

## Introduction:

This page contains the Programmer's API for the OGSQlai.xla Excel Add-In. The Add-In is the foundation for all second generation Excel based SQL reports. It is also a resource available to end users that develop their own Excel based SQL reports for OGWin.

The API is currently divided into three sections:

- *Common Functions* – These functions have been identified as those that are regularly used in the process of generating SQL based OGWin reports.
- *Selection Interfaces* – These subroutines provide reusable user interfaces for common entity selection dialogs such as Company Selection, Well Selection, etc.
- *Configuration Functions* – These items provide common configuration routines for SQL Connectivity and Site Specific Options.

**Note:** The OGSQlai.xla Add-In contains additional functionality not described in this document. These items are considered undocumented and are subject to change without notice.

## Common Functions:

### AdjKey Syntax:

OGaiAdjKey(byval sIn as String, ByVal iLen As Integer) As String

Description:

This function justifies and lengthens the string contained in *sIn* to the proper length as indicated by *iLen*. The function will left justify and pad with blanks any alphanumeric string. The function will right-justify and pad with blanks any numeric string. Leading and trailing whitespace characters are not considered when determining numeric status. Using AdjKey where Len(sIn) > iLen is undefined.

This function is normally used when a user supplied value is used in query generation in order to compensate for the storage of key values in the SQL schema.

Example:

```
sSQL = "SELECT * FROM Company WHERE Company_Code = " & _
```

```
OGaiAdjKey(sCompanyCode, 12) & ""
```

### GetAddress

Syntax:

OGaiGetAddress(ByVal iRow As Long, ByVal iCol As Integer) As String

Description:

This function returns the Excel Relative Address for the cell pointed to by iRow and iCol. This is useful when dynamically creating Excel formulas.

Example:

```
Cells(iMasterRowPtr, kNETVAL).Value = "=SUM(" & _
```

```
OGaiGetAddress(ITotalStart, kNETVAL) & ":" & _
```

```
OGaiGetAddress(ITotalEnd, kNETVAL) & ")"
```

### **GetFunctionalCurrency**

Syntax:

```
GetFunctionalCurrency() As String
```

Description:

This function connects to the database, determines and returns the functional currency code for the system. This function is useful when authoring reports that will be used on multiple multicurrency databases that have different functional currencies. Calling this function on a non-multicurrency system is undefined.

Example:

```
sFunCur = OGaiGetFunctionalCurrency()
```

### **PopTerm**

Syntax:

```
OGaiPopTerm(ByRef sTerms as String) As String
```

Description:

This function removes and returns the next term in the string sTerms. NOTE: sTerms is a ByRef parameter and will be altered by PopTerm. See PushTerm for more details.

Example:

```
sMainAccount = OGaiPopTerm(sAccountList)
```

### **PushTerm**

Syntax:

OGaiPushTerm(ByRef sTerms As String, ByVal sTerm As String)

Description:

This subroutine adds the term in sTerm to the list of terms contained in sTerms.  
NOTE: sTerms is a ByRef parameter and will be altered by PushTerm.

The pair PushTerm and PopTerm may used to store a list of string items in a single string capable object (string variable, cell, etc.). This is useful in situations where multiple items must be stored in one column. The list of terms operates as a *stack* such that the first item *pushed* will be the last item *popped* . Strings containing the *tilde* and *pipe* characters (~,|) may not be used with PopTerm and PushTerm. Such use is undefined.

Example:

OGaiPushTerm(sTerms,sMainAccount)

### **Selection Interfaces**

#### **SelectAccountV1**

Syntax:

OGaiSelectAccountV1(ByRef sMainAccount As String, \_  
ByRef sSubAccount As String) As Boolean

Description:

This function displays a *modal* user interface allowing the selection of an OGWin Account from the Chart of Accounts. If the user cancels this process, the function returns False. All other outcomes (which result an account selection) will return True. The Account is split into the Main Account and Sub Account components and stored in the ByRef parameters sMainAccount and sSubAccount Respectively.

Example:

If OGaiSelectAccountV1(sMain, sSub) = True Then

txtMain.Text = sMain

txtSub.text = sSub

End If

#### **SelectAFEV1**

Syntax:

OGaiSelectAFEV1(ByRef sAFECode As String, \_

Optional(ByVal sFilterByWell As String = "") As Boolean

Description:

This function displays a *modal* user interface allowing the selection of an AFE Code from the list of AFEs in the database. If the *Optional* parameter sFilterByWell is provided, the list of available AFE codes will be filtered to those relevant to that Well Code. If the user cancels the process, False will be returned. Otherwise True will be returned.

Example:

' Example 1, without filter

If OGaiSelectAFEV1(sAFE) = True Then

txtAFE.Text = sAFE

End If

' Example 2, with filter

sWell = "TX1000"

If OGaiSelectAFEV1(sAFE, sWell) = True Then

txtAFE.Text

End If

### **SelectCompanyV1**

Syntax:

OGaiSelectCompanyV1(ByRef sCompCode As String) As Boolean

Description:

This function displays a *modal* user interface allowing the selection of an OGWin Company from the list of Companies in the database. If the user cancels this process, the function returns False. All other outcomes (which result in a company selection) will return True. The Company Code is stored in the *ByRef* parameter sCompCode.

Example:

If OGaiSelectCompanyV1(sComp) = True Then

```
txtComp.Text = sComp
```

```
End If
```

### **SelectCurrencyV1**

Syntax:

```
OGaiSelectCurrencyV1(ByRef sCurCode As String) As Boolean
```

Description:

This function displays a *modal* user interface allowing the selection of a Currency Code from the list of Currencies in the database. If the user cancels this process, the function returns False. All other outcomes (which result in a currency code selection) will return True. The Currency Code will be stored in the *ByRef* parameter sCurCode. Use of this function on a non-multicurrency database is undefined.

Example:

```
If OGaiSelectCurrency(sCurCode) = True Then
```

```
txtCurrency.Text = sCurCode
```

```
End If
```

### **SelectLineFormatV1**

Syntax:

```
OGaiSelectLineFormatV1(ByRef sFormatCode As String) As Boolean
```

Description:

This function displays a *modal* user interface allowing the selection of a Line Format from the list of Financial Line Formats in the database. If the user cancels this process, the function returns False. All other outcomes (which result in a currency code selection) will return True. The Format Code will be stored in the *ByRef* parameter sFormatCode.

Example:

```
If OGaiSelectLineFormat(sFormatCode) = True Then
```

```
txtLF.Text = sFormatCode
```

```
End If
```

---

## SelectWellIV1

Syntax:

```
OGaiSelectWellIV1(ByRef sWellCode As String, _  
ByVal sFilterByAFE As String) As Boolean
```

Description:

This function displays a *modal* user interface allowing the selection of a Well Code from the list of Wells in the database. If the *Optional* parameter sFilterByAFE is provided, the list of available Well codes will be filtered to those relevant to that AFE Code. If the user cancels the process, False will be returned. Otherwise True will be returned.

Example:

```
' Example 1, without filter
```

```
If OGaiSelectWellIV1(sWell) = True Then
```

```
txtWell.Text = sWell
```

```
End If
```

```
' Example 2, with filter
```

```
sAFE = "01D001"
```

```
If OGaiSelectAFEV1(sWell, sAFE) = True Then
```

```
txtWell.Text
```

```
End If
```

## Configuration Functions:

### CheckSQLConnectivity

Syntax:

```
OGaiCheckSQLConnectivity() As Boolean
```

Description:

This function tests SQL Connectivity based on the server and database information contained in the Add-In. True is returned if the database connection test was successful; False otherwise

Example:

```
If OGaiCheckSQLConnectivity() = False Then
```

```
MsgBox "SQL Server Connectivity Failed!"
```

```
End If
```

---

### **GetConnectionString**

Syntax:

```
OGaiGetConnectionString() As String
```

Description:

This function returns an ADODB connection string constructed based on the SQL Server Settings contained in the Add-In. This string is used to configure an ADODB.Connection object.

Example:

```
oCon.ConnectionString = OGaiGetConnectionString()
```

---

### **Has[OPTION]**

Syntax:

```
OGaiHas<OPTION_NAME>() As Boolean
```

Description:

This series of functions return True if the current settings indicate that the given option is enabled. False is returned otherwise. These are useful when dealing with site specific options that have an impact on the schema.

Examples:

```
OGaiHasAFE()
```

```
OGaiHasFixedAssets()
```

```
OGaiHasMultiCurrency()
```

OGaiHasProduction()

---

### **SetSiteOptions**

Syntax:

OGaiSetSiteOptions()

Description:

This sub displays a *modal* user interface allowing the setting of site specific options.

Example:

```
Public Sub ChangeOptions()
```

```
OGaiSetSiteOptions
```

```
End Sub
```

---

### **SetSQLConnection**

Syntax:

OGaiSetSQLConnection()

Description:

This sub displays a modal user interface allowing for the entry of SQL Server connectivity information to be saved to the Add-In

Example:

```
Public Sub SetSQLConnectivity()
```

```
OGaiSetSQLConnection
```

```
End If
```