Method:API Documentation 2.0

User Guide

Contents

Overview

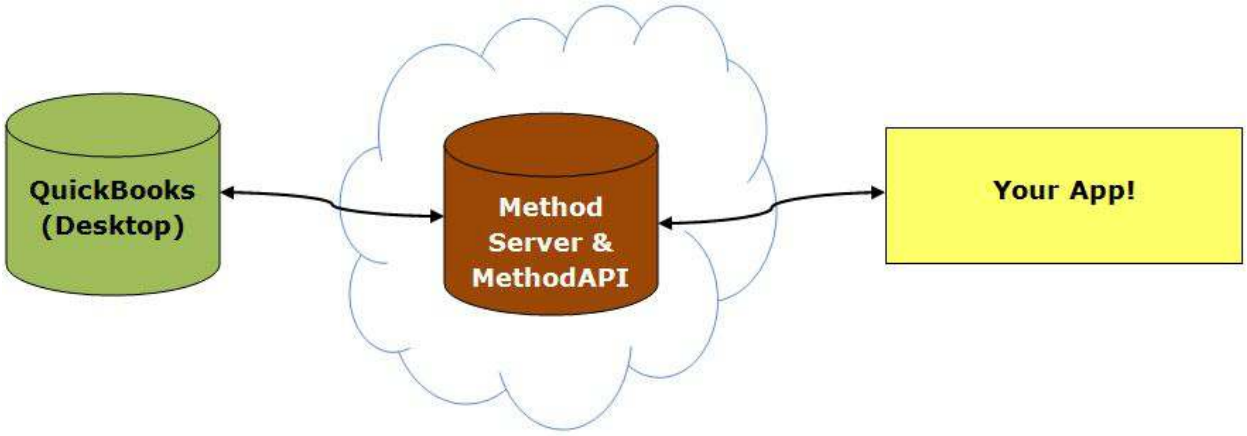
The Method:API is an extremely easy interface for accessing, adding and editing account data for a Method Integration account. It's a simple yet powerful way for programmers to integrate their web and desktop applications with their Method data.

Sample potential uses:

- *Importing tools* that take data from another program, like Excel, and send it to Method.
- *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 web service calls.
- *Web developers* who want their user's website to interact with their Method data as an alternative to the built-in Method Third Party Portals.

Diagram

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



Limits

The Method:API is intended to be a tool to allow external applications to integrate with Method applications, as an extension of the Method user experience. API Limits allow you to integrate your external applications with Method, in such a way that it does not cause performance degradation for other Method users.

Limits are set per account, per user, per min, per hour, per 24 hours and per 30 days. To optimize your usage, we suggest:

1. Removing any unnecessary calls to the API.
2. Ensuring that your code handles API errors and responses, and doesn't re-attempt the same failed calls repeatedly.

FAQ

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

No. The Method web platform is for end-users who develop apps using a drag and drop interface, without coding. The Method:API, on the other hand, is for software programmers who 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, standard pricing for Method starts at $25 per month. Based on how your app plans on using Method, volume and restrictive license pricing may be available. If you plan to sign up several of your own users, consider becoming a Method Solution Provider (www.methodintegration.com/web/consultants.aspx).

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 version of that database on our web server. 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. Go to http://methodintegration.com/cs/forums/15.aspx and post a question on the forum. Questions are answered here by The Method Team, and by other Method users and developers.

Q. Why do the dates and times not match what is seen within Method?

Date and time is stored on the Method server in UTC time. Whenever a date is displayed within the Method interface it always adjusts the date to show in local time.

In the Method:API, however, all dates and times are viewed and written in UTC time. For example if a field has the value of August 20th 10:00 PM in Eastern Time, and you are doing an insert or update, you should adjust the time to August 21st, 3:00 AM, which is 5 hours later in UTC time.

Data Library

How do I get a list of tables and fields available through the Method:API?

There are two API calls you can make that give you a list of the tables and fields available to the API. MethodAPITableListV2 is an API call you can make to get a list of all tables in Method, and whether the tables support additions and modifications. MethodAPIFieldListV2 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 8.0 = QB 2009 US
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 Method:API doesn't have....yet

The API is currently setup to access the data of a user who has a Method account. 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 set account and syncing preferences

Authentication

In order for you to use the Method:API, 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 Method:API.
- 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”. You must also have either their “Password” or the current SessionID of the signed in user of their registered Method account.

Note: a unique SessionID is re-generated automatically each time a user logs in. It can be retrieved by running an action on a Method screen, such as Assign Value to Action Result and using Value From Session: Session ID.

- As a security feature, the User must have Method:API access 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.

MethodAPITableListV2

Overview

This is a simple API operation 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. strSessionID – If you know the unique current SessionID of a signed in user, you can pass this instead of strPassword. This can be found from a Value From Session: Session ID action from Method (especially useful if you are calling a web page or web service from within Method).

Response

- If the account passes verification, and a successful response is generated, the returned XML response value will be "Success".
- If the request failed, the returned XML response value will be an error message.
- The actual list of tables returned will be in the <MethodIntegration> tag.

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.MethodAPITableListV2(sCompanyAccount, sUserName, sPassword, "")
Set objSclient = Nothing
```

Which returns the following XML:

```
<?xml version="1.0" encoding="windows-1252" ?>
<MethodAPI response = "Success" responsemessage="">
<MethodIntegration>
<Record>
<TableName>Account</TableName>
<SupportsAdd>True</SupportsAdd>
<SupportsEdit>True</SupportsEdit>
</Record>
<Record>
<TableName>AccountAccountType</TableName>
<SupportsAdd>False</SupportsAdd>
<SupportsEdit>False</SupportsEdit>
</Record>
...
```

MethodAPIFieldListV2

Overview

This is a simple API operation 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. strSessionID – If you know the unique current SessionID of a signed in user, you can pass this instead of strPassword. This can be found from a Value From Session: Session ID action from Method (especially useful if you are calling a web page or web service from within Method).
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 XML response value will be “Success”.
- If the request failed, the returned XML response value will be an error message.
- The actual list of fields returned will be in the <MethodIntegration> tag.

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.MethodAPIFieldListV2(sCompanyAccount, sUserName, sPassword, "", sTableName)
Set objSclient = Nothing
```

Which returns the following XML:

```
<?xml version="1.0" encoding="windows-1252" ?>
<MethodAPI response = "Success" responsemessage="">
<MethodIntegration Table='Account'>
<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_DataSetV2

Overview

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

Note that while MethodAPISelect_DataSetV2 is easy and convenient, it can typically be used only by .NET applications. Also, there are common issues experience by all web developers getting the correct date/time returned by server-side Microsoft .NET datasets. For this reason, we recommend using MethodAPISelect_XMLV2 when necessary, instead of MethodAPISelect_DataSetV2.

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. strSessionID – If you know the unique current SessionID of a signed in user, you can pass this instead of strPassword. This can be found from a Value From Session: Session ID action from Method (especially useful if you are calling a web page or web service from within Method).
5. datReturnedDataSet (ByRef) – A local dataset variable passed, ByRef, to the web service to capture the returned dataset.
6. strTable – The name of the table you would like to get records from.
7. 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.

8. 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"

9. 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".

10. 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.

11. 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 XML response value will be "Success".

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

- Note: the actual dataset returned is passed, ByRef, 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_DataSetV2(sCompanyAccount, sUserName, sPassword, _
"", datResponse, sSelectFrom, sSelectFields, sSelectWhere, sSelectGroupBy, sSelectHaving, sSelectOrderBy)
wbsMethodAPI = Nothing
```

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

MethodAPISelect_XMLV2

Overview

This API operation 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_DatasetV2, except that there is no datReturnedDataSet parameter.

Response

- If the account passes verification, and a successful response is generated, the returned XML response value will be “Success”.
- If the request failed, the returned XML response value will be an error message.
- The actual XML returned will be in the <MethodIntegration> tag.

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_XMLV2(sCompanyAccount, sUserName, sPassword, "", sSelectFrom, _
sSelectFields, sSelectWhere, sSelectGroupBy, sSelectHaving, sSelectOrderBy)
Set objSclient = Nothing
```

Which returns the following to the local sXML variable:

```
<?xml version="1.0" encoding="windows-1252" ?>
<MethodAPI response = "Success">
<MethodIntegration Table="invoice">
<Record>
      <TxnDate>6/24/2008 5:00:00 PM</TxnDate>
      <Subtotal>44.0000</Subtotal>
      <RefNumber>33176</RefNumber>
      <Customer>Bob Crenshaw</Customer>
      <BalanceRemaining>44.0000</BalanceRemaining>
</Record>
<Record>
      <TxnDate>7/22/2008 5:00:00 PM</TxnDate>
      <Subtotal>188.0000</Subtotal>
      <RefNumber>33976</RefNumber>
      <Customer>Nat Chapman </Customer>
      <BalanceRemaining>50.0000</BalanceRemaining>
</Record>
...
```

MethodAPIUpdateV2

Overview

This API operation 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. strSessionID – If you know the unique current SessionID of a signed in user, you can pass this instead of strPassword. This can be found from a Value From Session: Session ID action from Method (especially useful if you are calling a web page or web service from within Method).
5. strTable – The name of the table you would like to update.
6. arrUpdateFieldsArray – An array containing the names of fields you wish to update.
7. 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"
```

8. 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_XMLV2 or MethodAPISelect_DatasetV2 call to find it.

Response

- If the account passes verification, and the update successfully completes, the returned XML response value will be either "SucessSendToDesktop", "Success" or "SuccessNotConnected". 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 returned XML response value will be an error message.

ASP.NET Code Example

Check out the full ASP.NET Code example, MethodAPIUpdateV2 Sample, which takes values entered from a webpage and sends the update using the Method:API. The main portion of the example, though, is:

```
'build the array of fields to update - in reality, there is no limit to the size of the array and
'  number of fields you can update at once, but here we just use 2 fields
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.MethodAPIUpdateV2(sCompanyAccount, sUserName, sPassword, "", _
 sUpdateTable, arrUpdateFieldsArray, arrUpdateValueArray, intRecordID)
wbsMethodAPI = Nothing
```

MethodAPIInsertV2

Overview

This API operation 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. strSessionID – If you know the unique current SessionID of a signed in user, you can pass this instead of strPassword. This can be found from a Value From Session: Session ID action from Method (especially useful if you are calling a web page or web service from within Method).
5. strTable – The name of the table you would like to insert into.
6. arrInsertFieldsArray – An array containing the names of fields you wish to insert.
7. 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"
```

Response

- If the account passes verification, and the insert successfully completes, the returned XML response value will be either “SucessSendToDesktop”, “SuccessNotConnect” 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”.
- The XML response will contain a value for the newly generated RecordID.
- If the request failed, the returned XML response value will be an error message.

ASP.NET Code Example

Check out the full ASP.NET Code example, MethodAPIInsertV2 Sample, which takes values entered from a webpage and creates a new record using the Method:API. 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 and
'  number of fields you can insert at once, but here we just use 2 fields
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.MethodAPIInsertV2(sCompanyAccount, sUserName, sPassword, _
"", sInsertTable, arrInsertFieldsArray, arrInsertValueArray)
wbsMethodAPI = Nothing
```

MethodAPIDeleteV2

Overview

This API operation 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. strSessionID – If you know the unique current SessionID of a signed in user, you can pass this instead of strPassword. This can be found from a Value From Session: Session ID action from Method (especially useful if you are calling a web page or web service from within Method).
5. strTable – The name of the table you would like to delete from.
6. 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_XMLv2 or MethodAPISelect_Datasetv2 call to find it.

Response

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

ASP.NET Code Example

Check out the full ASP.NET Code example, MethodAPIDeleteV2 Sample, which takes values entered from a webpage and sends the delete request using the Method:API. 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.MethodAPIDeleteV2(sCompanyAccount, sUserName, sPassword, _
 "", sDeleteTable, intRecordID)
wbsMethodAPI = Nothing
```

MethodAPIInsertFile

Overview

This API operation inserts a single file into one or many fields of type “Picture” or “FileAttachement” into a record of a 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. strSessionID – If you know the unique current SessionID of a signed in user, you can pass this instead of strPassword. This can be found from a Value From Session: Session ID action from Method (especially useful if you are calling a web page or web service from within Method).
5. strTable – The name of the table you would like to insert into.
6. arrInsertFieldsArray – An array containing the names of fields you wish to insert.
7. fileAttachment – A byte array containing a file you wish insert.

Response

- If the account passes verification, and the insert successfully completes, the returned XML response value will be “Success”.
- The XML response will contain a value for the newly generated RecordID.
- If the request failed, the returned XML response value will be an error message.

ASP.NET Code Example

Check out the full ASP.NET Code example, MethodAPIInsertFile Sample, which takes an image file located on drive C:\, converts it to byte array and creates a new record with the retrieved file data using the Method:API.

```
'Read the file into byte array
Dim file As Byte() = Nothing
Dim fs As New FileStream("C:\icon.png", FileMode.Open, FileAccess.Read)
Dim br As New BinaryReader(fs)
Dim numBytes As Long = New FileInfo("C:\icon.png").Length
file = br.ReadBytes(CInt(numBytes))
'Insert the file into the record
Dim result As String = _methodApi.MethodAPIInsertFile("CompanyName", "UserName", _
    "Password", "", "UploadFileTable", New String() {"UploadField"}, _
    file, "helpIcon.png")
```

MethodAPIUpdateFile

Overview

This API operation updates a single file into one or many fields of type “Picture” or “FileAttachement” into a specified table record.

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. strSessionID – If you know the unique current SessionID of a signed in user, you can pass this instead of strPassword. This can be found from a Value From Session: Session ID action from Method (especially useful if you are calling a web page or web service from within Method).
5. strTable – The name of the table you would like to update.
6. arrUpdateFieldsArray – An array containing the names of fields you wish to update (only fields of type “Picture” or “FileAttachement” are allowed).
7. fileAttachment – A byte array containing a file you wish update.
8. 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_XMLV2 or MethodAPISelect_DatasetV2 call to find it.

Response

- If the account passes verification, and the insert successfully completes, the returned XML response value will be “Success”.
- If the request failed, the returned XML response value will be an error message.

ASP.NET Code Example

Check out the full ASP.NET Code example, MethodAPIUpdateFile Sample, which takes an image file located on drive C:\, converts it to byte array and updates a record with the file data using the Method:API.

```
'Read the file into byte array
Dim file As Byte() = Nothing
Dim fs As New FileStream("C:\icon.png", FileMode.Open, FileAccess.Read)
Dim br As New BinaryReader(fs)
Dim numBytes As Long = New FileInfo("C:\icon.png").Length
file = br.ReadBytes(CInt(numBytes))
'Update the file in the record
Dim result As String = _methodApi.MethodAPIUpdateFile("CompanyName", "UserName", "Password", _
        "", "UploadFileTable", New String() {"UploadField"}, file, "icon.png", "1")
```

MethodAPIActionSendToDesktopV2

Overview

This API operation simulates the action found within the Method web platform called “Send To Desktop”. Normally, it is not necessary for you to use this API operation, 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 SendToDesktopV2 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. strSessionID – If you know the unique current SessionID of a signed in user, you can pass this instead of strPassword. This can be found from a Value From Session: Session ID action from Method (especially useful if you are calling a web page or web service from within Method).
5. strTable – The name of the table you would like to update.
6. 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_XMLv2 or MethodAPISelect_Datasetv2 call to find it.

Response

- If the account passes verification, the returned XML response value will be either “SucessSendToDesktop”, “SuccessNotConnected” 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 respond with

"Success". If the request failed, the returned XML response value will be an error message.

ASP.NET Code Example

Check out the full ASP.NET Code example, MethodAPIActionSendToDesktopV2 Sample. This is the most advanced of all examples, as it first inserts an invoice, then inserts the invoice lines, and then finally calls SendToDesktopV2 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.SelectedDate.Month, "0#") & "-" & Format(Me.calTxnDate.SelectedDate.Day, "0#")
'add universal time of noon.  If you wanted to put in 5:25:33 PM Central you would put " 23:25:33",
' since 5:25 PM Central is 11:25 PM at night in universal time
arrInsertValueArray_Invoice(1) &= " 12:00:00"
arrInsertFieldsArray_Invoice(2) = "ARAccount"
arrInsertValueArray_Invoice(2) = Me.txtARAccount.Text

'MethodAPI to insert the invoice
sResult = wbsMethodAPI.MethodAPIInsertV2(sCompanyAccount, sUserName, sPassword, _
 "", "Invoice", arrInsertFieldsArray_Invoice, arrInsertValueArray_Invoice)

Dim xmlDoc As New System.Xml.XmlDocument
Dim objNodeList As System.Xml.XmlNodeList
Dim objAttribute As System.Xml.XmlAttribute

'START: Make sure the request was successful
If InStr(1, sResult.ToUpper, "MethodAPI response = ""Success".ToUpper) = 0 Then
  'failed, give a message
  If Strings.Left(sResult, Len("<?xml")) = "<?xml" Then
    xmlDoc.LoadXml(sResult)
    objNodeList = xmlDoc.GetElementsByTagName("MethodAPI")
    For Each objAttribute In objNodeList.Item(0).Attributes
      If objAttribute.Name = "response" Then
        sResult = objAttribute.Value
      End If
    Next
  Else
    sResult = "Error"
  End If
  'failed, give a message
  wbsMethodAPI = Nothing
  Me.divResponse.InnerHtml = sResult
  Exit Sub
Else
  'all good
  xmlDoc.LoadXml(sResult)
  objNodeList = xmlDoc.GetElementsByTagName("MethodAPI")
  For Each objAttribute In objNodeList.Item(0).Attributes
    If objAttribute.Name.ToUpper = "RecordID".ToUpper Then
      intInvoiceRecordID = objAttribute.Value
    End If
  Next
  sResult = "Success"
End If
'END: Make sure the request was successful
```

```
'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.StartsWith("Success".ToUpper) = False 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 QuickBooks
sResult = wbsMethodAPI.MethodAPIActionSendToDesktop(sCompanyAccount, sUserName, sPassword, _
 "Invoice", intInvoiceRecordID)
```

MethodAPIActionChargeCreditCard_PsiGateV2

Overview

This API operation 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. strSessionID – If you know the unique current SessionID of a signed in user, you can pass this instead of strPassword. This can be found from a Value From Session: Session ID action from Method (especially useful if you are calling a web page or web service from within Method).
5. strGateway – Should be set to "PSIGATEWAY".
6. strGatewayUserName – Your merchant account user name.
7. strGatewayPassword – Your merchant account password.
8. strGatewayCustomerName – The first and last name of the customer being charged.
9. strGatewayCustomerAddress – The customer’s street address.
10. strGatewayCustomerCity – The customer’s city.
11. strGatewayCustomerState – The customer’s state or province.
12. strGatewayCustomerZipCode – The customer’s zip or postal code.
13. strGatewayCustomerEmail – The customer’s email address.
14. strGatewayCustomerCountry – The customer’s country.
15. strGatewayCustomerCVV – The 3 or 4 digit security code on the credit card.

16. strGatewayCustomerSubTotal – The subtotal of the shopping cart before tax.
17. strGatewayCustomerTax – The amount of tax to be added to the subtotal.
18. strGatewayCustomerCreditCardNumber – The credit card number on the credit card.
19. strGatewayCustomerExpiryMonth – The expiry month of the credit card.
20. strGatewayCustomerExpiryYear – The expiry year on the credit card.
21. strGatewayCustomerTotal – The total amount to be charged, which must be subtotal plus tax.

Response

- If the account passes verification, and the request is successfully sent to PSIGate the returned XML response value will be “Success”. Check GatewayApproved, which will be “APPROVED” or a failure code. Check GatewayResponse, for response details from the gateway.
- If the request failed, the returned XML response value will be an error message.

MethodAPIActionChargeCreditCard_ElectraCashV2

Overview

This API operation 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. strSessionID – If you know the unique current SessionID of a signed in user, you can pass this instead of strPassword. This can be found from a Value From Session: Session ID action from Method (especially useful if you are calling a web page or web service from within Method).
5. strGateway – Should be set to "ELECTRACASH".
6. strGatewayMerchantID – Your merchant MerchantID with Electracash.
7. strGatewayAuthKey – Your merchant account key, supplied by Electracash.
8. strGatewayCustomerName – The First and Last Name of the customer being charged if a personal account, or the company name if a business account.
9. strGatewayCustomerAddress – The customer’s street address.
10. strGatewayCustomerCity – The customer’s city.
11. strGatewayCustomerState – The customer’s state or province.
12. strGatewayCustomerZipCode – The customer’s zip or postal code.
13. strGatewayCustomerEmail – The customer’s email address.
14. strGatewayCustomerCountry – The customer’s country.

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

Response

- If the account passes verification, and the request is successfully sent to Electracash the returned XML response value will be "Success". Check GatewayApproved, which will be "APPROVED" or a failure code. Check GatewayResponse, for response details from the gateway.
- If the request failed, the returned XML response value will be an error message.

MethodAPIActionSendEmailV2

Overview

This API operation 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. strSessionID – If you know the unique current SessionID of a signed in user, you can pass this instead of strPassword. This can be found from a Value From Session: Session ID action from Method (especially useful if you are calling a web page or web service from within Method).
5. strToEmail – The email address of the recipient.
6. strFromEmail – The email address of the person sending the email.
7. strFromName – The name of the person sending the email.
8. strCCEmail – The CC (Carbon Copy) address, if you choose to set one.
9. strBCCEmail – The BCC (Blind Carbon Copy) address, if you choose to set one.
10. strSubject – The email subject.
11. strBody – The email body.
12. strPriority – The priority of the email. Values can be “Normal”,”High” or “Low”.
13. strAttachment – The URL of an attachment.
14. strServerAddress – The location of the mail server, for example, mail.ispname.com.

15. strServerPassword – The password of the mail server user account.
16. 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 XML response value will be “Success”.
- If the request failed, the returned XML response value will be an error message.

MethodAPICreateTableV2

Overview

This API operation allows you to create a new Method database Table. Once the table is created, you can then use MethodAPICreateFieldV2 to add fields to the 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. strSessionID – If you know the unique current SessionID of a signed in user, you can pass this instead of strPassword. This can be found from a Value From Session: Session ID action from Method (especially useful if you are calling a web page or web service from within Method).
5. strTable – The name of the table you want to create.

Response

- If your user account has sufficient permissions to create new tables, and the request is successful, the returned XML response value will be “Success”.
- If the request failed, the returned XML response value will be an error message.

MethodAPICreateFieldV2

Overview

This API operation sends to add new fields to an existing Method database 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. strSessionID – If you know the unique current SessionID of a signed in user, you can pass this instead of strPassword. This can be found from a Value From Session: Session ID action from Method (especially useful if you are calling a web page or web service from within Method).
5. strTable – The name of the table you are adding fields to.
6. strFieldName – The name of the field you are adding.
7. strFieldType – The type of field you are adding. Values can be “Text”, “Decimal”, “Integer”, “DateTime”, “Dropdown”, “Money”, “YesNo”, “FileAttachement” or “Picture”.
8. strFieldSize – If a Text field, the number of characters permitted. If a Decimal field the number of decimal places permitted. All other field types can be left as 0.
9. blnUnique – Can be True or False. Whether values entered into this field need to be unique. Note that you need to be careful setting this to True since if the table already exists and is not empty, the existing records by default will violate the unique requirement.
10. blnRequired – Can be True or False. Whether records must have a value entered into this field.

11. strDropdownToTable – In the case of a dropdown field, choose which table the dropdown values will be drawn from. If this is not a dropdown field, just pass an empty string “”.

12. strDropdownToField– In the case of a dropdown field, choose the display field containing unique values that will be displayed in the dropdown. If this is not a dropdown field, just pass an empty string “”.

Response

- If your user account has sufficient permissions to edit this table, and the request is successful, the returned XML response value will be “Success”.
- If the request failed, the returned XML response value will be an error message.

MethodAPIAppointment_CopySeries

Overview

This API operation allows you to copy the settings and dates of an existing recurring appointment to another appointment. If the other appointment is a one-time occurrence, it will be converted into a series. In Method, this is typically performed on the Activities table, but it can be performed on any table that is the base of a calendar object.

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 want to create.
5. intCopySeriesFromRecordID – The RecordID of the series that you are copying from.
6. intCopySeriesToRecordID - The RecordID of the one time appointment or series you are copying to. The record must already exist. If the record is a one time appointment it will be automatically converted into a series.
7. strCopySeriesRecurringDatesOption – The option for whether you want to copy just the dates, the dates and the series settings, or just the series settings. Values can be:
   "Insert Dates" (default) – dates will be generated based on the 'Basis' criteria of the series.
   "Insert Dates Copy Exact Dates" – exact dates from the 'Copy From' series will be copied to 'Copy To' series no matter what the Basis criteria is of the series.
   "Copy Exact Dates Greater Than Today" – same as Insert Dates Copy Exact Dates, but will not copy occurrences older than today.
   "Copy No Dates" - no occurrences will be created.
8. strCopySeriesDeleteExistingOccurrences – Values can be "True" or "False. The option of whether you want to delete all existing unchanged

occurrences of the series, or whether your changes will copy dates in addition to what already exists. If occurrences have already been edited, they will no longer be considered “unchanged” and will not be deleted, regardless of your choice.

Response

- If the account passes verification, and the request is processed successfully the returned XML response value will be “Success”.
- If the request failed, the returned XML response value will be an error message.

MethodAPIAppointment_InsertDate

Overview

This API operation allows you to insert an occurrence appointment into an existing series of appointments. In Method, this is typically performed on the Activities table, but it can be performed on any table that is the base of a calendar object.

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 want to create.
5. intAppointmentRecordID – The RecordID of the series you are inserting into.
6. strAppointmentStartDateAndTime – The starting date and time of the appointment.
7. strAppointmentEndDateAndTime – The ending date and time of the appointment.

Response

- If the account passes verification, and the request is processed successfully the returned XML response value will be “Success”.
- If the request failed, the returned XML response value will be an error message.

MethodAPICreateMethodAccountV3

Overview

This API operation allows you to create a new Method account. This is designed for developers that have their own registration process or shopping cart on their site, and are registered Method Solution Providers that need to sign up their own clients. Only developers registered in the Method Solution Providers program can use this operation.

Web Service URL

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

Parameters

1. strNewCompanyAccount – The desired unique "Company Account" name.
2. strNewCustomerLogin – The "User Name" value of the admin user.
3. strNewCustomerPassword – The "Password" value of the admin user.
4. strSecurityNewCustomerQuestion – The security question, should the admin user forget their password.
5. strSecurityNewCustomerAnswer – The security answer, should the admin user forget their password.
6. blnUseExistingAccount – Whether this strNewCompanyAccount actually already exists, or whether a new account should be created. For 3rd party developers, this is always be False, as only internal Method developers are able to insert company accounts prior to calling this Method API operation.
7. strAppName – The name of the application on the Method Library the user is signing up for. Note that this can only be an application designed by the Method team, or an application that you developed. You cannot sign up users for applications that other developers created.
8. strDeveloper – The Developer Name assigned to you as part of the Method Solution Providers program.
9. strDeveloperGUID – The Developer GUID assigned to you as part of the Method Solution Providers program.

10. strNewCustomerFirstName – The first name of the billing contact at the company using Method.
11. strNewCustomerLastName - The last name of the billing contact at the company using Method.
12. strNewCustomerAddress - The address of the company using Method.
13. strNewCustomerAddressCity - The city of the company using Method.
14. strNewCustomerAddressState - The state or province of the company using Method.
15. strNewCustomerAddressCountry - The country of the company using Method.
16. strNewCustomerAddressZipCode - The zip / postal code of the company using Method.
17. strNewCustomerCompanyName - The company name of the company using Method.
18. strNewCustomerContact - The full name of the billing contact at the company using Method. This is typically just the same as first and last name that was used above.
19. strNewCustomerPhone - The phone number of the billing contact at the company using Method.
20. strNewCustomerEmail - The email address of the billing contact at the company using Method.

Response

- If the account passes verification, and the request is processed successfully the returned XML response value will be “Success”.
- If the request failed, the returned XML response value will be an error message.

MethodAPICreateMethodTabGroupRoleV2

Overview

This API operation allows Method Solution Providers to create a new Tab Group / Role. This is equivalent to going to Customize > Tab Groups / Roles in Method and adding a new Tab Group / Role manually. This is useful in situations where a Method Solution Provider wishes to sign up a new account and then configure the account automatically before the user signs in.

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. strSessionID – If you know the unique current SessionID of a signed in user, you can pass this instead of strPassword. This can be found from a Value From Session: Session ID action from Method (especially useful if you are calling a web page or web service from within Method).
5. strDeveloper – The Developer Name assigned to you as part of the Method Solution Providers program.
6. strDeveloperGUID – The Developer GUID assigned to you as part of the Method Solution Providers program.
7. strTabGroupRoleName – The name of the Tab Group / Role you want to create.
8. arrAssignToUsersArray – An array of users, listed by their User Name, that will have access to this Tab Group / Role.
9. arrAssignTabsArray – An array of Tabs that will belong to this Tab Group / Role, listed in the order they should appear.
10. blnAllowedToIntegrateAccounting – Whether users assigned to this Tab Group / Role are permitted to sync using the Method Integration Engine.

11. blnAllowedToDesignReports – Whether users assigned to this Tab Group / Role are permitted to design reports with the Method Report Designer.

Response

- If the account passes verification, and the request is processed successfully the returned XML response value will be “Success”.
- If the request failed, the returned XML response value will be an error message.

MethodAPICreateMethodTabV2

Overview

This API operation allows Method Solution Providers to create a new Tab in the Method account.

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. strSessionID – If you know the unique current SessionID of a signed in user, you can pass this instead of strPassword. This can be found from a Value From Session: Session ID action from Method (especially useful if you are calling a web page or web service from within Method).
5. strDeveloperGUID – The Developer GUID assigned to you as part of the Method Solution Providers program.
6. strTabGroupRoleName – The name of the Tab Group / Role you want to create.
7. strTabName – The name of the Tab you want to create.
8. strTabGroupRoleNameArray – An array of Tab Groups that this Tab will be added to.
9. arrTabLinkNameArray – An array of Tab Links that will be added to this Tab. It is used together with arrTabLinkScreenArray.
10. arrTabLinkScreenArray – An array of screen names associated with the Tab Links. For example, you may create two Tab Links for “Customers” and “Create Invoices” for the screens “Customer List” and “Invoice” with the following arrays:
    arrTabLinkNameArray(0) = “Customers”
    arrTabLinkScreenArray(0) = “Customer List”

arrTabLinkNameArray(1) = “Create Invoices”
arrTabLinkScreenArray(1) = “Invoice”

Response

- If the account passes verification, and the request is processed successfully the returned XML response value will be “Success”.
- If the request failed, the returned XML response value will be an error message.

MethodAPICreateMethodUserV2

Overview

This API operation allows Method Solution Providers to create new users in their customers' Method accounts.

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. strSessionID – If you know the unique current SessionID of a signed in user, you can pass this instead of strPassword. This can be found from a Value From Session: Session ID action from Method (especially useful if you are calling a web page or web service from within Method).
5. strDeveloperGUID – The Developer GUID assigned to you as part of the Method Solution Providers program.
6. strTabGroupRoleName – The name of the Tab Group / Role you want to create.
7. strTabGroupRoleNameArray – An array of Tab Group / Roles this user will belong to.
8. strNewUserName – The User Name of this new user.
9. strNewUserPassword – The Password of this new user.
10. blnActive – Whether this new user is active.
11. strEmail – The email address of this new user.
12. strSecurityQuestion – The security question, should the user forget their password.

13. strSecurityAnswer – The security answer, should the user forget their password.
14. strDefaultTabGroupRole – The default Tab Group / Role this user will first see when they sign in.
15. strEntityFullName – Optional. The Employee or Vendor associated with this user in QuickBooks.
16. strEmailSignature – Optional. The email signature for this user.
17. blnAllowedToAddEditAllTables – Whether this user is permitted to edit all tables.
18. blnAllowedToAddEditAllScreens – Whether this user is permitted to add and edit all screens.
19. blnAllowedToCreateNewTables – Whether this user is permitted to create new tables.
20. blnAllowedToCreateNewScreens – Whether this user is permitted to create new screens.
21. blnAllowedAccessToQuickBooksSynchronize – Whether this user is permitted to synchronize with QuickBooks using the Method Integration Engine.
22. blnAllowedAccessToQuickBooksResolveConflicts – Whether this user has access to the QuickBooks > Resolve Conflicts area of Method.
23. blnAllowedAccessToQuickBooksAuditTrail – Whether this user has access to the QuickBooks > Audit Trail area of Method.
24. blnAllowedAccessToCustomizeMyAccount – Whether this user has access to the Customize > My Account area of Method.
25. blnAllowedAccessToCustomizeDisplay – Whether this user has access to the Customize > Display area of Method.
26. blnAllowedAccessToCustomizeTablesFields – Whether this user has access to the Customize > Tables / Fields area of Method.
27. blnAllowedAccessToCustomizeScreens – Whether this user has access to the Customize > Screens area of Method.
28. blnAllowedAccessToCustomizeTabs – Whether this user has access to the Customize > Tabs area of Method.

29. blnAllowedAccessToCustomizeUsers – Whether this user has access to the Customize > Users area of Method.

30. blnAllowedAccessToCustomizeTabGroupsRoles – Whether this user has access to the Customize > Tab Groups / Roles area of Method.

31. blnAllowedAccessToCustomizeThirdPartyPortal – Whether this user has access to the Customize > Third Party Portals area of Method.

32. blnAllowedToConnectToMethodAPI – Whether this user has permission to run applications that connect to Method using the Method API.

33. blnCanLogIn – Whether this user has the right to sign into Method. If they have the right to sign into Method, it will typically mean there will be an increase in the monthly subscription for this account. Most accounts are using Named licensing, which means an additional cost per user that can sign in. Some accounts use Concurrent licensing, where the cost does not increase by how many named users are permitted to sign in, there is only a cost according to how many users are able to remain signed in at the same time.

Response

- If the account passes verification, and the request is processed successfully the returned XML response value will be “Success”.
- If the request failed, the returned XML response value will be an error message.

MethodAPIDeleteMethodTabLinkV2

Overview

This API operation allows Method Solution Providers to delete Tab Links in their customers' Method accounts.

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. strSessionID – If you know the unique current SessionID of a signed in user, you can pass this instead of strPassword. This can be found from a Value From Session: Session ID action from Method (especially useful if you are calling a web page or web service from within Method).
5. strDeveloperGUID – The Developer GUID assigned to you as part of the Method Solution Providers program.
6. strTabGroupRoleName – The name of the Tab Group / Role you want to create.
7. arrTabLinkNameArray – An array of Tab Links you want to delete. This is used in conjunction with arrTabNameArray.
8. arrTabNameArray – An array containing the Tabs of the associated Tab Links. For example, if you wanted to delete the "Reports" Tab Link found in the "Home" Tab, and also the "Estimates" Tab Link found in the "Customer Center" Tab, you would have the following:
   arrTabLinkNameArray(0) = "Reports"
   arrTabNameArray(0) = "Home"
   arrTabLinkNameArray(1) = "Estimates"
   arrTabNameArray(1) = "Customer Center"

Response

- If the account passes verification, and the request is processed successfully the returned XML response value will be “Success”.
- If the request failed, the returned XML response value will be an error message.

MethodAPIGetScreenURLV2

Overview

This API operation allows you to get a URL that will display a Method screen, based on the Tab Link provided. This has limited purpose, since the user must still be signed into Method for the session to be active, otherwise an error message will be returned.

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. strSessionID – If you know the unique current SessionID of a signed in user, you can pass this instead of strPassword. This can be found from a Value From Session: Session ID action from Method (especially useful if you are calling a web page or web service from within Method).
5. strTabLinkName – The name of the Tab Link associated with the screen you wish to open.
6. strTabName – The name of the Tab associated with the Tab Link that you wish to open.
7. intRecordID – The Record ID of the base table on the screen associated with the Tab Link that you want to have displayed.

Response

- If the account passes verification, and the request is processed successfully the returned XML response value will be "Success".
- If the request failed, the returned XML response value will be an error message.

MethodAPIGetCSS

Overview

This API operation retrieves current user's custom CSS display settings. It returns font size, font name, font measurement type, color of the active tab, color of a grid's header and footer, color of the section bar and color of inactive tab.

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. strSessionID – If you know the unique current SessionID of a signed in user, you can pass this instead of strPassword. This can be found from a Value From Session: Session ID action from Method (especially useful if you are calling a web page or web service from within Method).

Response

- If the account passes verification, and the request is processed successfully the returned XML response value will be "Success".
- The actual XML returned with all of the settings will be in the <MethodIntegration> tag.
- If the request failed, the returned XML response value will be an error message.