MethodAPI Documentation 1.0

User Guide

Contents

Overview

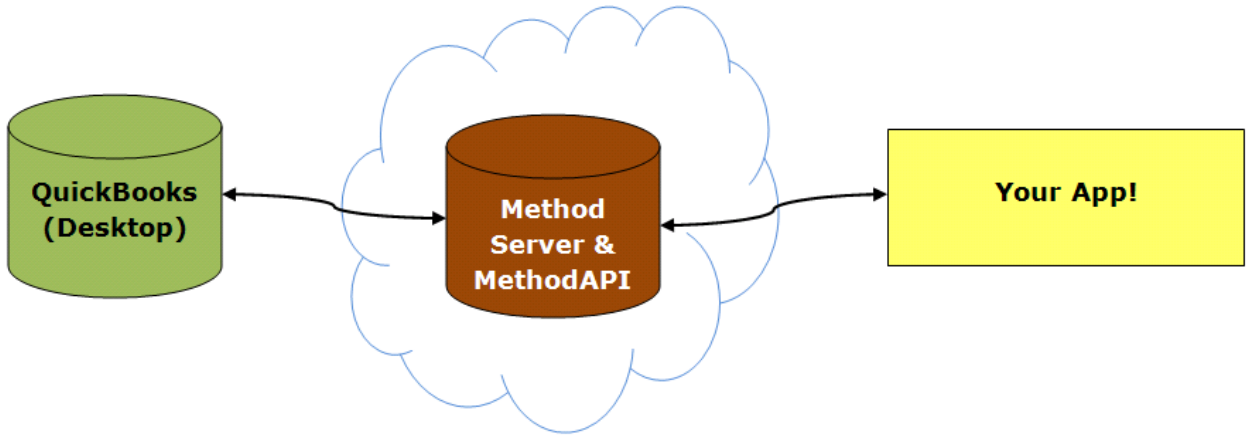
The MethodAPI is an extremely easy interface for accessing, adding and editing account data for a Method Integration account. It's a simple yet powerful way for programmers to integrate their web and desktop applications with QuickBooks and Method data.

Sample potential uses:

- *Importing tools* that take data from another program, like Excel, and send it to Method (and therefore QuickBooks).
- *Exporting tools* that take data from Method and place it in another application like Excel or Access.
- *Advanced tools for running complex functions* on Method and QuickBooks data and writing the results back to Method.
- *3rd party SAAS (Software As A Service) applications* that want to interact with their user's QuickBooks data using simple webservice calls.
- *Web developers* who want their user's web site to interact with their Method and QuickBooks data as an alternative to the built-in Method Third Party Portals.

Diagram

This is a simplified flow chart. Here your app uses the MethodAPI to read from and write to the Method Web Server. In turn, the Method Web Server synchronizes its data with QuickBooks on the desktop.



FAQ

Q. Do I need to use the MethodAPI to develop Method apps?

No. You do not need to use the MethodAPI to build Method apps. Furthermore, you do not even need to be a programmer to build Method apps. Method is first and foremost a web-based platform that allows non-programmers to drag and drop their way to creating their own rich and powerful business applications. With that said, the MethodAPI is available to those who are actually programmers that want to either add additional functionality not found within Method, or want to use Method as a syncing tool for their own web applications.

Q. Do my users also have to be users of Method?

Yes. They need to sign up for a Method account. Currently, the lowest level of Method starts at $25 per month, however, starting in September 2008 there is a promotion to create a 90 day unlimited Method account for only $5. If you plan to sign up several of your own users, consider becoming a Value Added Reseller (www.methodintegration.com/web/consultants.aspx). Based on how your app plans on using Method, volume and restrictive license pricing may be available.

Q. Where is the data stored on the web, and how does it get to QuickBooks?

Method takes data from your user's QuickBooks database and maintains a replicated copy of that database on our web server (hosted by Rackspace). A real-time synchronization is then kept so that any changes to QuickBooks are updated in Method, and any changes in Method are updated back in QuickBooks. If QuickBooks happens to be closed or disconnected from the internet any changes in either database are synchronized when it becomes online again – so therefore you can read and write to Method even when QuickBooks is closed.

Q. I'm a developer, how do I get a "developer account"?

There are currently no "developer" accounts of Method. Simply sign up for Method like a regular user at https://www.methodintegration.com/web/SignUp.aspx.

Q. How do I get help with questions, problems or suggestions?

Method is all about community. So just go to http://methodintegration.com/cs/forums/13.aspx and post a question on the forum. Questions are answered here by The Method Team, and by other Method users and developers.

Data Library

How do I get a list of tables and fields available through the MethodAPI?

There are two API calls you can make that give you a list of the tables and fields available to the API. *MethodAPITableList* is an API call you can make to get a list of all tables in Method, and whether the tables support additions and modifications. *MethodAPIFieldList* is an API call that gives you all the fields within a specified table, the data type and size of the field, as well as whether it is required, and whether it can be written to during an addition or modification.

How do I get more information about the meaning of these fields and tables?

The fields and tables from Method that come from QuickBooks conform to the QuickBooks SDK. You can visit http://developer.intuit.com/qbsdk-current/Common/newOSR/index.html and go through the OnScreen Reference guide to find out what is supported from each version of QuickBooks. Note that the following SDK versions correspond to these versions of QuickBooks:

SDK version 7.0 = QB 2008 US

SDK version 6.0 = QB 2007 US, QB 2008 UK/CA/AU

SDK version 5.0 = QB 2006 US

SDK version 4.0 = QB 2004 US

SDK version 3.0 = QB 2004 US

What the MethodAPI doesn't have....yet

The API is currently setup to access the data of a user who has a Method account. If you are planning on just using Method's syncing engine so that your application can integrate with QuickBooks, you'll find it has a few limitations, as the user will still need to sign into their Method account to perform certain functions. The following are API calls we plan to add in the future, but are not currently available.

- Access to the Audit Trail
- Access to resolve conflicts
- Ability to create new users
- Ability to create new fields and tables
- Ability to set account and syncing preferences

Authentication

In order for you to use the MethodAPI, the following is required:

- You, as a developer, must read and agree to http://www.methodintegration.com/web/terms-of-service.aspx before developing an application. If you do not agree to these terms, do not use the MethodAPI.
- Your customer must have signed up for a Method account, and synchronized with QuickBooks. They can sign up at: https://www.methodintegration.com/web/SignUp.aspx.
- Your customer must present you with the "Company Account", "User Name" and "Password" of their registered Method account.
- The User Name must have MethodAPI enabled. To do this, have the user go to Customize > Users, click Edit beside the User Name, advance to Step 5, and ensure that 'This user is allowed to connect to Method API' is selected.
- The User Name must have the appropriate table permissions. Under Step 5 of editing the settings for the User Name, your customer can select which database tables this User Name has permissions on. The

User Name should have either "This user is allowed access to all existing tables" selected, or if "This user only has access to specific existing tables" is selected, your customer should carefully ensure that all appropriate tables are granted permission.

Sample Apps

Yes, there are sample applications – which, for most developers, are far more valuable that the document you are reading!

To work with the **VBA (Visual Basic for Applications) Excel** Sample apps for Excel:

1. Download the file: http://www.methodintegration.com/documentation/MethodAPISample_VBA.zip
2. Unzip the file to any folder on your computer.
3. Open either MethodAPISample_GetInvoices.xls for an example on how to list outstanding invoices from QuickBooks, or MethodAPISample_GetTablesFields.xls for an example on how to get a list of available tables and fields.
4. Make sure you pay close attention to the instructions on the top of the Excel spread sheet.

To work with the **ASP.NET** Sample apps for web development:

1. Download the file: http://www.methodintegration.com/documentation/MethodAPISample_ASP.zip
2. Unzip the file to a new, empty folder on your computer.
3. Open Visual Studio 2005, or greater.
4. Click File > Open Web Site, find the folder you unzipped to, and click Open.
5. Click Debug > Start Debugging, to launch the website in a browser.

MethodAPITableList

Overview

This is a simple API call that gives a list of all the database tables available to you. Since Method users can create their own tables, there is no finite list. And since different versions of QuickBooks have different levels of access, the list of available tables and the addition & modification rights will vary.

Web Service URL

https://www.methodintegration.com/MethodAPI/service.asmx

Parameters

1. strCompanyAccount – The "Company Account" name of the registered Method Account.
2. strLogin – The "User Name" value of the registered Method Account.
3. strPassword – The "Password" value of the registered Method Account.
4. strReturnedXml (ByRef) – A local string variable passed, ByRef, to the web service to hold the returned XML values.

Response

- If the account passes verification, and a successful response is generated, the returned value will be "Success".
- If the request failed, the response will be an error message.
- Note: the actual list of tables returned is XML passed to your strReturnedXml local string variable.

VBA Code Example

Check out the full VBA Code example for "Get Table List". The main portion of the example, though, is:

```
'Point the SOAP API to the web service that we want to call...
Set objSclient = New SoapClient30
Call objSclient.mssoapinit(par_WSDLFile:="https://www.methodintegration.com/MethodAPI/service.asmx?wsdl")

'Call the web service to get list of tables available
sResult = objSclient.MethodAPITableList(sCompanyAccount, sUserName, sPassword, sXML)
Set objSclient = Nothing
```

Which returns the following to the local sXML variable:

```
<Record>
<TableName>Account</TableName>
<SupportsAdd>True</SupportsAdd>
<SupportsEdit>True</SupportsEdit>
</Record>
<Record>
<TableName>AccountAccountType</TableName>
<SupportsAdd>False</SupportsAdd>
<SupportsEdit>False</SupportsEdit>
</Record>
...
```

MethodAPIFieldList

Overview

This is a simple API call that gives a list of all the fields available for a specified table.

Web Service URL

https://www.methodintegration.com/MethodAPI/service.asmx

Parameters

1. strCompanyAccount – The "Company Account" name of the registered Method Account.
2. strLogin – The "User Name" value of the registered Method Account.
3. strPassword – The "Password" value of the registered Method Account.
4. strReturnedXml (ByRef) – A local string variable passed, ByRef, to the web service to hold the returned XML values.
5. strTable – The name of the table you would like to get a list of fields from.

Response

- If the account passes verification, and a successful response is generated, the returned value will be "Success".
- If the request failed, the response will be an error message.

- Note: the actual list of fields returned is XML passed to your strReturnedXml local string variable.

VBA Code Example

Check out the full VBA Code example for "Get Field List". The main portion of the example, though, is:

```
'Point the SOAP API to the web service that we want to call...
Set objSclient = New SoapClient30
Call objSclient.mssoapinit(par_WSDLFile:="https://www.methodintegration.com/MethodAPI/service.asmx?wsdl")

'Call the web service to get list of fields available
Dim sTableName As String
sTableName = InputBox("Please type out the name of the table you want to get fields from.", "Enter Table Name")
sResult = objSclient.MethodAPITableList(sCompanyAccount, sUserName, sPassword, sXML, sTableName)
Set objSclient = Nothing
```

Which returns the following to the local sXML variable:

```
<Record>
        <FieldName>AccountNumber</FieldName>
        <SupportsAdd>True</SupportsAdd>
        <SupportsEdit>True</SupportsEdit>
        <IsRequired>False</IsRequired>
        <MaxSize>50</MaxSize>
        <DataType>Text</DataType>
</Record>
<Record>
        <FieldName>AccountType</FieldName>
        <SupportsAdd>True</SupportsAdd>
        <SupportsEdit>True</SupportsEdit>
        <IsRequired>True</IsRequired>
        <MaxSize>50</MaxSize>
        <DataType>Dropdown</DataType>
</Record>
...
```

MethodAPISelect_DataSet

Overview

This API call provides a dataset containing specified fields and records from a named table.

Web Service URL

https://www.methodintegration.com/MethodAPI/service.asmx

Parameters

1. strCompanyAccount – The "Company Account" name of the registered Method Account.

2. strLogin – The "User Name" value of the registered Method Account.

3. strPassword – The "Password" value of the registered Method Account.

4. datReturnedDataSet (ByRef) – A local dataset variable passed, ByRef, to the web service to capture the returned dataset.

5. strTable – The name of the table you would like to get records from.

6. strFields – A list of fields you wish to return in the dataset, separated by commas. It is important to specify only the fields you need, as it will affect how long it takes to download the dataset from the server. For example, say you want to get a list of customers' phone numbers and names, your strFields value could be "Name,Phone,AltPhone". Note that you really are writing SQL, so you could also use an alias "Name,Phone as MainPhone,AltPhone", or if you wanted a total of PurchaseOrderLine Amounts, by Item, you could use a summary "Sum(Amount) as SumOfAmount,Item", or whatever SQL you like.

7. strWhereClause – Specify the WHERE statement you would like to use, if any. Do your best to limit the dataset to only the records you need so that you reduce download time from the server. For example, to return a list of customer's whose balance is greater than 0, specify a strWhereClause of "Balance > 0". Or if you want to return a list of invoices where the TxnDate is equal to December 25th, 2008 and the BalanceRemaining is greater than 0 and the Class is 'Commercial', you would write: " (TxnDate='2008-12-25') AND (BalanceRemaining > 0) AND (Class='Commercial')".
   Note: be careful when using single quotes (') in the strWhereClause parameter. SQL syntax requires that single quotes are replaced two single quotes (''). So instead of writing "Customer='O'Neil' and Balance=0" write "Customer='O''Neil' and Balance=0"

8. strGroupByClause – Specify the GroupBy, if any. You would typically only be doing this in conjunction with a Sum field in your strFields. So if you wanted a total of all Invoices, by customer, your strFields would be "Sum(Amount) as SumOfAmount,Customer" and the strGroupByClause would be "Customer".

9. strHaving – Specify the Having, if any. For example, if you had a strGroupByClause value of "Customer", you might have a value of "Customer<>'Bob Crenshaw'" for strHaving.

10. strOrderBy – Specify the Order By, if any. For example, if you are getting a list of your customers' phone numbers, you might order your dataset by state, and then by phone number, so strOrderBy would be "BillAddressState, Phone". If you wanted to order by state in "Descending" order (Z to A), your strOrderBy would be "BillAddressState DESC, Phone".

Response

- If the account passes verification, and a successful response is generated, the returned value will be "Success".
- If the request failed, the response will be an error message.
- Note: the actual dataset returned is passed to your datReturnedDataSet local dataset variable.

ASP.NET Code Example

Check out the full ASP.NET Code example, for Visual Studio 2005, for "Display DataSet Results". The main portion of the example, though, is:

```
'Write my SQL statement to get any outstanding invoices
Dim sSelectFields As String
Dim sSelectFrom As String
Dim sSelectWhere As String
Dim sSelectGroupBy As String
Dim sSelectHaving As String
Dim sSelectOrderBy As String
sSelectFields = Me.txtSelectFields.Text
sSelectFrom = Me.txtSelectTable.Text
sSelectWhere = Me.txtSelectWhere.Text
sSelectGroupBy = Me.txtSelectGroupBy.Text
sSelectHaving = Me.txtSelectHaving.Text
sSelectOrderBy = Me.txtSelectOrderBy.Text

'Call the MethodAPI to get the dataset
sResult = wbsMethodAPI.MethodAPISelect_DataSet(sCompanyAccount, sUserName, sPassword, _
datResponse, sSelectFrom, sSelectFields, sSelectWhere, sSelectGroupBy, sSelectHaving, sSelectOrderBy)
wbsMethodAPI = Nothing
If sResult.Trim.ToUpper <> "Success".ToUpper Then
  'failed, give a message
  Me.divResponse.InnerHtml = sResult
  Exit Sub
End If
```

This returns a dataset to the local datResponse variable, containing a list of the specified fields and records to the screen.

MethodAPISelect_XML

Overview

This API call provides an XML response containing specified fields and records from a named table. This is the same as MethodAPISelect_DataSet, except that XML is returned instead of a dataset.

Web Service URL

https://www.methodintegration.com/MethodAPI/service.asmx

Parameters

The parameters are the same as MethodAPISelect_Dataset, except that the fourth parameter is strReturnedXml (ByRef), instead of datReturnedDataSet (ByRef).

Response

- If the account passes verification, and a successful response is generated, the returned value will be "Success".
- If the request failed, the response will be an error message.
- Note: the actual XML returned is passed to your strReturnedXml dataset variable.

VBA Code Example

Check out the full VBA Code example, List OverDue Invoices, which grabs all outstanding invoices from a Method account and displays them in Excel. The main portion of the example, though, is:

```
'Point the SOAP API to the web service that we want to call...
Set objSclient = New SoapClient30
Call objSclient.mssoapinit(par_WSDLFile:="https://www.methodintegration.com/MethodAPI/service.asmx?wsdl")

'Call the MethodAPI web service
sResult = objSclient.MethodAPISelect_XML(sCompanyAccount, sUserName, sPassword, sXML, sSelectFrom, _
sSelectFields, sSelectWhere, sSelectGroupBy, sSelectHaving, sSelectOrderBy)
Set objSclient = Nothing

If UCase(sResult) <> "SUCCESS" Then
    'failed, give a message
    MsgBox sResult, vbExclamation, "Error returned from MethodAPI"
    Exit Sub
End If
```

Which returns the following to the local sXML variable:

```
<Record>
        <TxnDate>6/24/2008 12:00:00 AM</TxnDate>
        <Subtotal>44.0000</Subtotal>
        <RefNumber>33176</RefNumber>
        <Customer>Bob Crenshaw</Customer>
        <BalanceRemaining>44.0000</BalanceRemaining>
</Record>
<Record>
        <TxnDate>7/22/2008 12:00:00 AM</TxnDate>
        <Subtotal>188.0000</Subtotal>
        <RefNumber>33976</RefNumber>
        <Customer>Nat Chapman </Customer>
        <BalanceRemaining>50.0000</BalanceRemaining>
</Record>
...
```

MethodAPIUpdate

Overview

This API call updates one or many fields of a specified record in a table with a new value. If the updates are to a QuickBooks table, they will then be sent automatically to QuickBooks.

Web Service URL

https://www.methodintegration.com/MethodAPI/service.asmx

Parameters

1. strCompanyAccount – The "Company Account" name of the registered Method Account.
2. strLogin – The "User Name" value of the registered Method Account.
3. strPassword – The "Password" value of the registered Method Account.
4. strTable – The name of the table you would like to update.
5. arrUpdateFieldsArray – An array containing the names of fields you wish to update.
6. arrUpdateValueArray – An array containing the values of the fields you wish to update. For example, if you wanted to update the Customer

table and update the Name to "BobCrenshaw2" and the Phone to "555-123-456", you would write the following code:
Dim arrUpdateFieldsArray(1) As String
Dim arrUpdateValueArray(1) As String
arrUpdateFieldsArray(0) = "Name"
arrUpdateValueArray(0) = "BobCrenshaw2"
arrUpdateFieldsArray(1) = "Phone"
arrUpdateValueArray(1) = "555-123-456"

7. intRecordID – The RecordID of record you wish to update. Every table has a unique "RecordID" field that is used to identify each record. If you do not know the RecordID of the field, you must search for it using a MethodAPISelect_XML or MethodAPISelect_Dataset call to find it.

Response

- If the account passes verification, and the update successfully completes, the returned value will be either "SucessSendToDesktop" or "Success". If the update satisfies requirements for being ready to be sent to QuickBooks, it will receive "SuccessSendToDesktop", however, if it is not ready (such as when it requires line items to be entered still), it will just response with "Success".

- If the request failed, the response will be an error message.

ASP.NET Code Example

Check out the full ASP.NET Code example, MethodAPIUpdate Sample, which takes values entered from a webpage and sends the update using the MethodAPI. The main portion of the example, though, is:

```
'Get the update table from the screen
Dim sUpdateTable As String
Dim intRecordID As Integer
sUpdateTable = Me.txtUpdateTable.Text
intRecordID = Me.txtRecordID.Text

'build the array of fields to update - in reality, there is no limit to the size of the array
Dim arrUpdateFieldsArray(1) As String
Dim arrUpdateValueArray(1) As String
arrUpdateFieldsArray(0) = Me.txtUpdateField1.Text
arrUpdateValueArray(0) = Me.txtUpdateValue1.Text
arrUpdateFieldsArray(1) = Me.txtUpdateField2.Text
arrUpdateValueArray(1) = Me.txtUpdateValue2.Text

'Call the MethodAPI to update the record
sResult = wbsMethodAPI.MethodAPIUpdate(sCompanyAccount, sUserName, sPassword, _
 sUpdateTable, arrUpdateFieldsArray, arrUpdateValueArray, intRecordID)
wbsMethodAPI = Nothing
```

MethodAPIInsert

Overview

This API call inserts one or many fields into a record of a table. If the insert is into a QuickBooks table, it will then be sent automatically to QuickBooks.

Web Service URL

https://www.methodintegration.com/MethodAPI/service.asmx

Parameters

1. strCompanyAccount – The "Company Account" name of the registered Method Account.
2. strLogin – The "User Name" value of the registered Method Account.
3. strPassword – The "Password" value of the registered Method Account.
4. strTable – The name of the table you would like to insert into.
5. arrInsertFieldsArray – An array containing the names of fields you wish to insert.
6. arrInsertValueArray – An array containing the values of the fields you wish to insert. For example, if you wanted to insert a statement Charge for "Bob Crenshaw", you could write the following code:
   Dim arrInsertFieldsArray (4) As String
   Dim arrInsertValueArray (4) As String
   arrInsertFieldsArray (0) = "Customer"
   arrInsertValueArray (0) = "Bob Crenshaw"
   arrInsertFieldsArray (1) = "ARAccount"
   arrInsertValueArray (1) = "Accounts Receivable"
   arrInsertFieldsArray (2) = "Item"
   arrInsertValueArray (2) = "Service Call"
   arrInsertFieldsArray (3) = "Rate"
   arrInsertValueArray (3) = "100.55"
   arrInsertFieldsArray (4) = "TxnDate"
   arrInsertValueArray (4) = "2008-09-26 5:00:00"

7. intRecordID (ByRef) – An integer variable that is passed, ByRef, to the API call so that it will be assigned the newly generated RecordID of the new record.

Response

- If the account passes verification, and the insert successfully completes, the returned value will be either "SucessSendToDesktop" or "Success". If the insert satisfies requirements for being ready to be sent to QuickBooks, it will receive "SuccessSendToDesktop", however, if it is not ready (such as when it requires line items to be entered still), it will just response with "Success".
- It will return the new RecordID to the local variable assigned to the intRecordID parameter.
- If the request failed, the response will be an error message.

ASP.NET Code Example

Check out the full ASP.NET Code example, MethodAPIInsert Sample, which takes values entered from a webpage and creates a new record using the MethodAPI. The main portion of the example, though, is:

```
'Get the Method Account values from web page
Dim sCompanyAccount As String
Dim sUserName As String
Dim sPassword As String
sCompanyAccount = Me.txtCompanyAccount.Text
sUserName = Me.txtUserName.Text
sPassword = Me.txtPassword.Text

'Get the insert table from the screen
Dim sInsertTable As String
Dim intRecordID As Integer
sInsertTable = Me.txtInsertTable.Text

'build the array of fields to insert - in reality, there is no limit to the size of the array
Dim arrInsertFieldsArray(1) As String
Dim arrInsertValueArray(1) As String
arrInsertFieldsArray(0) = Me.txtInsertField1.Text
arrInsertValueArray(0) = Me.txtInsertValue1.Text
arrInsertFieldsArray(1) = Me.txtInsertField2.Text
arrInsertValueArray(1) = Me.txtInsertValue2.Text

'Call the MethodAPI to insert the records
sResult = wbsMethodAPI.MethodAPIInsert(sCompanyAccount, sUserName, sPassword, _
sInsertTable, arrInsertFieldsArray, arrInsertValueArray, intRecordID)
wbsMethodAPI = Nothing
```

MethodAPIDelete

Overview

This API call marks a single record for deletion, using a specified RecordID. If the deletion is on a QuickBooks record, the request will then be sent to QuickBooks for final approval. If the deletion is on a non-QuickBooks record, the deletion will occur right away.

Web Service URL

https://www.methodintegration.com/MethodAPI/service.asmx

Parameters

1. strCompanyAccount – The "Company Account" name of the registered Method Account.
2. strLogin – The "User Name" value of the registered Method Account.
3. strPassword – The "Password" value of the registered Method Account.
4. strTable – The name of the table you would like to delete from.
5. intRecordID – The RecordID of the record you wish to delete. Every table has a unique "RecordID" field that is used to identify each record. If you do not know the RecordID of the field, you must search for it using a MethodAPISelect_XML or MethodAPISelect_Dataset call to find it.

Response

- If the account passes verification, and the mark for deletion successfully completes, the returned value will be either "SucessSendToDesktop" in the case of a QuickBooks table or "Success" in the case of a non-QuickBooks table.
- If the request failed, the response will be an error message.

ASP.NET Code Example

Check out the full ASP.NET Code example, MethodAPIDelete Sample, which takes values entered from a webpage and sends the delete request using the MethodAPI. The main portion of the example, though, is:

```
'Get the Method Account values from web page
Dim sCompanyAccount As String
Dim sUserName As String
Dim sPassword As String
sCompanyAccount = Me.txtCompanyAccount.Text
sUserName = Me.txtUserName.Text
sPassword = Me.txtPassword.Text

'Get the values from the screen
Dim sDeleteTable As String
Dim intRecordID As Integer
sDeleteTable = Me.txtDeleteTable.Text
intRecordID = Me.txtRecordID.Text

'Call the MethodAPI to delete the record
sResult = wbsMethodAPI.MethodAPIDelete(sCompanyAccount, sUserName, sPassword, _
 sDeleteTable, intRecordID)
wbsMethodAPI = Nothing
```

MethodAPIActionSendToDesktop

Overview

This API call simulates the action found within the Method web platform called "Send To Desktop". Normally, it is not necessary for you to use this API call, as most API calls to Update, Delete or Insert will automatically send the changes to QuickBooks. However, in some circumstances it does not, especially on transactions (for example Invoices) that require line items (such as InvoiceLine). In those scenarios, you would first insert the transaction, then insert the line items, and then finally call SendToDesktop to let Method know that you are finished adding line items and it is ready to go to QuickBooks.

Web Service URL

https://www.methodintegration.com/MethodAPI/service.asmx

Parameters

1. strCompanyAccount – The "Company Account" name of the registered Method Account.
2. strLogin – The "User Name" value of the registered Method Account.
3. strPassword – The "Password" value of the registered Method Account.

4. strTable – The name of the table you would like to update.

5. intRecordID – The RecordID of the record you wish to update. Every table has a unique "RecordID" field that is used to identify each record. If you do not know the RecordID of the field, you must search for it using a MethodAPISelect_XML or MethodAPISelect_Dataset call to find it.

Response

- If the account passes verification, the returned value will be either "SucessSendToDesktop" or "Success". If the record satisfies requirements for being ready to be sent to QuickBooks, it will receive "SuccessSendToDesktop", however, if it is not ready (such as when it requires line items to be entered still), it will just response with "Success". If the request failed, the response will be an error message.

ASP.NET Code Example

Check out the full ASP.NET Code example, MethodAPIActionSendToDesktop Sample. This is the most advanced of all examples, as it first inserts an invoice, then inserts the invoice lines, then calls SendToDesktop to post the invoice to QuickBooks:

```
'Build array fields to insert into invoices
Dim arrInsertFieldsArray_Invoice(2) As String
Dim arrInsertValueArray_Invoice(2) As String
Dim intInvoiceRecordID As Integer
arrInsertFieldsArray_Invoice(0) = "Customer"
arrInsertValueArray_Invoice(0) = Me.txtCustomer.Text
arrInsertFieldsArray_Invoice(1) = "TxnDate"
arrInsertValueArray_Invoice(1) = CStr(Me.calTxnDate.SelectedDate.Year) & "-" & Format(Me.calTxnDate.Select
arrInsertValueArray_Invoice(1) &= " 12:00:00" 'add universal time of noon.  If you wanted to put in 5:25:3
arrInsertFieldsArray_Invoice(2) = "ARAccount"
arrInsertValueArray_Invoice(2) = Me.txtARAccount.Text

sResult = wbsMethodAPI.MethodAPIInsert(sCompanyAccount, sUserName, sPassword, _
 "Invoice", arrInsertFieldsArray_Invoice, arrInsertValueArray_Invoice, intInvoiceRecordID)
If sResult.Trim.ToUpper <> "Success".ToUpper Then
  'failed, give a message
  wbsMethodAPI = Nothing
  Me.divResponse.InnerHtml = sResult
  Exit Sub
End If

'Build array fields to insert invoice lines into invoices
Dim arrInsertFieldsArray_InvoiceLine(2) As String
Dim arrInsertValueArray_InvoiceLine(2) As String
Dim intInvoiceLineRecordID As Integer
arrInsertFieldsArray_InvoiceLine(0) = "Item"
arrInsertValueArray_InvoiceLine(0) = Me.txtItem.Text
arrInsertFieldsArray_InvoiceLine(1) = "Rate"
arrInsertValueArray_InvoiceLine(1) = Me.txtRate.Text
arrInsertFieldsArray_InvoiceLine(2) = "InvoiceRecordID"
arrInsertValueArray_InvoiceLine(2) = intInvoiceRecordID

'MethodAPI to insert the invoiceline, now that we know the invoice's recordid
sResult = wbsMethodAPI.MethodAPIInsert(sCompanyAccount, sUserName, sPassword, _
 "InvoiceLine", arrInsertFieldsArray_InvoiceLine, arrInsertValueArray_InvoiceLine, intInvoiceLineRecordID)
If sResult.Trim.ToUpper <> "Success".ToUpper Then
  'failed, give a message
  wbsMethodAPI = Nothing
  Me.divResponse.InnerHtml = sResult
  Exit Sub
End If

'MethodAPI to call the invoice to send to QuickBooks.  It wouldn't have gone through to QuickBooks
'when the invoice was first inserted since there weren't any line items.
'this SendToDesktop lets the invoice know that it is finished adding line items and can finally go to QB
sResult = wbsMethodAPI.MethodAPIActionSendToDesktop(sCompanyAccount, sUserName, sPassword, _
 "Invoice", intInvoiceRecordID)
```

MethodAPIActionChargeCreditCard_PsiGate

Overview

This API charges a credit card for merchants who have a merchant account using the PSIGate gateway.

Web Service URL

https://www.methodintegration.com/MethodAPI/service.asmx

Parameters

1. strCompanyAccount – The "Company Account" name of the registered Method Account.
2. strLogin – The "User Name" value of the registered Method Account.
3. strPassword – The "Password" value of the registered Method Account.
4. strGatewayResponse (ByRef) – The variable set to receive response details from the gateway.
5. strGatewayApproved (ByRef) – The variable set to receive "APPROVED" or a failure code.
6. strGateway – Should be set to "PSIGATEWAY".
7. strGatewayUserName – Your merchant account user name.
8. strGatewayPassword – Your merchant account password.
9. strGatewayCustomerName – The first and last name of the customer being charged.
10. strGatewayCustomerAddress – The customer's street address.
11. strGatewayCustomerCity – The customer's city.
12. strGatewayCustomerState – The customer's state or province.
13. strGatewayCustomerZipCode – The customer's zip or postal code.
14. strGatewayCustomerEmail – The customer's email address.

15. strGatewayCustomerCountry – The customer's country.
16. strGatewayCustomerCVV – The 3 or 4 digit security code on the credit card.
17. strGatewayCustomerSubTotal – The subtotal of the shopping cart before tax.
18. strGatewayCustomerTax – The amount of tax to be added to the subtotal.
19. strGatewayCustomerCreditCardNumber – The credit card number on the credit card.
20. strGatewayCustomerExpiryMonth – The expiry month of the credit card.
21. strGatewayCustomerExpiryYear – The expiry year on the credit card.
22. strGatewayCustomerTotal – The total amount to be charge, which must be subtotal plus tax.

Response

- If the account passes verification, and the request is successfully sent to PSIGate the returned value will be "Success". Check strGatewayApproved to see if the request is approved, and check strGatewayResponse for response details.
- If the request failed, the response will be an error message.

MethodAPIActionChargeCreditCard_ElectraCash

Overview

This API charges an electronic check for merchants who have a merchant account with Electracash.

Web Service URL

https://www.methodintegration.com/MethodAPI/service.asmx

Parameters

1. strCompanyAccount – The "Company Account" name of the registered Method Account.
2. strLogin – The "User Name" value of the registered Method Account.
3. strPassword – The "Password" value of the registered Method Account.
4. strGatewayResponse (ByRef) – The variable set to receive response details from the gateway.
5. strGatewayApproved (ByRef) – The variable set to receive "APPROVED" or a failure code.
6. strGateway – Should be set to "ELECTRACASH".
7. strGatewayMerchantID – Your merchant MerchantID with Electracash.
8. strGatewayAuthKey – Your merchant account key, supplied by Electracash.
9. strGatewayCustomerName – The First and Last Name of the customer being charged if a personal account, or the company name if a business account.
10. strGatewayCustomerAddress – The customer's street address.
11. strGatewayCustomerCity – The customer's city.
12. strGatewayCustomerState – The customer's state or province.
13. strGatewayCustomerZipCode – The customer's zip or postal code.

14. strGatewayCustomerEmail – The customer's email address.
15. strGatewayCustomerCountry – The customer's country.
16. strGatewayCustomerSubTotal – The subtotal of the shopping cart before tax.
17. strGatewayCustomerTax – The amount of tax to be added to the subtotal.
18. strGatewayCustomerECAccountNumber – The account number of the customer's bank account.
19. strGatewayCustomerECRoutingNumber – The 9 digit routing number of the customer's bank account.
20. strGatewayCustomerECBankType – Values should be one of "Business Checking", "Business Savings", "Personal Checking" or "Personal Savings".
21. strGatewayCustomerTotal – The total amount to be charge, which must be subtotal plus tax.

Response

- If the account passes verification, and the request is successfully sent to Electracash the returned value will be "Success". Check strGatewayApproved to see if the request is approved, and check strGatewayResponse for response details.
- If the request failed, the response will be an error message.

MethodAPIActionSendEmail

Overview

This API sends an email to the specified recipients using your account with your ISP.

Web Service URL

https://www.methodintegration.com/MethodAPI/service.asmx

Parameters

1. strCompanyAccount – The "Company Account" name of the registered Method Account.
2. strLogin – The "User Name" value of the registered Method Account.
3. strPassword – The "Password" value of the registered Method Account.
4. strToEmail – The email address of the recipient.
5. strFromEmail – The email address of the person sending the email.
6. strFromName – The name of the person sending the email.
7. strCCEmail – The CC (Carbon Copy) address, if you choose to set one.
8. strBCCEmail – The BCC (Blind Carbon Copy) address, if you choose to set one.
9. strSubject – The email subject.
10. strBody – The email body.
11. strPriority – The priority of the email. Values can be "Normal","High" or "Low".
12. strAttachment – The URL of an attachment.
13. strServerAddress – The location of the mail server, for example, mail.ispname.com.
14. strServerPassword – The password of the mail server user account.
15. strServerUserName – The user name of the mail server user account.

Response

- If the account passes verification, and the request is successfully sent to your ISP the returned value will be "Success".
- If the request failed, the response will be an error message.