal = gw.CreateCust(1234)
If lnRetVal = 1 Then
    ' Created
    gw.open("customer", "a_customer_add")
    gw.locate("account=1234")
    If gw.reclock() = 1 Then

        'record locked ready to update

        gw.replace("Name", "My Name")

        gw.recunlock()

    Else

        ' not likley to happen

    End If

    gw.close("a_customer_add")

Else

    ' Account Number already exists or some other error, account not created.
End If
```

You can get the next system generated number by calling the following function. Then use it in the example above in place of the 1234

```
lnNewAccount = gw.nextcust(0)
```

Any value other than zero should be treated as the next account number. Zero indicates that auto numbering is not active or some other access issue.

Create an Invoice

```
Dim nInvNum As Integer

 Dim dDate As Date

 dDate = New DateTime(2010, 10, 1).Date

 gw.setpath("c:\infusion\data")
 gw.PINVNUMBER = gw.nextinv(0).ToString.Trim
 gw.PACCOUNT = 1234
 gw.PDATE = dDate
 gw.PLOCATION = 1

 nInvNum = gw.PINVNUMBER

 Dim nInvLink As Integer

 nInvLink = gw.CreateInvoice()

 If nInvLink <> 0 Then

     gw.varreset()
     gw.PLINK = nInvLink
     gw.PAUTOPRICE = "Y"
     gw.PPCODE = “ABC”
     gw.PQTY = 2
     gw.InvoiceAddLine()

     gw.varreset()
     gw.PLINK = nInvLink
     gw.PAUTOPRICE = "Y"
     gw.PPCODE = “DEF”
     gw.PQTY = 3
     gw.InvoiceAddLine()

     gw.varreset()
     gw.PLINK = nInvLink
     gw.PAUTOPRICE = "Y"
     gw.PPCODE = “AAA”
     gw.PQTY = 4
     gw.InvoiceAddLine()

 End If

 gw.TotalInvoice()
```