IBSheet7 Method, Event API Document

Table of Contents

Chapter 1. IBSheet Events 11

1. IBSheet Events 13

1.1 Using events 13

2.2 Event List 14

OnAfterColumnMove Event 14

OnAfterEdit Event 14

OnAfterExpand Event 15

OnBeforeCheck Event 16

OnBeforeColumnMove Event 17

OnBeforeEdit Event 18

OnBeforeExpand Event 18

OnBeforePaste Event 19

OnBeforeSave Event 20

OnBeforeSearch Event 21

OnCellDropEnd Event 22

OnChange Event 23

OnChangeFilter Event 24

OnChangeSum Event 25

OnCheckAllEnd Event 25

OnClick Event 26

OnDblClick Event 27

OnDebugMsg Event 28

OnDecryption Event 29

OnDownFinish Event 29

OnDragStart Event 30

OnDropEnd Event 31

OnEditValidation Event 32

OnEncryption Event 33

OnFilterEnd Event 34

OnHScroll Event 35

OnKeyDown Event 36

OnKeyUp Event 37

OnLoadData Event 38

OnLoadExcel Event 38

OnLoadText Event 40

OnMessage Event 40

OnMouseDown Event 42

OnMouseMove Event 42
OnMouseUp Event 44
OnPageRequest Event 45
OnPopupClick Event 45
OnReadyExcelDown Event 46
OnResize Event 47
OnRowSearchEnd Event 47
OnSaveEnd Event 48
OnSearchEnd Event 49
OnSelectMenu Event 50
OnSelectCell Event 53
OnSmartResize Event 54
OnSort Event 54
OnTreeChild Event 55
OnUserResize Event 57
OnValidation Event 57
OnVScroll Event 59
OnWaitTimeOut Event 60

Chapter 2. IBSheet Methods 61

2. IBSheet Methods 62

2.1 Using methods 62
2.2 Method List 62
ActionMenu Method 62
AllowCheck Method 65
AllowEvent4CheckAll Method 66
AllowExpand Method 66
AutoRowHeight Method 68
AutoSumPosition Method 69
BasicImeMode Method 70
CellAlign Method 72
CellBackColor Method 73
CellComboItem Method 75
CellEditable Method 76
CellFont Method 78
CellFontBold Method 80
CellFontColor Method 81
CellFontItalic Method 83
CellFontName Method 84

CellFontSize Method	85
CellFontStrike Method	87
CellFontUnderline Method	88
CellImage Method	90
CellSearchValue Method	92
CellVAlign Method	93
CellText Method	94
CellValue Method	96
CheckAll Method	100
CheckedRows Method	101
CheckReverse Method	102
ClearHeaderCheck Method	103
ClipCopyMode Method	104
ClipPasteMode Method	105
ColBackColor Method	107
ColCondProperty Method	108
ColEditable Method	110
ColFontColor Method	112
ColFontUnderline Method	113
ColHidden Method	114
ColLeft Method	116
ColSaveName Method	116
ColumnSort Method	117
ColValueDup Method	119
ColValueDupRows Method	120
ColWidth Method	123
ComboOpenMode Method	124
ComputeSum Method	125
ConfirmOK Method	126
CountFormat Method	128
CountPosition Method	130
CreatePivotTable Method	131
CumulateBackColor Method	133
Data2Clipboard Method	134
DataAlternateBackColor Method	135
DataAutoTrim Method	136
DataBackColor Method	137
DataCopy Method	138
DataFontColor Method	139
DataInsert Method	140

DataLinkMouse Method	142
DataMove Method	143
DataRowHeight Method	144
DataRowMerge Method	146
DirectDown2Excel Method	147
DirectLoadExcel Method	151
DoAllSave Method	154
DoPrint Method	156
DoRowSearch Method	157
DoSave Method	158
DoSearch Method	161
DoSearchChild Method	162
DoSearchPaging Method	164
Down2Excel Method	166
Down2ExcelUrl Method	174
Down2ExcelBuffer Method	175
Down2Pdf Method	176
Down2PdfUrl Method	181
Down2Text Method	182
Down2TextUrl Method	185
DownloadingImage Method	186
DragMode Method	187
Editable Method	188
EditEnterBehavior Method	189
EditableColorDiff Method	191
EditArrowBehavior Method	192
EditTabBehavior Method	194
Ellipsis Method	195
Enable Method	196
EnterBehavior Method	197
EtcData Method	199
ExtendLastCol Method	203
FindCheckedRow Method	204
FindStatusRow Method	205
FindSubSumRow Method	206
FindSumRow Method	207
FindText Method	208
FitColWidth Method	209
FitSize Method	210
FocusAfterProcess Method	211

FocusAfterRowTransaction Method 213
FocusEditMode Method 214
FrozenRows Method 215
GetCellProperty Method 217
GetChildNodeCount Method 218
GetChildRows Method 218
GetComboInfo Method 219
GetCurrentPage Method 221
GetDataRows Method 221
GetEditText Method 222
GetFilterParam Method 222
GetFirstChildRow Method 224
GetGroupCol Method 224
GetLastChildRow Method 225
GetMergedEndCell Method 226
GetMergedStartCell Method 226
GetNextSiblingRow Method 227
GetSheetPageLength Method 228
GetParentRow Method 228
GetPrevSiblingRow Method 229
GetSaveData Method 230
GetSaveJson Method 231
GetSaveString Method 232
GetSearchData Method 235
GetSelectionCols Method 236
GetSelectionRows Method 237
GetSheetHtml Method 237
GoToFirstPage Method 238
GoToLastPage Method 239
GoToNextPage Method 240
GoToPageNum Method 241
GoToPrevPage Method 241
HeaderActionMenu Method 242
HeaderBackColor Method 244
HeaderCheck Method 245
HeaderFontBold Method 246
HeaderFontColor Method 247
HeaderRows Method 249
HeaderRowHeight Method 249
HideFilterRow Method 250

HideProcessDlg Method 251
HideSubSum Method 251
HighlightAfterSort Method 252
IBCloseCalendar Method 254
IBShowCalendar Method 254
ImageList Method 257
InitCellProperty Method 259
InitColumns Method 262
InitComboNoMatchText Method 277
InitHeaders Method 278
IsDataModified Method 280
IsHaveChild Method 280
KeyFieldImage Method 281
LastCol Method 283
LastRow Method 283
LeftCol Method 284
LoadSaveData Method 285
LoadSearchChildData Method 286
LoadSearchData Method 288
LoadExcel Method 289
LoadExcelBuffer Method 292
LoadExcelUrl Method 294
LoadText Method 295
LoadTextUrl Method 297
MergeSheet Method 298
MaximumValue Method 300
MinimumValue Method 302
MouseCol Method 303
MousePointer Method 304
MouseRow Method 306
MoveColumnFail Method 306
MoveColumnPos Method 307
RangeBackColor Method 308
RangeFontBold Method 310
RangeFontColor Method 312
RangeText Method 313
RangeValue Method 315
RedrawSum Method 317
RemoveAll Method 319
RenderSheet Method 319

ReNumberSeq Method	320
Reset Method	321
ReturnCellData Method	322
ReturnColumnPos Method	322
ReturnData Method	323
RowBackColor Method	324
RowBackColorD Method	325
RowBackColorI Method	326
RowBackColorU Method	328
RowCount Method	329
RowDelete Method	330
RowDraggable Method	331
RowEditable Method	332
RowExpanded Method	334
RowFontColor Method	335
RowHeight Method	336
RowHeightMax Method	337
RowHeightMin Method	339
RowHidden Method	340
RowJson Method	341
RowLevel Method	343
RowMerge Method	344
RowSaveStr Method	345
RowTop Method	346
SaveNameCol Method	346
SavingImage Method	347
SearchingImage Method	348
SearchRows Method	349
SelectCell Method	350
SelectCol Method	351
SelectionMode Method	352
SelectRow Method	354
SendComboData Method	355
SetBlur Method	356
SetCellImageStyle Method	357
SetColProperty Method	358
SetConfig Method	361
SetDown2ExcelConfig Method	368
SetEndEdit Method	370
SetFilterOption Method	371

SetFilterValue Method 372
SetFindDialog Method 373
SetFocus Method 374
SetMergeCell Method 375
SetRowHaveChildValue Method 376
SheetWidth Method 376
SheetHeight Method 377
ShowButtonImage Method 379
ShowCalendar Method 381
ShowDebugMsg Method 382
ShowColumnPopup Method 383
ShowFilterRow Method 384
ShowFindDialog Method 385
ShowGroupRow Method 386
ShowMsgMode Method 387
ShowProcessDlg Method 388
ShowSortArrow Method 389
ShowSubSum Method 391
ShowToolTip Method 393
ShowTreeSubSum Method 394
SubSumBackColor Method 396
ShowSum Method 397
ShowTreeLevel Method 398
SumBackColor Method 399
SumFontBold Method 400
SumFontColor Method 401
SumRowHidden Method 402
SumValue Method 403
Theme Method 404
ToolTipText Method 406
TopRow Method 407
TotalRows Method 408
TreeActionMode Method 409
UnicodeByte Method 411
UseDefaultTime Method 412
UserAgent Method 413
UploadingImage Method 414
ValidateFail Method 415
Visible Method 416
WaitImage Method 418

WaitImageVisible Method 419
WaitTimeOut Method 420

Chapter 10. IBSheet Global Function 10. IBSheet Global Function 422

10. IBSheet Global Function 423

10.1 Use Global Function 423
10.2 Function List 424
IBCalendarSetTheme Method 424
IBShowCalendar Method 424
IBCloseCalendar Method 428

Appendix 429

1. Configurable properties by column type 430

1.1. Text format type 430
1.2. Date format type 432
1.3. Number format type 434
1.4. Checkbox format type 436
1.5. Combo format type 438
1.6. Other format types 440

2. Data format allowed for Fx (Formatted) search mode 443

3. Creating a theme 445

4. Apply Merge setting to excel download 446

Chapter 1. IBSheet Events

1. IBSheet Events

1.1 Using events

IBSheet offers a wide range of events to support users to customize or configure a function as they want.

```
function Object ID_Event name(Parameter, ...) { }
```

For example, if you need to use OnChange event for a new function, you may create the following statement, assuming the IBSheet ObjectID is "mySheet":

```
function mySheet_OnChange(Row, Col, Value) {

    alert(Row + "," + Col + " values have been updated.");

}
```

The followings are the list of events offered by IBSheet:

2.2 Event List

OnAfterColumnMove Event

➢ **Purpose**

Event fires when a user has successfully moved a column using mouse drag or MoveColumnPos function.

➢ **Syntax**

Syntax	**function Object ID_OnAfterColumnMove(Col, NewPos) { }**

➢ **Parameters**

Parameter	Type	Description
Col	Long	Index of the original column
NewPos	Long	Index of the new column location

➢ **Example**

```
//Event fires after a column is moved.

function mySheet_OnAfterColumnMove(Col, NewPos) {

    alert(Col + " location has been moved to " + NewPos + " location..");

}
```

➢ **See Also**

OnBeforeColumnMove , **MoveColumnPos** , **MoveColumnFail**

OnAfterEdit Event

➢ **Purpose**

Event fires promptly after a cell value is edited.

➢ **Syntax**

Syntax	**function Object ID_OnAfterEdit**(Row, Col) { }

➢ **Parameters**

Parameter	Type	Description
Row	Long	Row index of the cell
Col	Long	Column index of the cell

➢ **Example**

```
//Event fires promptly after a cell value is edited.

function mySheet_OnAfterEdit(Row, Col) {

      alert("Exit editing.");

}
```

➢ **See Also**

OnBeforeEdit

OnAfterExpand Event

➢ **Purpose**

Event fires when you collapse or expand a tree-type sheet by clicking + or - buttons.

➢ **Syntax**

Syntax	**function ObjectID_OnAfterExpand(Row, Expand) { }**

➢ **Parameters**

Parameter	Type	Description
Row	Long	Row index of the cell
Expand	Long	0 : Expand 2 : Collapse

➢ **Example**

```
function mySheet_OnAfterExpand(Row, Expand) {

    alert( Row + "Row, "+ Expand +"'s status");

}
```

- **See Also**

  OnBeforeExpand, RowExpand, RowLevel, ShowTreeLevel

OnBeforeCheck Event

- **Purpose**

  Event fires promptly before user starts editing a checkbox value (by mouse-clicking or pushing the space button).

  As the event timing is immediately before checkbox is updated, conditional processing is available upon checking.

  The following data types are supported:
  - DelCheck
  - CheckBox
  - Radio
  - DummyCheck

- **Syntax**

Syntax	**function Object ID_OnBeforeCheck(Row, Col) { }**

- **Parameters**

Parameter	Type	Description
Row	Long	Row index of the cell
Col	Long	Column index of the cell

- **Example**

```
//Event fires when a box is about to be checked.

//Display a warning window if a row to check is an Insert row.

function mySheet_OnBeforeCheck(Row, Col) {

    if(mySheet.GetCellValue(Row,"status") == "I") {

            alert("Check : " + Row + " is an Insert row.");

     }

 }
```

OnBeforeColumnMove Event

- **Purpose**
  Event fires when the user starts moving a column using mouse or MoveColumnPos method. You can block column movements by using MoveColumnFail(1) method.

- **Syntax**

Syntax	**function Object ID_OnBeforeColumnMove(Col, NewPos) { }**

- **Parameters**

Parameter	Type	Description
Col	Long	Index of the source column
NewPos	Long	Index of the target column location

- **Example**

```
//Cancel column movement if column 0 is moved to a location past column
3

function mySheet_OnBeforeColumnMove(Col, NewPos) {

    if(Col==0 && NewPos > 3) {

           mySheet.MoveColumnFail(1);

 }
}
```

OnBeforeEdit Event

- **Purpose**

  Event fires immediately before a cell value is edited.

- **Syntax**

Syntax	**function Object ID_OnBeforeEdit**(Row, Col) { }

- **Parameters**

Parameter	Type	Description
Row	Long	Row index of the cell
Col	Long	Column index of the cell

- **Example**

```
//Event fires after a column is moved.

function mySheet_OnBeforeEdit(Row, Col) {

      alert("Start editing.");

}
```

OnBeforeExpand Event

- **Purpose**

  In a tree structure, event fires when a tree expands or collapses upon clicking.

- **Syntax**

Syntax	**function Object ID_OnBeforeExpand**(Row, Expand) { }

- **Parameters**

Parameter	Type	Description
Row	Long	Row index of the cell
Expand	Long	0 : Expand 2 : Collapse

- **Example**

```
function mySheet_OnBeforeExpand(Row, Expand) {
    if(Row == 1 && Expand == 2) {
        mySheet.SetAllowExpand(0);
        alert("You can expand Row 1 but cannot collapse it.");
    }
}
```

OnBeforePaste Event

- **Purpose**

  Event fires promptly before data are pasted into IBSheet.

- **Syntax**

Syntax	**function Object ID_OnBeforePaste**(text) { }

- **Parameters**

Parameter	Type	Description
text	String	Text to paste into IBSheet

- **Remarks**

  This event will fire before data are pasted: you can cancel pasting or edit the text before pasting as necessary. Pasting will be calceled in the event of a false return. When a string is returned, it will replace the paste target text.

➢ **Example**

```
function mySheet_OnBeforePaste(text) {

    if(text == “not allow text”) {

        alert("Pasting is cancelled.");

        return false;

    }

}



function mySheet_OnBeforePaste(text) {

    if(text == “not allow text”) {

        return “allow text”;

    }

}
```

OnBeforeSave Event

➢ **Purpose**

Event fires promptly before Ajax communication when a saving method is called.

➢ **Syntax**

Syntax	**function Object ID_OnBeforeSave**() { }

➢ **Parameters**

Parameter	Type	Description
N/A		

➢ **Remarks**

Event fires before Ajax communication when DoSave or DoAllSave method is called.

Use this event when you want to customize the saving waiting image.

- **Example**

```
function mySheet_OnBeforeSave() {

    alert("Saving.");

}
```

OnBeforeSearch Event

- **Purpose**

  Event fires promptly before Ajax communication when a search method is called.

- **Syntax**

Syntax	**function Object ID_OnBeforeSearch**() { }

- **Parameters**

Parameter	Type	Description
N/A		

- **Remarks**

  Event fires before Ajax communication when DoSearch, DoSearchChild, DoSearchPaging or DoRowSearch method is called.

  Use this event when you want to customize the search waiting image.

- **Example**

```
function mySheet_OnBeforeSearch() {

    alert("Search is in progress.");

}
```

OnCellDropEnd Event

- **Purpose**

  Event fires upon drop after the user starts dragging cells

- **Syntax**

Syntax	**function Object ID_OnCellDropEnd**(FromSheet, FromRow, FromCol, ToSheet, ToRow, ToCol, X, Y) { }

- **Parameters**

Parameter	Type	Description
FromSheet	Object	Drag location sheet object
FromRow	Long	Row index of the drag location sheet object
FromCol	Long	Column index of the drag location sheet object
ToSheet	Object	Drop location sheet object
ToRow	Long	Row index of the drop location sheet object
ToCol	Long	Column index of the drop location sheet object
X	Long	X coordinate of the drop location
Y	Long	Y coordinate of the drop location

- **Example**

Set the drag cell value into the cell at the drop target location.

```
function mySheet_OnCellDropEnd(FromSheet, FromRow, FromCol, ToSheet, ToRow,
ToCol) {

    var bValue = ToObj.GetCellValue(ToRow, ToCol);

    var aValue = Obj.GetCellValue(Row, Col);

    if (ToRow < 0) {

        ToRow = ToObj.DataInsert(ToRow);
```

```
            ToCol = ToObj.MouseCol();
        }
        if (ToObj && ToRow > 0 && ToCol >= 0) {
            ToObj.SetCellValue(ToRow, ToCol, aValue);
            if (bValue) {
                Obj.SetCellValue(Row, Col, bValue);
            } else {
                Obj.SetCellValue(Row, Col, "");
            }
        }
}
```

OnChange Event

- **Purpose**

  Event fires when the cell editing is completed and the previous value has been updated.

  In addition to value update by the user, other methods may also fire this event for each of the updated Checkbox in each data row, such as when CellValue method is used, when the Check All checkbox value has been updated in the header area, or CheckAll function is used. If the flag parameter value of CellValue method is set as 0, this event will not fire.

- **Syntax**

Syntax	**function Object ID_OnChange**(Row, Col, Value, OldValue, RaiseFlag) { }

- **Parameters**

Parameter	Type	Description
Row	Long	Row index of the cell
Col	Long	Column index of the cell
Value	String	Updated value;

		Value used for saving without formatting
OldValue	String	Value before update
RaiseFlag	Integer	Event fire option (0: manual editing, 1: method, 2: paste)

- **Example**

```
function mySheet_OnChange(Row, Col, Value) {
    if(Col == 3 && mySheet.GetCellValue(Row, 2) == 'Korean won' && Value
== '9') {
        alert("You should select code 10 for Korean won.");
    }
}
```

OnChangeFilter Event

- **Purpose**

  Event fires when a cell value in the filter row or an option has been updated.

  In addition to value editing by the user, use of SetFilterOption function also fires this event.

- **Syntax**

Syntax	**function Object ID_OnChangeFilter**() { }

- **Parameters**

Parameter	Type	Description
None		

- **Example**

```
//When SearchMode is set as 3, search if filtering condition is changed
function mySheet_OnChangeFilter() {
  // Filter row QueryString update

  var fp = mySheet.GetFilterParam(0,1);

  var info = {PageParam: "page", Param: "id=ibleaders&seq=1&" + fp};

  mySheet.DoSearchPaging("list.jsp",info);

}
```

OnChangeSum Event

- **Purpose**

  Event fires when the value in the sum row changes.

- **Syntax**

Syntax	**function Object ID_OnChangeSum**(Row,Col) { }

- **Parameters**

Parameter	Type	Description
Row	Long	Index of the row above the sum row
Col	Long	Col index of the updated cell

- **Example**

```
function mySheet_OnChangeSum(Row,   Col) {

    //When the value in the sum row changes, display calculation result in another cell of the same row:

    mySheet.SetSumText(2, mySheet.GetSumValue(Col) / 100 + " %");

}
```

OnCheckAllEnd Event

- **Purpose**

  Event fires when CheckAll action is completed for a checkbox type column.

➢ **Syntax**

Syntax	**function Object ID_OnCheckAllEnd**(Col, Value) { }

➢ **Parameters**

Parameter	Type	Description
Col	Integer	Index of the column
Value	Boolean	Checked or not

➢ **Example**

```
function mySheet_OnCheckAllEnd(Col, Value) {

    alert(CheckAll action for Col + "th column has been completed.");

}
```

OnClick Event

➢ **Purpose**

Event fires when user clicks a cell in data area.

➢ **Syntax**

Syntax	**function Object ID_OnClick**(Row, Col, Value, CellX, CellY, CellW, CellH) { }

➢ **Parameters**

Parameter	Type	Description
Row	Long	Row index of the cell
Col	Long	Column index of the cell
Value	String	Cell value where the event fired
CellX	Long	X coordinate of the cell

CellY	Long	Y coordinate of the cell
CellW	Long	Width of the cell
CellH	Long	Height of the cell

- **Example**

```
function mySheet_OnClick(Row, Col, Value, CellX, CellY, CellW, CellH) {

    //Set to move to another page when a row is clicked

    location.href = "link.jsp?key=" + Value;

}
```

OnDblClick Event

- **Purpose**

  Event fires when the user double clicks a cell in the data area.

- **Syntax**

Syntax	**function Object ID_OnDblClick**(Row, Col, Value, CellX, CellY, CellW, CellH) { }

- **Parameters**

Parameter	Type	Description
Row	Long	Row index of the cell
Col	Long	Column index of the cell
Value	String	Value of the cell
CellX	Long	X coordinate of the cell
CellY	Long	Y coordinate of the cell
CellW	Long	Width of the cell
CellH	Long	Height of the cell

➢ **Example**

```
function mySheet_OnDblClick(Row, Col, CellX, CellY, CellW, CellH) {

    //Set to move to another page when a row is double-clicked

    location.href = "link.jsp?key=" + mySheet.GetCellValue(Row, Col + 1) ;

}
```

OnDebugMsg Event

➢ **Purpose**

Event fires when a debugging message occurs during any processing.

If ShowDebugMsg property is set as 0, debugging message will be called through this event. If the property is set as 1, the message will appear in an pop-up window for the user to see.

➢ **Syntax**

Syntax	**function Object ID_OnDebugMsg**(Msg) { }

➢ **Parameters**

Parameter	Type	Description
Msg	String	Debug message

➢ **Example**

```
function mySheet_OnDebugMsg(Msg) {

     txtErr.value = txtErr.value + "₩n>>>>" + Msg;

}



//Create a text area to display debug message

<textarea name="txtErr" rows=10 cols=70></textarea>
```

OnDecryption Event

- **Purpose**

  Event fires for each cell or EtcData element when you run a search and need to edit data before they populate cells, or interface the data with encryption module.

  .

- **Syntax**

Syntax	**function Object ID_OnDecryption**(Row,Col,SaveName,Value){}

- **Parameters**

Parameter	Type	Description
Row	Long	Row index of the cell (-1 for EtcData)
Col	Long	Col index of the cell (-1 for EtcData)
SaveName	String	SaveName of the column or KEY of EtcData
Value	String	Data to populate cells or EtcData Value

- **Remarks**

  If you need to display B instead of the actual value A you retrieved through search,

  or if data are encrypted and need decryption to display, you can use this event.

  See the source below for detailed instructions.

- **Example**

```
function mySheet_OnDecryption(Row, Col, SaveName, Value){

/* Call and return the Dec user method which performs decryption using
value as a parameter.

   return Dec(Value);

}
```

OnDownFinish Event

- **Purpose**

Event fires when excel or text file download is complete.

.

- **Syntax**

Syntax	**function Object ID_OnDownFinish**(downloadType, result){}

- **Parameters**

Parameter	Type	Description
downloadType	String	File format between excel and text: "EXCEL" or "TEXT"
result	Boolean	Download error flag. Success: true, Failure: false

- **Remarks**

- **Example**

```
function mySheet_OnDownFinish(downloadType, result) {

    alert(downloadType + 'Download is complete. Download result: ' +
result);

}
```

OnDragStart Event

- **Purpose**

Event fires when the user starts dragging one or more rows or cells.

- **Syntax**

Syntax	**function Object ID_OnDragStart**(Row, Col) { }

- **Parameters**

Parameter	Type	Description
Row	Long	Row index of the drag location
Col	Long	Column index of the drag location

- **Example**

```
var dragValue = "";

function mySheet_OnDragStart(Row, Col) {

    // Save the CellValue of the position where drag started

    dragValue = mySheet.GetCellValue(Row, Col) ;

}
```

OnDropEnd Event

- **Purpose**

  Event fires upon drop after the user starts dragging one or more rows.

  Among event parameters, the Type parameter will deliver detailed drop location value in a tree structure.

  Type parameter may have the following values:

Type	Description
1	Upper part of the Drop location row
2	Middle part of the Drop location row
3	Lower part of the drop location row

- **Syntax**

Syntax	**function Object ID_OnDropEnd**(FromSheet, FromRow, ToSheet, ToRow, X, Y, Type) { }

- **Parameters**

Parameter	Type	Description
FromSheet	Object	Drag location sheet object
Row	Long	Row index of the drag location sheet object
ToSheet	Object	Drop location sheet object
ToRow	Long	Row index of the drop location sheet object
X	Integer	X coordinate of the drop location
Y	Integer	Y coordinate of the drop location
Type	Integer	Drop location type in case of a tree structure

- **Example**

```
// Add the drag row to the drop location and delete from the drag sheet.

function mySheet_OnDropEnd(FromSheet, FromRow, ToSheet, ToRow, X, Y, Type) {

    var NewRow = ToObj.DataInsert(ToRow);

    for (var c = 0; c <= Obj.LastCol(); c++) {

      ToObj.SetCellValue(NewRow, c, Obj.GetCellValue(Row, c));

    }

    Obj.RowDelete(Row);

}
```

OnEditValidation Event

- **Purpose**

  Event fires when cell editing completes so that validation check may run for the updated value.

  If validation fails, set as ValidateFail(1) and revert to the original values.

- **Syntax**

Syntax	**function Object ID_OnEditValidation**(Row, Col, Value) { }

- **Parameters**

Parameter	Type	Description
Row	Long	Row index of the cell
Col	Long	Column index of the cell
Value	Variant	This value is used for saving without a designated format.

- **Example**

```
function mySheet_OnEditValidation(Row, Col, Value) {
 switch(Col) {
  case 2:
  if(Value=="Korean won" && mySheet.GetCellValue(Row,Col+1) >=
  10000000) {
     alert("When the currency is Korean won, the amount cannot exceed ten
  million Korean won.");
     mySheet.ValidateFail(1);
  } else if(Value=="Foreign currency" && mySheet.GetCellValue(Row,Col+1) <
  10000000) {
     alert("When the currency is a foreign one, the amount must be ten
  million or greater.");
     mySheet.ValidateFail(1);
  }
 }
}
```

OnEncryption Event

- **Purpose**

When IBSheet saves data, any change will be adjoined into a QueryString. This event may be used when the data requires encryption or the user needs to change a value before saving.

- **Syntax**

Syntax	**function Object ID_OnEncryption(Row, Col, Value) { }**

- **Parameters**

Parameter	Type	Remark
Row	Long	Row Index of the cell
Col	Long	Col Index of the cell
SaveName	String	SaveName of the column
Value	String	Current data in the cell

- **Remarks**

If you need validation check before saving data, you may use OnValidation event to determine whether to save the data or not if validation fails. However, if you must save the data regardless of the validation result by changing any invalid values found during validation, you may use this event. See the source below for detailed instructions.

- **Example**

```
function mySheet_OnEncryption(Row, Col, SaveName, Value){

/* Call and return the Enc user function which performs encryption using the
values to save as a parameter.

   return Enc(Value);

}
```

OnFilterEnd Event

- **Purpose**

Event fires after filtering is completed.

- **Syntax**

Syntax	**function Object ID_ OnFilterEnd**(RowCnt, FirstRow) { }

- **Parameters**

Parameter	Type	Description
RowCnt	Long	Number of rows after filtering
FirstRow	Long	Index of the first row post filtering

- **Example**

```
function mySheet_ OnFilterEnd(RowCnt, FirstRow) {

}
```

OnHScroll Event

- **Purpose**

Event fires upon horizontal scroll.

- **Syntax**

Syntax	**function Object ID_OnHScroll**(hpos, oldhpos, isLeft, isRight, section) { }

- **Parameters**

Parameter	Type	Description
hpos	Long	Horizontal scroll position
oldhpos	Long	Previous horizontal scroll position
isLeft	Boolean	Whether the horizontal scroll bar is located at the far left
isRight	Boolean	Whether the horizontal scroll bar is located at the far right
section	Int	Scroll section value (Always return 1)

- **Example**

```
function mySheet_OnHScroll(hpos, oldhpos, isLeft, isRight, section) {

}
```

OnKeyDown Event

- **Purpose**

  Event fires when a cell value is being edited or a key is pressed on a selected cell.

  As KeyCode is an ASCII value, transcode before use.

- **Syntax**

Syntax	**function Object ID_OnKeyDown**(Row, Col, KeyCode, Shift) { }

- **Parameters**

Parameter	Type	Description
Row	Long	Row index of the cell
Col	Long	Column index of the cell
KeyCode	Integer	ASCII value of the keyboard
Shift	Integer	1 : When a Shift key is pressed 2 : When a Ctrl key is pressed 0 : Others

- **Example**

```
function mySheet_OnKeyDown(Row, Col, KeyCode, Shift) {

    //If the Enter key is pressed in the last column, put focus on the first part of the next row.

    if(KeyCode == 13 && Col == mySheet.LastCol()) {

        mySheet.SelectCell(Row + 1, 2);
    }
}
```

OnKeyUp Event

- **Purpose**

  Event fires when a cell value is being edited or a pushed key is unpressed on a selected cell. This event immediately follows OnKeyDown event.

  As KeyCode is an ASCII value, transcode before use.

- **Syntax**

Syntax	**function Object ID_OnKeyUp**(Row, Col, KeyCode, Shift) { }

- **Parameters**

Parameter	Type	Description
Row	Long	Row index of the cell
Col	Long	Column index of the cell
KeyCode	Integer	ASCII value of the keyboard
Shift	Integer	1 : When a Shift key is pressed 2 : When a Ctrl key is pressed 0 : Others

- **Example**

```
function mySheet_OnKeyUp(Row,Col,KeyCode,Shift) {
    //If the Enter key is pressed in the last column, put focus on the first part of the next row.
    if(KeyCode ==13 && Col == mySheet.LastCol()
                    && Row < mySheet.RowCount()) {
        mySheet.SelectCell(Row + 1, 2);
```

```
   }

}
```

OnLoadData Event

- **Purpose**

  Event fires when the data received from the server are loaded to IBSheet after a data search method or a data save method is called.

  This event can be used when the data received from the server requires editing or interface with encryption module.

- **Syntax**

Syntax	**function Object ID_OnLoadData**(Data) { }

- **Parameters**

Parameter	Type	Description
Data	String	Search XML or JSON charcter string/object

- **Example**

```
function mySheet_OnLoadData(data) {

      // Decryption

      var decrypt_data = fnDecryption(data);



      // Return decrypted data

      return decrypt_data;

}
```

OnLoadExcel Event

- **Purpose**

  Event fires after LoadExcel has been completed.

- **Syntax**

Syntax	**function Object ID_OnLoadExcel**(result) { }

- **Parameters**

Parameter	Type	Description
result	Boolean	Loading result. Success : true, Failure : false

- **Example**

```
function mySheet_OnLoadExcel(result) {

   if(result) {

      alert('Excel file loading is completed.');

   } else {

      alert('An error occured while loading an excel file.');

   }

}
```

OnLoadText Event

- **Purpose**

  Event fires after LoadText has been completed.

- **Syntax**

Syntax	**function Object ID_OnLoadText**(result) { }

- **Parameters**

Parameter	Type	Description
result	Boolean	Loading result. Success : true, Failure : false

- **Example**

```
function mySheet_OnLoadText(result) {

   if(result) {

      alert('Text file loading is completed.');

   } else {

      alert('An error occured while loading a text file.');

   }

}
```

OnMessage Event

- **Purpose**

  In case SetShowMsgMode is set as 0, a confirmation or warning message will fire this event instead of displaying a system pop-up. In case the trigger was a confirmation message, IsConfirm = 1. Always use ConfirmOK function to return a response to the sheet for such cases.

- **Syntax**

Syntax	**function Object ID_OnMessage**(Msg, Level, IsConfirm) { }

➢ **Parameters**

Parameter	Type	Description
Msg	String	Message
Level	String	Message level code "U" – message is for end users "E" – message is for developers "D" – message is concerned with server connection function page "X" – MXL parser message of inquiry or save XML
IsConfirm	Boolean	Confirmation message or not

➢ **Example**

```
//Configure the message mode.
mySheet.ShowMsgMode=0;


//Process OnMessage event.
function mySheet_OnMessage(Msg, Level, IsConfirm) {
     //Display message
     var win_result = window.showModalDialog(
         "sheet_message.jsp?Msg=" + Msg + "&IsConfirm=" + IsConfirm,
          'modalResult',
          'dialogWidth:200px;dialogHeight:200px;center:yes;help:no;status:no;');
     //Return the message response to the sheet.
     if(IsConfirm) mySheet.ConfirmOK(win_result);
}
```

OnMouseDown Event

- **Purpose**

  Event fires when mouse is clicked.

  To find out cell location on which the mouse is clicked, use MouseRow and MouseCol functions.

- **Syntax**

Syntax	**function Object ID_OnMouseDown**(Button, Shift, X, Y) { }

- **Parameters**

Parameter	Type	Description
Button	Integer	0 : Left, 2 : Right (Mouse button)
Shift	Integer	1 : When a Shift key is pressed 2 : When a Ctrl key is pressed 0 : Others
X	Long	X coordinate
Y	Long	Y coordinate

- **Example**

```
function mySheet_OnMouseDown(Button, Shift, X, Y) {
    //Check the column of the clicked cell
    alert(mySheet.MouseRow()  +  "Row"  +  mySheet.MouseCol()  +
"Column is clicked");
}
```

OnMouseMove Event

- **Purpose**

  Event fires when mouse pointer is moving on the sheet.

  To find out the cell location on which mouse pointer is moving, use MouseRow and MouseCol functions.

- **Syntax**

Syntax	**function Object ID_OnMouseMove**(Button, Shift, X, Y) { }

- **Parameters**

Parameter	Type	Description
Button	Integer	0 : Left, 2 : Right (Mouse button)
Shift	Integer	1 : When a Shift key is pressed 2 : When a Ctrl key is pressed 0 : Others
X	Long	X coordinate
Y	Long	Y coordinate

- **Example**

```
function mySheet_OnMouseMove(Button, Shift, X, Y) {

  //Fetch the row and column values of the mouse pointer location.

  var Row = mySheet.MouseRow();

  var Col = mySheet.MouseCol();

  var sText = mySheet.GetCellText(Row, Col);



  //Set the mouse pointer style

  //Set a hand mouse pointer only for Column 2 with cell text 2011-07-14

  if(Col == 2 && sText == "2011-07-14") {

      mySheet.SetMousePointer("Hand");

  } else {

      mySheet.SetMousePointer("Default");   // Default

  }
}
```

OnMouseUp Event

- **Purpose**

  Event fires when a clicked mouse button is unclicked.

  To find out the cell location on which the mouse is unclicked, use MouseRow and MouseCol methods.

- **Syntax**

Syntax	**function Object ID_OnMouseUp**(Button, Shift, X, Y) { }

- **Parameters**

Parameter	Type	Description
Button	Integer	0 : Left, 2 : Right (Mouse button)
Shift	Integer	1 : When a Shift key is pressed 2 : When a Ctrl key is pressed 0 : Others
X	Long	X coordinate
Y	Long	Y coordinate

- **Example**

```
function mySheet_OnMouseUp(Button, Shift, X, Y) {

    //Check the column where the mouse is unclicked

    alert(mySheet.MouseRow() + "Row" + mySheet.MouseCol() + "Column is
clicked");

}
```

OnPageRequest Event

- **Purpose**

  Event fires for server paging searches when a request for further loading is sent to the server in order to reload result page as the scroll reaches the end.

- **Syntax**

Syntax	**function Object ID_OnPageRequest**(page) { }

- **Parameters**

Parameter	Type	Description
page	Integer	Page number to request

- **Example**

```
function mySheet_OnPageRequest(page) {

      //Display a warning before page request is sent to the server.

      alert(page + " page will be requested to the server.");

}
```

OnPopupClick Event

- **Purpose**

  Event fires when user clicks the pop-up button in the cell that appears when focus is put on the cell, or tries to edit the cell, given that a cell type is either Popup or PopupEdit.

  User should add a popup invoking logic to this event to enable it.

- **Syntax**

Syntax	**function Object ID_OnPopupClick**(Row, Col) { }

- **Parameters**

Parameter	Type	Description

Row	Long	Row index of the cell
Col	Long	Column index of the cell

- **Example**

```
function mySheet_OnPopupClick(Row,Col) {

      //Open pop-up.

      window.open("Sheet_Popup.jsp?row="+Row+"&col="+Col,   "list",

                              "scrollbars=no,fullscreen=no,width=250,height=350");

}
```

OnReadyExcelDown Event

- **Purpose**

  After calling Down2Excel to download data into excel format and downloaded data are fetched, event fires before the form is submitted. Excel download can be successfully completed only when the form is submitted within this event.

- **Syntax**

Syntax	**function Object ID_OnReadyExcelDown**(frm) { }

- **Parameters**

Parameter	Type	Description
frm	Object	Html Form object

- **Example**

```
function mySheet_OnReadyExcelDown(frm) {

      //Submit the excel download form.

      frm.submit();

}
```

OnResize Event

- **Purpose**

  Event fires when the IBSheet width or height has been changed when the width is set as a relative value to height in percentage.

- **Syntax**

Syntax	**function Object ID_OnResize**(Width, Height) { }

- **Parameters**

Parameter	Type	Description
Width	Integer	Overall width
Height	Integer	Overall height

- **Example**

```
function mySheet_OnResize(Width, Height) {

    //Readjust the width of columns according to updated information.

    mySheet.FitColWidth();

}
```

OnRowSearchEnd Event

- **Purpose**

  Event fires for each row while data is being searched using DoSearch or LoadSearchData methods.

  This event comes in handy when you want to color font or background of certain cells based on the data contained in the row.

  This event will fire for every row during search. If it is used for complicated event logic or loop statement, it may result in slowness of performance.

➢ **Syntax**

Syntax	**function Object ID_OnRowSearchEnd** (row) { }

➢ **Parameters**

Parameter	Type	Description
row	Integer	Index of the row

➢ **Example**

```
function mySheet_OnRowSearchEnd(row) {
    //Change the font color of column 6 to red if column 3 is checked and
  column 4 value is bigger than 100.
    if( mySheet.GetCellValue(row,3) == 1 && mySheet.GetCellValue(row,4) >
  100){
        mySheet.SetCellFontColor(row ,6 ,"#FF0000");
    }
 }
```

OnSaveEnd Event

➢ **Purpose**

Event fires when saving is completed using saving function and other internal processing has been also completed.

If an error message occurs during saving, it will be set as code, a event parameter. Program an error processing logic for any code value smaller than 0.

If no result is returned due to network error, send the code value as -3.

This event can fire when DoSave or DoAllSave function is called.

➢ **Syntax**

Syntax	**function Object ID_OnSaveEnd**(Code, Msg, StCode, StMsg) { }

➢ **Parameters**

Parameter	Type	Description
Code	Long	Processing result code (0 or higher is success, others should be processed as error)
Msg	String	Processing result message
StCode	Integer	HTTP response code
StMsg	String	HTTP response message

➢ **Example**

```
function mySheet_OnSaveEnd(code, msg) {

      if(code >= 0) {

            alert(msg);   // saving success message

            mySheet.DoSearch("list1.jsp");

      } else {

            alert(msg);   // saving failure message

            }

}
```

```
<?xml version='1.0' ?>

<SHEET><Result Code="-1" Message= "The slip has been posted and cannot be edited."/></SHEET>
```

```
<?xml version='1.0' ?>

<SHEET><Result Code = "0" Message = "" / ></SHEET>
```

OnSearchEnd Event

➢ **Purpose**

Event fires when search is completed using a search function and other internal data processing are also completed.

If an error message occurs during searching, it will be set as code, an event parameter. Program an error processing logic for any code value smaller than 0.

If no result is returned due to network error, set the code value as -3.

- **Syntax**

Syntax	**function Object ID_OnSearchEnd**(Code, Msg, StCode, StMsg) { }

- **Parameters**

Parameter	Type	Description
Code	Long	Processing result code (0 is success, others should be processed as error)
Msg	String	Processing result message
StCode	Integer	HTTP response code
StMsg	String	HTTP response message

- **Example**

```
function mySheet_OnSearchEnd(code, message) {

    if(code == 0) {

        alert(message);

        //Perform post-search tasks

    } else {

        alert("An error occured during search..");

    }

}
```

OnSelectMenu Event

- **Purpose**

Given the menu displayed is not a column popup menu but the one set in ActionMenu, event fires when the user selects a certain menu item.

Texts returned by this event are the same as the characters displayed in the menu list, and they perform the corresponding functions.

- **Syntax**

Syntax	**function Object ID_OnSelectMenu**(Text, Code) { }

- **Parameters**

Parameter	Type	Description
Text	String	Selected menu string
Code	String	Selected menu code string

- **Example**

```
// Menu setting

mySheet.SetActionMenu("Insert|Copy row|-|Delete row|Clear|Download to
excel");



function mySheet_ OnSelectMenu(Text, Code) {

   // Perform action using the text or code value

   switch(text) {

            case "Insert data to the first row" :

                 mySheet.DataInsert(0);

                 break;



            case "Insert data to the last row" :

                 mySheet.DataInsert(-1);
```

```
                break;


            case "Insert" :

                mySheet.DataInsert();

                break;


            case "Copy row":

                mySheet.DataCopy();

                break;


            case "Delete row":

                mySheet.RowDelete();

                break;


            case "Clear": //RemoveAll

                        mySheet.RemoveAll();

                        break;


            case "Download to excel":

                        mySheet.Down2Excel();

                        break;


            Default=

                        break;

            }

}
```

OnSelectCell Event

- **Purpose**

  Event fires when a certain cell is selected.

  This returns both the previously selected cell and the newly selected cell information, so you can create logics for the selected cells.

- **Syntax**

Syntax	**function Object ID_OnSelectCell**(OldRow, OldCol, NewRow, NewCol,isDelete) { }

- **Parameters**

Parameter	Type	Description
OldRow	Long	Row Index of the previously selected cell
OldCol	Long	Column Index of the previously selected cell
NewRow	Long	Row Index of the currently selected cell
NewCol	Long	Column Index of the currently selected cell
isDelete	Boolean	When a newly added row is deleted using DelCheck, OnSelectCell event would not recognize the deletion and fires for the location. This value is to mark the deleted row.

- **Example**

```
function mySheet_OnSelectCell(OldRow, OldCol, NewRow,
NewCol,isDelete) {

   alert("Cell("+ OldRow + "," + OldCol + ")has been deselected and
now cell(" + NewRow + "," + NewCol +")is selected.");

}
```

OnSmartResize Event

- **Purpose**

  Event fires when there is no recurrence of change in IBSheet width or height from the initial change for a certain length of time (300ms). If you need FitColWidth upon Resizing, using this instead of OnResize event will help improve performance.

  This event will fire only when "SmartResize" property is set as 0 (false) in a global or SetConfig method. If "SmartResize" property is set as 1 (true), OnResize event will fire the same way as explained above instead of this event.

- **Syntax**

Syntax	**function Object ID_OnSmartResize**(Width, Height) { }

- **Parameters**

Parameter	Type	Description
Width	Integer	Overall width
Height	Integer	Overall height

- **Example**

```
function mySheet_OnSmartResize(Width, Height) {

    //Readjust the width of columns according to updated information.

    mySheet.FitColWidth();

}
```

OnSort Event

- **Purpose**

  Event fires when data sorting by clicking the header is completed. Col parameter is column index number. SortArrow is the sorting direction, which is expressed as in the table below. If SortEventMode is set as 1, column information of all sorted columns will be adjoined with “|” and returned.

SortArrow	Description

"ASC"	Ascending order
"DESC"	Descending order

- **Syntax**

Syntax	**function Object ID_OnSort**(Col, SortArrow) { }

- **Parameters**

Parameter	Type	Description
Col	Long	Index of sorted columns Adjoin with "|" if SortEventMode is 1
Arrow	String	Sort direction string Adjoin with "|" if SortEventMode is 1

- **Example**

```
function mySheet_OnSort(Col, SortArrow) {

   if(SortArrow =="ASC")

         alert(Col + "th column has been sorted in the ascending order.");

   else

         alert(Col + "th column has been sorted in the descending order.");

}
```

OnTreeChild Event

- **Purpose**

Event fires when you select to extend tree on a parent node without searching the children.

You can search the children nodes where this event fires using DoSearchChild function.

To run a search on this event, set <TR HAVECHILD="1"> in XML file and set TR HaveChild parameter value in JSON file in advance.

This event will fire only for the data with HaveChild value so make sure child level information is set when creating a search XML or JSON.

This event may be also used when you do not need to search the entire tree of tree-type data and instead want to search child level data only when you click the expand button.

- **Syntax**

Syntax	**function Object ID_OnTreeChild**(Row)

- **Parameters**

Parameter	Type	Description
Row	Long	Index of the parent row to expand

- **Example**

```
// Search the child data

<script type="text/Javascript">

 function mySheet_OnTreeChild(Row){

    var url = "";

    // Column 4 : Tree column

    switch(mySheet.GetCellValue(Row, 4)){

      case "Seoul" :

         url = " type15_dat(1).xml";

         break;

      case "Incheon":
         url = "type15_data(2).xml";

         break;
    }

    mySheet.DoSearchChild(Row, url, "", 1);

  }

</script>
```

OnUserResize Event

- **Purpose**

  Event fires when the end user changes the column width on the header using mouse.

- **Syntax**

Syntax	**function Object ID_OnUserResize**(Col, Width) { }

- **Parameters**

Parameter	Type	Description
Col	Long	Index of the column
Width	Long	Width of the column

- **Example**

```
function mySheet_OnUserResize(Col, Width) {

    alert(Col + "The column width has been changed to " + Width + "".)

}
```

OnValidation Event

- **Purpose**

  Event fires before you call a saving function to process saving to enable validation check for the data to save.

  Some basic required field values or overall entry checks are supported by IBSheet before saving, but this event can be used for various validations that may be required for specific business logics.

Set ValidateFail as 1 to stop saving when validation fails.

As this event will fire for each data cell, if you use “For” to run validation for all cells, it may take too much time. Run any validation for the entire data before saving function is called, and restricts use of this event to logics in individual cells.

- **Syntax**

Syntax	**function Object ID_OnValidation**(Row, Col, Value) { }

- **Parameters**

Parameter	Type	Description
Row	Long	Row index of the cell
Col	Long	Column index of the cell
Value	Variant	This value is used for saving without a designated format.

- **Example**

```
function mySheet_OnValidation(Row, Col, Value) {
 switch(Col) {
  case 2:
  if(Value=="Korean won" && mySheet.GetCellValue(Row,Col+1) >=
  10000000) {
     alert("When the currency is Korean won, the amount cannot exceed ten
  million Korean won.");
     mySheet.ValidateFail(1);
     mySheet.SelectCell(Row, Col+1);
  } else if(Value=="Foreign currency" && mySheet.GetCellValue(Row,Col+1) <
  10000000) {
     alert("When the currency is a foreign one, the amount must be ten
  million or greater.");
     mySheet.ValidateFail(1);
     mySheet.SelectCell(Row, Col+1);
  }
 }
}
```

OnVScroll Event

- **Purpose**

  Event fires upon vertical scroll.

- **Syntax**

Syntax	**function Object ID_OnVScroll**(vpos, oldvpos, isTop, isBottom) { }

- **Parameters**

Parameter	Type	Description
vpos	Long	Vertical scroll position
oldvpos	Long	Previous vertical scroll position
isTop	Boolean	Whether the vertical scroll bar is located at the topmost position
isBottom	Boolean	Whether the vertical scroll bar hit the bottom

- **Example**

```
function mySheet_OnVScroll(vpos, oldvpos, isTop, isBottom) {
}
```

OnWaitTimeOut Event

- **Purpose**

  Event fires when a server process has been aborted because of timeout.

- **Syntax**

Syntax	**function Object ID_OnWaitTimeOut**(Sec) { }

- **Parameters**

Parameter	Type	Description
Sec	Long	Time until timeout (unit: second)

- **Example**

```
function mySheet_OnWaitTimeOut(Sec) {

  // in case when a timeout occurs

  alert("Waiting time has run out. ");

}
```

Chapter 2. IBSheet Methods

2. IBSheet Methods

2.1 Using methods

You can use the following syntax to use methods offered by IBSheet.

Syntax	Object ID.Method name()

Any method parameters set between **[~]** are optional. If you call the method without setting those parameter, default values will apply.

The next section lists all the methods offered by IBSheet.

2.2 Method List

ActionMenu Method

- **Purpose**

  Check or configure menus used to perform certain actions, not a menu popup used to change values of a certain column like a column pop-up Menu. The action menu will appear as a pop-up upon mouse right-click. If a column pop-up is set for the column in the mouse click location, column pop-up will display. If you set menu text as "*-" and connect the menu option text with "|", a section separator will be inserted into the menu.

  OnSelectMenu event fires when a menu option is selected from the displayed action menu.

  If you set both the text and code or set the configuration parameter as a JSON object, GetActionMenu return value will be returned to the JSON object.

- **Syntax**

Syntax	Get	ObjId.**GetActionMenu**()

- **Info**

Return	String(Object), Set pop-up menu (In case of Get Method)

Parameter	Type	Required Y/N	Description

- **Example**

```
// Set menu only

mySheet.GetActionMenu();
```

- **Syntax**

Syntax	Set	ObjId.**SetActionMenu**(Text, Code)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Text	String	Required	Pop-up menu string to set
Code	String	Optional	Pop-up menu code string to set Default="Text charcter string"

- **Example**

```
// Set menu only

mySheet.SetActionMenu("Insert|Copy   row|*-|Delete   row|Clear|Download   to
excel");



// Set menu and code

var Text = "Insert|Copy row|*-|Delete row|Clear|Download to excel";

var Code = "Ins|Copy||Del|Clear|Download";

mySheet.SetActionMenu(Text, Code);
```

```
// Set JSON object
var Menu = [
      {Text: "Insert", Code: "Ins"},
      {Text: "Copy row", Code: "Copy"},
      {Text: "*-"},
      {Text: "Delete row", Code: "Delete"},
      {Text: "Clear", Code: "Clear"},
      {Text: "Download to excel", Code: "Download"}
];
mySheet.SetActionMenu(Menu);


// Set JSON object hierarchy
var Menu = [
      {Text: "Insert", Code: "Ins",
         Items : [
            { Text: "Insert first row", Code: "FIns"},
            { Text: "Insert last row", Code: "LIns"}
         ]
      },
      {Text: "Copy row", Code: "Copy"},
      {Text: "*-"},
      {Text: "Delete row", Code: "Delete"},
      {Text: "Clear", Code: "Clear"},
      {Text: "Download to excel", Code: "Download"}
];
```

```
mySheet.SetActionMenu(Menu);
```

AllowCheck Method

- **Purpose**

  Check or configure whether to accept a change when the user changes checkbox value.

  When OnBeforeCheck Event fires, you can control permission for checkbox changes using AllowCheck method.

- **Syntax**

Syntax	Get	ObjId.**GetAllowCheck**()

- **Info**

Return	Boolean, set value (In case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
// Check the checkbox value.

mySheet.GetAllowCheck();
```

- **Syntax**

Syntax	Set	ObjId.**SetAllowCheck**(Val)

- **Info**

Return	None		
Parameter	Type	Required	Description

		Y/N	
Val	Boolean	Required	Whether to accept change of value. Default=true

➢ **Example**

```
// Not change checkbox value.

mySheet.SetAllowCheck(false);
```

AllowEvent4CheckAll Method

➢ **Purpose**

Configure whether to run OnChangeEvent when CheckAll action is run. When there are many data rows, running CheckAll function will fire OnChangeEvent for individual checkboxes, which may result in delayed performance. If OnChangeEvent is not required for every checkbox, you can set AllowEvent4CheckAll as 0 to improve CheckAll performance. The default value is 1.

➢ **Syntax**

Syntax	ObjId.**AllowEvent4CheckAll**(Val)

➢ **Info**

Return	None.		
Parameter	Type	Required Y/N	Description
Val	Boolean	Required	OnChangeEvent fire Y/N

➢ **Example**

```
// Do not fire event upon CheckAll

sheetObj.AllowEvent4CheckAll(0);
```

AllowExpand Method

➢ **Purpose**

Use this method if you want to block the actual expansion or collapse right after OnBeforeExpand event fires.

AllowExpand value will be reset to 1 upon every OnBeforeExpand event.

Only when you set this value to 0 within an OnBeforeExpand event you can stop tree control of the user.

- **Syntax**

Syntax	Get	ObjId.**GetAllowExpand**()

- **Info**

Return	Boolean, set value (In case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
mySheet.GetAllowExpand();
```

- **Syntax**

Syntax	Set	ObjId.**SetAllowExpand**(Expand)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Expand	Boolean	Required	Whether to allow tree expansion

- **Example**

```
function mySheet_OnBeforeExpand(Row,Expand) {

    if(Row == 1 && Expand == 2) {

        mySheet.SetAllowExpand(0);

        alert("You can expand Row 1 but cannot collapse it.");

    }

}
```

AutoRowHeight Method

- **Purpose**

  Check or configure whether to automatically adjust data row height.

  If this is set as 1, row height will automatically heighten or lower to display all the text in the row. Row height may be adjusted in the following cases:

  1) When the MultiLineText property in InitColumns is set as MultiLineText:1 and the user set multi lines by pressing Shift + Enter keys while editing on keyboard;
  2) When the Wrap property in InitColumns is set as Warp: 1 and the text is displayed in multiple lines automatically as column width is adjusted;
  3) When a line out of multiple text lines is deleted and the overall height of the row is diminished;
  4) Multiple line display is set using SetCellText or SetCellValue; or
  5) When multi line text is loaded by search or upload from excel.

  If this method is set as 0, row height will be fixed to single line height configured in SetDataRowHeight.

- **Syntax**

Syntax	Get	ObjId.**GetAutoRowHeight**()

- **Info**

Return	Boolean, set value (In case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Check the configuration

mySheet.GetAutoRowHeight();
```

- **Syntax**

Syntax	Set	ObjId.**SetAutoRowHeight**(Flag)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Flag	Boolean	Required	Whether to automatically adjust data row height Default=1

- **Example**

```
// Fix row height to single line and disallow automatic height adjustment.

mySheet.SetAutoRowHeight(0);
```

AutoSumPosition Method

- **Purpose**

  Check or configure the display location of the sum/average.

  If Datatype is "dtAutoSum", "dtAutoSumEx", "dtAutoAvg", or "dtAutoAvgEx", the sum or average value is calculated and displayed. The default value is 1.

  The display location is as below:

Value	Description
0	Fix the display at the top right below the header
1	Fix the display at the bottom of the screen

- **Syntax**

Syntax	Get	ObjId.**GetAutoSumPosition**()

- **Info**

Return	Integer, set value (In case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Display a Sum row at the top

mySheet.GetAutoSumPosition();
```

- **Syntax**

Syntax	Set	ObjId.**SetAutoSumPosition**(Position)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Position	Integer	Required	Location value of the sum row

- **Example**

```
//Display a Sum row at the top

mySheet.SetAutoSumPosition(0);
```

BasicImeMode Method

- **Purpose**

Check or configure the keyboard to use when focus is set on an editable cell. Values available are as follows:

Value	Value	Description
imeAuto	0	Use the latest keyboard
imeHan	1	Set the Korean keyboard as default
imeEng	2	Set the English keyboard as default

However, SetBasicImeMode() method can be used only before search or when data is initialized using RemoveAll() method. Also it works only in Internet Explorer browser.

- **Syntax**

Syntax	Get	ObjId.**GetBasicImeMode**()

- **Info**

Return	Integer, set value (In case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Use Korean keyboard as default

mySheet.GetBasicImeMode();
```

- **Syntax**

Syntax	Set	ObjId.**SetBasicImeMode**(Val)

- **Info**

Return	None		
Parameter	Type	Required	Description

		Y/N	
Val	Integer	Required	ImeMode value

- **Example**

```
//Use Korean keyboard as default

mySheet.SetBasicImeMode(1);
```

CellAlign Method

- **Purpose**

Check or configure text alignment in a cell. This simply realign what appears in the screeen unlike alignment set by InitColumns function. Values available are as follows:

Left alignment	Center alignment	Right alignment
"Left"	"Center"	"Right"

- **Syntax**

Syntax	Get	ObjId.**GetCellAlign**(Row, Col)

- **Info**

Return	String, current alignment value (in case of Get Method)		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row index of the cell
Col	Long /String	Required	Column index of the cell or SaveName

- **Example**

```
// Check the alignment value of the cell.

var align = mySheet.GetCellAlign(1, 1);
```

```
alert("Alignment value of the cell is " + align + ".");
```

- **Syntax**

Syntax	Set	ObjId.**SetCellAlign**(Row, Col, Align)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row index of the cell
Col	Long /String	Required	Column index of the cell or SaveName
Align	String	Required	Alignment value

- **Example**

```
// Change alignment of the cell to Center alignment.

mySheet.SetCellAlign(1, 1, "Center");
```

CellBackColor Method

- **Purpose**

Check or configure background color of the cell.

You can set or check a background color of a cell whether the cell is in a data row or header row. If the cell does not exist, it will not return any error message and just cancels the background color setting. Select the color using WebColor or BaSIC 16 Color string.

- **Syntax**

Syntax	Get	ObjId.**GetCellBackColor**(Row, Col)

➢ **Info**

Return	String, set color value (in case of Get Method)		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row index of the cell
Col	Long /String	Required	Column index of the cell or SaveName

➢ **Example**

```
// Check the color value.

var color = mySheet.GetCellBackColor(1,1);

alert("Color value of the cell is " + color + ".");
```

➢ **Syntax**

Syntax	Set	ObjId.**SetCellBackColor**(Row, Col, Color)

➢ **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row index of the cell
Col	Long /String	Required	Column index of the cell or SaveName
Color	String	Required	Color value

➢ **Example**

```
//Change the background color of the first cell in the header row to red

mySheet.SetCellBackColor(0, 0, "#FF0000");           // WebColor
```

```
//Set the background color of data row cells as the same color as that of the first cell in the header row

mySheet.SetCellBackColor(1,0, mySheet.GetCellBackColor(0, 0));

//Change the background color of those cells with "amt" as SaveName in the header row to red

mySheet.SetCellBackColor(0,"amt", "#FF0000");
```

CellComboItem Method

- **Purpose**

  If Combo items of a particular cell are different from those of other cells, you can customize Combo items for the cell.

  Set the Combo items of all columns using InitColumns function. Use this function only when certain cells have different Combo items from the rest.

- **Syntax**

Syntax	ObjId.**CellComboItem**(Row, Col,info)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row index of the cells
Col	Long / String	Required	Column index of the cells or SaveName
info	Object	Required	Create a string of update Combo items using "|" as separator and set values using ComboCode and ComboText.

- **Example**

```
//Changing combo items of a particular cell

var info = {"ComboCode":"President|Manager|Assistant Manager","ComboText":"A|B|C"};

mySheet.CellComboItem(1,2,info);
```

CellEditable Method

- **Purpose**

Check or configure editability of cell

Whether a cell is editable is determined by: the overall editability set in Editable; Edit configuration set by InitColumns function; RowEditable function value; transaction status; and application of automatic calculation.

All automatic calculation cells are uneditable regardless of CellEditable value.

If the transaction status is "Deleted", all data except for those whose column type is DelCheck are uneditable. Also data type of Status, Image and Seq are uneditable regardless of other settings.

Except for the above cases, editable status of data will be affected as follows by this method:

Editable	ColEditable	RowEditable	**CellEditable**	**Cell editable Y/N**
No	No impact	No impact	**No impact**	**No**
Yes	No	No	**Yes/No**	**Yes/No**
Yes	Yes	Yes/No	**Yes/No**	**Yes/No**

- **Syntax**

Syntax	Get	ObjId.**GetCellEditable**(Row, Col)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row index of the cell
Col	Long /String	Required	Column index of the cell or SaveName

- **Example**

```
//If Row 1 Column 2 is editable, change the Column 3 as editable.

if(mySheet.GetCellEditable(1, 2) == 1) {

      mySheet.SetCellEditable(1, 3, 1);

}
```

- **Syntax**

Syntax	Set	ObjId.**SetCellEditable**(Row, Col, Edit)

- **Info**

Return	Boolean, set editability value (in case of Get Method)		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row index of the cell
Col	Long /String	Required	Column index of the cell or SaveName
Editable	Boolean	Required	Whether it is editable

- **Example**

```
//If Row 1 Column 2 is editable, change the Column 3 as editable.
```

```
if(mySheet.GetCellEditable(1, 2) == 1) {

      mySheet.SetCellEditable(1, 3, 1);

}
```

CellFont Method

- **Purpose**

  Check or configure the font type, size, color, italic, bold, underline, etc. of a cell or area. Font properties available for flag are as follows:

FLAG property	Description
"FontName"	Font type, String
"FontSize"	Font size, Integer
"FontColor"	Font color, String
"FontBold"	Whether the font is bold, Boolean
"FontItalic"	Whether the font is italic, Boolean
"FontUnderline"	Whether the font is underlined, Boolean
"FontStrike"	Whether the font is stricken, Boolean

- **Syntax**

Syntax	Get	ObjId.**GetCellFont**(Flag, Row1, Col1, Row2, Col2)

- **Info**

Return	Boolean / String, set parameter value (in case of Get Method)		
Parameter	Type	Required Y/N	Description
Flag	String	Required	Font attribute
Row1	Long	Required	Row index of the first cell of the area
Col1	Long / String	Required	Column index of the first cell of the area

			or SaveName
Row2	Long	Required	Row index of the last cell of the area
Col2	Long / String	Required	Column index of the last cell of the area or SaveName

- **Example**

```
// Change font size to 9 if larger than 10

if (mySheet.GetCellFont("FontSize", 2,1,2,1) >= 10) {

  mySheet.SetCellFont("FontSize", 2,1,2,1,9));

}
```

- **Syntax**

Syntax	Set	ObjId.**SetCellFont**(Flag, Row1, Col1, Row2, Col2, Value)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Flag	String	Required	Font attribute
Row1	Long	Required	Row index of the first cell of the area
Col1	Long / String	Required	Column index of the first cell of the area or SaveName
Row2	Long	Required	Row index of the last cell of the area
Col2	Long / String	Required	Column index of the last cell of the area or SaveName

Value	Boolean / String	Required	Values of the font attributes

➢ **Example**

```
//Bolden text in selected area

mySheet.SetCellFont("FontBold", 1,1,2,3,1) ;



//Strike text in selected area

mySheet.SetCellFont("FontStrike", 1,1,2,3,1) ;
```

CellFontBold Method

➢ **Purpose**

Check or configure that fonts in a cell is boldened.

➢ **Syntax**

Syntax	Get	ObjId.**GetCellFontBold**(Row, Col)

➢ **Info**

Return	Boolean, set value (In case of Get Method)		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row index of the cell
Col	Long /String	Required	Column index of the cell or SaveName

➢ **Example**

```
// Check whether boldening has been processed successfully for a cell.

alert(mySheet.GetCellFontBold(1, 1));
```

- **Syntax**

Syntax	Set	ObjId.**SetCellFontBold**(Row, Col, Bold)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row index of the cell
Col	Long /String	Required	Column index of the cell or SaveName
Bold	Boolean	Required	Bold or not

- **Example**

```
//Bolden fonts in a certain cell.

mySheet.SetCellFontBold(1, 1,1)
```

CellFontColor Method

- **Purpose**

Check or configure font color of a cell.

You can set font color of a cell whether the cell is in a data or header area. If the cell does not exist, it will not return any error message and just cancels the color setting.

Select the color using WebColor or BaSIC 16 Color string.

- **Syntax**

Syntax	Get	ObjId.**GetCellFontColor**(Row, Col)

- **Info**

Return	String, set color value (in case of Get Method)		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row index of the cell
Col	Long /String	Required	Column index of the cell or SaveName

- **Example**

```
//If the amount is smaller than 0, the numbers will display in red.
if(mySheet.GetCellValue(1, 2) < 0 ) {
    mySheet.SetCellFontColor(1,2,"#FF0000") ;
//If the amount is bigger than 0, the numbers will display in black.
} else {
    mySheet.SetCellFontColor(1,2, "#000000");
}
```

- **Syntax**

Syntax	Set	ObjId.**SetCellFontColor**(Row, Col, Color)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row index of the cell
Col	Long /String	Required	Column index of the cell or SaveName
Color	String	Required	Color value

- **Example**

```
//If the amount is smaller than 0, the numbers will display in red.

if(mySheet.GetCellValue(1, 2) < 0 ) {

    mySheet.SetCellFontColor(1,2,"#FF0000") ;

//If the amount is bigger than 0, the numbers will display in black.

} else {

    mySheet.SetCellFontColor(1,2, "#000000");

}
```

CellFontItalic Method

- **Purpose**

  Check or configure that fonts in a cell is italicized.

- **Syntax**

Syntax	Get	ObjId.**GetCellFontItalic**(Row, Col)

- **Info**

Return	Boolean, set value (In case of Get Method)		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row index of the cell
Col	Long /String	Required	Column index of the cell or SaveName

- **Example**

```
// Check whether a certain cell has been italicized.

alert(mySheet.GetCellFontItalic(1, 1));
```

- **Syntax**

Syntax	Set	ObjId.**SetCellFontItalic**(Row, Col, Italic)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row index of the cell
Col	Long /String	Required	Column index of the cell or SaveName
Italic	Boolean	Required	Italic or not

- **Example**

```
//Italicize fonts in a certain cell.

mySheet.SetCellFontItalic(1, 1,1)
```

CellFontName Method

- **Purpose**

  Check or configure font type for a cell.

  Use the font name used as in the style attribute of reference tag for confirmation return value and configuration parameter.

- **Syntax**

Syntax	Get	ObjId.**GetCellFontName**(Row, Col)

- **Info**

Return	String, set value (In case of Get Method)

Parameter	Type	Required Y/N	Description
Row	Long	Required	Row index of the cell
Col	Long /String	Required	Column index of the cell or SaveName

- **Example**

```
//Check the font type
alert(mySheet.GetCellFontName(1, 1));
```

- **Syntax**

Syntax	Set	ObjId.**SetCellFontName**(Row, Col, FontName)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row index of the cell
Col	Long /String	Required	Column index of the cell or SaveName
FontName	String	Required	Font type

- **Example**

```
//Change the font to Gungsuh.
mySheet.SetCellFontName(1, 1, "Gungsuh")
```

CellFontSize Method

- **Purpose**

Check or configure font size for a cell.

Configuration or confirmation value is set for each pixel.

- **Syntax**

Syntax	Get	ObjId.**GetCellFontSize**(Row, Col)

- **Info**

Return	Integer, set value (In case of Get Method)		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row index of the cell
Col	Long /String	Required	Column index of the cell or SaveName

- **Example**

```
//Check the font size.

alert(mySheet.GetCellFontSize(1,1));
```

- **Syntax**

Syntax	Set	ObjId.**SetCellFontSize**(Row, Col, FontSize)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row index of the cell
Col	Long /String	Required	Column index of the cell or SaveName

FontSize	Integer	Required	Font size to set

- **Example**

```
// Change the font size to 20px.

mySheet.SetCellFontSize(1, 1, 20)



// Change the font size of cells with "sName" as SaveName to 20.

mySheet.SetCellFontSize(1,"sName", 20);
```

CellFontStrike Method

- **Purpose**

  //Strike fonts in a certain cell.

  Display sample) Text

- **Syntax**

Syntax	Get	ObjId.**GetCellFontStrike**(Row, Col)

- **Info**

Return	Integer, set value (In case of Get Method)		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row index of the cell
Col	Long /String	Required	Column index of the cell or SaveName

- **Example**

```
//Check whether fonts in a cell has been stricken.

alert(mySheet.GetCellFontStrike(1,1));
```

➢ **Syntax**

Syntax	Set	ObjId.**SetCellFontStrike**(Row, Col, FontStrike)

➢ **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row index of the cell
Col	Long /String	Required	Column index of the cell or SaveName
FontStrike	Integer	Required	Font size to set

➢ **Example**

```
//Set Strikeout for fonts.

mySheet.SetCellFontStrike(1, 1, 1)

// Strike text in cells with SaveName "sName"

mySheet.SetCellFontStrike(1,"sName", 1);
```

CellFontUnderline Method

➢ **Purpose**

Check or set underline for fonts in a cell.

You can underline fonts in a cell whether the cell is in a data or header area. If the cell does not exist, it will not return any error message and just cancels the underline setting.

➢ **Syntax**

Syntax	Get	ObjId.**GetCellFontUnderline**(Row, Col)

➢ **Info**

Return	Boolean,current set value (in case of Get Method)		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row index of the cell
Col	Long /String	Required	Column index of the cell or SaveName

➢ **Example**

```
//If the amount is smaller than 0, underline the number.
if(mySheet.GetCellValue(1, 2) < 0 ) {
      mySheet.SetCellFontUnderline(1, 2, 1);


//If the amount is bigger than 0, do not underline the number.
} else {
      mySheet.SetCellFontUnderline(1, 2, 0);
}
```

➢ **Syntax**

Syntax	Set	ObjId.**SetCellFontUnderline**(Row, Col, Underline)

➢ **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row index of the cell
Col	Long /String	Required	Column index of the cell

			or SaveName
Underline	Boolean	Required	Underlined or not

- **Example**

```
//If the amount is smaller than 0, underline the number.

if(mySheet.GetCellValue(1, 2) < 0 ) {

      mySheet.SetCellFontUnderline(1, 2, 1);



//If the amount is bigger than 0, do not underline the number.

} else {

      mySheet.SetCellFontUnderline(1, 2, 0);

}
```

CellImage Method

- **Purpose**

  Check or configure image for a cell if the cell Type is Image or Image property has been used.

  Value you set for this method will configure the actual image file path. If the type does not fit or image is not inserted into the cell, the setting will be canceled.

  You can also check and configure the value using CellValue function.

- **Syntax**

Syntax	Get	ObjId.**GetCellImage**(Row, Col)

- **Info**

Return	String, set image path (in case of Get Method)		
Parameter	Type	Required Y/N	Description

Row	Long	Required	Row index of the cell
Col	Long /String	Required	Column index of the cell or SaveName

- **Example**

```
// Create a new data row and change the cell's image and strings.

  var Row=mySheet.DataInsert();

  mySheet.GetCellImage(Row, 1);
```

- **Syntax**

Syntax	Set	ObjId.**SetCellImage**(Row, Col, Image)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row index of the cell
Col	Long /String	Required	Column index of the cell or SaveName
Image	String	Required	Actual image path

- **Example**

```
// Create a new data row and change the cell's image and strings.

  var Row=mySheet.DataInsert();

  mySheet.SetCellImage(Row, 1, "../image/myImage1.jpg");



  // If SaveName of Column 1 is "btnAction"
  mySheet.SetCellImage(Row ,"btnAction", "../image/myImage1.jpg");
```

CellSearchValue Method

- **Purpose**

  Check the cell value as searched If the row has been newly created or one of the following types, the value will be null.

  Status, DelCheck, Seq, Image

- **Syntax**

Syntax	ObjId.**CellSearchValue**(Row, Col)

- **Info**

Return	String, cell value upon search		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row index of the cell
Col	Long /String	Required	Column index of the cell or SaveName

- **Example**

```
// Check the cell value as searched, and revert to the original value.

var OrgValue = mySheet.CellSearchValue(Row, Col);



if(OrgValue != mySheet.GetCellValue(Row, Col)) {

     mySheet.SetCellValue(Row, Col, OrgValue);

}
```

CellVAlign Method

- **Purpose**

  Check or configure vertical alignment of the cell.

- **Syntax**

Syntax	Get	ObjId.**GetCellVAlign**(Row, Col)

- **Info**

Return	String, vertical alignment value		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row index of the cell
Col	Long /String	Required	Column index of the cell or SaveName

- **Example**

```
// Check the vertical alignment setting of a cell.

alert(mySheet.GetCellVAlign(1,1));
```

- **Syntax**

Syntax	Set	ObjId.**SetCellVAlign** (Row, Col, valign)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row index of the cell
Col	Long /String	Required	Column index of the cell

			or SaveName
valign	String	Required	Vertical alignment setting of a cell (top / middle / bottom)

- **Example**

```
// Set top vertical alignment for a cell.

mySheet.SetCellVAlign(1, 1, "top");

// Set bottom vertical alignment for a cell.

mySheet.SetCellVAlign(1, 1, " bottom");
```

CellText Method

- **Purpose**

Check or configure the cell values as formatted and displayed on the screen.

If cell Type is Status, and if the value displayed on the screen is "Insert", using this method will read Insert. If you use CellValue instead, it will return a code value of "I".

All Type values like Status will be read and set as displayed on the screen. For cells with DelCheck or CheckBox type which uses a checkbox, you can check the actual checkbox texts using this method. Checking status of those boxes may be checked using CellValue.

- **Syntax**

Syntax	Get	ObjId.**GetCellText**(Row, Col)

- **Info**

Return	String, value set for the cell (In case of Get Method)		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row index of the cell

Col	Long /String	Required	Column index of the cell or SaveName

- **Example**

```
//Date cell, search value is "2011/07/15"

alert(mySheet.GetCellText(1,1));
```

- **Syntax**

Syntax	Set	ObjId.**SetCellText**(Row, Col, Text)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row index of the cell
Col	Long /String	Required	Column index of the cell or SaveName
Text	String	Required	Value to set to the cell

- **Example**

```
//Set a space

mySheet.SetCellText(1, 0, "" );



//Set value to date cell

mySheet.SetCellText(1, 1, "2011/07/15");



//Set value to a number cell
```

```
mySheet.SetCellText(1, 2, 1,234,567);



//Set combo text to a combo cell instead of combo code

mySheet.SetCellText(1, 3, "Korean won");
```

CellValue Method

- **Purpose**

  Check or configure cell value.

  Check or configure the cell values to be used for saving without formatting. Using this method to set value to a cell will fire OnChange event. If Flag parameter value of CellValue function is 0, OnChange event will not fire and the value will be updated.

  Cell values by Type and Format are as follows:

Type	Description	
Seq Pass	Configuration not allowed	
Status	Insert = "I", Update = "U", Delete = "D", 조회 = ""	
CheckBox, DummyCheck, RadioCheck	1 or 0	
Combo ComboEdit	Combo code	
Image	Actual path of the image	
Other types	Format	Description
	Ymd	Display 8 digit numbers in "yyyy.mm.dd format"; return 8 digit number upon checking

	Ym	Display 6 digit numbers in "yyyy.mm" format; return 6 digit number upon checking
	Md	Display 4 digit numbers in "mm.dd" format; return 4 digit number upon checking
	Hms	Display 6 digit numbers in "hh:mm.ss" format; return 6 digit number upon checking
	Hm	Display 4 digit numbers in "hh:mm" format; return 4 digit number upon checking
	YmdHms	Display 14 digit "numbers in "yyyy.mm.dd hh:mm:ss" format; return 14 digit number upon checking
	YmdHm,	Display 12 digit "numbers in "yyyy.mm.dd hh:mm" format; return 12 digit number upon checking
	IdNo	Display 13 digit number in "######-#######" format; return 13 digit number upon checking
	SaupNo	Display 10 digit number in "###-##-#####" format; return 10 digit number upon checking
	CardNo	Display 14 digit number in "####-####-####-####" format; return 14 digit number upon checking
	PostNo	Display 6 digit number in "###-###" format; return 6 digit number upon checking
	Integer	Display in "#,##0"; return numbers upon checking

	NullInteger	Display in "#,###"; return numbers upon checking
	Float	Display in "#,##0."+Point number; return numbers upon checking
	NullFloat	Display in "#,###."+Point number; return numbers upon checking

- **Syntax**

Syntax	Get	ObjId.**GetCellValue**(Row, Col)

- **Info**

Return	String, value set for the cell (In case of Get Method)		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row index of the cell
Col	Long /String	Required	Column index of the cell or SaveName

- **Example**

```
//Check value of a date cell; the result is 2011/07/15.

alert(mySheet.GetCellValue(1, 4));
```

- **Syntax**

Syntax	Set	ObjId.**SetCellValue**(Row, Col, Value)

- **Info**

Return	None		
Parameter	Type	Required	Description

		Y/N	
Row	Long	Required	Row index of the cell
Col	Long /String	Required	Column index of the cell or SaveName
Value	String	Required	Value to set to the cell
Flag	Boolean	Optional	Whether OnChange event fires (Default=1)

- **Example**

```
//Set the Status cell as "Deleted" status

mySheet.SetCellValue(1, 0, "D");



//Set the CheckBox as checked

mySheet.SetCellValue(1, 1, 1);



//Set a value to a number cell; the display value is 12,345

mySheet.SetCellValue(1, 2, 12345);



//Set a value to a combo cell; the display value is combo text

mySheet.SetCellValue(1, 3, "01");



//Set a value to a date cell; the display value is "2011/07/15"

mySheet.SetCellValue(1, 4, "2011/07/15");



//Set a value to a decimal point number cell; if there are 3 decimal places,
the display value is 123.450.
```

```
// OnChange Event fires

mySheet.SetCellValue(1,5, 123.450);



// OnChange event does not fire

mySheet.SetCellValue(1,5, 123.450, 0);
```

CheckAll Method

- **Purpose**

  Check or uncheck all checkboxes in a column where checkbox cell exists.

  This method will perform the same job as the user clicking on the CheckAll checkbox in the header. If the value is 0, all checkboxes are unchecked. If 1, all boxes are checked. Any values other than 0 or 1 will reverse the existing check setting for all boxes.

  This method apply to DelCheck and CheckBox types only, and only checkboxes in editable cells. When checking is completed, OnChange event will fire for each row.

  This method will apply to all rows if Col parameter is set as index and two or more rows are grouped as a unit data row. Apply only to rows with set SaveName If Col is set as SaveName.

  According to the value, CheckBoxes will be handled as follows:

Value	Description
0	Uncheck all
1	Check all
Others	Reverse the current check status

- **Syntax**

Syntax	ObjId.**CheckAll**(Col, Value, [OnChangeEvent])

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Col	Long / String	Required	Column index or SaveName of a particular column
Value	Integer	Required	0 : Uncheck all 1 : Check all Others: Reverse the current check status
OnChangeEvent	Boolean	Optional	Whether OnChange event fires Default=1

- **Example**

```
// Check all

mySheet.CheckAll(1, 1);



// Uncheck all

mySheet.CheckAll(1, 0);
```

CheckedRows Method

- **Purpose**

  Return the number of checked rows out of all checkboxes of the column

  This method will apply to all rows if Col parameter is set as index and two or more rows are grouped as a unit data row. Apply only to rows with set Savename If Col is set as SaveName.

- **Syntax**

Syntax	ObjId.**CheckedRows**(Col)

➢ **Info**

Return	Long, number of rows checked for a column		
Parameter	Type	Required Y/N	Description
Col	Long / String	Required	Column index of a particular column or SaveName

➢ **Example**

```
var RowCnt1 = mySheet.CheckedRows(1);

alert("The number of checked rows is " + RowCnt1 + ".");



var RowCnt2 = mySheet.CheckedRows("chkData");

alert("The number of checked rows is " + RowCnt2 + ".");
```

CheckReverse Method

➢ **Purpose**

Uncheck all the checked checkboxes and check all the unchecked ones in a column.

This method will apply to all rows if Col parameter is set as index and two or more rows are grouped as a unit data row. Apply only to rows with set Savename If Col is set as SaveName.

➢ **Syntax**

Syntax	ObjId.**CheckReverse**(Col, [Editable], [Event])

➢ **Info**

Return	None		
Parameter	Type	Required	Description

		Y/N	
Col	Long / String	Required	Column index of a particular column or SaveName
Editable	Boolean	Optional	Check editability and update only when editing is allowed. Default=0
Event	Boolean	Optional	Whether to fire OnChange event when check is updated, Default=0

- **Example**

```
// Reverse checks in Column 1.

mySheet.CheckReverse(1);



// Check the editability and fire event.

mySheet.CheckReverse(1, 1, 1);
```

ClearHeaderCheck Method

- **Purpose**

  Initialize all the checkbox values of the header to uncheck.

  This method does not perform CheckAll function, like HeaderCheck, and performs uncheck only.

- **Syntax**

Syntax	ObjId.**ClearHeaderCheck**()

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
N/A			

➢ **Example**

```
// Initialize all checkboxes before RemoveAll is called.

mySheet.ClearHeaderCheck();

mySheet.RemoveAll();
```

ClipCopyMode Method

➢ **Purpose**

You can select a string and press Ctrl + C to copy the selected text to clipboard. This method will check or configure how you would like to copy the selected to clipboard using Ctrl + C keys.

Available options and values are as follows:

Value	Description
0	Copy focus cell only when focus is on a single cell; when multiple cells are selected, copy multiple cells. Default
1	Copy the focus row when focus is on a single cell; when multiple cells are selected, copy multiple cells.

➢ **Syntax**

Syntax	Get	ObjId.**GetClipCopyMode**()

➢ **Info**

Return	Integer, set value (In case of Get Method)		
Parameter	Type	Required Y/N	Description

➢ **Example**

```
mySheet.GetClipCopyMode();
```

- **Syntax**

Syntax	Set	ObjId.**SetClipCopyMode**(Mode)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Mode	Integer	Required	Copy scope

- **Example**

```
// Copy all focus cell or selected cell values

mySheet.SetClipCopyMode(0);



// Copy all focus row or selected cell values

mySheet.SetClipPasteMode(1);
```

ClipPasteMode Method

- **Purpose**

  You can select a string and press Ctrl + C to copy the selected text to clipboard. This method will check or configure how you would like to paste the copied data to a sheet from clipboard using Ctrl + V keys. If the data type does not fit to the pasting cells or there are other error potentials, paste will fail.

  Available options and values are as follows:

Value	Description
-1	Not use paste function
0	Paste to a selected cell, Default
1	Paste to multiple cells starting from the selected cell As pasting copied characters in excel, new-line character("₩r₩n") separates rows and tap character ("₩t") separates columns.

2	Same as value 1; if there are not enough data rows to paste data into, add rows to paste all the data.
3	Add as many new rows as the pasting data rows.

➢ **Syntax**

Syntax	Get	ObjId.**GetClipPasteMode**()

➢ **Info**

Return	Integer, set value (In case of Get Method)		
Parameter	Type	Required Y/N	Description

➢ **Example**

```
mySheet.GetClipPasteMode();
```

➢ **Syntax**

Syntax	Set	ObjId.**SetClipPasteMode**(Mode)

➢ **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Mode	Integer	Required	how to paste

➢ **Example**

```
// Paste into single cell mode

mySheet.SetClipPasteMode(0);
```

```
// paste into multiple cells mode

mySheet.SetClipPasteMode(1);
```

ColBackColor Method

- **Purpose**

  Check or configure background color of the entire column. Background color will change only for data rows not the header row.

  If the column does not exist, it will not return any error message and just cancels the background color setting.

  Select the color using WebColor or BaSIC 16 Color string.

- **Syntax**

Syntax	Get	ObjId.**GetColBackColor**(Col)

- **Info**

Return	String, set color value (in case of Get Method)		
Parameter	Type	Required Y/N	Description
Col	Long /String	Required	Column index of a particular column or SaveName

- **Example**

```
// Check background color of Column 2.

mySheet.GetColBackColor(2);
```

- **Syntax**

Syntax	Set	ObjId.**SetColBackColor**(Col, Color)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Col	Long /String	Required	Column index of a particular column or SaveName
Color	String	Required	WebColor color value

- **Example**

```
//Set the column background color as gray.
mySheet.SetColBackColor(1, "#ADADAD");
 //Set the Column1 background color as the same color as Column 2 background.
mySheet.SetColBackColor(2, mySheet.GetColBackColor(1));
```

ColCondProperty Method

- **Purpose**

  If each cell data value meets the condition for a Number Type column, Check or configure certain attributes.

  The following properties may be configured using this method.

Property	Type	Description
BackColorT	String	Background color if True
BackColorF	String	Background color if False
FontColorT	String	Font color if True
FontColorF	String	Font color if False
EditT	Boolean	Editability if True
EditF	Boolean	Editability if False

CursorT	String	Mouse pointer style if True
CursorF	String	Mouse pointer style if False

Mouse pointer styles available are as follows:

Value	Mouse pointer style
Default	Basic arrow style
Pointer	Finger style

➢ **Syntax**

Syntax	Get	ObjId.**GetColCondProperty**(Col)

➢ **Info**

Return	String, set condition (In case of Get Method)		
Parameter	Type	Required Y/N	Description
Col	Long / String	Required	Index of the target column or SaveName

➢ **Example**

```
// Check the set condition.

mySheet.GetColCondProperty(13)
```

➢ **Syntax**

Syntax	Set	ObjId.**SetColCondProperty**(Col, Cond, Prop)

➢ **Info**

Return	None

Parameter	Type	Required Y/N	Description
Col	Long / String	Required	Index of the target column or SaveName
Cond	String	Required	Condition to set (ex: "%d >1000")
Prop	Object	Optional	Target column attribute

- **Example**

```
// If any cell data value exceeds 1000 for the 13th column

mySheet.SetColCondProperty(13,"%d>1000",{BackColorT:"#00ff00",FontColorT
:"#ffff00", EditT : false, BackColorF : "#0000ff",FontColorF:"#ffffff", EditF : true,
CursorT:"Pointer", CursorF:"Default"})
```

ColEditable Method

- **Purpose**

  Check or configure editable status of a particular column.

  You can change the editable status of a column only if all cell's editability status is editable. If ColEditable value is Not Allowed, RowEditable configuration will be ignored.

  Editability of a particular column will be determined as follows:

Editable	**ColEditable**	RowEditable	CellEditable	**Cell editable Y/N**
No	**No impact**	No impact	No impact	**No**
Yes	**No**	No	Yes/No	**Yes/No**
Yes	**Yes**	Yes/No	Yes/No	**Yes/No**

- **Syntax**

Syntax	Get	ObjId.**GetColEditable**(Col)

- **Info**

Return	Boolean, editability status (in case of Get Method)		
Parameter	Type	Required Y/N	Description
Col	Long /String	Required	Column index of a particular column or SaveName

- **Example**

```
//Check editability status of Column 5.

mySheet. GetColEditable (5);
```

- **Syntax**

Syntax	Set	ObjId.**SetColEditable**(Col, Editable)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Col	Long /String	Required	Column index of a particular column or SaveName
Editable	Boolean	Required	Editability status of a particular column (Use only for Set)

- **Example**

```
// Check editability status of Column 5 as not editable.

mySheet.SetColEditable(5,0);
```

```
// Check editability status of Column 5 as editable.

mySheet. SetColEditable (5,1);
```

ColFontColor Method

- **Purpose**

  Check or configure font color of the entire column. Font color will change only for data rows not the header row.

  If the column does not exist, it will not return any error message and just cancels the background color setting.

  Select the color using WebColor.

- **Syntax**

Syntax	Get	ObjId.**GetColFontColor**(Col)

- **Info**

Return	String, set color value (in case of Get Method)		
Parameter	Type	Required Y/N	Description
Col	Long /String	Required	Column index of a particular column or SaveName

- **Example**

```
// Check font color of Column 2.

alert(mySheet.GetColFontColor(2));
```

- **Syntax**

Syntax	Set	ObjId.**SetColFontColor**(Col, Color)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Col	Long /String	Required	Column index of a particular column or SaveName
Color	String	Required	Color value

- **Example**

```
//Set the column font color as gray.
mySheet.SetColFontColor(1, "#FF0000");
//Set the Column1 font color as the same color as Column 2 font.
mySheet. SetColFontColor(2, mySheet.GetColFontColor(1));
```

ColFontUnderline Method

- **Purpose**

  Check or configure text underline status of the entire column. Underline status update will apply only for data rows not the header row.

  If the column does not exist, it will not return any error message and just cancels the underline setting.

- **Syntax**

Syntax	Get	ObjId.**GetColFontUnderline**(Col)

- **Info**

Return	Boolean, set underline value (in case of Get Method)		
Parameter	Type	Required Y/N	Description
Col	Long /String	Required	Column index of a particular column

			or SaveName

➢ **Example**

```
//Check whether texts in a column has been underlined.

alert(mySheet.GetColFontUnderline(1));
```

➢ **Syntax**

Syntax	Set	ObjId.**SetColFontUnderline**(Col, Underline)

➢ **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Col	Long /String	Required	Column index of a particular column or SaveName
Underline	Boolean	Required	Underlined or not

➢ **Example**

```
//Underline all text for a column.

mySheet.SetColFontUnderline(1, 1);
```

ColHidden Method

➢ **Purpose**

Check or configure whether to hide column.

➢ **Syntax**

Syntax	Get	ObjId.**GetColHidden**(Col)

- **Info**

Return	Boolean, hiding setting value (in case of Get Method)		
Parameter	Type	Required Y/N	Description
Col	Long /String	Required	Column index of a particular column or SaveName

- **Example**

```
// Check hiding status of columns and unhide any hidden columns.
if(mySheet.GetColHidden(1) == 1) {
      mySheet.SetColHidden(1, 0);
}
```

- **Syntax**

Syntax	Set	ObjId.**SetColHidden**(Col, Hidden)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Col	Long /String	Required	Column index of a particular column or SaveName
Hidden	Boolean	Required	Hidden or not

- **Example**

```
// Check hiding status of columns and unhide any hidden columns.
if(mySheet.GetColHidden(1) == 1) {
      mySheet.SetColHidden(1, 0);
}
```

ColLeft Method

- **Purpose**
  Check left of a column

- **Syntax**

Syntax	ObjId.**ColLeft(**Col**)**

- **Info**

Return	Long, left location value of a particular column		
Parameter	Type	Required Y/N	Description
Col	Long /String	Required	Column index of a particular column or SaveName

- **Example**

```
//Identify the left location of a column

var iLeft = mySheet.ColLeft(1);
```

ColSaveName Method

- **Purpose**
  Check the SaveName set in InitColumns function that corresponds to Index of a particular column.

- **Syntax**

Syntax	ObjId.**ColSaveName**(Col)

- **Info**

Return	String, SaveName of a particular column		
Parameter	Type	Required Y/N	Description
Col	Long	Required	Column index of a particular column

- **Example**

```
//Fetch SaveName of a column.

var sSaveName = mySheet.ColSaveName(1);
```

ColumnSort Method

- **Purpose**

Sort data in a single or multiple columns.

If you are sorting data for multiple columns, connect columns in the sorting order with "|"so that sorting can apply sequentially. In default, sorting will apply to columns with smaller Column Index first.

If ColSort parameter value is set as null, all columns will be sorted according to the sorting direction order set in Sort parameter. Use this if you want to set different sorting direction for different columns.

KeepColOrder parameter means the sorting order for columns set in Col parameter. If this parameter value is 0, sorting order will be the same as column index order. If the parameter value is set as 1, the sorting order will be according to the setting. The default value of this parameter is 0.

If Col parameter is set as empty or null, column sorting will be initialized.

- **Syntax**

Syntax	ObjId.**ColumnSort**(Col, [Sort], [ColSort],[KeepColOrder])

- **Info**

Return	None		
Parameter	Type	Required	Description

		Y/N	
Col	Long / String	Required	String of Column Index or SaveName of columns to sort, connected with "|"
Sort	String	Optional	"ASC" or "DESC", Default ="ASC"
ColSort	String	Optional	String of sorting directions for each column, connected with "|"
KeepColOrder	Boolean	Optional	Whether to sort according to Cols parameter order; Default=0

➢ **Example**

```
//Sort column 6 in the descending order

mySheet.ColumnSort("6", "DESC")



//Sort in the ascending order with column 4 as the centerpiece, and sort column 5 in the ascending order separately within

mySheet.ColumnSort("4|5")



//Sort in the order of column 2,3 and 4 in the descending order.

mySheet.ColumnSort("2|3|4", "DESC");



//Sort column 3,2 and 4 in the order of 2,3 and 4 (Column index) in descending, ascending and ascending order, respectively.

mySheet.ColumnSort("3|2|4", "DESC","ASC|DESC|ASC");



//Sort column 3,2 and 4 in the order of 3, 2 and 4 in ascending, descending and ascending order, respectively.

mySheet.ColumnSort("3|2|4", " DESC ","ASC|DESC|ASC", 1);
```

```
//Initialize all sorting settings.

mySheet.ColumnSort();
```

ColValueDup Method

- **Purpose**

  Check whether there are redundant value within a particular column.

  If there is a redundancy, return the row index of the second one.

  You can use this for just one column. To use for multiple columns, use "|" as connector. If redundancies are found in multiple columns, the row index will be returned.

  If IncludeDelRow parameter is 1, include rows with transaction status of Deleted for redundancy check. If 0, those rows will be excluded from the redundancy check.

  If there is no redundancy, –1 will be returned.

- **Syntax**

Syntax	ObjId.**ColValueDup**(ColStr, [IncludeDelRow], [Division])

- **Info**

Return	Long, row number		
Parameter	Type	Required Y/N	Description
ColStr	Long/ String	Required	Combination of Column Indexes or SaveNames Connect with "|"
IncludeDelRow	Boolean	Optional	Whether to include rows with Transaction status of Deleted Default=1

Division	Boolean	Optional	Case sensitivity setting Default=1

- **Example**

```
//Fetch the row numbers with redundant value in Column 1

var Row = mySheet.ColValueDup("1");



//Fetch the row numbers with redundant value in Column 2, 3 and 7

var Row = mySheet.ColValueDup("2|3|7");



//Run redundancy check excluding deleted rows

var Row = mySheet.ColValueDup("2|3|7", 0);
```

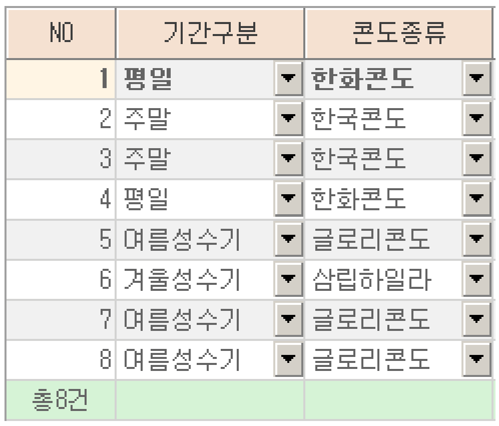
ColValueDupRows Method

- **Purpose**

  Combine all the redundant rows with "," and return in string.

  ColValueDup method can check the index of the first redundant row, but using this you can check all redundant row numbers as well as the first redundant row.

  For example, if a sheet is composed of two columns as "Period" and "Condo name", ColValueDup=4. The return result will be row 4, as it is the first redundancy for row 1. With every other condition the same, using this method will return "3, 4, 7, 8".



NO	기간구분	콘도종류
1	평일	한화콘도
2	주말	한국콘도
3	주말	한국콘도
4	평일	한화콘도
5	여름성수기	글로리콘도
6	겨울성수기	삼립하일라
7	여름성수기	글로리콘도
8	여름성수기	글로리콘도
총8건		

IncludeDelRow parameter sets whether to include rows with transaction status Deleted. If this parameter is true, include deleted rows. If false, not include. If row 1 is deleted, and set IncludeDelRow parameter as false, ColValueDup=3, and this method will return "3, 7, 8".

IncludeFirstRow parameter sets whether to include the first row of overlapping rows in the result. If this parameter is set as false, the first row is not included. If set true, the first rows are put into a string connected with ",". Other redundant rows will be also put into a string separated with ",". The two strings will be adjoined with "|" and returned.

In the sample above, the starting rows are "1, 2, 5" and redundant rows are "3, 4, 7, 8". Therefore "1,2,5|3,4,7,8" will be returned.

StartRow parameter and **EndRow** parameter sets the areas for redundancy check as row index. In default, all data rows are subject to redundancy check. In the sample above, the return result will be "3, 4" if StartRow=1 and EndRow=4.

If there is no redundancy, “” will be returned.

- **Syntax**

Syntax	ObjId.**ColValueDupRows**(ColStr, [IncludeDelRow], [IncludeFirstRow], [StartRow], [EndRow])

- **Info**

Return	String, string of all redundant rows separated with ","

Parameter	Type	Required Y/N	Description
ColStr	Long/ String	Required	Combination of Column Indexes or SaveNames Connect with "|"
IncludeDelRow	Boolean	Optional	Whether to include rows with Transaction status of Deleted Default=1
IncludeFirstRow	Boolean	Optional	Whether to include the starting row of subsequent redundant rows Default=0
StartRow	Long	Optional	First row index of the area for redundancy check Default="First row"
EndRow	Long	Optional	Last row index of the area for redundancy check Default="Last row"

➢ **Example**

```
//Character redundancies in Column 6 and 7 (including deleted rows, excluding the first occurrence, check for all data areas)

  var duprows1 = mySheet.ColValueDupRows("6|7");



  //Redundancy check for column 4 and 5 between row 1 - 50 (excluding deleted rows, including the first occurrence, Row 1-50)

  var duprows2 = mySheet.ColValueDupRows("4|5",false,true,1,50);



  //Create an array using the fetched rows

  var arrRow = duprows1.split(",");

  for (idx=0; idx<arrRow.length-1; idx++){ alert(arrRow[idx] + "행"); }
```

ColWidth Method

- **Purpose**

  Check or configure width of a particular column.

  You can set width by pixel. If set 0, width will be automatically adjusted to fit the longest text within the column.

  If the column does not exist, it will not return any error message and just cancels the setting.

- **Syntax**

Syntax	Get	ObjId.**GetColWidth**(Col)

- **Info**

Return	Integer, number of pixels for width of a particular column (in case of Get Method)		
Parameter	Type	Required Y/N	Description
Col	Long/ String	Required	Column index of a particular column or SaveName

- **Example**

```
// Check width of Column 1

mySheet.GetColWidth(1);
```

- **Syntax**

Syntax	Set	ObjId.**SetColWidth**(Col, Width)

- **Info**

Return	None		
Parameter	Type	Required	Description

		Y/N	
Col	Long/ String	Required	Column index of a particular column or SaveName
Width	Integer	Required	너비 픽셀 값

➢ **Example**

```
//Change the width to 50 pixels

mySheet.SetColWidth(1, 50);



//Automatic adjustment to fit the widest text within the column

mySheet.SetColWidth(2, 0);



//Set the width of Column 3 to the same width as Column 2

mySheet.SetColWidth(3, mySheet.GetColWidth(2));
```

ComboOpenMode Method

➢ **Purpose**

Check or configure whether to expand Combo or ComboEdit column with one click.

➢ **Syntax**

Syntax	Get	ObjId.**GetComboOpenMode**()

➢ **Info**

Return	Boolean,current set value (in case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Check the value set in ComboOpenMode.

mySheet.GetComboOpenMode();
```

- **Syntax**

Syntax	Set	ObjId.**SetComboOpenMode**(mode)

- **Info**

Return	None			
Parameter	Type	Required Y/N	Description	
mode	Boolean	Required	0 (Default)	Single Click on combo will get focus without expanding. (Default)
			1	Expand combo with one click.

- **Example**

```
//Open combo with one click.

mysheet.SetComboOpenMode(1);
```

ComputeSum Method

- **Purpose**

Calculate the sum of a particular area and return.

This method may calculate sum of a particular column, or sum of values calculated using an equation. If you do not designate a particular area, all data will be summed up in return.

- **Syntax**

Syntax	ObjId.**ComputeSum**(CalcuLogic,[FirstRow],[LastRow],[isFullSum])

- **Info**

Return	Double, sum of values in a particular area		
Parameter	Type	Required Y/N	Description
CalcuLogic	String	Required	equation; If values from other column is used in an equation, surround it with "|"
FirstRow	Long	Optional	Start row index in the areas subject to summing, Default=-1
LastRow	Long	Optional	Last row index in the areas subject to summing, Default=-1
isFullSum	Boolean	Optional	Whether to include partial sum rows in the equation, Default=1 1 : exclude partial sum rows 0 : include partial sum rows

- **Example**

```
//Calculate sum of column 3

var Sum3 = mySheet.ComputeSum("|3|");

//From row 1-10, calculate sum of "Column 3 * Column 4 / 100"

var Sum4 = mySheet.ComputeSum("|3| * |4| / 100", 1, 10);
```

ConfirmOK Method

- **Purpose**

  If IsConfirm = 1 in OnMessage Event, this method will display confirmation window and return the response to the sheet.

  In case ShowMsgMode is set as 0, OnMessage Event will fire when a message occurs in a sheet instead of a system pop-up. If the message is a confirmation message, OnMessage Event IsConfirm parameter value will be 1.

  A warning message will complete as the message window closes, but confirmation message requires user response returned to the sheet. This method can be used only within OnMessage

Event to return response to the sheet.

- **Syntax**

Syntax	ObjId.**ConfirmOK**(Val)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Val	Boolean	Required	Selected value on the confirmation window

- **Example**

```
//Configure the message mode.

mySheet.ShowMsgMode=0;



//Process OnMessage event.

Function mySheet_OnMessage(grid,msg, level, isconfirm)

    //Display message

    var win_result = window.showModalDialog(

        "sheet_message.jsp?Msg=" + msg + "&IsConfirm=" + isconfirm,

         'modalResult',

         'dialogWidth:200px;dialogHeight:200px;center:yes;help:no;status:no;');

    //Return the message result to the sheet.

    if(IsConfirm) mySheet.ConfirmOK(win_result);

</script>
```

CountFormat Method

- **Purpose**

  Check or configure the count format. Count information is a combination of available reserved words. The following is the list of available reserved words.

Reserved word	Description
"BOTTOMDATA"	Row index of the bottom data
"TOTALROWS"	Data count of the entire DB 1) all data count returned 2) user setting Among the two steps above, the last set value will display as TOTALROWS.
"SEARCHROWS"	Data count returned (Count only those with status value "")
"INSERTROWS"	Inserted count (Count onlyu those with status value "I")
"UPDATEROWS"	Updated count (Count onlyu those with status value "U")
"DELETEROWS"	Deleted count (Count onlyu those with status value "D")
"ROWCOUNT"	All (Insert + Updated + Deleted) count
"SELECTDATAROW"	Record order of the currently focused row 1) Include hidden rows in counting 2) Exclude sum and partial sum from record count 3) If a record comprises two or more rows, display in record order not row order.

  The default format is "[BOTTOMDATA / TOTALROWS]".

- **Syntax**

Syntax	Get	ObjId.**GetCountFormat**()

- **Info**

Return	String, display format as set(in case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Check the count format.

mySheet.GetCountFormat();
```

- **Syntax**

Syntax	Set	ObjId.**SetCountFormat**(Format)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Format	String	Required	Display format to set

- **Example**

```
//Set count format.

mySheet.SetCountFormat("Currently BOTTOMDATA / Total TOTALROWS");



//Display in focus.

mySheet.SetCountFormat("Select SELECTDATAROW row / Total count
ROWCOUNT");
```

CountPosition Method

- **Purpose**

  Set count data to display in a particular area of IBSheet.

  Count data will display in the format set in the CountFormat method. In default, the current location and final count will display. As the scroll bar moves or rows are added, display will be updated.

  Count display area by setting is as follows:

Value	0	1	2	3	4
Display location	No display	Upper left	Upper right	Bottom left	Bottom right

- **Syntax**

Syntax	Get	ObjId.**GetCountPosition**()

- **Info**

Return	Integer, display location value as set (In case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Display count info in the top left corner is currently not displaying.
if(mySheet.GetCountPosition() == 0) {
   mySheet.SetCountPosition(1);
}
```

- **Syntax**

Syntax	Set	ObjId.**SetCountPosition**(Position)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Position	Integer	Required	Display location value to set

- **Example**

```
//Display count info in the top left corner is currently not displaying.

if(mySheet.GetCountPosition() == 0) {

   mySheet.SetCountPosition(1);

}
```

CreatePivotTable Method

- **Purpose**

Pivot table is a kind of dialogue table. You can perform computing such as sum or count depending on the data listing format.

Even if IBSheet with data is updated, the pivot table will not be updated automatically. You need to call this method again as necessary.

- **Syntax**

Syntax	ObjId.**CreatePivotTable**(Info, DataSheet)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Info	json	Required	Pivot table configuration object group

			(See details)
DataSheet	Object	Required	Object of IBSheet with source data

Details

You can set the following parameters in Info.

Property	Type	Description
Cols	String	Column Index or SaveName separated with '|' to set in column label field
DefaultView	String	String to show in cell with null value
Rows	String	Column Index or SaveName separated with '|' to set in row label field
SortRow	Boolean	Whether to sort first when summing row label. if 0, create row label with returned data without sorting. If 1, sort the returend data to create a row label. (Default: 1)
Value	String	Column Index or SaveName separated with '|' to sum up or count.
ValueType	String	Type of summed rows separated with '|' Must be the same as count in ('Sum', 'Count') value.

- **Example**

```
//Set column 5, 6 and 7 as row label field, and 8, 9 and 10 as column label field. Display column 13 in summary (sum).

mySheet2.CreatePivotTable({Rows:'5|6|7', Cols:'8|9|10', Value:'13', ValueType:'Sum'}, mySheet);



//Set column 5, 6 and 7 as row label field, set column 2 as column label field. Display column 13 in summary (count).
```

```
mySheet2.CreatePivotTable({Rows:'5|6|7', Cols:'2', Value:'13', ValueType:'Count'},
mySheet);
```

CumulateBackColor Method

- **Purpose**

  Check or configure background color of aggregate row.

  Select the color using WebColor.

- **Syntax**

Syntax	Get	ObjId.**GetCumulateBackColor**()

- **Info**

Return	String, current aggregate color value (in case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Check background color of aggregate row.

mySheet.GetCumulateBackColor();
```

- **Syntax**

Syntax	Set	ObjId.**SetCumulateBackColor**(Color)

- **Info**

Return	String, current aggregate color value (in case of Get Method)

Parameter	Type	Required Y/N	Description
Color	String	Required	WebColor value

- **Example**

```
//Set background color of aggregate row as green.

mySheet.SetCumulateBackColor("#00FF00");
```

Data2Clipboard Method

- **Purpose**

  Copy all the data on IBSheet to ClipBoard. Calling this function will copy all data including the header, data area and sums. Columns will be separated with a tab and rows with new line in ClipBoard.

  This function can be used only on Internet Explorer because of ClipBoard security restrictions.

- **Syntax**

Syntax	ObjId.**Data2Clipboard**()

- **Info**

Return	String, string copied in ClibBoard		
Parameter	Type	Required Y/N	Description
N/A			

- **Example**

```
//Copy to ClibBoard.

mySheet.Data2Clipboard();
```

DataAlternateBackColor Method

- **Purpose**

  Check or configure default background color of even-number data rows.

  Use this to set different colors for even and odd-number data rows. Background color of data rows will be determined by this and DataBackColor function.

  Select the color using WebColor.

- **Syntax**

Syntax	Get	ObjId.**GetDataAlternateBackColor**()

- **Info**

Return	String, current even number row color value (in case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
// Check the current color value.

var color = mySheet.GetDataAlternateBackColor(1,1);

alert("Color value of the even number rows is " + color + ".");
```

- **Syntax**

Syntax	Set	ObjId.**SetDataAlternateBackColor**(Color)

- **Info**

Return	None		
Parameter	Type	Required	Description

		Y/N	
Color	String	Required	Color value to set

- **Example**

```
//Change the background color of the even number rows to red

mySheet.SetDataAlternateBackColor("#FF0000");          // WebColor
```

DataAutoTrim Method

- **Purpose**

  Check or configure whether to trim space in data for search or save.

  In default, spaces leading or trailing the data will be trimmed in search or save; if this attribute is set as 0, the empty spaces in data will be returned in search and saved. This setting will apply not just for search or save but to all attribute such as CellValue. In default, data space trimming is set.

- **Syntax**

Syntax	Get	ObjId.**GetDataAutoTrim**()

- **Info**

Return	Boolean, data space trim setting value (in case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Check automatic trim setting

mySheet.GetDataAutoTrim();
```

➢ **Syntax**

Syntax	Set	ObjId.**SetDataAutoTrim**(Trim)

➢ **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Trim	Boolean	Required	Data space trim value to set

➢ **Example**

```
//Search using automatic trimming

mySheet.SetDataAutoTrim(1);

mySheet.DoSearch("list.jsp");

//Save data without trimming

mySheet.SetDataAutoTrim(0);

mySheet.DoSave("save.jsp");
```

DataBackColor Method

➢ **Purpose**

Check or configure default background color of odd-number data rows.

Use this to set different colors for even and odd-number data rows. Background color of data rows will be determined by this and DataAlternateBackColor function.

Select the color using WebColor.

➢ **Syntax**

Syntax	Get	ObjId.**GetDataBackColor**()

➢ **Info**

Return	String, current odd number row color value (in case of Get Method)		
Parameter	Type	Required Y/N	Description

➢ **Example**

```
//Check the current background color of odd-number data rows.

mySheet.GetDataBackColor();
```

➢ **Syntax**

Syntax	Set	ObjId.**SetDataBackColor**(Color)

➢ **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Color	String	Required	Color value to set

➢ **Example**

```
//Set the background color of odd-number data rows to white.

mySheet.SetDataBackColor("#FFFFFF");
```

DataCopy Method

➢ **Purpose**

Copy the data row content that is last selected to create a new row, and return the row index of the new row. In case of a tree structure and the copy target row includes child levels, copy all child level rows if IncludeChild parameter is 1.

Transaction status of new row will be "Insert".

➢ **Syntax**

Syntax	ObjId.**DataCopy**([IncludeChild])

➢ **Info**

Return	Long, Row Index of the row copied		
Parameter	Type	Required Y/N	Description
IncludeChild	Boolean	Optional	Whether to copy child level rows Default=0

➢ **Example**

```
//Copy the row and change the transaction status of copied row to "Search".

//'Status' is the status column SaveName

var Row = mySheet.DataCopy();

mySheet.SetCellValue(Row, "Status", "R");

//Copy child level rows

mySheet.DataCopy(1);
```

DataFontColor Method

➢ **Purpose**

Check or configure font color of all data rows.

Select the color using WebColor.

➢ **Syntax**

Syntax	Get	ObjId.**GetDataFontColor**()

➢ **Info**

Return	String, set font color value (in case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
// Check font color of a data row.

mySheet.GetDataFontColor();
```

- **Syntax**

Syntax	Set	ObjId.**SetDataFontColor**(Color)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Color	String	Required	Color value to set

- **Example**

```
//Set the data row font color as red.

mySheet.SetDataFontColor("#FF0000");
```

DataInsert Method

- **Purpose**

Create a new data row, and return the row index of the new row.

Row parameter and Level parameter will be set as follows:

Row setting	New row location	Level
Row < 0	Create as the last	Level 0

	row	
Row >= All rows	Create as the last row	Level 0
Row < First data row	Create as the first row	Level 0
Others	Create in the row	Set level
Default	Create below the selected row	If there is no setting, child level of the selected row

In case of a tree structure, the defaul is to create a row as child of selected row if Level parameter is not set. If the selected row is deleted, an error message is returned and creation will be aborted.

- **Syntax**

Syntax	ObjId.**DataInsert**([Row], [Level])

- **Info**

Return	Long, Row index of the new row		
Parameter	Type	Required Y/N	Description
Row	Long	Optional	Location of the new row, Default="immediately below the last row selected"
Level	Long	Optional	Tree level of the new row, Default="immediately above one level than the last row selected, create as a child"

- **Example**

```
//Create as the first row

mySheet.DataInsert(0);



//Create as the last row

mySheet.DataInsert(-1);



//Create immediately below the currently selected row

mySheet.DataInsert();



//Create as Row 7

mySheet.DataInsert(7);
```

DataLinkMouse Method

- **Purpose**

  Check or configure whether a data row links a page.

  If a page is linked to a data row which opens upon click or DbleClick, use this method to change mouse pointer to finger style to indicate a link.

  If Link parameter is set as 1, mouse pointer will change to finger style for the column.

- **Syntax**

Syntax	Get	ObjId.**GetDataLinkMouse**(Col)

- **Info**

Return	Boolean, set link value (In case of Get Method)		
Parameter	Type	Required Y/N	Description
Col	Long	Required	Column index of a particular column

	/String		or SaveName

- **Example**

```
// Check a data row for page link.

mySheet.GetDataLinkMouse(1);
```

- **Syntax**

Syntax	Set	ObjId.**SetDataLinkMouse**(Col, Link)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Col	Long /String	Required	Column index of a particular column or SaveName
Link	Boolean	Required	Set whether link exists

- **Example**

```
//Set to allow link for column 1 and 2 only.

mySheet.SetDataLinkMouse(1, 1);

mySheet.SetDataLinkMouse(2, 1);
```

DataMove Method

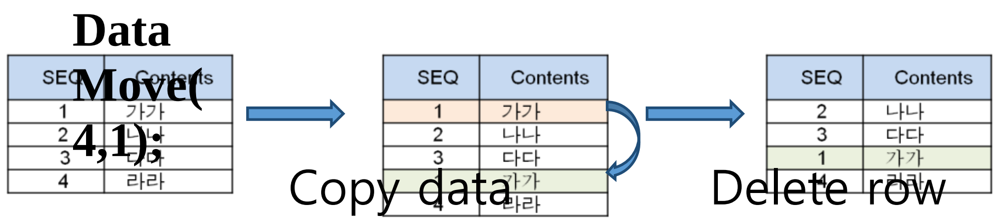
- **Purpose**

  Move data row to a desired location. In case of a tree type, any child level rows of the selected row will be moved as well.

  This function does not create new rows like DetaInsert or DataCopy but simply relocate existing rows. However, internally there will be copy and delete.

Therefore, attention should be paid to the pasting location if ToRow is bigger than FromRow.

For example, if ToRow is bigger than FromRow, it will work like below:



- **Syntax**

Syntax	ObjId.**DataMove**(ToRow, [FromRow], [RowLevel])

- **Info**

Return	Long, Top Row index of the relocated rows		
Parameter	Type	Required Y/N	Description
ToRow	Long	Required	Row Index of the new location
FromRow	Long	Optional	Row Index of the selected data Default=-1
RowLevel	Integer	Optional	Tree level of the relocated data Default="Original level"

- **Example**

```
//Move row 12 to row 10.

mySheet.DataMove(10, 12);
```

DataRowHeight Method

- **Purpose**

Check or configure row height of all data rows.

This property can be set at pixels. Default value is 21 pixels.

- **Syntax**

Syntax	Get	ObjId.**GetDataRowHeight**()

- **Info**

Return	Integer, set height value (In case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Check the height of all data rows.

mySheet.GetDataRowHeight();
```

- **Syntax**

Syntax	Set	ObjId.**SetDataRowHeight**(Height)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Height	Integer	Required	Data row height to set

- **Example**

```
//Set the height of all data rows at 22 pixels.

mySheet.SetDataRowHeight(22);
```

DataRowMerge Method

- **Purpose**

  Check or configure whether to horizontally merge all data rows.

  The default is that horizontal merging is not allowed unless the value is set as 0.

- **Syntax**

Syntax	Get	ObjId.**GetDataRowMerge**()

- **Info**

Return	Boolean, set value (In case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Whether to allow horizontal merge for all data rows

mySheet.GetDataRowMerge();
```

- **Syntax**

Syntax	Set	ObjId.**SetDataRowMerge**(Merge)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Merge	Boolean	Required	Whether to allow merge

- **Example**

```
//Allow horizontal merge for all data rows

mySheet.SetDataRowMerge(1);
```

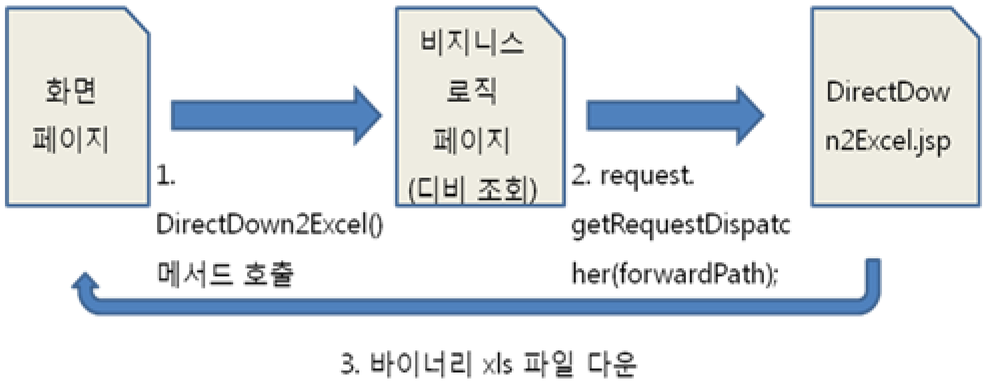
DirectDown2Excel Method

- **Purpose**

  This download data into excel the same way as Down2Excel. The different is that the data to download into excel is not data displayed in IBSheet but the data created in the server. This method will quickly create an excel file using the data.

  Therefore, the only data received from the IBSheet on the screen are header titles. The actual data will be transferred to DirectDown2Excel.jsp file by putting the data into request object named "SHEETDATA" in java.util.List(java.util.Map) structure, which is then downloaded into an excel file.

  The process is as follows:


화면 페이지
1. DirectDown2Excel() 메서드 호출
비지니스 로직 페이지 (디비 조회)
2. request. getRequestDispatcher(forwardPath);
DirectDown2Excel.jsp
3. 바이너리 xls 파일 다운

```
// Screen page

var cols = [

  {Type:"Text",Width:85,SaveName:"POSTNO",Format:"PostNo",Align:"center"},

        {Type:"Text",Width:70,SaveName:"SIDO"},

        {Type:"Text",Width:80,SaveName:"SIGUNGU"},

        {Type:"Text",Width:80,SaveName:"LEE"},

        {Type:"Text",Width:300,SaveName:"ADDRESS"}
];

mySheet.InitColumns(cols);



var param = {

       URL:"/bus/bussinessList.jsp"    //Business logic page

       ,ExtendParam:"sa_nm=양진열&sa_no=980123"

       ,FileName:"PersonList.xls"

};

sheet.DirectDown2Excel(param);
```

```
// bussinessList.jsp page

// 1. Receive search parameters from the screen.
String sa_name = request.getParameter("sa_nm");

String sa_no = request.getParameter("sa_no');

// 2. Inquire the data to download into excel from DB

String query

= "SELECT POSTNO, SIDO, SIGUNGU,   LEE, ADDRESS FROM POSTNO";

Class.forName(driver);

conn = DriverManager.getConnection(url,id,pwd);

pstmt = conn.prepareStatement(query);

rs = pstmt.executeQuery();

// 3. Convert data to List(Map).

java.util.List li = new java.util.List();

java.util.Map mp = null;

while(rs.next()){
    mp = new Java.util.Map();
    mp.put("POSTNO",rs.getString("POSTNO"));
    mp.put("SIDO",rs.getString("SIDO"));
    ..
    ..
    li.add(mp);
}
// Put data into object. Must use SHEETDATA as object name.
request.setAttribute("SHEETDATA", li);


// 4. Forward data to DirectDown2Excel.jsp page
System.out.println("All counts:"+li.size());
String forwardPath = "./DirectDown2Excel.jsp";
if(!"".equals(forwardPath)){
RequestDispatcher rd = request.getRequestDispatcher(forwardPath);
rd.forward(request,response);
}
```

The parameters to send are JSON type. Put configuration information into JSON format and send it to the server.

Ex)

var params = { FileName : "myFile.xls", SheetName : "Sheet"} ;

URL parameter is used to mark the page path where excel display data is populated. (**Required**)

ex) URL:"/bus/displayList.do"

FileName parameter is used to set the downloaded excel file name. If file extension is set as xls, excel 2003 format file is downloaded. If xlsx, excel 2007 format file is downloaded. If no value is set, an xls file is downloaded.

ex) FileName:"NameCard.xls"

SheetName parameter sets Worksheet names within an excel file. If the data is large enough to exceed the max number of rows allowed for a single Worksheet, which is 65536, () and index will be added to the name. For example, if the set name is "January", any data exceeding the limit will be put into a new spreadsheet in names "January (1)" and "January (2)".

ex) SheetName:"SupportTeam"

DownCols parameter is a string connecting all downloading columns using "|". You can use either SaveName or column index. If null, all columns are downloaded.

DownHeader parameter sets whether to include header when downloading. Default is 1(include header).

Merge parameter determines whether to merge columns if adjacent header data cells contain same letters. The default is 0.

SheetDesign parameter determines whether to download header color. The default is 0.

ExcelFontSize parameter can determine particular font size within the excel file, independently of SheetFontSize.

ExtendParam parameter is used to create a get method QueryString of search conditions to send to the server, which can be retrieved using request.getParameter() method from the page set in URL parameter.

ex)   ExtendParam:"name=shkim&sa_no=980123&enter_date=19980222";

- **Syntax**

Syntax	ObjId.**DirectDown2Excel**([parameters])

- **Example**

```
// Prepare and download data directly from the server.

var param = {URL:"/sub/ex/bussDeptList.jsp"

   ,ExtendParam:"DECNO=3422&PartMngNO=982211"

   ,FileName:"OrgList.xls"};

mySheet. DirectDown2Excel(param);
```

DirectLoadExcel Method

- **Purpose**

This method reads excel document like LoadExcel, but it does not put the excel contents into IBSheet but transfers them to a page designated by the server.

A page where excel data should be transferred should be developed separately, and designate forwarding page path (**FP=/jsp/excelsave.jsp**) in ExtendParam.

Forwarding page receives excel content using "SHEETDATA" as a name of a request object. SHEETDATA contents are composed of List(Map). Map keys are the SaveName of each IBSheet column.

ex) //forwarding page..

```
 List sheet = (List)request.getAttribute("SHEETDATA");

 for(int i=0;i<sheet.size();i++){

    Map mp =(Map)sheet.get(i);

 }
```

Also all the data transferred to ExtendParam can be loaded to the screen using getAttribute to the forwarding page.

p.s : You need to set DirectLoadExcel path within ibsheet.cfg to use this method.

- **Syntax**

Syntax	ObjId.**DirectLoadExcel**([parameters])

- **Info**

Parameter	Type	Required Y/N	Description
ColumnMapping	String	Optional	Excel column number Default=""
EndRow	String	Optional	Last row number after excel loading is completed Default="0"
ExtendParam	String	Required	Put parameters to send to server in QueryString **FP to perform saving is required**
FileExt	String	Optional	File extensions that can be uploaded Default=""
Mode	String	Optional	Loading options (header matching, etc.) Default="HeaderMatch"
StartRow	String	Optional	Excel loading row number Default="1"
WorkSheetNo	String	Optional	Excel spreadsheet number, Default="1"

WorkSheetName	String	Optional	Excel Worksheet name Default=""

- **Example**

```
// Reference function

function makeExParam(key,data){

   return   "&"+encodeURIComponent(key)+"="+encodeURIComponent(data);

}



// Forwarding page to transfer excel contents (except for ContextRoot name.)

var param =makeExParam( “FP" ,"/bu/MassSave.jsp");

param += makeExParam("sname","chris");

param += makeExParam("date","20091221”);

var parameters = { Mode : mch,   StartRow: "1", ExtendParam:param}

// Load from excel for immediate processing by the server

mySheet. DirectLoadExcel(parameters);

-----------------------------------------------------------------------------------------------

// MassSave.jsp page content

// 1. Check excel contents. (Check data)

String PRINT_STR = "";

ArrayList keys = new ArrayList();

List li = (List)request.getAttribute("SHEETDATA");

for(int i=0;i<li.size();i++){

            Map mp = (Map)li.get(i);

            // Send header row only.

            if(i==0){
                      Iterator it = mp.keySet().iterator();

                      while(it.hasNext()){

                                String key = (String)it.next();

                                PRINT_STR += key+"₩t";

                                keys.add(key);

                      }

                      PRINT_STR += "₩₩n";

            }

            // Send data

            for(int c=0;c<keys.size();c++){

                      PRINT_STR += mp.get(keys.get(c))+"₩t";

            }

            PRINT_STR += "₩₩n";

}

// Check from server console

System.out.println(PRINT_STR);



// 2. Check for the content sent to ExtendParam.

System.out.println(   request.getAttribute("sname"));

System.out.println(   request.getAttribute("date"));



// 3. Return the result to the screen.

out.println("<script>alert('Total data count :"+li.size()+"  has been₩nsaved.');</script>");
```

DoAllSave Method

- **Purpose**
  This method calls pages to save all data regardless of data transaction status.

If there is no data, a warning message will appear and process will be aborted.

OnValidation event will fire in the processing collecting data to save. Depending on the custom logic, failure of OnValidation may result in abortion of saving.

Call the save page using URL and complete saving to read saving XML. Then OnSaveEnd event fires and the whole process completes.

You can set how to combine Query String using Mode parameter.

Query String options based on parameter values are as follows:

Mode	Description	Ex)
1	Combine by cell (Array by SaveName)	sSeq=1&sStatus=R..
2	Combine by column (Combine using separator by SaveName)	sSeq=1|2&sStatus=R|U..

You can set property values for "Selection" parameter in JSON format. (See the sample)

- **Syntax**

Syntax	ObjId.**DoAllSave**(PageUrl, [Param], [UrlEncode], [Mode], [Delim])

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description

PageUrl	String	Required	Page file name to save
Param	String	Optional	Parameter for saving, Default=""
UrlEncode	Boolean	Optional	Whether to encode data on IBSheet, Default=1
Mode	Integer	Optional	How to combine string in QueryString, Mode=1, Mode=2 (Default=1)
Delim	String	Optional	Decide separator for Mode=2 (Default ="|")

➢ **Example**

```
// Save all
var Result = mySheet.DoAllSave("save.jsp", "id=khlee&seq=1");


//If saving fails, return an error message. If succeed, do search.
if(!Result){
    alert("Saving failed. Try again");
} else {
    mySheet.DoSearch("list.jsp");
}


//Set values for "Selection" parameter in JSON format
mySheet.DoAllSave(PageUrl, { UrlEncode:0, Mode:2, Delim:"$"});
```

DoPrint Method

➢ **Purpose**

Print all the data that are displayed.

Browser printing configuration is used to process printing. In order to print the background color or image, you need to change print configuration of the browser.

- **Syntax**

Syntax	ObjId.**DoPrint**()

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
N/A			

- **Example**

```
// Print

mySheet.DoPrint();
```

DoRowSearch Method

- **Purpose**

Search cell data of a particular row.

- **Syntax**

Syntax	ObjId.**DoRowSearch**(Row, PageUrl, [Param], [Opt])

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Index of the row
PageUrl	String	Required	URL of the page
Param	String	Optional	Search parameter Query String, Default ""

Opt.Wait	Boolean	Optional	Whether to display waiting image, Default =1
Opt.Sync	Boolean	Optional	Whether to sync search Default=0 (SearchSync value if SearchSync is set)

- **Example**

```
//Read column 3 data from DB if the data in column 3 has been updated

function mySheet_OnChange(Row, Col, Value) {

   if (Col == 3) {

          var opt = { Wait : 1, Sync : 1 };

          mySheet.DoRowSearch(Row, "grid_rowdata.html",   "" , opt);

   }

}

// 1. Do not display image, async search

var opt = { Wait : 0, Sync : 0 };

mySheet.DoRowSearch(Row, "grid_rowdata.html",   "" , opt);



// 2. Display image, sync search

var opt = { Wait : 1, Sync : 1 };

mySheet.DoRowSearch(Row, "grid_rowdata.html",   "" , opt);
```

DoSave Method

- **Purpose**

  Save data based on data transaction status or column.

  If Col parameter is not set, data rows whose transaction status is not “Search” is saved. If there is a particular parameter set in Col, data with values in the designated column will be saved.

If the column is in CheckBox format, only checked boxes will be saved.

If there is no data to save, a warning message will appear and the process is dropped.

OnValidation event will fire in the processing collecting data to save. Depending on the custom logic, failure of OnValidation may result in abortion of saving.

Call the save page using URL and complete saving to read saving XML. Then OnSaveEnd event fires and the whole process completes.

You can set how to combine Query String using Mode parameter.

Query String options based on parameter values are as follows:

Mode	Description	Ex)
1	Combine by cell (Array by SaveName)	sSeq=1&sStatus=R..
2	Combine by column (Combine using separator by SaveName)	sSeq=1|2&sStatus=R|U..

"Selection" parameter can be set in JSON format. (See the example)

➢ **Syntax**

Syntax	ObjId.**DoSave**(PageUrl, [Param], [Col] , [Quest], [UrlEncode], [Mode], [Delim])

➢ **Info**

Return	None		
Parameter	Type	Required Y/N	Description
PageUrl	String	Required	Page file name to save
Param	String	Optional	Parameter for saving, Default=""
Col	Long / String	Optional	Column to save, or SaveName Default=Status column(-1)
Quest	Boolean	Optional	Whether to display confirmation message before saving, Default=1
UrlEncode	Boolean	Optional	Whether to encode data on IBSheet, Default=1
Mode	Integer	Optional	How to combine strings for Query String, Mode=1, Mode=2 (Default=1)
Delim	String	Optional	Set the separator for Mode=2 (Default="|")

➢ **Example**

```
//Save only the data with transaction happening

mySheet.DoSave("Save.jsp" ,"id=khlee&seq=1");



//Save checked checkboxes in Column 2

mySheet.DoSave("Save.jsp" ,"id=khlee&seq=1", 2);



//Set “Selection” parameter value in json format

mySheet.DoSave(PageUrl, {UrlEncode:0, Mode:2, Delim:"$"});
```

DoSearch Method

- **Purpose**

  Connect to search page to read search XML, and then load XML data internally in IBSheet

  Param parameter can be set by connecting conditions using "=" and "&", as in "Condition name=value 1&condition name 2=value 2".

  In Opt parameter, an object-type parameter, you can set whether to do Sync search (Sync) and Append search (Append).

  Sync parameter is sync/async search mode. Async search means when there are multiple calls sent, following calls for search will be ignored if the first search is not complete. If you need to run multiple calls and all searches must be complete, use sync mode.

  If you set Append parameter as 1, you can append the existing data to the current search data to run search.

  Call the search page using URL and complete data representation by reading search data. Then OnSearchEnd event fires and the whole process completes.

- **Syntax**

Syntax	ObjId.**DoSearch**(PageUrl, [Param], [Opt])

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
PageUrl	String	Required	Search XML page file name
Param	String	Optional	Search condition Query String, Default=""
Opt.Sync	Boolean	Optional	Sync search or not Default=0 (SearchSync value when SearchSync is set)

Opt.Append	Boolean	Optional	Append search result or not, Default=0

- **Example**

```
// 1. General search

mySheet.DoSearch("list.jsp", "p1=aa&p2=bb");



// 2. Sync search

var opt = { Sync : 1 };

mySheet.DoSearch("list.jsp", "p1=aa&p2=bb", opt);



// 3. Append search

var opt = { Append : 1 };

mySheet.DoSearch("list.jsp", "p1=aa&p2=bb", opt);



// 4. Sync && Append search

var opt = { Sync : 1, Append : 1 };

mySheet.DoSearch("list.jsp", "p1=aa&p2=bb", opt);
```

DoSearchChild Method

- **Purpose**

  In a tree data structure, this method connects to child data search page within OnTreeChild event to read search XML and JSON, and append XML and JSON data as child.
  Row parameter transfers parent parameter to append child data through OnTreeChild event.

  Call the page to run child data search using **URL** and read search XML and JSON to complete data representation. OnSearchEnd event fires and the whole process is completed.

  Param parameter can be configured by connecting search conditions using "=" and "&" as in the following format: "Condition name 1=Value 1&Condition **Name 2= Value 2".**

Wait parameter can be used to set whether to display waiting image during search.

- **Syntax**

Syntax	ObjId.**DoSearchChild**(Row, PageUrl, [Param], [Opt])

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Index of the row received from OnTreeChild event
PageUrl	String	Required	File name of search XML and JSON pages
Param	String	Optional	Search parameter Query String, Default=""
Opt.Wait	Boolean	Optional	Whether to display waiting image Default=1
Opt.Sync	Boolean	Optional	Sync search or not Default=0 (SearchSync value when SearchSync is set)

- **Example**

```
// Search the child data

<script type="text/Javascript">
```

```
 function mySheet_OnTreeChild(Row){

    var url = "";

    // Column 4 : Tree column

    switch(mySheet.GetCellValue(Row, 4)){

     case "Seoul" :

        url = " type15_dat(1).xml";

        break;

     case "Incheon":
        url = "type15_data(2).xml";

        break;

    }

    var opt = { Wait : 1, Sync : 0 };

    mySheet.DoSearchChild(Row, url, "", opt);

 }

</script>



// 1. Do not display image, async search

var opt = { Wait : 0, Sync : 0 };

mySheet.DoSearchChild(Row, url, "", opt);


// 2. Display image, sync search

var opt = { Wait : 1, Sync : 1 };

mySheet.DoSearchChild(Row, url, "", opt);
```

DoSearchPaging Method

➢ **Purpose**

When searching a large amount of data, search only partial data corresponding to the IBSheet scroll location to display on the screen.

As the page scrolls, the Url configured as a parameter is called to display further search result data corresponding to the new scroll location.

When this function is used, SearchMode should be set as 3 in the initialization method SetConfig, and Page property must be configured.

The total scroll size will be determined by Total property value of the search data. Total property value must be set.

When this function is used for search, there may be some restrictions to Row and Cell properties or use of multi-transaction function.

- **Syntax**

Syntax	ObjId.**DoSearchPaging**(Url, [Info])

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Url	String	Required	Search page Url
Info.PageParam	String	Optional	Parameter name to receive page index, Default="ibpage"
Info.Param	String	Optional	Search parameter Query String, Default=""
Info. OrderbyParam	String	Optional	Parameter name to receive header sorting information, Default="iborderby" As in “SIDO|SIGUNGU ^ASC|DESC”, the SaveName and sorting direction are separated by “^” and

			each name with “|”.
Info. UseWaitImage	Boolean	Optional	When WaitImageVisible setting is true, whether to display waiting image for search of 2 pages or more. Default=0
Info.Sync	Boolean	Optional	Sync search or not Default=0 (SearchSync value when SearchSync is set)

- **Example**

```
//Default setting (Set page size at 100)

var cfg = {SearchMode:3, Page:100};

mySheet.SetConfig(cfg);



// Real-time search

var info = {PageParam: "page", OrderbyParam:"orderbyParam", Param: "id=ibleaders&seq=1"};

mySheet.DoSearchPaging("list.jsp",info);



// Sync search

var info = {PageParam: "page", OrderbyParam:"orderbyParam", Param: "id=ibleaders&seq=1", Sync : 1};

mySheet.DoSearchPaging("list.jsp",info);
```

Down2Excel Method

- **Purpose**

IF there are any search result data returned, download the data displayed in IBSheet into an excel file.

The parameters to send are JSON type. Put configuration information into JSON format and send it to the server.

Sample)

```
var params = { FileName : "myFile.xls",   SheetName : "Sheet"} ;

mySheet.Down2Excel(params);
```

FileName parameter is used to set the downloaded excel file name. If file extension is set as xls, excel 2003 format file is downloaded. If xlsx, excel 2007 format file is downloaded. If no value is set, an xls file is downloaded.

SheetName parameter sets Worksheet names within an excel file. If the data is large enough to exceed the max number of rows allowed for a single Worksheet, which is 65536, () and index will be added to the name. For example, if the set name is "January", any data exceeding the limit will be put into a new spreadsheet in names "January(1)" and "January(2)".

DownRows parameter is a string connecting all downloading rows using "|". If this parameter is null, all rows are downloaded. If the value is "Visible", all rows that are currently visible will be downloaded, excluding hidden rows. If this parameter is not null, **DownTreeHide** parameter will be set as false.

DownCols parameter is a string connecting all downloading columns using "|". You can use either SaveName or column index. If this is null, all columns are downloaded.

HiddenColumn parameter downloads hidden columns as hidden. The columns will not be visible initially but you can unhide them by using excel "unhide" menu option.

DownHeader parameter sets whether to include header when downloading. Default is 1(include header).

DownSum parameter sets whether to include sum rows to download. Default is 1(include sum).

Merge parameter determines whether to apply IBSheet merge status to excel document. Using this parameter may result in further delay of performance. Default is 0 (not use merging).

※ For use of merge settings, see Appendix#4. Applying Merge Setting for Excel Download.

SheetDesign parameter reflects IBSheet design. Font name, font size and background color are available. Using this parameter may result in further delay of performance. Default is 0 (Not apply design). Font color will always display in black. Multiple font coloring is not supported. For cell background color, up to 48 colors may be used concurrently for a single excel file. If one IBSheet includes a larger number of colors, some of the colors will display to the closest color among the 48 available colors. If this parameter is set as 2, design will be still reflected as in 1, but cell borders will not be lined.

CheckBoxOnValue parameter is used to define other values than 1 if you have checked checkbox and radio box. Default is 1.

CheckBoxOffValue parameter can be used to define values other than 0 if you have unchecked a checkbox or radio box. Default is 0.

DownCombo parameter allows you to download optional attributes of combo and combo edit in TEXT or CODE format. The default is "TEXT". If you change the value to "CODE", download format will be CODE instead of TEXT.

TitleText parameter allows the user to customize title at the top of grid or other texts. If you enter A|B|C₩r₩nD|E|F for this parameter, three cells in the first row will be populated with A, B and C respectively, and those in the second row with D, E and F. To include 'enter' within value, insert ₩r or ₩n. If the total ₩r₩n count is 10, it will take up 11 rows. IBSheet data will continue from the 12th row.

If cell data put into this parameter have more columns than the columns displayed in excel, those values will be ignored. If the # of columns displayed in an excel file is 10 in total, you can only configure 10 titles or other texts. (No such restrictions to the title row)

UserMerge parameter is configured to apply merge to cells entered using TitleText or cells from IBSheet. One merge statement is composed of four numbers and commas. Multiple merges can be combined using space character as separator. If the parameter value is set as "0,0,2,2 0,2,1,8", 2X2 merge is applied to the first row, first column of the excel (0, 0). Then another, a 1x8 merge, will apply to (0,2), a cell in the first row and Column 2(=third column) .

OnlyHeaderMerge parameter restricts merge in the data area to improve performance.

ExcelFontSize parameter can designate font size in excel document separately from SheetFontSize.

ExcelRowHeight parameter is uesd to fix all the rows to a particular pixel size only in excel document. If this is set as "auto", excel row height is automatically adjusted.

The page configured in **URL** parameter is called before Down2Excel.jsp page when you put a URL to process a logic when there are any server side jobs that should be processed by the server along with Down2Excel (logging, for example). Therefore, configuration page must send a request to Down2Excel.jsp page after the jobs are completed.

Sample)

```
var param = { URL:"/ibsheet7_down2excel_extendparam/fp.jsp"};

mySheet.Down2Excel(param);
```

Server side page)

```
RequestDispatcherrd=
request.getRequestDispatcher("/ibsheet7_down2excel_extendparam/IBSheet/Down2Excel.jsp");
```

rd.forward(request,response);

ExtendParam parameter is used when you have instructions to send to the server. Connect them into a QueryString of GetMethod and set it in this parameter to deliver it to the same page as set in URL.

Sample)

param = { ExtendParam:"**sawon_name=shkim&sawon_no=12345**",

URL:"/ibsheet7_down2excel_extendparam/fp.jsp };

ExtendParamMethod determines whether to send ExtendParam contents using GET or POST method.

TextToGeneral parameter is used when IBSheet Text type columns are downloaded in an excel file and you need to change them into general or text type.

DownTreeHide parameter is used to download the rows collapsed in a tree structure.

If this parameter is set true, hidden rows are all downloaded using SetRowHidden.

KeyFieldMark parameter is used to download KeyFIeld mark (*) in KeyField column.

TreeLevel parameter is used to download tree levels in a tree-structure column.

WordWrap parameter is used to configure whether to allow multilines for a text cell.

AutoSizeColumn parameter is used to adjust column width to fit the column text. Still, automatic adjustment may not be precise.

ExcludeSubSum parameter can be used if you want to exclude sub sum or aggregate total rows from downloaded rows.

ComboValidation parameter is used to set whether to create combo type columns in the data area (excluding the header, sub sum, aggregate total or sum rows) as a drop-down list (data validation). The default value is 0, which means data validation will not be done.

When this parameter is set as 1, the dropdown list items will use either the ComboCode as set in the column if DownCombo value is "Code", and ComboText if "Text".

Combo list updated using InitCellProperty will not be updated individually to excel's dropdown list, but the values will display intact when downloaded in excel.

- **Syntax**

Syntax	ObjId.**Down2Excel**([parameters])

- **Info**

Parameter	Type	Required Y/N	Description
AutoSizeColumn	bool	Optional	Default=0(do not auto adjust column width)
CheckBoxOffValue	String	Optional	Value when checkbox is unchecked Default="0"
CheckBoxOnValue	String	Optional	Value when checkbox is checked Default="1"
ComboValidation	bool	Optional	Whether to download combo dropdown format Default=0
DownCols	String	Optional	Connect columns to download using | Default=""(Download all)

DownCombo	String	Optional	Whether to download combo in TEXT/CODE format Default="TEXT"
DownHeader	bool	Optional	Whether to download header Default=1
DownRows	String	Optional	Connect rows to download using | Default=""(Download all)
DownSum	bool	Optional	Whether to download sums Default=1
DownTreeHide	bool	Optional	Whether to download collapsed rows in a tree Default=0(Not download)
ExcelFontSize	Integer	Optional	Font size setting Default=0
ExcelRowHeight	String	Optional	Default=""(Not use)
ExcludeSubSum	Integer	Optional	Whether to exclude partial sum/aggregate rows 1: exclude partial sums only 2: exclude aggregate only 3: exclude both partial sum/aggregates Default=0 (Include both partial sum and aggregates)
ExtendParam	String	Optional	Default=""(Not use)
ExtendParamMethod	String	Optional	Default="GET"
FileName	String	Optional	File name to save

			Default="Excel.xls"
HiddenColumn	bool	Optional	Whether to apply hiding status to download file Default=0
KeyFieldMark	bool	Optional	Whether to download Keyfield mark(*) Default=1(Download)
Merge	bool	Optional	Whether to apply merge to download Default=0
OnlyHeaderMerge	bool	Optional	Whether to merge the header only Default=0
SheetDesign	Integer	Optional	Whether to apply IBSheet design concept to download file Default=0
SheetName	String	Optional	Excel worksheet name, Default="Sheet"
TextToGeneral	bool	Optional	Text type excel format Default=1(General)
TitleText	String	Optional	Default=""(Not use)
TreeLevel	bool	Optional	Default=0(Not download)
URL	String	Optional	Default=""(Not use)
UserMerge	String	Optional	Default=""(Not use)
WordWrap	bool	Optional	Default=1(allow multi-line)

➢ **Example**

```
// Download in excel

mySheet.Down2Excel();



// Set the download file name as excel2, and worksheet name as sheet-test.

mySheet.Down2Excel({FileName:'excel2',SheetName:' sheet-test'});



//Apply sheet color and merge to download file and use code for combo and Y/N for check. Exclude Header and sum, and download three columns starting from the left.

mySheet.Down2Excel({SheetDesign:1, Merge:1, DownCombo:'CODE',
  CheckBoxOnValue:'Y', CheckBoxOffValue:'N',   DownRows:'', DownCols:'0|1|2',
  DownHeader:0, DownSum:0});
```

Down2ExcelUrl Method

➢ **Purpose**

Check and configure server page path to process excel download.

➢ **Syntax**

Syntax	Get	ObjId.**GetDown2ExcelUrl**()

➢ **Info**

Return	String, set path value (in case of Get Method)		
Parameter	Type	Required Y/N	Description

➢ **Example**

```
//Check the excel download path.
```

```
var url = mySheet.GetDown2ExcelUrl();
```

- **Syntax**

Syntax	Set	ObjId.**SetDown2ExcelUrl**(Url)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Url	String	Required	Server page URL to set

- **Example**

```
// Configure excel download path.

mySheet.SetDown2ExcelUrl("/jsp/Down2Excel.jsp");
```

Down2ExcelBuffer Method

- **Purpose**
  Download multiple sheets into a single excel document.

If buffer parameter in Down2ExcelBuffer is set as true, all Down2Excel executed afterwards will not actually run and buffered to internal memory.

Afterwards, when the buffer parameter of Down2ExcelBuffer is changed to false, all the sheets buffered will be downloaded as separate spreadsheet within a single excel file.

Among the excel file names you set during buffering, the first name and excel file format will be effective. If worksheet name is redundant, name will change automatically adding brackets and serial numbers.

➢ **Syntax**

Syntax	ObjId.**Down2ExcelBuffer**(IsBuffer)

➢ **Info**

Parameter	Type	Required Y/N	Description
IsBuffer	bool	Required	Buffering or not

➢ **Example**

```
//Buffer afterwards. Do nothing.
firstSheet.Down2ExcelBuffer(true);


//Reserve downloading into the first worksheet.
firstSheet.Down2Excel({FileName:'excel2',SheetName:'sheet1'});


//Reserve downloading into the second worksheet.
secondSheet.Down2Excel({FileName:'excel2',SheetName:' sheet2'});


// Download all buffered sheet data into a single excel document immediately.
firstSheet.Down2ExcelBuffer(false);
```

Down2Pdf Method

➢ **Purpose**

If there are any returned data, convert IBSheet contents into a PDF file to download.

The parameters to send are in JSON type. Put configuration information into JSON format and send it to the server.

Sample)

var params = { FileName:"myPDF.pdf"};

mySheet.Down2Pdf(params);

FileName parameter sets the file name for downloaded PDF file. File extension must be pdf. You can skip adding extension and then .pdf will be added automatically when file is downloaded.

DownCols parameter is a string of all download columns separated with "|" You can use either SaveName or Column Index. If this parameter is null, all column data will be downloaded.

Sample)

var params = { DownCols:"4|5|6|7|8|9|10"};

mySheet.Down2Pdf(params);

Paper parameter sets the print orientation for PDF file. Vertical: landscape, horizontal: portrait

Sample)

var params = { Paper:" landscape"};

mySheet.Down2Pdf(params);

Dpi parameter sets the enlargement scale for PDF file. Value may be set between 50 and 32800. Bigger the value is, smaller the print size gets.

Sample)

var params = { Dpi:1800};

mySheet.Down2Pdf(params);

Title parameter sets the title to print in PDF file.

TitleStyle parameter sets css style to apply to title to print in PDF file.

Sample)

var params = {Title:"IBSheet PDF file", TitleStyle:"color:red;size:12pt;" };

mySheet.Down2Pdf(params);

The page configured in **URL** parameter is called before Down2Pdf.jsp page when you put a URL to process a logic when there are any server side jobs that should be processed by the server along with Down2Pdf (logging, for example). Therefore, configuration page must send a request to Down2Pdf.jsp page after the jobs are completed.

Sample)

var param = { URL:"/ibsheet7_down2pdf/fp.jsp"};

mySheet.Down2Pdf(param);

Server side page)

RequestDispatcher rd= request.getRequestDispatcher("/ibsheet7_down2pdf/ Down2Pdf.jsp");

rd.forward(request,response);

ExtendParam parameter is used when you have instructions to send to the server. Connect them into a QueryString of GetMethod and set it in this parameter to deliver it to the same page as set in URL.

Sample)

param = { ExtendParam:"**sawon_name=shkim&sawon_no=12345**",

URL: "/ibsheet7_down2pdf/fp.jsp"};

mySheet.Down2Pdf(param);

ExtendParamMethod determines whether to send ExtendParam contents using GET or POST method.

FontTo parameter sets the Korean font to use in PDF. When the sheet includes Korean characters, set either Gulim or Gothic for font, as they are supported by PDF conversion module. (Gulim or Gothic)

Sample)

var params = { FontTo:" Gulim"};

mySheet.Down2Pdf(params);

- **Syntax**

Syntax	ObjId.**Down2Pdf**([parameters])

- **Info**

Parameter	Type	Required Y/N	Description
DownCols	String	Optional	Connect columns to download using | Default=""(Download all)
Dpi	Integer	Optional	Enlargement scale. Bigger the print image is the smaller the value is. Value may be set between 50 and32840. Default = 2000

ExtendParam	String	Optional	If there are any instructions to send to the server, connect them into a Get Method QueryString. Default=""
ExtendParam	String	Optional	Default=""(Not use)
ExtendParamMethod	String	Optional	Default="GET"
FileName	String	Optional	File name to save Default="IBSheet.pdf"
FontTo	String	Optional	Default = "Gothic"
Paper	String	Optional	Print orientation Landscape or portrait Default = "landscape"
Title	String	Optional	Default = ""
TitleStyle	String	Optional	Default = ""
URL	String	Optional	Default=""(Not use)

- **Example**

```
// Download to PDF.

mySheet.Down2Pdf();



// Set the download file name as text and download the file.

mySheet.Down2Pdf({FileName:'text' });



// Set the download columns and download them into myPDF.pdf.

mySheet.Down2Pdf({FileName:"myPDF", DownCols:"7|8|9|4|5|6|10"});
```

Down2PdfUrl Method

- **Purpose**

  Check and configure server page path to process PDF download.

- **Syntax**

Syntax	Get	ObjId.**GetDown2PdfUrl**()

- **Info**

Return	String, set path value (in case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Check the PDF download path.

var url = mySheet.GetDown2PdfUrl();
```

- **Syntax**

Syntax	Set	ObjId.**SetDown2PdfUrl**(Url)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Url	String	Required	Server page URL to set

- **Example**

```
// Configure PDF download path.

mySheet.SetDown2PdfUrl("/jsp/Down2Pdf.jsp");
```

Down2Text Method

- **Purpose**
  If there are any returned data, convert IBSheet contents into a text file to download.

The parameters to send are in JSON type. Put configuration information into JSON format and send it to the server.

Sample)

```
var params = { FileName : "myFile.txt "};

mySheet.Down2Text(params);
```

FileName parameter sets the file name for downloaded text file. File extension must be txt. You can skip adding extension and then .txt will be added automatically when file is downloaded.

DownRows parameter is a string of all download rows separated with "|" If this parameter is null, all row data will be downloaded.

DownCols parameter is a string of all download columns separated with "|" You can use either SaveName or Column Index. If this parameter is null, all column data will be downloaded.

DownHeader parameter configures whether to include header for download. Default is 1 (Include header).

DownSum parameter configures whether to include sums for download. Default is 1 (Include Sums).

DownCombo parameter allows you to download optional attributes of combo and combo edit in TEXT or CODE format. The default is "TEXT". If you change the value to "CODE", download format will be CODE instead of TEXT.

ExtendParam parameter is used when you have instructions to send to the server. Connect them into a QueryString of GetMethod and set it in this parameter to deliver it to the same page as set in URL.

Sample)

param = {**ExtendParam**:"**sawon_name=shkim&sawon_no=12345**",

URL:"/ibsheet7_down2text_extendparam/fp.jsp };

DownTreeHide parameter is used to download the rows collapsed in a tree structure.

- **Syntax**

Syntax	ObjId.**Down2Text**([parameters])

- **Info**

Parameter	Type	Required Y/N	Description
FileName	String	Optional	File name to save Default="Test.txt"
RowDelim	String	Optional	Record separator to display in row data Default="₩n" (Enter format)
ColDelim	String	Optional	Column separator to display in row data Default=" " (Space format)
DownRows	String	Optional	Connect rows to download using | Default=""(Download all)

DownCols	String	Optional	Connect columns to download using | Default=""(Download all)
DownHeader	bool	Optional	Whether to download header Default=1
DownSum	bool	Optional	Whether to download sums Default=1
DownCombo	String	Optional	Whether to download combo in TEXT/CODE format Default="TEXT"
ExtendParam	String	Optional	If there are any instructions to send to the server, connect them into a Get Method QueryString. Default=""
DownTreeHide	bool	Optional	Whether to download collapsed rows in a tree Default=0(Not download)

- **Example**

```
// Download to text.

mySheet.Down2Text();



// Set the download file name as text and download the file.

mySheet.Down2Text({FileName:'text' });



//Use code for combo and exclude header and sum. Download three
columns starting from the left.

mySheet.Down2Text({DownCombo:'CODE', DownRows:'', DownCols:'0|1|2',
DownHeader:0, DownSum:0});
```

Down2TextUrl Method

- **Purpose**

  Check and configure server page path to process text file download.

- **Syntax**

Syntax	Get	ObjId.**GetDown2TextUrl**()

- **Info**

Return	String, set path value (in case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Check the text download path.

var url = mySheet.GetDown2TextUrl();
```

- **Syntax**

Syntax	Set	ObjId.**SetDown2TextUrl**(Url)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Url	String	Required	Server page URL to set

- **Example**

```
// Configure text download path.

mySheet.SetDown2TextUrl("/jsp/Down2Text.jsp");
```

DownloadingImage Method

- **Purpose**

  Check or configure the waiting image file location to display during file download.

  This method can be used to customize waiting image as the user want which displays during file upload.

- **Syntax**

Syntax	Get	ObjId.**GetDownloadingImage**()

- **Info**

Return	String, currently set value (In case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Check the currently set image path of the download waiting image.

alert(mySheet.GetDownloadingImage());
```

- **Syntax**

Syntax	Set	ObjId.**SetDownloadingImage**(Url)

- **Info**

Return	None		
Parameter	Type	Required	Description

		Y/N	
Url	String	Required	Image URL

- **Example**

```
//Change the download waiting image.

mySheet.SetDownloadingImage( "/sheet/imgDownload.gif");
```

DragMode Method

- **Purpose**

  Check or configure mouse dragging mode.

  Available options and values are as follows:

Value	Details	
0 (Default)	General	Select a range of cells or rows
	Use Ctrl key	Drag rows
1	General	Drag rows
	Use Ctrl key	Select a range of cells or rows

- **Syntax**

Syntax	Get	ObjId.**GetDragMode**()

- **Info**

Return	Boolean, set value (In case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Check DragMode configuration

var mode = mySheet.GetDragMode();
```

➢ **Syntax**

Syntax	Set	ObjId.**SetDragMode**(Mode)

➢ **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Mode	Boolean	Required	Drag effect configuration (Default=0)

➢ **Example**

```
// DragMode configuration (configure how mouse dragging works on row dragging)

mySheet.SetDragMode(1);
```

Editable Method

➢ **Purpose**

Check or configure overall editability.

If overall editing is not allowed, all cells become uneditable regardless of other settings.

Overall editability is determined as follows:

Editable	ColEditable	RowEditable	CellEditable	**Cell editable Y/N**
No	No impact	No impact	No impact	**No**
Yes	No	No	Yes/No	**Yes/No**
Yes	Yes	Yes/No	Yes/No	**Yes/No**

➢ **Syntax**

Syntax	Get	ObjId.**GetEditable**()

- **Info**

Return	Boolean, set editability value (In case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Check editability status

mySheet.GetEditable();
```

- **Syntax**

Syntax	Set	ObjId.**SetEditable**(Edit)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Edit	Boolean	Required	Editability value to set

- **Example**

```
//Configure overall editability before initial load

mySheet.SetEditable(1);
```

EditEnterBehavior Method

- **Purpose**

  Check or configure enter key behavior after editing data

  Remember this method is different from EnterBehavior Method. EnterBehavior method is used to define enter key behavior when enter key is pressed when mouse is simply focused not in editing mode. This method is used to define enter key behavior when the key is pressed to complete editing.

When newline parameter is set, columns with MultiLineText parameter value set as 1 in InitColumns will become multiline columns. (This is not supported for Opera browser)

Value	Description
"tab"	Move horizontally to the next cell like Tab key behavior
"down"	Move down to the next cell like Down key behavior
"newline"	New line is created
"none"	Do nothing

- **Syntax**

Syntax	Get	ObjId.**GetEditEnterBehavior**()

- **Info**

Return	String, set parameter value (in case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
Check Enter Key behavior.

mySheet.GetEditEnterBehavior( );
```

- **Syntax**

Syntax	Set	ObjId.**SetEditEnterBehavior**(Mode)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Mode	String	Required	Value to define enter key behavior in editing mode Default="tab"

- **Example**

```
// Make enter key behave like down key when pressed after editing

mySheet.SetEditEnterBehavior( "down");
```

EditableColorDiff Method

- **Purpose**

  Check or configure whether to mark uneditable cells with different color.

  Available options and values are as follows:

Type	Description
0	Do not mark uneditable cells
1	Mark uneditable cells with the color configured in css
2	Mark uneditable cells with combination of color configured in css and basic background color

- **Syntax**

Syntax	Get	ObjId.**GetEditableColorDiff**()

- **Info**

Return	Integer, set value (In case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Check how uneditable cells are marked

mySheet.GetEditableColorDiff ();
```

- **Syntax**

Syntax	Set	ObjId.**SetEditableColorDiff**(Mode)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Mode	Integer	Required	Value

- **Example**

```
//Do not mark uneditable cells

mySheet.SetEditableColorDiff (0);
```

EditArrowBehavior Method

- **Purpose**

Check or configure arrow (up, down, left and right keys) behavior in editing mode.

Arrow key	Vertical movement to adjacent cells	Horizontal movement to adjacent cells
0	No	No
1	Yes	No
2	No	Yes
3	Yes	Yes

➢ **Syntax**

Syntax	Get	ObjId.**GetEditArrowBehavior**()

➢ **Info**

Return	Integer, set value (In case of Get Method)		
Parameter	Type	Required Y/N	Description

➢ **Example**

```
//Check the configuration

mySheet.GetEditArrowBehavior();
```

➢ **Syntax**

Syntax	Set	ObjId.**SetEditArrowBehavior**(behavior)

➢ **Info**

Return	None		
Parameter	Type	Required Y/N	Description

behavior	Integer	Required	Movement value to set

- **Example**

```
// Move focus to up, down, left or right cell when arrow keys are pressed in editing mode.

mySheet.SetEditArrowBehavior(3);
```

EditTabBehavior Method

- **Purpose**

  Check or configure tab key behavior after data editing

  If this method is set true, focus will move to the next cell even if it is uneditable.

  If set false, focus will move to the next editable cell.

- **Syntax**

Syntax	Get	ObjId.**GetEditTabBehavior**()

- **Info**

Return	String, set parameter value (in case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
// Check the tab key behavior.

mySheet.GetEditTabBehavior();
```

- **Syntax**

Syntax	Set	ObjId.**SetEditTabBehavior**(Mode)

➢ **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Mode	Boolean	Required	Value to define tab key behavior in editing mode Default="0"

➢ **Example**

```
// Set tab key behavior after editing to move to the next cell.

mySheet.SetEditTabBehavior(1);
```

Ellipsis Method

➢ **Purpose**

Check or configure whether to show ellipsis for overflowing test.

In default, overflowing text will be cut off if text within a cell is longer than the width of the column. Use this method to replace such overflowing text with ellipsis ("...").

To display overflowing text, you can also use Wrap:1 property of InitColumn to set auto multiline.

➢ **Syntax**

Syntax	Get	ObjId.**GetEllipsis**()

➢ **Info**

Return	Boolean, availability value (In case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Check the configuration

mySheet.GetEllipsis();
```

- **Syntax**

Syntax	Set	ObjId.**SetEllipsis**(Flag)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Flag	Boolean	Required	Whether to use ellipsis Default=0

- **Example**

```
// Use ellipsis

mySheet.SetEllipsis(1);
```

Enable Method

- **Purpose**

  If you set this property value at 0, all user interface functions using mouse or keyboard will be disabled. Methods or properties offered by other product are still available as they can be called by coding.

- **Syntax**

Syntax	Get	ObjId.**GetEnable**()

- **Info**

Return	Boolean, availability value (In case of Get Method)

Parameter	Type	Required Y/N	Description

➢ **Example**

```
//Check availability of user interface.

mySheet.GetEnable();
```

➢ **Syntax**

Syntax	Set	ObjId.**SetEnable**(Enable)

➢ **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Enable	Boolean	Required	WHether user interface is enabled

➢ **Example**

```
//Disable user interface.

mySheet.SetEnable(0);



//Enable user interface.

mySheet.SetEnable(1);
```

EnterBehavior Method

➢ **Purpose**

When focus is on a cell, pressing TAB key moves focus to the next cell, and pressing enter ey starts editing. This method can be used to set different enter behavior from this.

In default, enter key will start editing, If you want to use enter key like tab key to move focus, set this property as "tab".

Value	Description
“tab”	Move to the next cell like tab key behavior
“edit”	Start editing (Default)
“down”	Move down
“none”	Do nothing

- **Syntax**

Syntax	Get	ObjId.**GetEnterBehavior**()

- **Info**

Return	String, set parameter value (in case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Check the configuration.

mySheet.GetEnterBehavior();
```

- **Syntax**

Syntax	Set	ObjId.**SetEnterBehavior**(Mode)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description

Mode	String	Required	Property value to define enter key behavior Default="edit"

➢ **Example**

```
//Set enter key behavior as moving horizontally to the next cell

mySheet.SetEnterBehavior("tab");
```

EtcData Method

➢ **Purpose**

Check or configure information other than data information.

This property can be used to save any search result using a search method, or additional search result added on saving result as ecetra data.

Ecetra data are composed of key name and key value. Properties may be checked or configured directly.

Use <ETC-DATA> tag in XML format as follows:

Basic structure	**<ETC-DATA>**
	
	      <**ETC KEY**="key name">key value**</ETC**>
	
	      <**ETC KEY**="name">Gildong Hong**</ETC**>
	
	      <**ETC KEY**="age">30**</ETC>**
	
	</ETC-DATA>

Search	Set between <SHEET> tag and <DATA> tag. Sample) <?xml version='1.0' ?> **<SHEET>** <ETC-DATA>

	<ETC KEY="name">Gildong Hong</ETC>
	    <ETC KEY="age">30</ETC>
	  </ETC-DATA>
	  <DATA>
	    <TR>
	      <TD>CWOF-171</TD>
	      <TD>17x3.3 square meters</TD>
	      <TD>2040000</TD>
	      <TD>2101200</TD>
	    </TR>
	  </DATA>
	</SHEET>

Save	Set between <SHEET> tag and <RESULT> tag.
	Sample)
	<?xml version='1.0' ?>
	<SHEET>
	  <ETC-DATA>
	    <ETC KEY="name">Gildong Hong</ETC>
	    <ETC KEY="age">30</ETC>
	  </ETC-DATA>
	  <RESULT Code="0" Message="Saving successful" />"
	</SHEET>

Use the etc key in JSON format as follows:

Basic structure	**etc**:{ "Key name":"Key value", name: "Gildong Hong", age: 30}

Search	Sample)
	{
	    etc:{ name: " Gildong Hong ", age: 30},
	    data:[
	        {C1: "CWOF-171", C2: "17x3.3 square meters"},
	        ...
	    ]
	}

Save	Sample)
	{
	    etc:{ name: " Gildong Hong ", age: 30},
	    result:[
	        {Code:0, Message: "Saving successful"},
	        ...
	    ]
	}

- **Syntax**

Syntax	Get	ObjId.**GetEtcData**(KeyName)

- **Info**

Return	String, value set in the key (In case of Get Method)		
Parameter	Type	Required Y/N	Description
KeyName	String	Required	Etc data key name

- **Example**

```
// Set the etc data searched in XML to TextBox.

document.form1.txtAge.value = mySheet.GetEtcData("age")



// After completing saving, move page using Etc data.

mySheet.DoSave(“save.html”);

location.href = "/site/showmaster.html?keyinfo=" + mySheet.GetEtcData(“Slip
number”)
```

- **Syntax**

Syntax	Set	ObjId.**SetEtcData**(KeyName, Value)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
KeyName	String	Required	Etc data key name
Value	String	Required	Etc data key value

- **Example**

```
// Change the etc data value.

mySheet.SetEtcData("age", 40);



// Create new etc data.

mySheet.SetEtcData("Pay", 2000000);
```

ExtendLastCol Method

- **Purpose**

  Check or configure whether to automatically adjust width of the last column to fit the overall width setting.

  If the sum of width of all columns is smaller than the object width, you can extend the last column width to automatically to fit the object width using this property.

- **Syntax**

Syntax	Get	ObjId.**GetExtendLastCol**()

- **Info**

Return	Boolean, auto extension setting (in case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Check for auto extension setting.

mySheet.GetExtendLastCol()
```

- **Syntax**

Syntax	Set	ObjId.**SetExtendLastCol**(Extend)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Extend	Boolean	Required	Whether to extend last column width Default= 0

- **Example**

```
//Adjust the last column width to fit the overall object width

mySheet.SetExtendLastCol(1)
```

FindCheckedRow Method

- **Purpose**

Return row numbers checked for a specific column by connecting them with "|".

This method will apply to all rows if Col parameter is set as index and two or more rows are grouped as a unit data row. Apply only to rows with set Savename If Col is set as SaveName.

- **Syntax**

Syntax	ObjId.**FindCheckedRow**(Col)

- **Info**

Return	String, string of checked row numbers joined by "|" (Default="")		
Parameter	Type	Required Y/N	Description
Col	Long / String	Required	Column index of a particular column or SaveName

- **Example**

```
//Fetch checked row numbers.

//Parameter->1, Outcome->1|3|4|5|6|

var sRow = mySheet.FindCheckedRow(1);

var sRow = mySheet.FindCheckedRow("pass_yn");



//Create an array with the received outcome.

var arrRow = sRow.split("|");

for(idx=0; idx<arrRow.length-1; idx++){ alert(arrRow[idx]); }
```

FindStatusRow Method

- **Purpose**

Return row numbers that match a transaction status by joining them with ";".

Set the desired transaction status by connecting RIUD strings with "|". All rows with the transaction status will be identified and the row numbers will be joined with "|" and returned.

- **Syntax**

Syntax	ObjId.**FindStatusRow**(sStatus)

- **Info**

Return	String, string of row numbers with matching transaction status adjoined by ";" (Default="")		
Parameter	Type	Required Y/N	Description
sStatus	String	Required	String with transaction status code to locate adjoined by "|"

- **Example**

```
// Find all rows with Edited or Deleted status

//Parameter->U|D, Outcome->1;3;4;5;6;

var sRow = mySheet.FindStatusRow("U|D");



//Create an array with the received outcome.

var arrow = sRow.split(";");
```

FindSubSumRow Method

- **Purpose**

  Return a string of row numbers of sub sums marked using ShowSubSum by connecting them with "|". If StdCol parameter is not set, this method will find all sub sum rows. If the parameter is set, only the sub sum rows calculated in that column will be returned in string.

- **Syntax**

Syntax	ObjId.**FindSubSumRow**([StdCol])

- **Info**

Return	String, string of row numbers of sub sums adjoined by "|"		
Parameter	Type	Required Y/N	Description
StdCol	Long/String	Optional	Column index or SaveName of the reference column marking sub sums Default=""(All columns)

- **Example**

```
//Fetch row numbers of all marked sub sums.

var sRow = mySheet.FindSubSumRow();



//Fetch row numbers of marked sub sums using column 1 as reference.

var sRow = mySheet.FindSubSumRow(1);
```

FindSumRow Method

- **Purpose**

  Check the index of reference sum rows.

- **Syntax**

Syntax	ObjId.**FindSumRow**()

- **Info**

Return	Long, index of the sum rows (Default=-1)		
Parameter	Type	Required Y/N	Description
N/A			

- **Example**

```
//Fetch row numbers of sum rows.

var sumRow = mySheet.FindSumRow();
```

FindText Method

- **Purpose**

  Find specific text within a column and return row numbers.

  In default, you can find data row identical to search text from the start to the end, including case. Depending on the parameter setting, you can also find data row with the same first letter.

  If no identical string is found, -1 is returned.

- **Syntax**

Syntax	ObjId.**FindText**(Col,SearchText,[StartRow],[FullMatch],[CaseSensitive])

- **Info**

Return	Long, row numbers found Default=-1		
Parameter	Type	Required Y/N	Description
Col	Long / String	Required	Index of the column to search or SaveName
SearchText	String	Required	String to search
StartRow	Long	Optional	Index of the start row, Default="First row"
FullMatch	Integer	Optional	Matching characters, Default=-1
CaseSensitive	Boolean	Optional	Case sensitivity, Default=1

FullMatch parameter is an option for character search and works as follows:

FullMatch value	Purpose
-1	Find rows that are identical to SearchText
0	Find rows that have matching opening with SearchText
1	Find rows that have matching ending with SearchText
2	Find rows that have matching center with SearchText

- **Example**

```
// Find data rows that start with 'Korea' within two columns

var Row1 = mySheet.FindText(2, "Korea", 0, 0, 0);



// Find data rows that are 'Korea'

var Row1 = mySheet.FindText(2, "Korea", 0);



// Find data rows that end with "Bank"

var Row1 = mySheet.FindText(2, "Bank", 0, 1);



//Find rows with at least one "B"

var Row1 = mySheet.FindText(2, "B", 0, 2);



//Find without case sensitivity

var Row1 = mySheet.FindText(2, "Bank", 0, 2, 0);
```

FitColWidth Method

- **Purpose**

  Adjust width of all columns to fit the overall width so that no horizontal scroll is necessary.

  If the sheet width is 0 or there is no visible column, return -1 without any further processing.

To set a value to Width parameter, apply the relative value to overall width in percentage. Parameter values can be set by adjoining width of individual columns with "|", and column width will be readjusted by percentage according to the number of items in the parameter value.

If the number set in width parameter does not match the column count or the sum is bigger than 100 percent, horizontal scroll bar may appear.

If Width parameter value is "" or method is called without the parameter, column widths will be readjusted to fit the object overall width while maintaining the current relative column width percentage.

- **Syntax**

Syntax	ObjId.**FitColWidth**([Width])

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Width	String	Optional	Combination of individual column width expressed in % Default=""

- **Example**

```
// Adjust overall width while maintaining percentage of individual column width

mySheet.FitColWidth();



// Reconfigure in new %

mySheet.FitColWidth("10|20|40|30");
```

FitSize Method

- **Purpose**

Adjust height and width of all rows and columns.

If RowHeight parameter is set as 1, all row height is readjusted to fit the data height. If ColumnWidth parameter is set as 1, width of all columns will be readjusted to fit the widest text within the column.

- **Syntax**

Syntax	ObjId.**FitSize**(RowHeight, ColumnWidth)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
RowHeight	Boolean	Required	Whether to adjust row height
ColumnWidth	Boolean	Required	Whether to adjust column width

- **Example**

```
//Readjust row height only

mySheet.FitSize(1, 0);



//Readjust column width only

mySheet.FitSize(0, 1);



//Readjust both

mySheet.FitSize(1, 1);
```

FocusAfterProcess Method

- **Purpose**

Check or configure whether to put focus on data row after search.

In default, when data search using DoSearch function is complete, focus is put on the first data row. However, if this property is set false, focus will not be taken from the previous control even after search.

- **Syntax**

Syntax	Get	ObjId.**GetFocusAfterProcess**()

- **Info**

Return	Boolean, set value (In case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Focus location after search

mySheet.GetFocusAfterProcess();
```

- **Syntax**

Syntax	Set	ObjId.**SetFocusAfterProcess**(mode)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
mode	Boolean	Required	Whether to get focus (Default=1)

- **Example**

```
// Do not get focus after search

mySheet.SetFocusAfterProcess(0);
```

FocusAfterRowTransaction Method

- **Purpose**

  Check or configure whether to move focus after adding, deleting, moving or copying a row.

  When you are working on two or more rows, you can prevent unnecessary focus movement to improve performance.

- **Syntax**

Syntax	Get	ObjId.**GetFocusAfterRowTransaction**()

- **Info**

Return	Boolean, set value (In case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
// Check whether to move focus

mySheet.GetFocusAfterRowTransaction();
```

- **Syntax**

Syntax	Set	ObjId.**SetFocusAfterRowTransaction**(mode)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
mode	Boolean	Required	Whether to get focus (Default=1)

- **Example**

```
// Add 10 rows below the current focus row, and move focus to the last added row directly.

mySheet.SetFocusAfterRowTransaction(0);

var new Row = null;

for (var i = 0; I < 10; i++) {

   newRow = mySheet.DataInsert();

}

mySheet.SetSelectRow(newRow);
```

FocusEditMode Method

- **Purpose**

  When focus is set on an editable cell, check or configure whether to leave it as simple focus mode or change to edit mode.

  This property is used to set edit mode for editable cell or check such setting.

Value	Details
0	Leave as simple focus (Default)
1	Edit mode once focus is put
2	Leave column types of Combo and ComboEdit as simple focus Change all the others to edit mode

- **Syntax**

Syntax	Get	ObjId.**GetFocusEditMode**()

- **Info**

Return	Integer, set value (In case of Get Method)		
Parameter	Type	Required	Description

		Y/N	

- **Example**

```
//Check whether to make cell edit mode when focus is put

mySheet.GetFocusEditMode();
```

- **Syntax**

Syntax	Set	ObjId.**SetFocusEditMode**(Mode)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Mode	Integer	Required	Setting (Default=0)

- **Example**

```
// Maintain simple focus not edit mode when focus is put

mySheet.SetFocusEditMode(0);



// Use Edit mode once focus is set

mySheet.SetFocusEditMode(1);
```

FrozenRows Method

- **Purpose**

Check or configure whether to display frozen rows and other display options.

Display a set number of rows as frozen.

If you call this method before search, search results that are returned will be displayed as frozen rows.

Sum row and filter row cannot be frozen. When the rows set as frozen include sum or filter row, the frozen rows are created below such sum or filter rows.

Frozen rows do not support row selection using mouse or row movement by mouse dragging.

When you are using server page search, unit data row, tree sheet or sub sum, this function is not supported.

If you set all main section rows as frozen, or frozen rows are too many to display within the visible area, IBSheet may not work properly. (주의)

- **Syntax**

Syntax	Get	ObjId.**GetFrozenRows**()

- **Info**

Return	Integer, set value (In case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
// Check the frozen row setting.

alert(mySheet.GetFrozenRows());
```

- **Syntax**

Syntax	Set	ObjId.**SetFrozenRows**(Rows)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description

Rows	Integer	Required	Frozen row count to set (Default=0)

- **Example**

```
// Set 3 frozen rows.

mySheet.SetFrozenRows(3);
```

GetCellProperty Method

- **Purpose**

  Check property value set in InitColumns or InitCellProperty.

- **Syntax**

Syntax	ObjId.**GetCellProperty**(Row, Col, PropName)

- **Info**

Return	String/ Boolean/ Integer, property values of the set columns		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row index of the cells
Col	Long / String	Required	Column index or SaveName of a particular cell
PropName	String	Required	Property name to check

- **Example**

```
//Read data type

var iType = mySheet.GetCellProperty(1, 1, "Type");



//Read data SaveName
```

```
var sSaveName = mySheet.GetCellProperty(1, 1, "SaveName");
```

GetChildNodeCount Method

- **Purpose**

  In tree structure, check the node count below (child level) a particular row

- **Syntax**

Syntax	ObjId.**GetChildNodeCount**(Row)

- **Info**

Return	Integer, child node count		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row index of a particular row

- **Example**

```
//child node count to the first row

var childCount = mySheet.GetChildNodeCount(1);
```

GetChildRows Method

- **Purpose**

  In tree structure, combine and return index of child rows of a particular row by "|"

  When you set MaxLevel property, all child level rows down to that level will be returned. If the property is not set, child levels of all levels will be returned.

- **Syntax**

Syntax	ObjId.**GetChildRows**(Row, [MaxLevel])

➢ **Info**

Return	String, string of relevant child rows		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Index of relevant rows
MaxLevel	Integer	Optional	Maximum child row level to check (All level if -1), Default=-1

➢ **Example**

```
// Check all child rows of the second row

var childRows = mySheet.GetChildRows(2);



// Check child levels up to Level 3 of the second row

var childRows = mySheet.GetChildRows(2, 3);



// Check child levels of the second row from the bottom + 2 level

var myLevel = mySheet.GetRowLevel(2);

var childRows = mySheet.GetChildRows(2, myLevel+2);
```

GetComboInfo Method

➢ **Purpose**

Check for combo information of a particular cell. Flag parameter values are as follows:

Value	Description
"Text"	Combo text
"Code"	Combo code
"SelectedIndex"	Item index of the selected combo

- **Syntax**

Syntax	ObjId.**GetComboInfo**(Row,Col,Flag)

- **Info**

Return	String, combo text or code		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row index of the cells
Col	Long / String	Required	Column index or SaveName of a particular cell
Flag	String	Required	Select "Text" or "Code"

- **Example**

```
//Fetch combo code and text.
var sText = mySheet.GetComboInfo(0,2, "Text");
var sCode = mySheet.GetComboInfo(0,2, "Code");


//Create each into an array.
var arrText = sText.split("|");
var arrCode = sCode.split("|");


//Using combo code of Row 2, Column 2, fetch combo text.
for(i=0; i<arrCode.length; i++) {
    if(mySheet.GetCellValue(2,2) == arrCode[i]) {
        alert(arrText[i]);
        break;
    }

}
```

GetCurrentPage Method

- **Purpose**

  If search method is not smGeneral, return the current page number out of all data

- **Syntax**

Syntax	ObjId.**GetCurrentPage**()

- **Info**

Return	Integer, current page number.

- **Example**

```
// Current page number.

var pageNum = mySheet.GetCurrentPage();
```

GetDataRows Method

- **Purpose**

  Check configured unit data row count

- **Syntax**

Syntax	ObjId.**GetDataRows**()

- **Info**

Return	Integer, configured unit data row count		
Parameter	Type	Required Y/N	Description
N/A			

- **Example**

```
// Check for unit data row count.

var dataRows = mySheet.GetDataRows();.
```

GetEditText Method

- **Purpose**

  Check the character being edited.

- **Syntax**

Syntax	ObjId.**GetEditText**()

- **Info**

Return	String, character being edited (Default="")		
Parameter	Type	Required Y/N	Description
N/A			

- **Example**

```
// Check character being edited.

function mySheet_OnKeyUp(Row, Col, KeyCode, Shift){

  var editTxt= "character being edited = " + mySheet.GetEditText();

  editTxt += "Original character = " + mySheet.GetCellValue(Row,Col);

  alert(editTxt);

}
```

GetFilterParam Method

- **Purpose**

  SearchMode:3 fetches partial data from the database to run a search, so filtering cannot be used for this. In this case, filter value of the filtering rows and filtering options should be sent to the server to use for DB search. This function is to create QueryString to get the same filtered data on the Sheet. If AllFilter is set as 0, all columns are subject to this method. If 1, filtering columns only. Even if filter cell has a value, if the option value is 0(do not use) it will be ignored.

QueryString is composed in the format of SaveName=CellValue&SaveName_opt=OptionValue for each column, and columns are connected using "&".

- **Syntax**

Syntax	ObjId.**GetFilterParam**([AllFilter], [UrlEncode])

- **Info**

Return	String, search parameter Query String		
Parameter	Type	Required Y/N	Description
AllFilter	Boolean	Optional	Filter all or not, Default=0
UrlEncode	Boolean	Optional	UrlEncode or not, Default=1

- **Option**

0	Do not use	**1**	Equal
2	Not equal	**3**	Smaller
4	Equal or smaller	**5**	Bigger
6	Equal or bigger	**7**	Start with word
8	Not start with word	**9**	End with word
10	Not end with word	**11**	Include
12	Not include		

- **Example**

```
// Return all columns of filtered rows in Param-format string.

var FilterStr = mySheet.GetFilterParam(1);



// Return filtered columns in Param-format string.

var FilterStr = mySheet.GetFilterParam(0);
```

GetFirstChildRow Method

- **Purpose**

  In tree structure, check the index of the first child row of a particular row.

  If there is no child row, return -1.

- **Syntax**

Syntax	ObjId.**GetFirstChildRow**(Row)

- **Info**

Return	Long, first child row index		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Index of relevant rows

- **Example**

```
// Check the first child row of the second row.

var firstChild = mySheet.GetFirstChildRow(2);
```

GetGroupCol Method

- **Purpose**

  Check SaveName of group reference column currently set

  If there are two or more columns, a string will be returned with "|" as separator.

- **Syntax**

Syntax	ObjId.**GetGroupCol**()

- **Info**

Return	Stirng, currently set group reference column information		
Parameter	Type	Required Y/N	Description
N/A			

- **Example**

```
// Create group rows

mySheet.ShowGroupRow();



// Check currently set group reference column

var Cols = mySheet.GetGroupCol();
```

GetLastChildRow Method

- **Purpose**

In tree structure, check the index of the last child row of a particular row.

If there is no child row, return -1.

- **Syntax**

Syntax	ObjId.**GetLastChildRow**(Row)

- **Info**

Return	Long, last child row index		
Parameter	Type	Required	Description

		Y/N	
Row	Long	Required	Index of relevant rows

- **Example**

```
// Check the last child row of the second row.

var lastChild = mySheet.GetLastChildRow(2);
```

GetMergedEndCell Method

- **Purpose**

  Among all cells merged together, row and col information of the last cell is returned in Row, Col format string.

- **Syntax**

Syntax	ObjId.**GetMergedEndCell**(Row, Col)

- **Info**

Return	String, Row and Col information is returned in string of "Row, Col"		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Index of relevant rows
Col	Long / String	Required	Target column status or SaveName

- **Example**

```
var endMergeCell = mySheet.GetMergedEndCell(4,5);
```

GetMergedStartCell Method

- **Purpose**

  Among all cells merged together, row and col information of the starting cell is returned in

Row,Col format string.

- **Syntax**

Syntax	ObjId.**GetMergedStartCell**(Row, Col)

- **Info**

Return	String, Row and Col information is returned in string of "Row, Col"		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Index of relevant rows
Col	Long / String	Required	Target column status or SaveName

- **Example**

```
var startMergeCell = mySheet.GetMergedStartCell(4,5);
```

GetNextSiblingRow Method

- **Purpose**

  In tree structure, check the next row index of same parent and same level for a particular row.

  If there is no next row, return -1.

- **Syntax**

Syntax	ObjId.**GetNextSiblingRow**(Row)

- **Info**

Return	Long, index of the next same level row		
Parameter	Type	Required Y/N	Description

Row	Long	Required	Index of relevant rows

- **Example**

```
// Check the next same level row of the 6th row.

var next = mySheet.GetNextSiblingRow(6);
```

GetSheetPageLength Method

- **Purpose**

  Check for Page (number of rows to display in one page) property value set in SetConfig method.

- **Syntax**

Syntax	ObjId.**GetSheetPageLength**()

- **Info**

Return	Integer, Page property value set in SetConfig method.		
Parameter	Type	Required Y/N	Description

- **Example**

```
// Check Page property value.

mySheet.GetSheetPageLength();
```

GetParentRow Method

- **Purpose**

  Check index of parent row of a particular row.

  If there is no parent, return -1.

- **Syntax**

Syntax	ObjId.**GetParentRow**(Row)

- **Info**

Return	Long, index of the parent row		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Index of relevant rows

- **Example**

```
// Check parent row of the 5th row.

var parent = mySheet.GetParentRow(5);
```

GetPrevSiblingRow Method

- **Purpose**

  Check index of the previous row of the same parent and same level for a particular row.

  If there is no such previous row, return -1.

- **Syntax**

Syntax	ObjId.**GetPrevSiblingRow**(Row)

- **Info**

Return	Long, index of the previous same level row		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Index of relevant rows

- **Example**

```
// Check the previous same level row of the 6th row.
```

```
var previous = mySheet.GetPrevSiblingRow(6);
```

GetSaveData Method

- **Purpose**

  Call save page to complete saving, and return the outcome in string.

  This function is different from DoAllSave or DoSave in that it saves the saving target QueryString as parameter, and return data itself without processing the saving result. The saved data returned using this function can be represented in IBSheet if you use the data as a parameter for LoadSaveData function.

  If you need encoding for SaveString, you need to do it before calling the function.

- **Syntax**

Syntax	ObjId.**GetSaveData**(PageUrl, [SaveString], Param)

- **Info**

Return	String, String of saved data		
Parameter	Type	Required Y/N	Description
PageUrl	String	Required	URL of the page to save
SaveString	String	Optional	Query String to save, Default=""
Param	String	Optional	Saving configuration Query String, Default=""

- **Example**

```
//Fetch save string – when you want to save only those with transactions

var SaveStr = mySheet.GetSaveString();



//Validations for required fields and others will be done.

if (mySheet.IsDataModified && SaveStr == "") return;
```

```
// Read saving outcome

var rtnData = mySheet.GetSaveData("save.jsp", SaveStr);
//Load returned result to my sheet

mySheet.LoadSaveData(rtnData);
```

GetSaveJson Method

- **Purpose**

  Return saving target data as JSON object.

  Calling this function will create a saving object. After checking for required items, OnValidation event fires to process user validation logics.

  Col parameter is used when AllSave parameter is set as 0 to select saving target columns. If the data structure is a unit data row structure of two or more rows, setting index to Col parameter will result in creating individual rows into JSON object. If you set SaveName instead of index, each unit data rows will be created into a JSON object.

  If Validation check fails, result code and message will be returned.

  1. If target rows do not exist:

     Code : "IBS000", Message : "NoTargetRows"

  2. If a required item is missing:

     Code : "IBS010", Message : "KeyFieldError"

  3. If validation fails:

     Code : "IBS020", Message : "InvalidInputError"

- **Syntax**

Syntax	ObjId.**GetSaveJson**([AllSave], [Col])

- **Info**

Return	Object, JSON object of saving data		
Parameter	Type	Required Y/N	Description
AllSave	Boolean	Optional	Whether to save all, Default=0
Col	Long / String	Optional	Target column status or SaveName Default=Status column

- **Example**

```
//Receve data of all rows as object
var SaveJson = mySheet.GetSaveJson(1);


//Receive saving target data as object (rows with transactions)
var SaveJson = mySheet.GetSaveJson(0);


//Receive data of rows with checked 4th column as object
var SaveJson = mySheet.GetSaveJson(0, 4);
```

GetSaveString Method

- **Purpose**

  Return a string of data Query String used for saving

  This function can be used when security module is used for saving. It returns strings to save so that a security module can encrypt them before saving.

  Calling this method will create save string. After checking required items, OnValidation event fires to process user validation logics.

  When AllSave parameter is set as 1, the same saving string will be returned as DoAllSave method. If it is set 0, the same saving string will be returned as DoSave method.

If UrlEncode is set as 1, all Korean characters in Query String will be encoded.

Col parameter is used when AllSave parameter is set as 0 to select saving target columns.

You can set how to combine Query String using Mode parameter.

Query Strings depending on set values are as follows:

If Validation check fails, KeyFieldError string will be returned in default. If SYS_KeyFieldError in ibmsg has a value set, the value will be returned.

Mode	Description	Ex)
1	Combine by cell (Array by SaveName)	sSeq=1&sStatus=R..
2	Combine by column (Combine using separator by SaveName)	sSeq=1|2&sStatus=R|U..

"Selection" parameter can be set in JSON format. (See the example)

- **Syntax**

Syntax	ObjId.**GetSaveString**([AllSave], [UrlEncode], [Col], [Prefix] ,[Mode], [Delim])

- **Info**

Return	String, Query String to save		
Parameter	Type	Required Y/N	Description

AllSave	Boolean	Optional	Whether to save all, Default=0
UrlEncode	Boolean	Optional	UrlEncode or not, Default=1
Col	Long / String	Optional	Target column status or SaveName Default=Status column
Prefix	String	Optional	String to put before SaveName when saving, Default=""
Mode	Integer	Optional	Set how to combine string in QueryString, Mode=1, Mode=2 (Default=1)
Delim	String	Optional	When Mode=2, set the separator (Default ="|")

➢ **Example**

```
//Fetch the same save string as the one returned by DoAllSave method

var SaveStr = mySheet.GetSaveString(1);



//Fetch the same save string as the one returned by DoSave method

//– Save only rows with transactions

var SaveStr = mySheet.GetSaveString(0);



//Fetch the same save string as the one returned by DoSave method

//– Save only rows with Column 4 checked

var SaveStr = mySheet.GetSaveString(0, 1, 4);



//Fetch the same save string as the one returned by DoSave method

// Return value will be: pre_AA=1&pre_BB=2&pre_CC=3.

var SaveStr = mySheet.GetSaveString(0, 1, 1, “pre_”);
```

```
//Set property of "Selection" parameter in JSON format

mySheet.GetSaveString({AllSave : 1, UrlEncode:0, Mode:2, Delim:"$"})
```

see also

GetSaveJson Method

GetSearchData Method

- **Purpose**
  Call search page, complete search and return search result data in string. Unlike DoSearch, this method returns search result data itself without processing search result. Search result data returned by this method can be loaded to IBSheet if you use them as LoadSearchData parameter.

- **Syntax**

Syntax	ObjId.**GetSearchData**(PageUrl, [Param])

- **Info**

Return	String, string of search data		
Parameter	Type	Required Y/N	Description
PageUrl	String	Required	Page URL to search
Param	String	Optional	Search parameter Query String, Default=""

- **Example**

```
//Read search data

var sXml = mySheet.GetSearchData("list.jsp");
```

```
//Load search data

mySheet.LoadSearchData(sXml);
```

GetSelectionCols Method

- **Purpose**

  Return column numbers in Selection status by separating each with separator.

  If you do not set a separator, default "|" will be used.

- **Syntax**

Syntax	ObjId.**GetSelectionCols**([DeliChar])

- **Info**

Return	String, selected row number combination string		
Parameter	Type	Required Y/N	Description
DeliChar	String	Optional	Separator, Default="|"

- **Example**

```
Fetch selected row numbers using //"/" as separator.

var sColStr = mySheet.GetSelectionCols("/");



//Create Java script array.

var arr = sColStr.split("/");

for (i=0; i<arr.length(); i++) {

    alert(arr[i] + "Column have been selected");

}
```

GetSelectionRows Method

- **Purpose**

  Return selected row numbers separated by a set separator.

  If you do not set a separator, default "|" will be used.

- **Syntax**

Syntax	ObjId.**GetSelectionRows**([DeliChar])

- **Info**

Return	String, selected row number combination string		
Parameter	Type	Required Y/N	Description
DeliChar	String	Optional	Separator, Default="|"

- **Example**

```
Fetch selected row numbers using //"/" as separator.

var sRowStr = mySheet.GetSelectionRows("/");



//Create Java script array.

var arr = sRowStr.split("/");

for (i=0; i<arr.length(); i++) {

      alert(arr[i] + " rows have been selected");

}
```

GetSheetHtml Method

- **Purpose**

  Return Html source of the current sheet.

  Return two objects, one style with sheet style string and body with html source string.

* As this method returns html codes of the sheet as it shows in the current screen, hiding and column size will be maintained. Anything not loaded on the screen will not be included.

- **Syntax**

Syntax	ObjId.GetSheetHtml()

- **Info**

Return	Object, (object made of style and body)		
Parameter	Type	Required Y/N	Description
N/A			

- **Example**

```
// Fetch sheet Html

var code = mySheet.GetSheetHtml();
document.getElementById("styleText").value = code.style;
document.getElementById("htmlText").value += code.body;
```

GoToFirstPage Method

- **Purpose**

Move to the first page if paging mode is set.

To use this method, you need to set SearchMode of SetConfig method as paging mode.

- **Syntax**

Syntax	ObjId.**GoToFirstPage**()

- **Info**

Return	None.

Parameter	Type	Required Y/N	Description
N/A			

- **Example**

```
// Set paging mode

var cfg = {SearchMode:1, Page:10};

mySheet.SetConfig(cfg);



// Move to the first page

mySheet.GoToFirstPage();
```

GoToLastPage Method

- **Purpose**

  In case paging mode is set, move to the last page.

  To use this method, you need to set SearchMode of SetConfig method as paging mode.

- **Syntax**

Syntax	ObjId.**GoToLastPage**()

- **Info**

Return	Boolean		
Parameter	Type	Required Y/N	Description
N/A			

- **Example**

```
// Set paging mode
```

```
var cfg = {SearchMode:1, Page:10};

mySheet.SetConfig(cfg);



// Move to the last page

mySheet.GoToLastPage();
```

GoToNextPage Method

- **Purpose**

  Move to the next page if paging mode is set.

  To use this method, you need to set SearchMode of SetConfig method as paging mode.

- **Syntax**

Syntax	ObjId.**GoToNextPage**()

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
N/A			

- **Example**

```
// Set paging mode

var cfg = {SearchMode:1, Page:10};

mySheet.SetConfig(cfg);



// Move to the next page

mySheet.GoToNextPage();
```

GoToPageNum Method

- **Purpose**

  In case paging mode is set, move to the page set in a parameter.

  To use this method, you need to set SearchMode of SetConfig method as paging mode.

- **Syntax**

Syntax	ObjId.**GoToPageNum**(index)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
index	Integer	Required	Index of the page to move to

- **Example**

```
// Set paging mode

var cfg = {SearchMode:1, Page:10};

mySheet.SetConfig(cfg);



// Move to the 5th page

mySheet.GoToPageNum(5);
```

GoToPrevPage Method

- **Purpose**

  Move to the previous page if paging mode is set.

  To use this method, you need to set SearchMode of SetConfig method as paging mode.

- **Syntax**

Syntax	ObjId.**GoToPrevPage**()

➢ **Info**

Return	None.		
Parameter	Type	Required Y/N	Description
N/A			

➢ **Example**

```
// Set paging mode

var cfg = {SearchMode:1, Page:10};

mySheet.SetConfig(cfg);



// Move to the previous page

mySheet.GoToPrevPage();
```

HeaderActionMenu Method

➢ **Purpose**

Check or configure header context menu.

This method can be used only when you set UserHeaderActionMenu as 1 in SetConfig or ibsheet.cfg.

If you set UseHeaderActionMenu but not SetHeaderActionMenu, default menu will be displayed.

Put MenuText and MenuCode into string using "|" as a separator.

If you configure an OOTB frozen code as MenuCode, the function will be processed within IBSheet.

If MenuText is not set, default menu will be used.

- **Syntax**

Syntax	Get	ObjId.**GetHeaderActionMenu**()

- **Info**

Return	Object, Object composed of MenuText and MenuCode		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Check the current header menu configuration

var menu = mySheet.GetHeaderActionMenu();

alert(menu.MenuText); // Check the current MenuText configuration

alert(menu.MenuCode); // Check the current MenuCode configuration
```

- **Syntax**

Syntax	Set	ObjId.**SetHeaderActionMenu**(MenuText, MenuCode)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
MenuText	String	Optional	Menu name string adjoined by "|"
MenuCode	String	Optional	MenuCode string adjoined by "|" If you set OOTB frozen code, the function will be processed within IBSheet.

* Frozen code

Code	Description
_ibColHidden	Hide target column
_ibCancelColHidden	Display hidden column in header menu
_ibSaveColPosition	Save current column location (Location, column hiding status, width)
_ibResetColPosition	Delete saved column info
_ibShowFilter	Display filtered rows
_ibHIdeFilter	Hide filtered rows

➢ **Example**

```
// Set header menu so that filters display when aaa menu is clicked and filters are hidden when bbb menu is clicked.

mySheet.SetHeaderActionMenu("aaa|*-|bbb", "_ibShowFilter||_ibHideFilter");
```

HeaderBackColor Method

➢ **Purpose**

Check or configure background color of the header row.

Select the color using WebColor.

➢ **Syntax**

Syntax	Get	ObjId.**GetHeaderBackColor**()

➢ **Info**

Return	String, current background color value (in case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
// Check background color of the header

mySheet.GetHeaderBackColor();
```

- **Syntax**

Syntax	Set	ObjId.**SetHeaderBackColor**(Color)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Color	String	Required	Background color of the header row

- **Example**

```
//Set the header background color as red.

mySheet.SetHeaderBackColor("#FF0000");
```

HeaderCheck Method

- **Purpose**

  Check and configure Checkbox values of the header.

  You can configure simply either Check or Uncheck unlike mouse click.

- **Syntax**

Syntax	Get	ObjId.**GetHeaderCheck**(Row, Col)

- **Info**

Return	Boolean,current set value (in case of Get Method)		
Parameter	Type	Required	Description

		Y/N	
Row	Long	Required	Index of relevant rows
Col	Long / String	Required	Target column status or SaveName

- **Example**

```
// Check the Checkbox value of the 3rd column in the header row

var CheckValue = mySheet.GetHeaderCheck(0, 3);
```

- **Syntax**

Syntax	Set	ObjId.**SetHeaderCheck**(Row, Col, Value)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Index of relevant rows
Col	Long / String	Required	Target column status or SaveName
Value	Boolean	Required	Value to set to the checkbox

- **Example**

```
// Uncheck all the checkbox values in the thid column of the head row

mySheet.SetHeaderCheck(0, 3, 0);
```

HeaderFontBold Method

- **Purpose**

  Check or configure font bold status of the header row. In default, header font s not bold; you can use this method to bolden the header font.

- **Syntax**

Syntax	Get	ObjId.**GetHeaderFontBold**()

- **Info**

Return	Boolean,current set value (in case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
// Check the bold status of header font.

mySheet.GetHeaderFontBold();
```

- **Syntax**

Syntax	Set	ObjId.**SetHeaderFontBold**(Bold)

- **Info**

Return	Boolean,current set value (in case of Get Method)		
Parameter	Type	Required Y/N	Description
Bold	Boolean	Required	Font bold status of the header row

- **Example**

```
//Bolden the header font.

mySheet.SetHeaderFontBold(1);
```

HeaderFontColor Method

- **Purpose**

Check or configure font color of the header row.

- **Syntax**

Syntax	Get	ObjId.**GetHeaderFontColor**()

- **Info**

Return	String, currently set value (In case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Set the header row font color as black.

mySheet.GetHeaderFontColor();
```

- **Syntax**

Syntax	Set	ObjId.**SetHeaderFontColor**(Color)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Color	String	Required	WebColor to set

- **Example**

```
//Set the header row font color as black.

mySheet.SetHeaderFontColor("#000000");
```

HeaderRows Method

- **Purpose**

  Check header row count.

  This method returns the header row counts as set in InitHeaders() method.

- **Syntax**

Syntax	ObjId.**HeaderRows**()

- **Info**

Type	Long, header row count	
Parameter	Type	Description
N/A		

- **Example**

```
//Check header row count.

alert("header row count is " + mySheet.HeaderRows() + ".");
```

HeaderRowHeight Method

- **Purpose**

  Check or configure row height of the header rows. Set the value in pixel count.

- **Syntax**

Syntax	Get	ObjId.**GetHeaderRowHeight**()

- **Info**

Return	Integer, currently set value (In case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Check header row height.

mySheet.GetHeaderRowHeight();
```

- **Syntax**

Syntax	Set	ObjId.**SetHeaderRowHeight**(Height)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Height	Integer	Required	Pixel count to set

- **Example**

```
// Set the header row height at 30 pixels

mySheet.SetHeaderRowHeight(30);
```

HideFilterRow Method

- **Purpose**

  Delete the filter row from the frozen rows at the top of IBSheet

  If there are any filter row, it will be deleted. If there are none, nothing will happen.

- **Syntax**

Syntax	ObjId.**HideFilterRow**()

- **Info**

Return	None

Parameter	Type	Required Y/N	Description
N/A			

- **Example**

```
//Deleting filter row

mySheet.HideFilterRow();
```

HideProcessDlg Method

- **Purpose**

  Close waiting image from sheet

- **Syntax**

Syntax	ObjId.**HideProcessDlg**()

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
N/A			

- **Example**

```
// Close waiting image

mySheet.HideProcessDlg();
```

HideSubSum Method

- **Purpose**

Calling ShowSubSum method will insert sub sum rows to search data from the next search. Using this method will stop addition of subsum started by ShowSubSumto the next search.

This method will not operate immediately upon call, but work in the next search.

- **Syntax**

Syntax	ObjId.**HideSubSum**([StdCol])

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
StdCol	Long/String	Optional	Column index or SaveName of the reference column marking sub sums Default=""(All columns)

- **Example**

```
//Display sub sums for Column 1

var info = [{StdCol:1, SumCols:"3|4|5"}];

mySheet.ShowSubSum(info);



//Hide showing sub sums

mySheet.HideSubSum();
```

HighlightAfterSort Method

- **Purpose**

Check or configure highlighting after sorting.

If this property value is 0, focus will be reset after sorting. If it is 1, focus will move to the row originally highlighted after sorting.

- **Syntax**

Syntax	Get	ObjId.**GetHighlightAfterSort**()

- **Info**

Return	Integer, currently set value (In case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Check the configuration post sorting

mySheet.GetHighlightAfterSort();
```

- **Syntax**

Syntax	Set	ObjId.**SetHighlightAfterSort**(Sort)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Sort	Integer	Required	Sort setting (Default=1)

- **Example**

```
//After sorting, go to the data previously selected.

mySheet.SetHighlightAfterSort(1);
```

IBCloseCalendar Method

- **Purpose**

  Set to close calendar pop-up used by external control.

- **Syntax**

Syntax	ObjId.**IBCloseCalendar**()

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
N/A			

```
// Open calendar pop-up.
mySheet.IBShowCalendar("20121116", {Format:"yyyy/MM/dd", X:300, Y:600});


// Close calendar pop-up dialog.
mySheet.IBCloseCalendar();
```

IBShowCalendar Method

- **Purpose**

  Set to allow use of calendar pop-up by external control.

  When a CallBack method is configured, the first parameter is usually the selected date string. If there are other parameters to send by CallBack method, set it as CallBackParam. Object set in CallBackParam will be sent as the second parameter of CallBack method.

- **Syntax**

Syntax	ObjId.**IBShowCalendar**(val, [obj]);

- **Info**

Return	String, (Date value set in calendar pop-up dialog)		
Parameter	Type	Required Y/N	Description
val	String	Required	Date data value (Defaul=today's date)
obj	Object	Optional	Configure function into JSON format.

Details

Properties configurable by column are as follows:

Property	Type	Description
Format	String	Date format (Default="Ymd")
Target	String, Object	"Mouse" (When last mouse location is used, Default) or calendar button Object (when calendar button location is used)
Result	Object	Object where to include selected values (Input object)
X	Integer	(When coordinate values are used), X coordinate
Y	Integer	(When coordinate values are used), Y coordinate
CallBack	String	Function Name Method name to use upon return
CallBackParam	Object	Parameter object to receive from CallBack method
CalButtons	String	Configure the button to use for calendar popup

		Adjoin buttons to use with "|".	
		Value	Details
		Close	Cancel button
		Today	Enter today's date button
		Yesterday	Enter yesterday's date button
CalButtonAlign	String	Configure how to align buttons to use in calendar pop-up	
		Value	Details
		Left	Align left
		Center	Align center (Default)
		Right	Align right

- **Example**

```
// Enter date data value

var val = document.getElementById("DateText").value;



// Calendar pop-up dialog location: When X and Y coordinates are used

1.var obj = { Format: "Ymd", X:300, Y:600, CallBack: "Test" };



// Calendar pop-up dialog location: When last mouse location is used

2.var obj = { Format: "Ymd", Target:"Mouse", CallBack: "Test" };



// Calendar pop-up dialog location: When calendar button location is used

3.var obj = { Format:"Ymd", Target:document.getElementById("DateBtn"),
CallBack: "Test" };
```

```
// CalButtons property: when closing calendar button option setting is used

3.var obj = { Format:"Ymd", Target:document.getElementById("DateBtn"),
CallBack: "Test" , CalButtons : "Close"};



// Alignment setting when closing calendar button option setting is used:
Align left

4.var obj = { Format:"Ymd", Target:document.getElementById("DateBtn"),
CallBack: "Test", CalButtons : "Close", CalButtonAlign : "Left" };



// fnName: create a function using the method name

function Test (date){

// Return value when a date is selected from calendar pop-up dialogue

   document.getElementById("DateText").value = date;

}
```

ImageList Method

- **Purpose**

  Check or configure web server path of Nth image file

  This property is used to remember image file by configuring image path displayed in a cell with Image type so that it can be downloaded and reused later once.

  Can be used for search and new row entry.

  Image file path may be a relative path to the current page, or an absolute path below web server root.

  If you set index, start with 0 and set as many index as you want.

  [Instructions]

  mySheet.SetImageList(0, "../image/btn_search.gif");

mySheet.SetImageList(1,"/common/image/btn_cal.gif");

[List of methods used to change image-related values]

Image setting	Purpose	Image
ImageList	Change image	SetCellValue,SetCellImage SetSearchingImage,SetSavingImage, SetWaitImage,SetKeyFieldImage
	Change value	N/A

- **Syntax**

Syntax	Get	ObjId.**GetImageList**(Index)

- **Info**

Return	String, image path configured in the index (in case of Get Method)		
Parameter	Type	Required Y/N	Description
Index	Integer	Required	Image index

- **Example**

```
Check for image path configured in the index

mySheet.GetImageList(0);
```

- **Syntax**

Syntax	Set	ObjId.**SetImageList**(Index, Url)

- **Info**

Return	None (in case of Get Method)		
Parameter	Type	Required Y/N	Description
Index	Integer	Required	Image index
Url	String	Required	Image path to set

- **Example**

```
// Set image path in 0 index

mySheet.SetImageList(0, "../image/btn_search.gif”);



// Use the image saved in 0   index (Type:Image)

mySheet.SetCellValue(3, 5, 0);

mySheet.SetCellImage(3, 5, 0);

mySheet.SetSearchingImage(0);

mySheet.SetSavingImage(0);

mySheet.SetWaitImage(0);

mySheet.SetKeyFieldImage(0);
```

InitCellProperty Method

- **Purpose**

  After search or addition of a new row, use this method to define properties of a particular cell from default column property sets.

  (Date formats available are Ymd, Ym, Md, YmdHms and YmdHm.)

  If the column type defined in InitColumns method is Seq, Status, DelCheck, AutoSum or AutoAvg, you cannot use this method.Available types are as follows:

Type	Description

Text	Basic string data
CheckBox	Checkbox type data
Combo	Uneditable combo data
ComboEdit	Auto complete combo data
Image	Uneditable simple image representation data
Int	Integer type
Float	Float type
Date	Date type
Popup	Data using popup
PopupEdit	Editable data which uses pop-up

Properties configurable in this method are as follows:

Property	Type	Description
Type	String	Column data type (Required)
AcceptKeys	String	Accepted key configuraiton
Align	String	Data alignment
ApproximateType	Integer	Approximation type (1: Rounding, 2: Rounding down, 3: Rounding up)
ComboCode	String	Combo list code group
ComboText	String	Combo list text string group
Cursor	String	Mouse pointer style setting
CustomDate	Integer	Custom date availability
Edit	Boolean	Editability
EditLen	Integer	Editable data legnth
Format	String	Data mask application format

FormatFix	Boolean	Whether to return GetCellText values during GetCellValue (Default=0); if true, the cell value will be saved with format.
HoverUnderline	Boolean	Whether to underline when mouse is hovering over
Image	String	Image path
ImgAlign	String	Image display location
ImgHeight	Integer	Image height
ImgWidth	Integer	Image width
InputCaseSensitive	Number	Whether to use particular case automatically for alphabet inputs 0: No configuration (Default) 1: Replace with upper case 2: Replace with lower case
MultiLineText	Boolean	Whether to use multilines
PointCount	Integer	When column type is float, number of decimal places.

For detailed description of each property, see InitColumns Method desription.

- **Syntax**

Syntax	ObjId.**InitCellProperty**(Row, Col, info);

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Index of relevant rows
Col	Long /	Required	Index of the target column

	String		or SaveName
info	Object	Required	Cell property definition object

- **Example**

```
// Change to text column type

var info = {Type: "Text", Align: "Center", Edit: 0};

mySheet.InitCellProperty(2, "sCombo", info);
```

InitColumns Method

- **Purpose**

  Configure data type, format and functionality of each column.

- **Syntax**

Syntax	ObjId.**InitColumns**(cols);

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
cols	json	Required	Configure functionality of each column into JSON format.

- **Details**

  Properties configurable for each column are as follows:

  ※ To see which properties are configurable by column, see Appendix#1. Properties definable by column type.

Property	Type	Description

Type	String	Column data type (Required)
AcceptKeys	String	Accepted key configuraiton
Align	String	Data alignment
AllowNull	Boolean	Whether to allow empty value for columns of number-like status
ApproximateType	Number	Approximation type
BackColor	String	Background color
ButtonUrl	String/Number	Popup button's image path or image list index
CalcLogic	String	Calculation equation by column
CaseSensitive	Boolean	Case sensitivity in filtering (Default=1)
ColMerge	Boolean	Whether to allow column merging
ColSpan	Number	ColSpan scope (Can be used only for unit data row structure)
ComboCode	String	Combo list code group
ComboText	String	Combo list text string group
Cursor	String	Mouse pointer style setting
CustomDate	Boolean	Custom date availability
DefaultValue	String	Default applies for new entries
Edit	Boolean	Editability
EditLen	Number	Editable data legnth
Ellipsis	Boolean	Use of Ellipsis
ExcludeEmpty	Boolean	In case AutoAvg type
FalseValue	String	Configure False value other than 1 for CheckBox columns. If set "F", you may use F instead of 1 for a false value.

FontColor	String	Font color
Format	String	Data mask application format
FormatFix	Boolean	Whether to return GetCellText value upon GetCellValue (Default=0) If set as true, the format will be also saved upon saving.
FullInput	Boolean	When the column type is single row string, whether to input data to fill the full legnth (EditLen)
HeaderCheck	Boolean	Whether the CheckAll in the header is checked (Default=1)
Hidden	Boolean	Whether a column is hidden
HoverUnderline	Boolean	Whether to underline when mouse is hovering over
Image	String	Image representation URL
ImgAlign	String	Image alignment
ImgHeight	Number	Image height
ImgWidth	Number	Image width
InputCaseSensitive	Number	Whether to use particular case automatically for alphabet inputs 0: No configuration (Default) 1: Replace with upper case 2: Replace with lower case
InsertEdit	Boolean	Whether to allow data editing when transaction is in "Insert" state
KeyField	Boolean	Required fields
LevelSaveName	String	A parameter name used to save or search data of a particular tree level

MaximumValue	Number	Max value allowed for number format
MinimumValue	Number	Minimum value allowed for number format
MinWidth	Number	Minimum column width
MultiLineText	Boolean	Whether to use multilines
PointCount	Number	When column type is float, number of decimal places.
PopupCheckEdit	Boolean	Whether to allow editing when popup menu is configured
PopupCode	String	Pop-up menu code compilation
PopupText	String	Popup menu string compilation
RadioIcon	Boolean	Select a checkbox-type button image within data cell
RowSpan	Number	RowSpan (Can be used only for unit data row structure)
SaveName	String	A parameter name used to save or search data
ShowCal	Number	Set the column to display as column value when multi combo is set
Sort	Boolean	Whether to allow sorting by clicking on the header (Default=1)
ToolTip	Boolean	Whether to display tooltip in cell
ToolTipText	String	String text to set for tooltip in the header row
TreeCol	Boolean	Tree reference column
TrueValue	String	Set true value for checkbox colums whose true value is not 1. If set "M", "M" can be used as True value instead

		of 1.
UpdateEdit	Boolean	Whether to allow data editing when transaction is in "Search" state
VAlign	String	Column vertical alignment (Default="middle")
Width	Number	Column width
Wrap	Boolean	Auto multiline or not

You can set **types** of a given column as follows:

Type	Description
Text	Basic string data
Status	Data that display and contain transaction status
DelCheck	Checkbox type data that only checks deletion
CheckBox	Checkbox type data
Radio	One of many data rows is checked
Combo	Uneditable combo data
ComboEdit	Auto complete type combo data (Not supported for mobile devices)
AutoSum	Data for auto summing; default format is NullFloat
AutoAvg	Data for auto averaging; default format is NullFloat
Image	Uneditable simple image representation data
Int	Integer type
Float	Float type
Date	Date type
Popup	Data using popup
PopupEdit	Editable data which uses pop-up

Pass	Password-type data
Seq	Irreversibly increasing value like DB sequence; uneditable data
Html	Html tag type data
Result	Saving result display data

※ In mobile devices, ComboEdit type cells will display in Combo type instead.

Align is Left, Center or Right alignment setting of cell data.

DataAlign	Description
Left	Left alignment (Default)
Center	Center alignment
Right	Right alignment

AcceptKeys is where you configure key operations allowed for a column.

Set as "N" for numbers and "E" for english alphabet; if both numbers and English alphabets are allowed, connect the two values with "|".

BackColor is the background color of a column.

ButtonUrl property may be used when you wish to change the pop-up button image for column types Popup or PopupEdit. If this property is not configured, "popup.gif" in Theme will display in default. If date format is set, "calendar.gif" image will display. If you want to use an image other than those two, use this property to set the wanted image. To use a selected image, image path string or image list index should be configured as property value. Use the default theme path as reference when you configure a new image. Image size should be 12*12 to display correctly. For other than dynamic change, it is recommended to replace the theme image at the same location.

CalcLogic can be used to configure calculation formula for the data. If part of column data is referenced in a formula, surround the value with "|" within the formula. The default is "".

For example, if you need to multiply column 5 value by 2 and add column 3 value, the formula will be "|5| * 2 + |3|". Once a formula is configured, the value will be automatically updated upon search or when a column value is changed.

In addition to column name, you can also use SaveName of other columns in your formula. If SaveName of Column 5 is "pay" in the example above, the formula may be "|pay|*2+|3|" and still it will produce the same effect.

CaseSensitive is where you may configure whether to make filtering case-sensitive.

ColMerge configures whether to allow vertical merge for data columns. The default value is 1.

In **ColSpan,** you can set the scope of ColSpan to apply as fixed within a unit data rows for unit-data row structure. When this property is set, cell will display as merged regardless of whether the data are the same. This property is effective only for unit data row structure. To use this property, MergeSheet value must be 3 or 8. (Related property: RowSpan)

ComboCode is composed of code items adjoined by "|" for columns of Type "Combo" or "ComboEdit". The item count must be the same as the item count for ComboText above.

ComboText is composed of string items to display on the screen, adjoined by "|" for columns of Type "Combo" or "ComboEdit".

Cursor can be used to set the cursor image to display when mouse is over the column.

You may set this as "Default" or "Pointer". "Default" will be the default value.

DefaultValue can be used to configure the default value for any new entries. The default value is "".

Edit can be used to configure data editability. The default value is 1.

EditLen can be used to configure the maximum number of characters to allow for a piece of data.

Ellipsis can be used to set whether to display ellipsis when full column text cannot be displayed. The default value is 0.

ExcludeEmpty can be used to configure whether to include empty values to averaging for AutoAvg column type. If not configured, the default value of 0 will apply (include empty value). As this property involves identification of empty values, the format should be set as NullInteger or NullFloat to use this.

FontColor can be used to configure a column's font color.

Format can be used to configure whether to apply Mask to data. Available values are as in the table below. In addition to formats provided in OOTB solution, you may also create a custom format ((ex) Format="#,###,#0"). Formats will apply depending on the column Type.

Format	Description	Available types
Ymd	"Year, month, day" format. Follow SYS_d format of ibmsg.	Date,Text,Popup,PopupEdit
Ym	"Year Month" format. Follow SYS_m format of ibmsg.	Date,Text,Popup,PopupEdit
Md	"Month, day" format. Follow SYS_M format of ibmsg.	Date,Text,Popup,PopupEdit

Hms	"Hour, minute, second" format. Follow SYS_T format of ibmsg.	Date,Text,Popup,PopupEdit
Hm	"Hour, minute" format. Follow SYS_t format of ibmsg.	Date,Text,Popup,PopupEdit
YmdHms	"Year, month, day, hour, minute, second" format. Follow SYS_G format of ibmsg.	Date,Text,Popup,PopupEdit
YmdHm	"Year, month, day, hour, minute" format. Follow SYS_g format of ibmsg.	Date,Text,Popup,PopupEdit
Integer	Integer format, default is 0	Int,AutoSum,AutoAvg
NullInteger	Null integer type, Default is null	Int,AutoSum,AutoAvg
Float	Float format, default is 0	Float,AutoSum,AutoAvg
NullFloat	Null float type, Default is null	Float,AutoSum,AutoAvg
IdNo	Korean resident ID number format	Text
SaupNo	Korean business registration number format	Text
PostNo	Korean zip code format	Text
CardNo	Credit card number format	Text
PhoneNo	Phone number format	Text
Number	Number format (No particular format; accepts number key entries)	Text

※ Number format guide for users

In addition to OOTB formats, you can define a custom format by combining "#", "0", "," ".". Types that may use a custom format are Int, Float, AutoSum and AutoAvg.

Detailed instuctions are as in the table below:

Cell format	symbol	Description

Number	#	If a number format is set using #, display numbers for those digits and leave an empty digit if there is no number for the corresponding digit. When # is used after decimal point, numbers in decimal places will be rounded up if the number exceeds the decimal place count set by #.
		
		Sample)
		
		Input= 12345.678, Format=#,###.##,Output=12,345.68
		
		Input= 0.789 Format=#,###.##,Output=.79

	0	Display value as set. If there is no value to display, display 0 instead. This works the same way as # above in decimal places.
		
		Sample)
		
		Input=123456.7 Format=#,##0.00 Result=123,456.70
		
		Input= 0.7 Format=#,##0.00 Result=0.70

Decimal separator	.	When there is a decimal separator, the number format may be a float. The symbol for decimal separator will be represented with the value configured in SYS_DecimalSeparator of ibmsg.
Thousands separator	,	When thousands separator is configured, groups of thousands will be separated using the separator configured in SYS_GroupSeparator of ibmsg. Also, when thousands separator is used immediately before the decimal separator, the display value will be the number divided by 1000 as many times as the number of commas. Sample)

		Input= 123456.789 Format=0,.00 Result=123.46
		Input= 123456.789 Format=0,,.00 Result=0.12

Percentage	₩₩ %	When percentage symbol is used at the end, the input value will be automatically marked with the percentage symbol. Sample) Input=123.456 Format=#,##0.00₩₩% Result=123.46%

※ Character format guide for users

In addition to OOTB formats, you can define a custom format by combining "#", "*", or others. Types that may use a custom format are Text and PopupEdit.

Detailed instuctions are as in the table below:

Cell format	symbol	Description

Character	#	If a number format is set using #, display numbers for those digits and leave an empty digit if there is no number for the corresponding digit.
		Sample)
		Input=7907211022553,
		Format=######-****, Result=790721-****553

	*	
	Others	Characters or numbers other than # and * will be displayed as they are.

FormatFix can be used to configure whether to return GetCellText value upon GetCellValue. If set as true, the format will be also saved upon saving.

The default value is 0.

FullInput can be used when you need to use the full legnth of EditLen for columns with single row string as the type. The default value is 0.

HeaderCheck can be used to configure whether to check the CheckAll box in the header.

The default value is 1.

Hidden can be used to configure whether to hide a certain data column.

HoverUnderline can be used to configure whether to underline cell value when the mouse point is over the cell. The default value is 0.

Image can be used when an image should be displayed in the column. This may be used for the following column types: "Text", "Int", "Float", "AutoSum", "AutoAvg" The image configured here is the default image for the column. In Search Data Structure, you may set different images for individual cells.

ImgAlign can be used to configure the image location when you are using the Image property. If you set this as "Left", image will display left to text. "Right" will display image on the right side of text. If you choose not to configure, the default value is "Left".

ImgHeight can be used to configure the height of image if the column type is "Image" or you are using the "Image" property. The default is 0, which displays the image at its original size.

ImgWidth can be used to configure the width of image if the column type is "Image" or you are using the "Image" property. The default is 0, which displays the image at its original size.

InsertEdit can be used to configure editability of data the transaction status of which is Insert. The default value is 1.

KeyField can be used to configure whether to make a data cell a required field. If the value is set as 1 and the data cell is empty, a warning message appears when the saving method is called so as to encourage the user to fill the cell. The default value is 0.

LevelSaveName allows you to configure the tree-level parameter name to use when you save data for a tree-type search. If the name is not set, tree levels of the row will not be transferred to the server.

MaximumValue can be used to set the max value allowed during editing for number cell formats like Integer, Float, NullInteger or NullFloat.

MinimumValue can be used to set the min value allowed during editing for number cell formats like Integer, Float, NullInteger or NullFloat.

MultiLineText can be used to configure whether to allow multiple lines for columns with Type "Text". The default value is 0.

PointCount can be used to configure the number of demimal places to display for column type Float.

If a value is not set, the number of decimal places will be determined by Format value. If PointCount is set, it will prevail over Format value.

PopupCheckEdit can be used to check editability of selected rows when column pop-up is set. Set the value as 1 if you need to change only editable data. The default value is 0.

PopupCode is composed of code values mapped to PopupText, adjoined by "|". The value count must be the same as the value count in PopupText. Once this property is configured, when a pop-up menu option is selected, mapping code values display in the adjacent column.

PopupText is composed of popup menu options to display upon right click of mouse, adjoined by "|".

RadioIcon can be used to select the button or box image to display in data cells for CheckBox, DummyCheck, DelCheck and Radio column types. If set 1, the cells will display a radio button. If set 0, a checkbox. The default value is 1 for Radio column type and 0 for other column types.

RowSpan can be used to set the scope of RowSpan to apply as fixed within a unit data rows for unit-data row structure. When this property is set, cell will display as merged regardless of whether the data are the same. This property is effective only for unit data row structure. To use this property, MergeSheet value must be 3 or 8. (Related property: ColSpan)

SaveName can be used to configure the parameter names to use when saving data. If not configured, names will be given sequentially as in C1, C2 and so on.

Sort can be used to configure whether to allow sorting by clicking on the header cell. The default value is 1.

ToolTipText can be used to configure the text to display in tooltip at the header row. The default is empty value.

TreeCol can be used to configure whether to set a reference column for a tree-type search.

TrueValue and **FalseValue** allows user to use custom values in CheckBox columns rather than 0 and 1, which are default in OOTB. For example, if you set as TrueValue:"T" and FalseValue:"F", you can use "T" instead of 1 and "F" instead of 0 to represent values.

UpdateEdit can be used to configure editability of data the transaction status of which is Search. The default value is 1.

Width should be set as pixel count. Otherwise, the column width will be automatically adjusted to fit the header text.

Wrap can be used to configure whether to break lines of text automatically to display the text within the set column width. The default value is 0.

VAVlign can be used to configure vertical alignment option for columns. If set as "Top", text and images will be aligned to the top. If set as "Bottom", they will be aligned to the bottom. If this property is not configured, they will be aligned as "Middle".

- **Example**

```
//Configure functions of each column.

var cols = [

{Type:"Status",Width:60,SaveName:"sStatus",Align:"Center"},

{Type:"DelCheck",Width:60,SaveName:"sDelete",Align:"Center"},

{Type:"Text",Width:100,SaveName:"JOB",Align:"Center"},

{Type:"Combo",Width:100,SaveName:"DEPTNO",Align:"Center"

,ComboText:comboDataArr[0],ComboCode:comboDataArr[1]},

{Type:"Text",Width:60,SaveName:"EMPNO",Align:"Center"},

{Type:"Text",Width:150,SaveName:"ENAME",Align:""},

{Type:"Date",Width:120,SaveName:"HIREDATE",Format:"Ymd"

,Align:"Center",EditLen:8},

{Type:"Text",Width:120,SaveName:"MGR",Align:"Center"},

{Type:"Int",Width:120,SaveName:"SAL",Align:"Right",Format:"NullInteger"},

{Type:"Int",Width:60,SaveName:"COMM",Align:"Right",Format:"Integer"}

];

mySheet.InitColumns(cols);
```

InitComboNoMatchText Method

- **Purpose**

  Using this method, you can configure the characters to display when you configure or search for items that do not exist in Combo items.

  Assuming the Show parameter value is set as 1 (true), when the search data include values that do not exist in ComboCode, the value set at ShowText parameter will display.

  Assuming the InsertItem parameter is set as 1 (true), if a value that has never been set at InitDataCombo is searched, that value will be added as a new item. However, if there is a set value in ShowText, ShowText value will be added as an item.

- **Syntax**

Syntax	ObjId.**InitComboNoMatchText**(Show, [ShowText], [InsertItem])

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Show	Boolean	Required	Whether to show characters other than combo items.
ShowText	String	Optional	Characters to display when combo items are not found, Default=""
InsertItem	Boolean	Optional	If Default=0

- **Example**

```
//When there is no Combo item, display "No Item Found"

mySheet.InitComboNoMatchText(1, "No Item Found");
```

InitHeaders Method

- **Purpose**
  You can define the header title and function using this method.

- **Syntax**

Syntax	ObjId.**InitHeaders(Headers, [info])**

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Headers	json	Required	Define header title and alignment in JSON format.
info	json	Optional	Define header functions such as sorting and column movement permissions.

Detailed configurations

Parameters		Type	Description
headers	Text	String	String of texts to display in header, adjoined by "|"
	Align	String	How to align header text (Default = "Center")
info	Sort	Boolean	Whether to allow sorting by clicking on the header (Default=1)
	ColMove	Boolean	Whether to allow column movement in header (Default=1)
	ColResize	Boolean	Whether to allow resizing of column width (Default=1)

	HeaderCheck	Boolean	Whether the CheckAll in the header is checked (Default=1)

➢ **Example**

```
// Configure double line header title.

var headers = [

{Text:"Status|Delete|Employee Profile|Employee Profile|Employee Profile", Align:" Center"} ,

{Text:"Status|Delete|Employee Name|Employee Number|Date of Joining", Align:" Center"}

];

var info = {Sort:1, ColMove:1, ColResize:0, HeaderCheck:0};

mySheet.InitHeaders(headers,info);
```

IsDataModified Method

- **Purpose**

  Check transaction status of data rows.

  If any data is in transaction status other than "Search", 1 is returned; otherwise, 0 is returned.

- **Syntax**

Syntax	ObjId.**IsDataModified**()

- **Info**

Return	Boolean, whether transaction occurred in a data row		
Parameter	Type	Required Y/N	Description
N/A			

- **Example**

```
// If there is no transaction occurred, cancel saving after displaying a message.
if(!mySheet.IsDataModified()) {
    alert("Nothing to save. Saving is cancelled.");
// Save any transactions occurred
} else {
    mySheet.DoSave("sheet_save.jsp");
}
```

IsHaveChild Method

- **Purpose**

  In tree type data, check whether there are Child level rows to a particular row.

  If there are any child levels, 1 is returned; otherwise, 0 is returned.

IncludeDelRow parameter determines whether to include "deleted" child level rows for the row defined in Row parameter. If the value is set as 1, "Deleted" child rows will be included. If 0, they are excluded. Default=0.

- **Syntax**

Syntax	ObjId.**IsHaveChild**(Row, [IncludeDelRow])

- **Info**

Return	Boolean, whether child rows exist.		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row index of a particular row
IncludeDelRow	Boolean	Optional	Whether to include rows of "Deleted" status among the child level rows, Default=0

- **Example**

```
//Check for child level rows of Row 4.

if(mySheet.IsHaveChild(4)) {

     alert("Row 4 has child level.");

}else{

   alert("Row 4 does not have child level.");

}

//Check child rows of Row 4, including deleted child rows.

var bResult = mySheet.IsHaveChild(4, 1);
```

KeyFieldImage Method

- **Purpose**

Check or configure image file location for required field image.

This method can be used when the user wants to customize the required field image.

- **Syntax**

Syntax	Get	ObjId.**GetKeyFieldImage**()

- **Info**

Return	String, set value (In case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
// Check the required field item image.

mySheet.GetKeyFieldImage();
```

- **Syntax**

Syntax	Set	ObjId.**SetKeyFieldImage**(imgUrl)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
imgUrl	String	Required	Url to set

- **Example**

```
// Change the required field item image.

mySheet.SetKeyFieldImage("/sheet/imgKeyField.gif");
```

LastCol Method

- **Purpose**

  Return the column index of the last column.

- **Syntax**

Syntax	ObjId.**LastCol**()

- **Info**

Return	Long, last column index		
Parameter	Type	Required Y/N	Description
N/A			

- **Example**

```
//Check the column index of the last column.

alert("Index of the last column is " + mySheet.LastCol() + ".");
```

LastRow Method

- **Purpose**

  Return the row index of the last row.

  Using this method will return the index of the very last row, not just last data row or the last row as displayed in the screen.

  Note that the last row may be a sum row, data row or even a header row.

- **Syntax**

Syntax	ObjId.**LastRow**()

- **Info**

Return	Long, last row index	
Parameter	Type	Description
N/A		

- **Example**

```
//Check the row index of the last row.

alert("Index of the last row is " + mySheet.LastRow() + ".");
```

LeftCol Method

- **Purpose**

  Check or configure the leftmost column to display. If there are any frozen columns, the column you set using this method will display right to the frozen columns. Setting this property will not move focus, but as the column at the leftmost side is changed, horizontal scroll bar will be moved accordingly. LeftCol is set by counting distance from column 0 including hidden columns.

- **Syntax**

Syntax	Get	ObjId.**GetLeftCol**()

- **Info**

Return	Long, leftmost column index (in case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Check for left column setting

mySheet.GetLeftCol();
```

- **Syntax**

Syntax	Set	ObjId.**SetLeftCol**(Col)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Col	Long/ String	Required	Column index of a particular column or SaveName

- **Example**

```
//If there are frozen columns

//Configure that column 4 should display at the leftmost side of horizontal scroll

mySheet.SetLeftCol(4);



//When there are no frozne columns

//Configure that column 6 should display at the leftmost side of horizontal scroll

mySheet.SetLeftCol(6);
```

LoadSaveData Method

- **Purpose**

Fetch saved data as method parameter and process the result within IBSheet.

This method can be used to read encrypted save data when security module is used. Saved data are retrieved using GetSaveData method. The data retrieved are set as a method parameter. The results are displayed and OnSaveEnd event fires.

- **Syntax**

Syntax	ObjId.**LoadSaveData**(Content)

- **Info**

Return	None.		
Parameter	Type	Required Y/N	Description
Content	String	Required	Save XML or Save JSON string

- **Example**

```
//Apply ajax to saving

var rtnData = mySheet.GetSaveData("Action.do",param);



//Represent saving results

mySheet.LoadSaveData(rtnData);
```

LoadSearchChildData Method

- **Purpose**

This method performs the same function as DoSearchChild method but as object or string, not url; In tree structure, child node data (xml or json) are received as method parameter and loaded in IBSheet. This method can be used to read encrypted search data when security module is used. Search data are retrieved using GetSearchData method from the server. The data string fetched is set as a method parameter. The search data are loaded to IBSheet and OnSearchEnd event fires.

- **Syntax**

Syntax	ObjId.**LoadSearchChildData**(Row, Data, [Opt])

- **Info**

Return	None.		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Target parent row index
Data	String / Object	Required	Data string or JSON object to search
Opt	Object	Optional	Option parameter Configure function into JSON format.

The followings are the Opt option parameters:

WaitDlg	Boolean	Whether to display waiting image (Default=1)
Append	Boolean	Whether to append to the existing child data (Default=0)

- **Example**

```
//Read search data

var Row = mySheet.GetSelectRow();

var Data;

Data=mySheet.GetSearchData('list2_Child_Json.jsp',
'ChildLevel='+(parseInt(mySheet.GetRowLevel(Row))+1));

var Opt = {Append: 1, WaitDlg: 1};



//Load search data
```

```
mySheet.LoadSearchChildData(Row, Data, Opt);
```

LoadSearchData Method

- **Purpose**

  Get search data (xml or json) as a method parameter and load to IBSheet.

  This method can be used to read encrypted search data when security module is used.

  Search data are retrieved using GetSearchData method from the server. The data string fetched is set as a method parameter. The search data are loaded to IBSheet and OnSearchEnd event fires.

- **Syntax**

Syntax	ObjId.**LoadSearchData**(Content, [Opt])

- **Info**

Return	None.		
Parameter	Type	Required Y/N	Description
Content	String	Required	Search XML or Search JSON string
Opt.Append	Boolean	Optional	Append search result or not, Default=0

- **Example**

```
/* The below two rows represent each step of two-step processes of using DoSearch() method. */

//Read search data

var sXml = mySheet.GetSearchData(" list.jsp");



//Load search data
```

```
mySheet.LoadSearchData(sXml);



// Represent search result by appending to the exisisting data

var opt = { Append : 1 };

mySheet.LoadSearchData(sXml, opt);
```

LoadExcel Method

- **Purpose**

  Read data from excel to display in the data area.

  The parameters to send are in JSON type. Put configuration information into JSON format and send it to the server.

  Sample)

  ```
  var params = { Mode : "NoHeader ",   StartRow: "1"} ;

  mySheet.LoadExcel(params);
  ```

  If **Append** parameter is set as 1, you may append the current excel data to the existing data to load.

  ColumnMapping parameter is used when you want to load a selected range of excel columns to IBSheet from the first column onward. The default value is null. If not null, this parameter will prevail over Mode even though its value is HeaderMatch.

  FileExt parameter can be used to configure file extensions that can be uploaded. Extensions should be put without ".", such as "xls". The default value is "" and it has the same effect as "xls|xlsx". Multiple extensions should be separated with "|".

Mode parameter determines whether to match grid header title and excel header title when loading, and whether header is included in any excel file to load.

Mode value	Description
HeaderMatch	Match excel header titles with the grid header titles to load excel columns in the matching order, even though columns are mixed in the excel document.
NoHeader	Excel document does not have a header row above data rows, so keep the excel column order when loading.
HeaderSkip	Starting from the first row, the excel document has as many header rows as the grid header rows, but ignore those and instead load the rest data rows sequentially. When IBSheet has two header rows, load excel document from the third row, which is presumably the start of data rows.

StartRow and **EndRow** parameters define the starting and ending rows of the areas to fetch; if these are not set, all excel areas will be fetched.

Excel row number starts from 1.

WorkSheetName parameter can be used to set the names of excel worksheets to load.

Selection of excel worksheet will be based on WorkSheetName parameter first.

If reading the parameter fails, then WorkSheetNo parameter value will be considered.

(Sample)

ColumnMapping:'1|2|3' // Load the first, second and third columns of the excel file to the first, second and third columns of IBSheet.

ColumnMapping:'7||7' // Load the seventh excel column to the first and third columns of IBSheet.

- **Syntax**

Syntax	ObjId.**LoadExcel**([parameters])

- **Info**

Parameter	Type	Required Y/N	Description
Append	Boolean	Optional	Whether to append to the existing data Default=0
ColumnMapping	String	Optional	Excel column number Default=""
EndRow	String	Optional	Last row number after excel loading is completed Default="0"
ExtendParam	String	Optional	Define extra parameters to send to the server in a=1&b=2 format. Default="" Default=""
FileExt	String	Optional	File extensions that can be uploaded Default=""
Mode	String	Optional	Loading options (header matching, etc.) Default="HeaderMatch"
StartRow	String	Optional	Excel loading row number Default="1"
WorkSheetNo	String	Optional	Excel worksheet number Default="1"

WorkSheetName	String	Optional	Excel Worksheet name Default=""

- **Example**

```
// Load excel file

mySheet.LoadExcel();



// Compare the header titles to load the matching columns, but read only
from row 7 to 9.

mySheet.LoadExcel({Mode:'HeaderMatch',StartRow:'7',EndRow:'9'});



// Load the fifth excel column values to the IBSheet first column, and load
 the first excel column values to the 5th IBSheet column.

mySheet.LoadExcel({ColumnMapping:'5|4|3|2|1'});
```

LoadExcelBuffer Method

- **Purpose**

Load one excel document to multiple sheets.

When you set IsBuffer parameter of LoadExcelBuffer as true, all LoadExcel execution thereafter will not run but be buffered in memory.

File upload window will appear once you change the IsBuffer value to false.

All the FileExt property settings made during buffering will be ignored except for the last one.

- **Syntax**

Syntax	ObjId.**LoadExcelBuffer**(IsBuffer)

➢ **Info**

Parameter	Type	Required Y/N	Description
IsBuffer	bool	Required	Buffering or not

➢ **Example**

```
// Immediately load an excel file

mySheet.LoadExcel();



//Buffer afterwards. Do nothing.

mySheet.LoadExcelBuffer(true);



// Schedule an excel loading. Compare the header titles of my Sheet and the first spreadsheet of an excel file to load the same columns, but load only rows 7-9.

mySheet.LoadExcel({Mode:'HeaderMatch',StartRow:'7',EndRow:'9',
WorkSheetNo:1});



// Schedule an excel loading. Load the second spreadsheet of excel file to
  mySheet2. Load the fifth excel column values to the first IBSheet column,
  and load the first excel column values to the fifth IBSheet column.

mySheet2.LoadExcel({ColumnMapping:'5|4|3|2|1', WorkSheetNo:2});



// Bring up a file upload window to send all buffered excel and file loading to the server.

mySheet.LoadExcelBuffer(false);
```

LoadExcelUrl Method

- **Purpose**

  Check or configure the server page path to process excel upload.

- **Syntax**

Syntax	Get	ObjId.**GetLoadExcelUrl**()

- **Info**

Return	String, set path value (in case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Check the excel upload path.

var url = mySheet.GetLoadExcelUrl();
```

- **Syntax**

Syntax	Set	ObjId.**SetLoadExcelUrl**(Url)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Url	String	Required	Server page URL to set

- **Example**

```
// Configure excel upload path.

mySheet.SetLoadExcelUrl("/jsp/LoadExcel.jsp");
```

LoadText Method

- **Purpose**
  Read data from a text file and display them in the data area.

The parameters to send are in JSON type. Put configuration information into JSON format and send it to the server.

Sample)

```
var params = { Mode : "NoHeader ", Deli: "₩t"} ;

mySheet.LoadText(params);
```

If **Append** parameter is set as 1, you may append the current search result data to the existing data to run a search.

Deli parameter sets a column separator for the data. The default value is "₩t".

FileExt can be used to configure file extensions that can be uploaded. Extensions should be put without ".", such as "txt". The default value is "" and it has the same effect as "txt". Multiple extensions should be separated with "|".

Mode parameter determines whether to match grid header title and excel header title when loading, and whether header is included in any excel file to load.

Mode value	Description

HeaderMatch	Match text header titles with the grid header titles to load text columns in the matching order, even though columns are mixed in the text document.
NoHeader	Text document does not have a header row above data rows, so keep the text column order when loading.
HeaderSkip	Starting from the first row, the text document is presumed to have as many header rows as the grid header row count. However, those will be ignored and instead the rest data rows will load sequentially. When IBSheet has two header rows, load text document from the third row, which is presumably the start of data rows.

➢ **Syntax**

Syntax	ObjId.**LoadText**([parameters])

➢ **Info**

Parameter	Type	Required Y/N	Description
Mode	String	Optional	Loading options (header matching, etc.) Default="HeaderMatch"
Deli	String	Optional	Data column separator. Default="₩t"
Append	Boolean	Optional	Whether to append to the existing data Default=0
FileExt	String	Optional	File extensions that can be

			uploaded Default=""

- **Example**

```
// Load as text file

mySheet. LoadText();



// Compare the header titles to load the matching columns; column
    separator is'₩t'.

mySheet.Load Text({Mode:'HeaderMatch', Deli:'₩t'});
```

LoadTextUrl Method

- **Purpose**

  Check and configure server page path to process text file upload.

- **Syntax**

Syntax	Get	ObjId.**GetLoadTextUrl**()

- **Info**

Return	String, set path value (in case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Check the text upload path.

var url = mySheet.GetLoadTextUrl();
```

- **Syntax**

Syntax	Set	ObjId.**SetLoadTextUrl**(Url)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Url	String	Required	Server page URL to set

- **Example**

```
// Configure text upload path.

mySheet.SetLoadTextUrl("/jsp/LoadText.jsp");
```

MergeSheet Method

- **Purpose**

Check or configure the overall merge types permitted.

If 'Both' merge option is turned off, you cannot merge any cells. When 'Both' merge optioni is allowed, you can set to allow or disallow vertical or horizontal merge.

※ **Merged data will be divided by page size set in SetConfig. However, if merge action follows msPrevColumnMerge setting, you can display merged data based on previous column merge status by setting PrevColumnMergeMode:0 in SetConfig. This action may result in delay in performance if there are many merged rows, as it will display all merged rows on the screen.**

This method performs the same function as MergeSheet in SetConfig method.

Constant value	Type	Description
0	msNone	No merge
1	msAll	Allow merge for all

2	msPrevColumnMerge	Merge is allowed for rows where they are merged in the previous column.
3	msFixedMerge	Merge fixed cells in unit data row structure
5	msHeaderOnly	Allow merge in the header rows only
7	msPrevColumnMerge + msHeaderOnly	Merge is allowed for rows where they are merged in the previous column as well as header rows.
8	msFixedMerge + msHeaderOnlyl	Merge is allowed for fixed cells in unit data row structure and in header rows

- **Syntax**

Syntax	Get	ObjId.**GetMergeSheet**()

- **Info**

Return	Integer, currently configured merge type (in case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Check merge type

mySheet.GetMergeSheet();
```

- **Syntax**

Syntax	Set	ObjId.**SetMergeSheet**(Merge)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Merge	Integer	Required	Merge types

- **Example**

```
//Configure all merge types upon loading

mySheet.SetMergeSheet( msHeaderOnly);
```

MaximumValue Method

- **Purpose**

  Check or configure the max value to allow during editing when the cell format is Integer, Float, NullInteger, NullFloat or other number related format.

  When a cell with max value setting is edited with a value bigger than the set max value, a warning message will appear. (ibmsg file's SYS_MaximumBigValue)

  Min and max value configured as default are as follows:

Format	Min value	**Max value**
Integer NullInteger	-999999999999999	999999999999999
Float NullFloat	-999999999999999.00	999999999999999.00

- **Syntax**

Syntax	Get	ObjId.**GetMaximumValue**(Row, Col)

➢ **Info**

Return	Long, set max value (in case of Get Method)		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row index of the cell
Col	Long/ String	Required	Column index or SaveName

➢ **Example**

```
//Check the configured max value of number format.
mySheet.GetMaximumValue(1, 12);
```

➢ **Syntax**

Syntax	Set	ObjId.**SetMaximumValue**(Row, Col, Value)

➢ **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row index of the cell
Col	Long/ String	Required	Column index or SaveName
Value	Long	Required	Max value to configure

➢ **Example**

```
//Set the max value of number format.
mySheet.SetMaximumValue(1, 12, 5000);
```

MinimumValue Method

- **Purpose**

  Check or configure the min value to allow during editing when the cell format is Integer, Float, NullInteger, NullFloat or other number related format.

  When a cell with min value limit is edited with a value smaller than the set min value, a warning message will appear. (ibmsg file's SYS_MinimumBigValue)

  Min and max value configured as default are as follows:

Format	**Min value**	Max value
Integer NullInteger	-999999999999999	999999999999999
Float NullFloat	-999999999999999.00	999999999999999.00

- **Syntax**

Syntax	Get	ObjId.**GetMinimumValue**(Row, Col)

- **Info**

Return	Long, set min value (in case of Get Method)		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row Index of the selected data
Col	Long/ String	Required	Column Index of the selected data or SaveName

- **Example**

```
//Check the configured min value of number format.

mySheet.GetMinimumValue(1, 12);
```

- **Syntax**

Syntax	Set	ObjId.**SetMinimumValue**(Row, Col, Value)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row Index of the selected data
Col	Long/ String	Required	Column Index of the selected data or SaveName
Value	Long	Required	Min value to set

- **Example**

```
//Set the min value of number format.

mySheet.SetMinimumValue(1, 12, 1000);
```

MouseCol Method

- **Purpose**
  Return the selected column index when mouse button is clicked. If the mouse is clicked on non data area, return -1.

- **Syntax**

Syntax	ObjId.**MouseCol**()

- **Info**

Return	Long, column index of the mouse location		
Parameter	Type	Required Y/N	Description
N/A			

➢ **Example**

```
// Fetch the column index where mouse button is clicked.

function mySheet_OnMouseDown(Button, Shift, X, Y) {

    //Check the column of the clicked cell

    alert(mySheet.MouseRow() + "Row" + mySheet.MouseCol() + "Column is
  clicked");

}
```

MousePointer Method

➢ **Purpose**

Check or configure the mouse pointer style. Values available are as follows:

Value	Mouse pointer style
Default	Basic arrow style
Hand	Finger style

This property is different from DataLinkMouse property.

➢ **Syntax**

Syntax	Get	ObjId.**GetMousePointer**()

➢ **Info**

Return	String, mouse style configuration (In case of Get Method)		
Parameter	Type	Required	Description

		Y/N	

➢ **Example**

```
//Check the mouse pointer style.

mySheet.GetMousePointer("Default");
```

➢ **Syntax**

Syntax	Set	ObjId.**SetMousePointer**(Pointer)

➢ **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Pointer	String	Required	Mouse pointer style to set

➢ **Example**

```
function mySheet_OnMouseMove(Button, Shift, X, Y) {

  //Change the mouse pointer style to finter only when the mouse location
is Column 2.

  if(mySheet.MouseCol() == 2) {

    mySheet.SetMousePointer("Hand");

  } else {

    mySheet.SetMousePointer("Default");

  }

}
```

MouseRow Method

- **Purpose**

  Return the selected row index when mouse button is clicked. If the mouse is clicked on non data area, return -1.

- **Syntax**

Syntax	ObjId.**MouseRow**()

- **Info**

Return	Long, row index of the mouse location	
Parameter	Type	Description
N/A		

- **Example**

```
// Fetch the row index where mouse button is clicked.

function mySheet_OnMouseDown(Button, Shift, X, Y){

  //Check the row of the clicked cell

      alert(mySheet.MouseRow() + "row is clicked");

}
```

MoveColumnFail Method

- **Purpose**

  This method is used within OnBeforeColumnMove event, and it commands whether column movement will fail. If you set this property as 1 within OnBeforeColumnMove event, column movement will fail and OnAfterColumnMove will not fire.

- **Syntax**

Syntax	ObjId.**MoveColumnFail**(Flag)

➢ **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Flag	Boolean	Required	Whether column movement fails

➢ **Example**

Cancel column movement if column 0 is moved to a location past column 3.

```
function mySheet_OnBeforeColumnMove(Col, NewPos) {
    if(Col==0 && NewPos > 3) {
      mySheet.MoveColumnFail(1);
  }else{
      mySheet.MoveColumnFail(0);
  }
}
```

MoveColumnPos Method

➢ **Purpose**

Move a particular column to a new column location. If Event parameter value is set as 1, OnBeforeColumnMove event and OnAfterColumnMove event will fire, so you can use the column movement cancellation function within OnBeforeColumnMove event. If set 0, columns will be moved without events.

➢ **Syntax**

Syntax	ObjId.**MoveColumnPos**(Col, NewPos, [Event])

➢ **Info**

Return	Boolean, whether movement has successfully done		
Parameter	Type	Required Y/N	Description
Col	Long/String	Required	Index or SaveName of the column to move
NewPos	Long/String	Required	Column location index to move column to or SaveName
Event	Boolean	Optional	When the column is moved, whether to fire OnBeforeColumnMove Event and OnAfterColumn Event. Default=1

- **Example**

```
//Move column 1 to column 9.

mySheet.MoveColumnPos(1, 9);



//In case you use SaveName for each column

mySheet.MoveColumnPos("stockNm", "payAmt");
```

RangeBackColor Method

- **Purpose**

  Check or configure background color for a selected area of cells

  You may check or configure cell background color for either data or header area cell. If the area is set beyond header and data area, background color setting will be cancelled without an error message. If you are simply checking, 0 will be returned.

- **Syntax**

Syntax	Get	ObjId.**GetRangeBackColor**(Row1,Col1,Row2,Col2)

- **Info**

Return	String, currently set value (In case of Get Method)		
Parameter	Type	Required Y/N	Description
Row1	Long	Required	Row index of the first cell of the area
Col1	Long	Required	Column index of the first cell of the area
Row2	Long	Required	Row index of the last cell of the area
Col2	Long	Required	Column index of the last cell of the area

- **Example**

```
//Check background color of a selected area

mySheet.GetRangeBackColor(1,0,10,10);
```

- **Syntax**

Syntax	Set	ObjId.**SetRangeBackColor**(Row1,Col1,Row2,Col2,Color)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Row1	Long	Required	Row index of the first cell of the area

Col1	Long	Required	Column index of the first cell of the area
Row2	Long	Required	Row index of the last cell of the area
Col2	Long	Required	Column index of the last cell of the area
Color	String	Required	WebColor

- **Example**

```
//Set background color to a selected area

mySheet.SetRangeBackColor(1,0,10,10, "#FFFF00");
```

RangeFontBold Method

- **Purpose**

  Check or configure bold font for a selected area of cells

  You may check or configure bold font for either data or header area cell. If the area is set beyond header or data area, bold setting will be cancelled without an error message. If you are simply checking, 0 will be returned.

- **Syntax**

Syntax	Get	ObjId.**GetRangeFontBold**(Row1, Col1, Row2, Col2)

- **Info**

Return	Boolean, currently set value (In case of Get Method)		
Parameter	Type	Required Y/N	Description
Row1	Long	Required	Row index of the first cell of the area

Col1	Long	Required	Column index of the first cell of the area
Row2	Long	Required	Row index of the last cell of the area
Col2	Long	Required	Column index of the last cell of the area

- **Example**

```
//Check bold font usage in a selected area

mySheet.GetRangeFontBold(1, 0, 2, 2);
```

- **Syntax**

Syntax	Set	ObjId.**SetRangeFontBold**(Row1, Col1, Row2, Col2, Bold)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Row1	Long	Required	Row index of the first cell of the area
Col1	Long	Required	Column index of the first cell of the area
Row2	Long	Required	Row index of the last cell of the area
Col2	Long	Required	Column index of the last cell of the area
Bold	Boolean	Required	Bold font setting

- **Example**

```
//Bolden font in a selected area of cells

mySheet.SetRangeFontBold(1, 0, 2, 2, 1);
```

RangeFontColor Method

- **Purpose**

  Check or configure font color for a selected area of cells

  You may check or configure cell font color for either data or header area cell. If the area is set beyond header and data area, font color setting will be cancelled without an error message. If you are simply checking, 0 will be returned.

- **Syntax**

Syntax	Get	ObjId.**GetRangeFontColor**(Row1, Col1, Row2, Col2)

- **Info**

Return	String, currently set value (In case of Get Method)		
Parameter	Type	Required Y/N	Description
Row1	Long	Required	Row index of the first cell of the area
Col1	Long	Required	Column index of the first cell of the area
Row2	Long	Required	Row index of the last cell of the area
Col2	Long	Required	Column index of the last cell of the area

- **Example**

```
//Check font color of a selected area

mySheet.GetRangeFontColor(1,0,10,10);
```

- **Syntax**

Syntax	Set	ObjId.**SetRangeFontColor**(Row1, Col1, Row2, Col2, Color)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Row1	Long	Required	Row index of the first cell of the area
Col1	Long	Required	Column index of the first cell of the area
Row2	Long	Required	Row index of the last cell of the area
Col2	Long	Required	Column index of the last cell of the area
Color	String	Required	WebColor

- **Example**

```
//Set font color to a selected area

mySheet.SetRangeFontColor(1,0,10,10, "FF0000");
```

RangeText Method

- **Purpose**

Check or configure cell text of a selected area with the values as they appear on the screen including format configurations.

You may check or configure cell text for either data or header area cell. If the area is set beyond header or data area, cell text setting will be cancelled without an error message. If you are simply checking, 0 will be returned.

- **Syntax**

Syntax	Get	ObjId.**GetRangeText**(Row1,Col1,Row2,Col2,[ColSeparator], [RowSeparator])

- **Info**

Return	String, currently set string (In case of Get Method)		
Parameter	Type	Required Y/N	Description
Row1	Long	Required	Row index of the first cell of the area
Col1	Long	Required	Column index of the first cell of the area
Row2	Long	Required	Row index of the last cell of the area
Col2	Long	Required	Column index of the last cell of the area
ColSeparator	String	Optional	Column separator Default="|"
RowSeparator	String	Optional	Row separator Default="^"

- **Example**

```
//Check cell text of a selected area with the values as they appear on the screen including format configurations.

mySheet.GetRangeText(1, 1, 2, 2, "|", "^");
```

- **Syntax**

Syntax	Set	ObjId.**SetRangeText**(sData,Row1,Col1,Row2,Col2, [ColSeparator], [RowSeparator])

- **Info**

Return	None

Parameter	Type	Required Y/N	Description
sData	String	Required	String
Row1	Long	Required	Row index of the first cell of the area
Col1	Long	Required	Column index of the first cell of the area
Row2	Long	Required	Row index of the last cell of the area
Col2	Long	Required	Column index of the last cell of the area
ColSeparator	String	Optional	Column separator Default="|"
RowSeparator	String	Optional	Row separator Default="^"

- **Example**

```
//Set A in cell 1, 1 and D in cell 2, 2.

  mySheet.SetRangeText("A|B^C|D", 1, 1, 2, 2, "|", "^");



  //Set ** from cell 1, 7 to cell 5, 10.

    mySheet.SetRangeText("**", 1, 7, 5, 10);
```

RangeValue Method

- **Purpose**

  Check or configure the cell values of a particular area with values without formatting that are used for saving.

  You may check or configure cell values for either data or header area cell. If the area is set beyond header or data area, the setting will be cancelled without an error message. If you are simply checking, 0 will be returned.

➢ **Syntax**

Syntax	Get	ObjId.**GetRangeValue(**Row1,Col1,Row2,Col2,[ColSeparator],[RowSeparator])

➢ **Info**

Return	String, currently set string (In case of Get Method)		
Parameter	Type	Required Y/N	Description
Row1	Long	Required	Row index of the first cell of the area
Col1	Long	Required	Column index of the first cell of the area
Row2	Long	Required	Row index of the last cell of the area
Col2	Long	Required	Column index of the last cell of the area
ColSeparator	String	Optional	Column separator Default=|
RowSeparator	String	Optional	Row separator Default=^

➢ **Example**

```
//Check the cell values of a particular area with values without formatting that are used for saving.

mySheet.GetRangeValue("A|B^C|D", 1, 1, 2, 2, "|", "^");
```

➢ **Syntax**

Syntax	Set	ObjId.**SetRangeValue**(sData,Row1,Col1,Row2,Col2,[ColSeparator],[RowSeparator])

➢ **Info**

Return	None

Parameter	Type	Required Y/N	Description
sData	String	Required	String
Row1	Long	Required	Row index of the first cell of the area
Col1	Long	Required	Column index of the first cell of the area
Row2	Long	Required	Row index of the last cell of the area
Col2	Long	Required	Column index of the last cell of the area
ColSeparator	String	Optional	Column separator Default=|
RowSeparator	String	Optional	Row separator Default=^

- **Example**

```
//Set A in cell 1, 1 and D in cell 2, 2.

mySheet.SetRangeValue("A|B^C|D", 1, 1, 2, 2, "|", "^");



//Set ** from cell 1, 7 to cell 5, 10.

mySheet.SetRangeValue("**", 1, 7, 5, 10);
```

RedrawSum Method

- **Purpose**

  Check or configure whether to recalculate sums.

  If you change cell values of a column which include sum using SetCellValue method, data change performance will suffer. The reason is that at every update of a single data value, sum will be recalculated. You can improve data update performance using this method to recalculate sum only after all data have been changed. Note that after using this property by setting it at 0, you need to reset the value to 1.

- **Syntax**

Syntax	Get	ObjId.**GetRedrawSum**()

- **Info**

Return	Boolean, set value (In case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Check whether to sum data items.

mySheet.GetRedrawSum();
```

- **Syntax**

Syntax	Set	ObjId.**SetRedrawSum**(Redraw)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Redraw	Boolean	Required	Whether to calculate sum row

- **Example**

```
//Use when there are many data items to update

mySheet.SetRedrawSum(0);

for(var i=1; i<100; i++) mySheet.SetCellValue(i,1, 10000, 0);

//When RedrawSum is set as 1, updated sum and data will display together.
```

```
mySheet.SetRedrawSum(1);
```

RemoveAll Method

- **Purpose**

  Delete all data rows and leave the header row only.

- **Syntax**

Syntax	ObjId.**RemoveAll**()

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
N/A			

- **Example**

```
// Delete all data

mySheet.RemoveAll();
```

RenderSheet Method

- **Purpose**

  When you add multiple rows or hide multiple columns, you may use this method and do rendering once at the end to improve performance.

  Note! After using this property by setting it at 0(false), you need to reset the value to 1 (true). Otherwise, Sheet may display incorrectly or broken.

  You may expect improved performance when you use the following functions for multiple targets.

Method	Description
ColHidden	Hiding columns
DataInsert	Adding rows
InitCellProperty	Resetting cell properties

- **Syntax**

Syntax	ObjId.**RenderSheet**(Render)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Render	Boolean	Required	Rendering or not (Default=1)

- **Example**

```
// Improve performance when adding 30 rows, using RenderSheet method

mySheet.RenderSheet(0);

for (var i = 0; i <30; i++) {

mySheet.DataInsert();

}

mySheet.RenderSheet(1);
```

ReNumberSeq Method

- **Purpose**

  Renumber the data with data type Seq.

- **Syntax**

Syntax	ObjId.**ReNumberSeq()**

➢ **Info**

Return	None		
Parameter	Type	Required Y/N	Description
N/A			

➢ **Example**

```
//Renumber columns with column type Seq from 1.

mySheet.ReNumberSeq();
```

Reset Method

➢ **Purpose**

Remove all configurations in IBSheet and reset to OOTB state.

➢ **Syntax**

Syntax	ObjId.**Reset**()

➢ **Info**

Return	None		
Parameter	Type	Required Y/N	Description
N/A			

➢ **Example**

```
//Restore the original state

mySheet.Reset();
```

ReturnCellData Method

- **Purpose**

  In search result, restore a particular cell value to the original search result data before transaction is occurred.

  In the following cases, you cannot restore the original search data:

  1. When the data is not searched for; for example, when the data is in Insert state

  2. When there is no Status column

  3. When the data type is one of the followings: Status, DelCheck, Seq, Image

- **Syntax**

Syntax	ObjId.**ReturnCellData**(Row,Col)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row index of the data cell
Col	Long / String	Required	Column index of the data cell or SaveName

- **Example**

```
//Restore the original state

mySheet.ReturnCellData(Row,Col);
```

ReturnColumnPos Method

- **Purpose**

  Return the moved column to its original location

- **Syntax**

Syntax	ObjId.**ReturnColumnPos**()

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
N/A			

- **Example**

```
//Return to the original location

mySheet.ReturnColumnPos();
```

ReturnData Method

- **Purpose**

  Change the data of a particular row to search result string

  In the following cases, you cannot restore the original search data:

  1. When the data is not searched for; for example, when the data is in Insert state

  2. When there is no Status column

- **Syntax**

Syntax	ObjId.**ReturnData**(Row)

- **Info**

Return	None		
Parameter	Type	Required	Description

		Y/N	
Row	Long	Required	Row index of the data row

- **Example**

```
//Restore the original state

mySheet.ReturnData(2);
```

RowBackColor Method

- **Purpose**

  Check or configure background color of the entire row. Background color of the data area will be checked or configured. If the row does not exist, it will not return any error message and just cancels the background color setting. If you are simply checking, 0 will be returned.

  Configure the color using WebColor.

- **Syntax**

Syntax	Get	ObjId.**GetRowBackColor**(Row)

- **Info**

Return	String, background color (in case of Get Method)		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row Index

- **Example**

```
// Check background color of Row 1.

mySheet.GetRowBackColor(1);
```

- **Syntax**

Syntax	Set	ObjId.**SetRowBackColor**(Row,BackColor)

➢ **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row Index
BackColor	String	Required	WebColor color value

➢ **Example**

```
//Set the row background color as gray for Row 1.

mySheet.SetRowBackColor(1,"#C0C0C0");



//Set the row background color as red for Row 3.

mySheet.SetRowBackColor(1,"#FF0000");

//Set the Row 2 background color as the same color as Row 1 background.

mySheet.SetRowBackColor(2, mySheet.GetRowBackColor(1));
```

RowBackColorD Method

➢ **Purpose**

Check or configure background color of rows the transaction status of which is Deleted.

When this method applies, the background color will apply to newly added items and the background color of the existing data on the sheet will remain the same.

Configure the color using WebColor.

➢ **Syntax**

Syntax	Get	ObjId.**GetRowBackColorD**()

➢ **Info**

Return	String, set color (in case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
Check background color of rows the transaction status of which is Deleted.

mySheet.GetRowBackColorD();
```

- **Syntax**

Syntax	Set	ObjId.**SetRowBackColorD**(BackColor)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
BackColor	String	Required	WebColor값

- **Example**

```
//Set the background color of rows with transaction status Deleted as gray.

mySheet.SetRowBackColorD("#C0C0C0");



//Set the background color of rows with transaction status Deleted as the same color as Row 1 background.

mySheet.SetRowBackColorD(mySheet.GetRowBackColor(1));
```

RowBackColorI Method

- **Purpose**

Check or configure background color of rows the transaction status of which is Insert.

When this method applies, the background color will apply to newly added items and the background color of the existing data on the sheet will remain the same.

- **Syntax**

Syntax	Get	ObjId.**GetRowBackColorI**()

- **Info**

Return	String, set color (in case of Get Method)		
Parameter	Type	Required Y/N	Description
BackColor	String	Required	WebColor color value

- **Example**

```
//Check background color of rows the transaction status of which is Insert.

mySheet.GetRowBackColorI();
```

- **Syntax**

Syntax	Set	ObjId.**SetRowBackColorI**(BackColor)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
BackColor	String	Required	WebColor color value

- **Example**

```
//Set the background color of rows with transaction status Insert as gray.

mySheet.SetRowBackColorI("#C0C0C0");



//Set the background color of rows with transaction status Insert as the same
color as Row 1 background.
```

```
mySheet.SetRowBackColorI(mySheet.GetRowBackColor(1));
```

RowBackColorU Method

- **Purpose**

  Check or configure background color of rows the transaction status of which is Edit.

  When this method applies, the background color will apply to newly added items and the background color of the existing data on the sheet will remain the same.

- **Syntax**

Syntax	Get	ObjId.**GetRowBackColorU**()

- **Info**

Return	String, set color (in case of Get Method)		
Parameter	Type	Required Y/N	Description
BackColor	String	Required	WebColor color value

- **Example**

```
//Check background color of rows the transaction status of which is Edit.

mySheet.GetRowBackColorU();
```

- **Syntax**

Syntax	Set	ObjId.**SetRowBackColorU**(BackColor)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
BackColor	String	Required	WebColor color value

- **Example**

```
//Set the background color of rows with transaction status Edit as gray.

mySheet.SetRowBackColorU("#C0C0C0");



//Set the background color of rows with transaction status Edit as the same color as Row 1 background.

mySheet.SetRowBackColorU(mySheet.GetRowBackColor(1));
```

RowCount Method

- **Purpose**

  Check the total data row count.

  If the Status value is not set, this method will check for the total data row count, including search results and any newly added rows.

  Depending on the Status value, you may count the rows with different status, such as Search, Insert, Edit or Deleted.

- **Syntax**

Syntax	ObjId.**RowCount**([Status])

- **Info**

Return	Long, total or partial data row count with a particular transaction status		
Parameter	Type	Required Y/N	Description
Status	String	Optional	Transaction code. Default="Total count"

- **Example**

```
alert("Total count is " + mySheet.RowCount() + " rows. ");

alert("Search row count is " + mySheet.RowCount("R") + " rows. ");

alert("Insert row count is " + mySheet.RowCount("I") + " rows. ");

alert("Edit row count is " + mySheet.RowCount("U") + "   rows. ");

alert("Deleted row count is " + mySheet.RowCount("D") + "   rows. ");
```

RowDelete Method

- **Purpose**

  Delete one or more data rows.

  If Confirm parameter is set as 1, a confirmation message will appear before deletion.

  If Row parameter is not set, rows currently selected will be deleted.

- **Syntax**

Syntax	ObjId.**RowDelete**([Row],[Confirm])

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Row	Number/String	Optional	Index of the row to delete, or a string of row indexes adjoined by "|". Default="Currently selected row(s)"
Confirm	Boolean	Optional	Whether to show confirmation message before deletion Default=0

- **Example**

```
//Delete Row 1 without a confirmation message

mySheet.RowDelete(1, 0);



// Delete rows 3, 7 and 10

mySheet.RowDelete("3|7|10");
```

RowDraggable Method

- **Purpose**

  Check or configure whether to allow mouse dragging for a particular row

  Available options and values are as follows:

Value	Details
0	Row drag not allowed
1 (Default)	Row drag is allowed

- **Syntax**

Syntax	Get	ObjId.**GetRowDraggable**(row)

- **Info**

Return	Boolean, set value (In case of Get Method)		
Parameter	Type	Required Y/N	Description
row	Long	Optional	Row index of the row

- **Example**

```
// Check drag permission for a row

var drag = mySheet.GetRowDraggable(3);
```

- **Syntax**

Syntax	Set	ObjId.**SetRowDraggable**(row, drag)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
row	Long	Optional	Row index of the row
drag	Boolean	Required	Drag permission configuration value (Default=1)

- **Example**

```
// Allow dragging for Row 3 (Configure whether to allow mouse dragging on a row)

mySheet.SetRowDraggable(3, 1); // Drag is allowed

mySheet.SetRowDraggable(3, 0); // Drag is disallowed
```

RowEditable Method

- **Purpose**

Check or configure editable status of a particular row.

You can change the editable status of a row only when all cell's editability status is editable.

If ColEditable value is Not Allowed, RowEditable configuration will be ignored.

Whether a row is editable will be determined as follows:

Editable	ColEditable	**RowEditable**	CellEditable	**Cell editable Y/N**

No	No impact	**No impact**	No impact	**No**
Yes	No	**No**	Yes/No	**Yes/No**
Yes	Yes	**Yes/No**	Yes/No	**Yes/No**

- **Syntax**

Syntax	Get	ObjId.**GetRowEditable**(Row)

- **Info**

Return	Boolean, editability status (in case of Get Method)		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Index of the row

- **Example**

```
//Check editability status of Row 1.

mySheet.GetRowEditable(1,0);
```

- **Syntax**

Syntax	Set	ObjId.**SetRowEditable**(Row, Editable)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Index of the row
Editable	Boolean	Required	Row editable Y/N (Use only for Set)

- **Example**

```
//Set editability status of Row 1 as 0.

mySheet.SetRowEditable(1,0);
```

RowExpanded Method

- **Purpose**

  In tree type sheet, this method checks whether child levels of a row is expanded, or configures to expand child level.

- **Syntax**

Syntax	Get	ObjId.**GetRowExpanded**(Row)

- **Info**

Return	Boolean, whether child rows are expanded (in case of Get Method)		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row Index

- **Example**

```
//Check whether child rows of Row 2 are expanded, and if not, expand them.

if(mySheet.GetRowExpanded(2) == 0){

    mySheet.SetRowExpanded(2, 1);

}
```

- **Syntax**

Syntax	Set	ObjId.**SetRowExpanded**(Row, Expand)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row Index
Expand	Boolean	Required	Whether child rows are expanded (Set)

- **Example**

```
//Check whether child rows of Row 2 are expanded, and if not, expand them.

if(mySheet.GetRowExpanded(2) == 0){

    mySheet.SetRowExpanded(2, 1);

}
```

RowFontColor Method

- **Purpose**

  Check or configure font color of the entire row.

- **Syntax**

Syntax	Get	ObjId.**GetRowFontColor**(Row)

- **Info**

Return	String, set color (in case of Get Method)		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row Index

- **Example**

```
// Check font color of Row 1.

mySheet.GetRowFontColor(1);
```

- **Syntax**

Syntax	Set	ObjId.**SetRowFontColor**(Row,Color)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row Index
Color	String	Required	WebColor color value

- **Example**

```
//Set the row font color as gray for Row 1.

mySheet.SetRowFontColor(1, "192,192,192");



//Set the Row 2 font color as the same color as Row 1 font.

mySheet.SetRowFontColor(2,mySheet.GetRowFontColor(1));
```

RowHeight Method

- **Purpose**

  Check or configure height of a particular row.

  You can set row height by pixels. If the row does not exist, it will not return any error message and just cancels the setting.

- **Syntax**

Syntax	Get	ObjId.**GetRowHeight**(Row)

- **Info**

Return	Integer, current row height (In case of Get Method)		
Parameter	Type	Required	Description

		Y/N	
Row	Long	Required	Row index to set height for

- **Example**

```
//Check the Row 1 height.

mySheet.GetRowHeight(1);
```

- **Syntax**

Syntax	Set	ObjId.**SetRowHeight**(Row, Height)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row index to set height for
Height	Integer	Required	Row height to set

- **Example**

```
//Adjust the height to 50 pixels

mySheet.SetRowHeight(1, 50);



//Set the height of row 3 as the same height as Row 2

mySheet.SetRowHeight(3, mySheet.GetRowHeight(2));
```

RowHeightMax Method

- **Purpose**

Check or configure row height of all data rows.

This method will apply only when data row height is set as automatic. If the value is set smaller than DataRowHeight, the latter will apply.

This method is used when you want to set row height as automatic but do not want row height to be bigger than a certain level.

- **Syntax**

Syntax	Get	ObjId.**GetRowHeightMax**()

- **Info**

Return	Integer, row max height (In case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Check the set min height value.
mySheet.GetRowHeightMax();
```

- **Syntax**

Syntax	Set	ObjId.**SetRowHeightMax**(MaxHeight)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
MaxHeight	Integer	Required	Max height value to set for the row

- **Example**

```
//Set the minimum height at 50 pixels.

mySheet.SetRowHeightMax(50);
```

RowHeightMin Method

- **Purpose**
  Check or configure minimum height of all rows. Minimum neight cannot be 5 pixels or smaller, so choose a value bigger than 5 pixels. The default is set as 21 pixels.

- **Syntax**

Syntax	Get	ObjId.**GetRowHeightMin**()

- **Info**

Return	Integer, row min height (In case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Check the set min height value.

mySheet.GetRowHeightMin();
```

- **Syntax**

Syntax	Set	ObjId.**SetRowHeightMin**(MinHeight)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description

MinHeight	Integer	Required	Min height value to set for the row

- **Example**

```
//Set the minimum height at 10 pixels.

mySheet.SetRowHeightMin(10);
```

RowHidden Method

- **Purpose**

  Check or configure hiding status of rows.

  You can configure hiding status of multiple rows by adjoining row indexes with '|'. (However, Read function is not supported for multiple rows.)

- **Syntax**

Syntax	Get	ObjId.**GetRowHidden**(Row)

- **Info**

Return	Boolean, set value (If Get: 1, hidden; if 0, unhidden)		
Parameter	Type	Required Y/N	Description
Row	Long /String	Required	Row index of a particular row or string of row indexes adjoined by "|" (Use only for Set)

- **Example**

```
//Check hiding status of Row 1 and unhide if hidden.

if(mySheet.GetRowHidden(1)){

  mySheet.SetRowHidden(1,0);

}
```

➢ **Syntax**

Syntax	Set	ObjId.**SetRowHidden**(Row,Hidden)

➢ **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Row	Long /String	Required	Row index of a particular row or string of row indexes adjoined by "|" (Use only for Set)
Hidden	Boolean	Required	Hidden or not

➢ **Example**

```
//Check hiding status of Row 1 and unhide if hidden.
if(mySheet.GetRowHidden(1)){
  mySheet.SetRowHidden(1,0);
}
//Hide multiple rows simultaneously
mySheet.SetRowHidden("2|5|7|10", 1);
```

RowJson Method

➢ **Purpose**

Create and return a JSON object with row data using SaveName of each column, or change configuration.

For unit data row structure of two or more rows, the entire unit data rows are returned or configured.

- **Syntax**

Syntax	ObjId.**GetRowJson**(Row)

- **Info**

Return	Object, data object of the selected rows		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Index of the row

- **Example**

```
// Return JSON object of Row 1.

var rowJosn = mySheet.GetRowJson(1);
```

- **Syntax**

Syntax	ObjId.**SetRowJson**(Row, Data)

- **Info**

Return	Object, data object of the selected rows		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Index of the row
Data	Object	Required	Search JSON string

- **Example**

```
// Configure Row 1 as the selected JSON object.

var data = {Data:[[ {sName:"Gildong Hong", sAge:20} ]]};

var rowJosn = mySheet.SetRowJson(1, data);
```

RowLevel Method

- **Purpose**

  Check or configure tree levels of a row.

  When there is a tree sturucture, this method will Check or configure the tree level of row. Tree levels will start from 0.

- **Syntax**

Syntax	Get	ObjId.**GetRowLevel**(Row)

- **Info**

Return	Integer, tree level of the current row (In case of Get Method)		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Index of the row

- **Example**

```
// Check for tree level of Row 1

mySheet.GetRowLevel(1);
```

- **Syntax**

Syntax	Set	ObjId.**SetRowLevel**(Row,Level)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Index of the row
Level	Integer	Required	Tree level value of a row (Set)

- **Example**

```
//Move a row to upper tree level by 1 level

mySheet.SetRowLevel(1,mySheet.GetRowLevel(1) +1);
```

RowMerge Method

- **Purpose**

  Check or configure horizontal merge status of a row.

  Horizontal merge is allowed only when Merge All is allowed and the selected data row exists.

- **Syntax**

Syntax	Get	ObjId.**GetRowMerge**(Row)

- **Info**

Return	Boolean, horizontal merge permission of the current row (In case of Get Method)		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Index of the row

- **Example**

```
// Check for horizontal merge permission of Row 1.

mySheet.GetRowMerge(1);
```

- **Syntax**

Syntax	Set	ObjId.**SetRowMerge**(Row, Merge)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Index of the row
Merge	Boolean	Required	Whether horizontal merge is allowed

➢ **Example**

```
// Allow horizontal merge for Row 1.

mySheet.SetRowMerge(1, 1);
```

RowSaveStr Method

➢ **Purpose**

This method will combine and return row data using column SaveName in a query string format which is used for saving.

SaveName1=Value1&SaveName2=Value2&...

The return string will be in the above format. Korean characters will be UrlEncoded before being returned.

➢ **Syntax**

Syntax	ObjId.**RowSaveStr**(Row)

➢ **Info**

Return	String, row data		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Index of the row

➢ **Example**

```
// Fetch SaveString of Row 1

var RowParam = mySheet.RowSaveStr(1);
```

RowTop Method

- **Purpose**

  Check offsetTop value of a row.

  If Row parameter setting is not correct, return -1 without any further processing.

  The reference point will be the topmost location of the Sheet container (table). (Reference point: 0)

  Depending on the location of vertical scroll, the return value may vary for the same row.

- **Syntax**

Syntax	ObjId.**RowTop**(Row)

- **Info**

Type	Long, top location of a row	
Parameter	Type	Description
Row	Long	Row Index

- **Example**

```
//Find out the top location of a row.

var iTop = mySheet.RowTop(1);
```

SaveNameCol Method

- **Purpose**

  Check column index using SaveName set in InitColumns. If there is no matching column for the SaveName, -1 is returned.

- **Syntax**

Syntax	ObjId.**SaveNameCol**(SaveName)

- **Info**

Return	Long, column index		
Parameter	Type	Required Y/N	Description
SaveName	String	Required	Saved parameter name

- **Example**

```
// Fetch column index using SaveName.

var Col = mySheet.SaveNameCol("stockNm");
```

SavingImage Method

- **Purpose**

  Check or configure the location of image file of saving waiting image.

  This method can be used to customize waiting image as the user want which displays during saving.

- **Syntax**

Syntax	Get	ObjId.**GetSavingImage**()

- **Info**

Return	String, currently set value (In case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Check the currently set image path of saving waiting image.

alert(mySheet.GetSavingImage());
```

- **Syntax**

Syntax	Set	ObjId.**SetSavingImage**(Url)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Url	String	Required	Image URL

- **Example**

```
//Change the search waiting image.

mySheet.SetSavingImage( "/sheet/imgSave.gif");
```

SearchingImage Method

- **Purpose**

  Check or configure the location of image file of search waiting image.

  This method can be used to customize waiting image as the user want which displays during saving.

- **Syntax**

Syntax	Get	ObjId.**GetSearchingImage**()

- **Info**

Return	String, currently set value (In case of Get Method)

Parameter	Type	Required Y/N	Description

➢ **Example**

```
//Check the currently set image path of search waiting image.

alert(mySheet.GetSearchingImage());
```

➢ **Syntax**

Syntax	Set	ObjId.**SetSearchingImage**(Url)

➢ **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Url	String	Required	Image URL

➢ **Example**

```
//Change the search waiting image.

mySheet.SetSearchingImage( "/sheet/imgSearch.gif");
```

SearchRows Method

➢ **Purpose**

Check the total row count returned by a search XML.

➢ **Syntax**

Syntax	ObjId.**SearchRows**()

- **Info**

Return	Long, total return row count		
Parameter	Type	Required Y/N	Description
N/A			

- **Example**

```
//Check row count

alert(mySheet.SearchRows());
```

SelectCell Method

- **Purpose**

  Selet a particular cell. If the selected cell is editable and EditMode parameter is set as 1, the cell selected will turn to Edit mode as soon as it is selected. If you do not need the cell to be Edit Mode, set EditMode parameter as 0 before calling this method.

  EditModeText parameter is used only when EditMode parameter is 1. When the cell is turned to Edit Mode, you can configure the cell text to display in EditMode Text.

- **Syntax**

Syntax	ObjId.**SelectCell**(Row, Col, [EditMode], [EditModeText])

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row Index of the cell to select
Col	Long /	Required	Column Index of the selected cell

	String		or SaveName
EditMode	Boolean	Optional	Select or deselect Edit Mode for the selected cell, Default=0 (Not in edit mode)
EditModeText	String	Optional	Set the cell text when EditMode is 1. Default=""

- **Example**

```
// Select Row 2, and create a new row below the selected row

mySheet.SelectCell(2, 0);

mySheet.DataInsert();
```

SelectCol Method

- **Purpose**

  Check or configure the column index of the currently selected cell.

  Use this property along with SelectRow property, or you may use SelectCell method which uses both properties.

- **Syntax**

Syntax	Get	ObjId.**GetSelectCol**()

- **Info**

Return	Long, currently selected column index (In case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Check the selected column.

alert(mySheet.GetSelectCol());
```

- **Syntax**

Syntax	Set	ObjId.**SetSelectCol**(Col)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Col	Long/String	Required	Column index of the currently selected cell or SaveName

- **Example**

```
//Select the column whose column index is 2.

mySheet.SetSelectCol(2);
```

SelectionMode Method

- **Purpose**

Check or configure cell selection mode. Configurable options are as follows:

Type	Description
0	Select by cells
1	Select by rows
3	Select multiple rows from different locations using Ctrl key Selected row index can be checked using GetSelectionRows() method

- **Syntax**

Syntax	Get	ObjId.**GetSelectionMode**()

- **Info**

Return	Integer,current set value (in case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Check the current selection mode.

alert(mySheet.GetSelectionMode());
```

- **Syntax**

Syntax	Set	ObjId.**SetSelectionMode**(Mode)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Mode	Integer	Required	Selection mode type (0: Select by cell, 1: select by row)

- **Example**

```
//Configure select by cell.

mySheet.SetSelectionMode(0);
```

SelectRow Method

- **Purpose**

  Check or configure row index of the currently selected cell.

  Use this property along with SelectCol property, or you may use SelectCell method which uses both properties.

- **Syntax**

Syntax	Get	ObjId.**GetSelectRow**()

- **Info**

Return	Long, currently selected row index (In case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Check the selected row.

alert(mySheet.GetSelectRow());
```

- **Syntax**

Syntax	Set	ObjId.**SetSelectRow**(Row)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Index of the row to select

- **Example**

```
//Select the row whose row index is 2

mySheet.SetSelectRow(2);
```

SendComboData Method

- **Purpose**

  Check or configure the data transmitted to server when data type is Combo.

  When data of a column is to be transmitted to server using Saving method, usually the code configured in Combo is transmitted. However by using this method you may transmit text as displayed instead of code, or transmit both the code and text.

- **Syntax**

Syntax	Get	ObjId.**GetSendComboData**(DataRow, Col)

- **Info**

Return	String, transmission data format of the currently selected combo column (In case of Get Method)		
Parameter	Type	Required Y/N	Description
DataRow	Long	Required	Unit data row index
Col	Long / String	Required	Column index or SaveName of a particular column

- **Example**

```
//Check the combo transmission format of Column 3

mySheet.GetSendComboData(0,3);
```

- **Syntax**

Syntax	Set	ObjId.**SetSendComboData**(DataRow, Col, Type)

➢ **Info**

Return	String, transmission data format of the currently selected combo column (In case of Get Method)		
Parameter	Type	Required Y/N	Description
DataRow	Long	Required	Unit data row index
Col	Long / String	Required	Column index or SaveName of a particular column
Type	String	Optional	Server transmission option configuration ("Text" or "code"), Default="code"

➢ **Example**

```
  //Configure  the combo transmission format of Column 3 as both Code
and text.

  mySheet.SetSendComboData(0,3,"Code|Text");



  //Configure  the combo transmission format of Dept column as text
instead of code.

  mySheet.SendComboData(0,"Dept","Text");
```

SetBlur Method

➢ **Purpose**

Remove focus from the sheet

➢ **Syntax**

Syntax	ObjId.**SetBlur**()

➢ **Info**

Return	None		
Parameter	Type	Required Y/N	Description
N/A			

- **Example**

```
// Remove focus from the sheet.

mySheet.SetBlur();
```

SetCellImageStyle Method

- **Purpose**

Configure image when a cell uses Image property.

The following properties may be configured using this method.

Property	Type	Description
Image	Integer	Image path
ImgAlign	String	Image display location
ImgWidth	Integer	Width of the image
ImgHeight	Integer	Height of the image

- **Syntax**

Syntax	ObjId.**SetCellImageStyle**(Row, Col, Style);

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Index of relevant rows

Col	Logn/String	Required	Index or SaveName of the target column
Style	Object	Required	Cell's image related property object

➢ **Example**

```
//Display image of the third row, third column in the left side

var style = {Image: "myImage.gif", ImgHeight:20, ImgAlign:"Left"};

mySheet.SetCellImageStyle(3, 3, style);



// Display the first image of the Image List in Row 3, Column 3 in the right side.

mySheet.SetImageList(0, "image0.gif");

mySheet.SetImageList(1. "image1.gif");

var style = {Image: 1, ImgAlign:"Right"};

mySheet.SetCellImageStyle(3, 3, style);
```

SetColProperty Method

➢ **Purpose**

Use this method when you want to define a property of a particular column dynamically, after the property is set in InitColumns Method.

Properties that may be reconfigured using this method are as follows:

Property	Type	Description
AcceptKeys	String	Accepted key configuraiton
Align	String	Data alignment
AllowNull	Boolean	Whether to allow empty value for columns of number-like status

ApproximateType	Number	Approximation type (1: Rounding, 2: Rounding down, 3: Rounding up)
ButtonUrl	String/Number	Popup button's image path or image list index
ComboCode	String	Combo list code group
ComboText	String	Combo list text string group
CustomDate	Number	Custom date availability
DefaultValue	String	Default applies for new entries
Edit	Boolean	Editability
EditLen	Number	Editable data legnth
FalseValue	String	Set false value for checkbox colums whose false value is not 0. If set "F", you can use "F" as False value instead of 0.
Format	String	Data mask application format
FormatFix	Boolean	Whether to return GetCellText value upon GetCellValue. If set as true, the format will be also saved upon saving.
HoverUnderline	Boolean	Whether to underline when mouse is hovering over
Image	String	Image representation URL
ImgHeight	Number	Image height
ImgWidth	Number	Image width
InputCaseSensitive	Number	Whether to use particular case automatically for alphabet inputs 0: No configuration (Default) 1: Replace with upper case 2: Replace with lower case

KeyField	Boolean	Required fields
MinWidth	Number	Minimum column width
MultiLineText	Boolean	Whether to use multilines
PointCount	Number	Decimal place count when column type is float.
RadioIcon	Boolean	Select a checkbox-type button image within data cell
ShowCol	Number	Set the column to display as column value when multi combo is set
TrueValue	String	Set true value for checkbox colums whose true value is not 1. If set "M", "M" can be used as True value instead of 1.
Width	Number	Column width

For detailed description of each property, see InitColumns Method desription.

- **Syntax**

Syntax	ObjId.**SetColProperty**(DataRow, Col, Prop);

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
DataRow	Number	Required	Unit data row index
Col	Number / String	Required	Index or SaveName of the target column
Prop	Object	Required	Column property definition object

- **Example**

```
// Change the combo list of the third column

var info = {ComboText: "New|In Progress|Completed ", ComboCode: " 01|02|03" };

mySheet.SetColProperty(0, 3, info);



// Change the combo list of the 1st and 3rd column of the unit data rows

var info = {ComboText: "New|In Progress|Completed ", ComboCode: " 01|02|03" };

mySheet.SetColProperty(1, 3, info);
```

SetConfig Method

- **Purpose**

  In this method, you may configure how to fetch initialized sheet, location of frozen rows or columns and other basic configurations.

- **Syntax**

Syntax	ObjId.**SetConfig**([cfg])

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
cfg	json	Optional	The parameters to send are in JSON type. Put configuration information into JSON format and send it to the server.

< 세부내용>

Parameters	Type	Description

AutoCloseDialog	Boolean	Configure whether to automatically close Combo list, calendar pop-up, action menu popup. Default=0. When you set to close automatically, those pop-ups or list will close automatically when mouse moves out of cell.
ChildPage	Integer	In tree structure, page unit count of child rows (Default=0)
ComboMaxHeight	Integer	Combo list max height
DataRowMerge	Boolean	Whether to allow horizontal merge of the entire row (Default=0)
DragCell	Boolean	Whether to allow cell dragging Default=0.
DragMode	Integer	Drag effect (Default=0)
FrozenCol	Integer	Number of frozne columns in the left (Default=0)
FrozenColRight	Integer	Number of frozne columns in the right (Default=0)
HeaderCheckMode	Boolean	When Check All is checked whether to check filtered rows Default=0.
HeaderMergeMode	Boolean	Configure header merge mode Default=0.
KeyFieldPosition	String	Configure where to put required field marker (Default="Left")
MaxSort	Integer	Set the max number of columns to sort by clicking on the header (Default=3)
MergeSheet	Integer	Merge type (Default=0)

NewRowDeleteMode	Boolean	Whether to display confirmation window when a new row is deleted. Default=0.
NextPageCall	Integer	Configure when the next page is called in case the search mode is server paging, LazyLoad. The value should be set at a percentage value between 60-90.
Page	Integer	Number of rows to display in one page (Default=20)
PrevColumnMergeMode	Integer	Configure how to merge previous column in LazyLoad (0: entire page, 1: within one page (Default)
SearchMode	Integer	Configure search mode (Default: 2)
SizeMode	Integer	Configure sizing mode Default=0.
SmartResize	Boolean	Whether to use SmartResize (Default:0)
SortEventMode	Boolean	Whether to return information of all sorted columns when OnSort event fires. Default=0.
SumPosition	Boolean	Location of sum row (1: Frozen at the bottom (Default), 0: Frozen at the top)
ToolTip	Boolean	Display tooltip in a cell Default=0.
TouchScrolling	Integer	Configure touch scroll mode (0: Not use, 1: General use (Default), 2: Delayed movement)
UseHeaderActionMenu	Boolean	Configure whether to use header context menu If header menu is set as "use" and

		header menu is not configured, default menu will automatically display. (See HeaderActionMenu) 0: Not use-Default, 1: use
UseNoDataRow	Integer	Whether to display "There is no search result" message when there is no search result (0: Not use, 1: use (Default))
VScrollMode	Integer	Configure verical scroll bar display (0: Auto (Default), 1: Frozen)

ChildPage configures LazyLoad paging count unit for child rows in a stree structure. This applies only when a value is set.

(To use this property, SearchMode should be set smLazyMode.)

ComboMaxHeight is the maximum height of combo list of combo type. When set 0, the height will be automatically configured to fit the browser height. Height can be set by pixel.

DataRowMerge can be used to configure whether to allow horizontal merge of the entire row. The default value is 0.

DragMode is where you can configure effect of dragging. The default value is 0.

Value	Details
-1	- Not use drag
0 (Default)	- General: select a range of cells or rows - Use Ctrl key: Drag rows
1	- General: Drag rows

	- Use Ctrl key: select a range of cells or rows

FrozenCol is where you can select the frozen column count in the left. The default value is 0.

FrozenColRight is where you can select the frozen column count in the right. The default value is 0.

HeaderMergeMode is where you can configure horizontal merge mode of header. When set 0, merge will be allowed regardless of the ColMerge setting. When set 1, merge permission will follow ColMerge property setting.

KeyFieldPosition is where you can configure where to put required field marker. The default is Left.

Value	Description
Left	Left to the header text
Right	Right to the header text

MergeSheet is where you can configure merge styles. The default value is msNone.

Constant value	Value	Description
0	msNone	No merge is allowed; Default
1	msAll	Allow merge for all
2	msPrevColumnMerge	Merge is allowed for rows where they are merged in the previous column.
5	msHeaderOnly	Allow merge in the header rows only
7	msPrevColumnMerge + msHeaderOnly	Merge is allowed for rows where they are merged in the previous column + allow merge in the header

Page is where you may configure paging unit size for paging mode, LazyLoad mode or real-time server processing mode. The default value is 20.

SearchMode is where you can configure search mode by selecting one from General, Paging, LazyLoad or real-time server processing modes. The default value is 0.

Constant value	Value	Description
0	smGeneral	General search mode Search all data and display all result data on the screen.
1	smClientPaging	Paging mode Search all data and page the result according to Page property configuration.
2	smLazyLoad	LazyLoad mode Search all data and display search result data on the screen by page as set in Page property value according to the scroll location
3	smServerPaging	Real-time server processing mode Receive only partial search results corresponding to the scroll location from the server and display them on the screen. Data not displaying on the screen will be lost.

SizeMode is where you can configure how to handle size. You may apply the sizes configured when creating IBSheet object to consigure this property. The default value is sizeAuto.

Constant value	Value	Description

0	sizeAuto	Use the height and width as configured (Default)
1	sizeNoVScroll	Automatically adjust height to remove scroll Height value configured upon creation will be ignored.
2	sizeNoHScroll	Automatically adjust width to remove scroll Height value configured upon creation will be ignored.
3	sizeNoBothScroll	Automatically adjust width and height to remove scrolls Width and height value configured upon creation will be ignored.

SmartResize is an improvement to Resize. When this property is set as 1 (true), Sheetdml Resizing will be done and OnResize event will fire if there is no recurrence for a certain span of time (300ms) from the initial occurrence. This property may come in handy if you have any resizing performance issue for a sheet with a large amount of data.

SumPosition is where you can configure location of the sum row. The default value is posBottom.

Constant value	Value	Description
1	posBottom	Fix the sum row to the bottom row.
0	posTop	Fix the sum row to the top row.

ToolTip is where you can configure whether to display tooltip for a cell. The default value is 0. If you change the setting to 1, tips will display in a balloon when mouse hovers over the cell.

- **Example**

```
//Paging mode search sample

cfg= {SearchMode:1, Page:20};

mySheet.SetConfig(cfg);
```

SetDown2ExcelConfig Method

- **Purpose**

  In this method, you may configure some basic settings for excel download.

- **Syntax**

Syntax	ObjId.**SetDown2ExcelConfig**([cfg])

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
cfg	json	Optional	The parameters to send are in JSON type. Put configuration information into JSON format and send it to the server.

- **Details**

Parameter	Type	Required Y/N	Description
FileName	String	Optional	File name to save Default="Excel.xls"
SheetName	String	Optional	Excel worksheet name, Default="Sheet"

DownRows	String	Optional	Connect rows to download using | Default=""(Download all)
DownCols	String	Optional	Connect columns to download using | Default=""(Download all)
DownHeader	bool	Optional	Whether to download header Default=1
DownSum	bool	Optional	Whether to download sums Default=1
Merge	bool	Optional	Whether to apply merge to download Default=0
SheetDesign	bool	Optional	Whether to apply IBSheet design concept to download file Default=0
TitleText	String	Optional	Default=""(Not use)
UserMerge	String	Optional	Default=""(Not use)
OnlyHeaderMerge	bool	Optional	Whether to merge the header only Default=0

- **Example**

```
// Default setting

cfg= {FileName:"DownExcel.xls", Merge:1};

mySheet.SetDown2ExcelConfig(cfg);
mySheet.Down2Excel();
```

SetEndEdit Method

- **Purpose**

  End cell editing. If you want to save edited contents before exiting edit mode, set Save parameter as 1 (true). If you want to cancel editing before saving, set the parameter as 0 (false).

- **Syntax**

Syntax	ObjId.**SetEndEdit**(Save)

- **Info**

Return	Boolean, whether ending has been successfully done		
Parameter	Type	Required Y/N	Description
Save	Boolean	Required	Whether to save edited contents

- **Example**

```
// Method to process saving related logics

var result = mySheet.SetEndEdit(1);



if (!result) {

      return;

}



// Proceed with saving processing logics
```

SetFilterOption Method

- **Purpose**

  Use this method to configure column filtering options when filter rows are used

- **Syntax**

Syntax	ObjId.**SetFilterOption**(Col, Option)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Col	Long	Required	Column index or SaveName
Option	Integer	Required	Option value to set

- **Option Value**

0	**Do not use**		
1	**Equal**	**2**	**Not equal**
3	**Smaller**	**4**	**equal or smaller**
5	**Bigger**	**6**	**equal or bigger**
7	**Start with word**	**8**	**Not start with word**
9	**End with word**	**10**	**Not end with word**
11	**Include**	**12**	**Not include**

- **Example**

```
// When filter row is Row 1, configure filtering value for column 2

mySheet.SetCellValue(1, 2,  “Include”);
```

```
// Configure filter option for filter row column 2  – Filter any words that are
the same as string 'Include'

mySheet.SetFilterOption (2, 1);
```

SetFilterValue Method

- **Purpose**

  Use this method when setting filter value when filter row is used,

- **Syntax**

Syntax	ObjId.**SetFilterValue**(Col, Value, Option)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Col	Long	Required	Column index or SaveName
Value	String	Required	Value to set in filter
Option	Integer	Optional	Option value to set

- **Option Value**

0	**Do not use**		
1	**Equal**	**2**	**Not equal**
3	**Smaller**	**4**	**equal or smaller**
5	**Bigger**	**6**	**equal or bigger**
7	**Start with word**	**8**	**Not start with word**
9	**End with word**	**10**	**Not end with word**
11	**Include**	**12**	**Not include**

➢ **Example**

```
// Filter strings that include "Seoul" in column 5

mySheet.SetFilterValue(5,  “Seoul”, 11);
```

SetFindDialog Method

➢ **Purpose**

Configure default setting for search dialogue using Ctrl+Shift+F keys.

In FullMatch parameter, you can configure what types of items you may search. FindTextMethod is the same as FullMatch configuration. The available values are as follows:

FullMatch value	Purpose
-1	Find rows that are identical to SearchText
0	Find rows that have matching opening with SearchText
1	Find rows that have matching ending with SearchText
2	Find rows that have matching center with SearchText

➢ **Syntax**

Syntax	ObjId.**SetFindDialog**(Show,[Col],[FullMatch],[FirstStart],[CaseSensitive],[KeepDialog])

➢ **Info**

Return	None.		
Parameter	Type	Required Y/N	Description
Show	Boolean	Required	Whether to use Search dialogue
Col	Long/String	Optional	Index or SaveName of the search

			target columns If empty, all columns, Default=""
FullMatch	Integer	Optional	Configure text match option, Default=-1
FirstStart	Boolean	Optional	Configure start location, Default=1
CaseSensitive	Boolean	Optional	Configure case sensitivity, Default=0
KeepDialog	Boolean	Optional	Configure whether to close dialog after search, Default=0

➢ **Example**

```
//Use search dialog, target column: 5, start location: next row after the focus row, match option: center match)

mySheet.SetFindDialog(1, 5, 2, 0);
```

SetFocus Method

➢ **Purpose**

Configure focus on sheet

➢ **Syntax**

Syntax	ObjId.**SetFocus**()

➢ **Info**

Return	None		
Parameter	Type	Required Y/N	Description
N/A			

- **Example**

```
// Configure focus on sheet.

mySheet.SetFocus();
```

SetMergeCell Method

- **Purpose**

Force merge on a particular cell with adjacent cells

This method must be used for search screen.

In general, cells can be automatically merged only when the value of the adjacent cell is identical to the merging cell. However, if you call this method, merge will be forced to a particular cell with all the cells within the set size indiscriminately (even though cell values are different).

- **Syntax**

Syntax	ObjId.**SetMergeCell**(Row, Col, Rows, Cols)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row Index of the cell to force merge
Col	Long	Required	Column Index of the cell to force merge
Rows	Long	Required	Row count of the cells to force merge
Cols	Long	Required	Column count of the cells to force merge

- **Example**

```
// Merge and display 2X2 window cells from (1, 10) to (2, 11).

mySheet.SetMergeCell(1, 10, 2, 2);
```

SetRowHaveChildValue Method

- **Purpose**

  In this method, you can change HaveChild property setting for a stree structure.

  When HaveChild property is configured, lower node expansion icon is set depending on the configuration. Using this method, you may change the HaveChild property value.

- **Syntax**

Syntax	ObjId.**SetRowHaveChildValue**(Row, HaveChild)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Index of relevant rows
HaveChild	Boolean	Required	Property setting

- **Example**

```
// Change HaveChild configuration of Row 3 to 0.

mySheet.SetRowHaveChildValue(3, 0);
```

SheetWidth Method

- **Purpose**

  Check or configure the overall width. Set the value in pixel count.

- **Syntax**

Syntax	Get	ObjId.**GetSheetWidth**()

- **Info**

Return	Integer, width pixel counts (in case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
// Currently configured width value.

alert(mySheet.GetSheetWidth());
```

- **Syntax**

Syntax	Set	ObjId.**SetSheetWidth**(Width)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Width	Integer	Required	Sheet width pixel count to set

- **Example**

```
// Set the total width at 800 pixels

mySheet.SetSheetWidth(800);
```

SheetHeight Method

- **Purpose**

Check or configure the total height. Set the value in pixel count.

- **Syntax**

Syntax	Get	ObjId.**GetSheetHeight**()

- **Info**

Return	Integer, height pixel counts (in case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
// Check currently configured height value.

alert(mySheet.GetSheetHeight());
```

- **Syntax**

Syntax	Set	ObjId.**SetSheetHeight**(Height)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Height	Integer	Required	Sheet height pixel count to set

- **Example**

```
// Set the total height at 800 pixels.

mySheet.SetSheetHeight(800);
```

ShowButtonImage Method

- **Purpose**

  Check or configure the button image styles of popup and combo.

  Upon initial loading, button image display style should be configured.

  Style setting to configure are as follows:

Type	Description
0	Display combo, calendar and pop-up image only when is focused.
1	Display calendar and popup image when the cell is editable.
2	Always display calendar and pop-up image.
3	Display combo, calendar and popup image when the cell is editable (Default)
4	Always display combo, calendar and pop-up image

- **Syntax**

Syntax	Get	ObjId.**GetShowButtonImage**()

- **Info**

Return	Integer,current set value (in case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
// Check the displayed button image style of popup and combo.

mySheet.GetShowButtonImage();
```

- **Syntax**

Syntax	Set	ObjId.**SetShowButtonImage**(type)

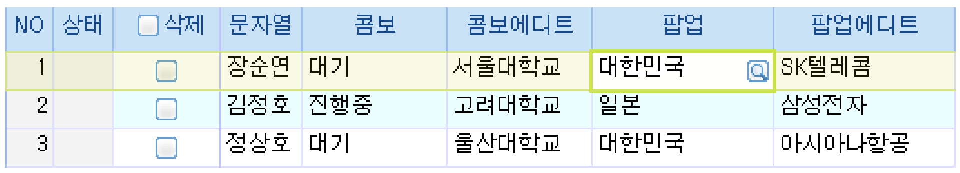
➢ **Info**

Return	None		
Parameter	Type	Required Y/N	Description
type	Integer	Required	Set value (Default=3)

➢ **Example**

// Combo and popup image will display only when focus is placed.

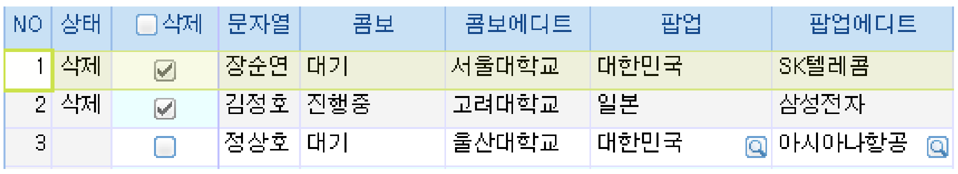
mySheet.SetShowButtonImage(0);



NO	상태	☐삭제	문자열	콤보	콤보에디트	팝업	팝업에디트
1		☐	장순연	대기	서울대학교	대한민국 🔍	SK텔레콤
2		☐	김정호	진행중	고려대학교	일본	삼성전자
3		☐	정상호	대기	울산대학교	대한민국	아시아나항공

// Popup image displays when cells are editable.

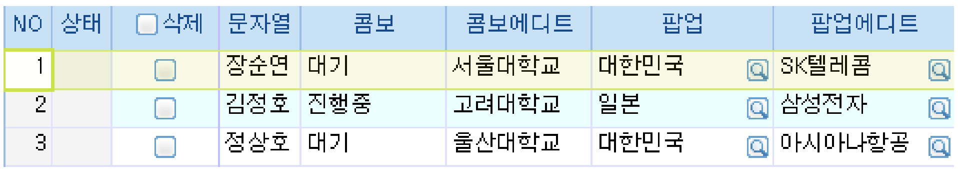
mySheet.SetShowButtonImage(1);



NO	상태	☐삭제	문자열	콤보	콤보에디트	팝업	팝업에디트
1	삭제	☑	장순연	대기	서울대학교	대한민국	SK텔레콤
2	삭제	☑	김정호	진행중	고려대학교	일본	삼성전자
3		☐	정상호	대기	울산대학교	대한민국 🔍	아시아나항공 🔍

// Popup images are always displayed.

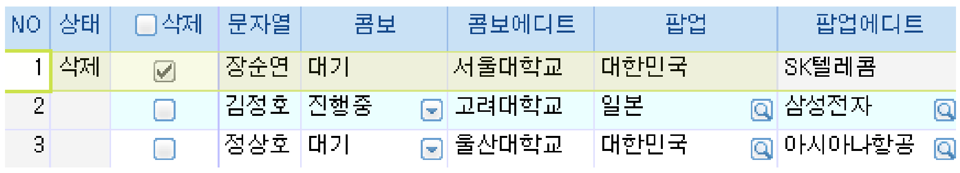
mySheet.SetShowButtonImage(2);



NO	상태	☐삭제	문자열	콤보	콤보에디트	팝업	팝업에디트
1		☐	장순연	대기	서울대학교	대한민국 🔍	SK텔레콤 🔍
2		☐	김정호	진행중	고려대학교	일본 🔍	삼성전자 🔍
3		☐	정상호	대기	울산대학교	대한민국 🔍	아시아나항공 🔍

// Combo and popup images are displayed when cells are editable.

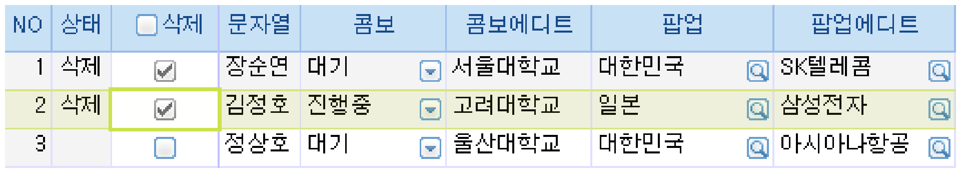
```
mySheet.SetShowButtonImage(3);
```



NO	상태	☐ 삭제	문자열	콤보	콤보에디트	팝업	팝업에디트
1	삭제	✓	장순연	대기	서울대학교	대한민국	SK텔레콤
2		☐	김정호	진행중	고려대학교	일본	삼성전자
3		☐	정상호	대기	울산대학교	대한민국	아시아나항공

```
// Combo and popup images are always displayed.

mySheet.SetShowButtonImage(4);
```



NO	상태	☐ 삭제	문자열	콤보	콤보에디트	팝업	팝업에디트
1	삭제	✓	장순연	대기	서울대학교	대한민국	SK텔레콤
2	삭제	✓	김정호	진행중	고려대학교	일본	삼성전자
3		☐	정상호	대기	울산대학교	대한민국	아시아나항공

ShowCalendar Method

- **Purpose**

  When date format is set in Text type column, calendar popup will display upon clicking on the column if you run ShowCalendar() in OnClick event.

  If you want to enter data of '0000-00-00' or '2013-00-00' format in date data type, you may use OnClick or OnPopupClick when CustomDate parameter is set as 1 for Text, Popup or PopupEdit columns.

  When you use CustomDate, you need to define user custom format. You may choose from Year-month-date(####-##-##, ####/##/## etc.), Year-month (####-##, ####/## etc.), or Month-date (##-##, ##/## etc.) types.

  The format will be determined by the number ouf #: Year-month-date (8), Year-month(6) or Month-date (4).

  Only 4 digit format is supported for year format, and the display order should be year-month-date.

  If there is a date data value, the date value displays as selected, If there is no data or if the data is not in date format, today's date displays as selected.

- **Syntax**

Syntax	ObjId. **ShowCalendar**()

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
N/A			

- **Example**

```
// Display calendar pop-up when text-type column is clicked

function mySheet_OnClick(Row, Col) {

            mySheet.ShowCalendar();

}



// Display calendar popup when OnPopup button is clicked in Popup or
PopupEdit column

function mySheet_OnPopupClick(Row,Col) {

            mySheet.ShowCalendar();

}
```

ShowDebugMsg Method

- **Purpose**

  Check or configure whether to display debugging message

  If you set to display debugging message, debugging message will display as a system popup message. If you set not to display the message, OnDebugMsg event fires so that the user can check the message as an event parameter.

  Setting options are as follows:

Type	Description
-1	Start system popup debugging
0	End all debugging

- **Syntax**

Syntax	ObjId.**ShowDebugMsg**(Msg)

- **Info**

Return	Boolean, 디버깅용 메시지 표시 여부		
Parameter	Type	Required Y/N	Description
Msg	Integer	Required	Whether to display debugging message Default=0.

- **Example**

```
//Show debugging message for saving process as a pop-up

mySheet.ShowDebugMsg(-1);



//Search

mySheet.DoSearch("list.xml");
```

ShowColumnPopup Method

- **Purpose**

This method is used to force open a pop-up in a column when column pop-up function is configured using InitColumns method.

When MousePos is set as 1, the popup will display at (X, Y) coordinates where the mouse was clicked for the last time. If MousePos is set as 0, popup will display below the (Row, Col) cell location.

- **Syntax**

Syntax	ObjId.**ShowColumnPopup**(Row, Col, [MousePos])

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row Index of the cell to select
Col	Long / String	Required	Column Index of the selected cell or SaveName
MousePos	Boolean	Optional	Whether to use the last mouse location Default=1 (Use the last location)

- **Example**

```
//Open a popup at the current cell location

mySheet.ShowColumnPopup(1, 4, 0)



//Open a popup at the last mouse click location

mySheet.ShowColumnPopup(1, 4,1)
```

ShowFilterRow Method

- **Purpose**

Add filter row as a frozen row at the top of IBSheet

Using this method will allow you to use filtering for column data.

You need to call this method after initialization before search processing.

- **Syntax**

Syntax	ObjId.**ShowFilterRow**()

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
N/A			

- **Example**

```
//Add a filter row

mySheet.ShowFilterRow();
```

ShowFindDialog Method

- **Purpose**

  Display search dialog when the user presses Ctrl + Shift + F on IBSheet.

- **Syntax**

Syntax	ObjId.**ShowFindDialog**()

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
N/A			

- **Example**

```
//Display search dialog.

mySheet.ShowFindDialog();
```

ShowGroupRow Method

- **Purpose**

  Add a group row at the top of IBSheet header.

  This method will create a group row at the top of header where columns can be dragged and dropped. By dragging and dropping columns to group, you can group data based on the column.

  You need to call this method after initialization before search processing.

- **Syntax**

Syntax	ObjId.**ShowGroupRow**([Cols])

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Cols	Long/ String	Optional	String of column index or SaveName to group, adjoined by "|"

- **Example**

```
//Add a group row

mySheet.ShowGroupRow();



//Add a group row for column index 1 and 3

mySheet.ShowGroupRow("1|3");
```

```
//Add a group row for column SaveName sName and sDept

mySheet.ShowGroupRow("sName|sDept");
```

ShowMsgMode Method

- **Purpose**

  Configure whether to use system pop-up or event for various IBSheet messages.

  If this property value is 1, all messages will be system popup. If 0, OnMessage event will fire. If you need to apply a design concept to message dialog or adjust buttons, set this property as 0, and use OnMessage event and ConfirmOK method together.

- **Syntax**

Syntax	Get	ObjId.**GetShowMsgMode**()

- **Info**

Return	Boolean,current set value (in case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Check the message mode.

mySheet.GetShowMsgMode();
```

- **Syntax**

Syntax	Set	ObjId.**SetShowMsgMode**(Mode)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Mode	Boolean	Required	Whether to display message

- **Example**

```
//Configure the message mode.

mySheet.SetShowMsgMode(0);



//Process OnMessage event.

function mySheet_OnMessage(Msg, Level, IsConfirm) {

    //Display message

    var win_result = window.showModalDialog(

        "sheet_message.jsp?Msg=" + Msg + "&IsConfirm=" + IsConfirm,

         'modalResult',

         'dialogWidth:200px;dialogHeight:200px;center:yes;help:no;status:no;');

    //Return the message result to the sheet.

    if(IsConfirm) mySheet.ConfirmOK( win_result);

}
```

ShowProcessDlg Method

- **Purpose**

Display waiting image at the center of IBSheet.

Depending on the parameters, you may display waiting image for search, saving, download or upload. Each waiting image will be the image configured in methods.

To close display image, use HideProcessDlg method.

Available images based on parameters are as follows:

Type	Image type	Method (Configuration/check)
Search	Search waiting image	SearchingImage
Save	Saving waiting image	SavingImage
Download	Download waiting image	DownloadingImage
Upload	Upload waiting image	UploadingImage

- **Syntax**

Syntax	ObjId.**ShowProcessDlg**(Type)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Type	String	Optional	Type of waiting image (Default:"Search")

- **Example**

```
// Display waiting image for saving.

mySheet.ShowProcessDlg("Search");
```

ShowSortArrow Method

- **Purpose**

  Check or configure whether to display sorting direction using an arrow when sorting by clicking is allowed on the header.

- **Syntax**

Syntax	Get	ObjId.**GetShowSortArrow**()

- **Info**

Return	Boolean, whether to display arrow image (in case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
// Check the setting whether sorting direction is displayed.

alert(mySheet.GetShowSortArrow());
```

- **Syntax**

Syntax	Set	ObjId.**SetShowSortArrow**(Arrow)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Arrow	Boolean	Required	Whether arrow is displayed

- **Example**

```
// Display sorting direction.

mySheet.SetShowSortArrow(1);
```

ShowSubSum Method

- **Purpose**

  Calculate sub sums and total sum for a column.

  This method should be called before data search method is called. When configuration is changed, you need to rerun a data search to apply it.

  SumCols is a string of column indexes where sub sum and total sum should be calculated, adjoined by“|”.

  If **ShowCumulate** is set as 1, sub sums are displayed and aggregate total of all sub sums are displayed in the row below. If set as 0, only sub sums are calculated and displayed.

  Total sum is calculated as follows: The first aggregate total is the same as the first sub sum. The second aggregate total is the sum of the first and second sub sums. The last aggregate total is sum of all previous sub sums.   Aggregate total is displayed only when sub sums are calculated.

  AvgCols parameter is a string of column indexes that average should be calculated, adjoined by “|”.

  CaptionText parameter can be used when you want to change different format for sub sum caption text.

  If this parameter is not set, it is set as “Sub sum : ” + reference value.

  When sub sum (aggregate total) is displayed, “%s” is used. You can use “%col” to set the format for reference value.

  For example, if you set the format as “%s = %col”, sub sum row will display sub sum = reference value, and the aggregate total row will display aggregate total = reference value. For example

  Sub sum rows will display “Sub sum : ” + reference value in the first column, and aggregate total rows will display “Aggregate total : ” + reference value.

  Set the parameter values in JSON format.

Sample) var info = {StdCol:2, SumCols:"2|3|4", ShowCumulate:1, AvgCols:"5|6"};

- **Syntax**

Syntax	ObjId.**ShowSubSum**(info)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
StdCol	Long/String	Required	Reference column index or SaveName
SumCols	String	Required	String of column indexes where sub sum should be calculated, adjoined by"|".
Sort	Boolean	Optional	Whether the reference column is sorted. To display sub sum, values in reference column must have been sorted. Default=1
ShowCumulate	Boolean	Optional	Whether to display aggregate total of sub sums Default=0
CaptionCol	Long	Optional	Column to set sub sum caption text as "Sub sum : " + reference value Default=The first unhidden column
CaptionText	String	Optional	Set sub sum caption text format Default=sub sum (aggregate total): + reference value

AvgCols	String	Optional	String of column indexes where average should be calculated, adjoined by"|". Default=""

- **Example**

```
//Calculate and display sub sums for Column 1 (volume)

var info = [

   {StdCol:1, SumCols:"2|3|4|5|6|7",   Sort:1}

];

mySheet.ShowSubSum(info);



// Display aggregate total as well

var info = [

  {StdCol:1, SumCols:"2|3|4|5|6|7", ShowCumulate:1,   Sort:1}

];

mySheet.ShowSubSum(info);
```

ShowToolTip Method

- **Purpose**

  Configure to show tooltip for all data cell.

  You may set the same effect using ToolTip parameter of initialization method SetConfig.

  You need to call this method after initialization before search processing.

- **Syntax**

Syntax	ObjId.**ShowToolTip**(ToolTip)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
ToolTip	Boolean	Required	Whether to display tooltips

- **Example**

```
//Configure to show tooltip

mySheet.ShowToolTip(1);
```

ShowTreeSubSum Method

- **Purpose**

In stree structure, calculate and display sub sum in a particular column

This method should be called before data search method is called. When configuration is changed, you need to rerun a data search to apply it.

SumCols is a string of column indexes or SaveNames where sub sum should be calculated, adjoined by"|".

AvgCols is a string of column indexes or SaveNames where average should be calculated, adjoined by"|".

CountCols is a string of column indexes or SaveNames where counting should be done, adjoined by"|".

Set the parameter values in JSON format.

Sample) var info = {SumCols:"2|3|4", AvgCols:"5|6", SumEx:1};

➢ **Syntax**

Syntax	ObjId.**ShowTreeSubSum**(Info)

➢ **Info**

Return	None		
Parameter	Type	Required Y/N	Description
SumCols	String	Optional	String of column indexes or SaveNames where aggregate total should be calculated, adjoined by"|". Default=""
AvgCols	String	Optional	String of column indexes or SaveNames where average should be calculated, adjoined by"|". Default=""
CountCols	String	Optional	String of column indexes or SaveNames where counting should be done, adjoined by"|". Default=""
SumEx	Boolean	Optional	Whether to include rows with status Deleted in calculation. Default=0
ExceptNull	Boolean	Optional	Whether to includ Null values to averaging Default=0

➢ **Example**

```
//Display tree sub sums for the 2nd column

var info = SumCols:"2};

mySheet.ShowTreeSubSum(info);



// Exclude deleted rows from calculation

var info = {SumCols:"2, SumEx:1};
```

```
mySheet. ShowTreeSubSum (info);
```

SubSumBackColor Method

- **Purpose**

  Check or configure background color of sub sum row.

- **Syntax**

Syntax	Get	ObjId.**GetSubSumBackColor**()

- **Info**

Return	String, currently set value (In case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Check background color of sub sum row.

mySheet.GetSubSumBackColor();
```

- **Syntax**

Syntax	Set	ObjId.**SetSubSumBackColor**(Color)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Color	String	Required	WebColor value to set

➢ **Example**

```
//Set background color of sub sum row as green.

mySheet.SetSubSumBackColor("#00FF00");
```

ShowSum Method

➢ **Purpose**

When aggregate total is not calculated due to performance issue, this method can be used to calculate sum of AutoSum columns to display once, or perform recalculation.

It may be faster to call ShowSum() after search without calculating the sum, rather than calculating sum from the start.

You can call ShowSum multiple times to recalculate the sum.

Sums will be calculated (recalculated) only when SetRedrawSum is 1. If it is 0, sums will not be calculated (recalculated) even though this method is called.

➢ **Syntax**

Syntax	ObjId.**ShowSum**()

➢ **Info**

Return	None		
Parameter	Type	Required Y/N	Description
N/A			

➢ **Example**

```
mySheet.ShowSum();
```

ShowTreeLevel Method

- **Purpose**

  Configure the tree levels to display when data is tree type.

  Available options are as follows:

Level	Description
0	Collapse all tree
-1	Expand all, Default
Others	Expand to that level

In ChildStatus, you can configure child status below the tree level to expand.

In ChildStatus, you can configure the following values:

ChildStatus	Description
0	Maintain the previous state, Default
1	Collapse all
2	Expand all

- **Syntax**

Syntax	ObjId.**ShowTreeLevel**([Level], [ChildStatus])

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Level	Integer	Optional	Tree level to show Default=-1
ChildStatus	Integer	Optional	Configure expansion status of the levels below the tree level to expand. Default=0

➢ **Example**

```
//Collapse all

mySheet.ShowTreeLevel(0, 1);



//Expand all

mySheet.ShowTreeLevel(-1);
```

SumBackColor Method

➢ **Purpose**

Check or configure background color of sum row.

Select the color using WebColor.

➢ **Syntax**

Syntax	Get	ObjId.**GetSumBackColor**()

➢ **Info**

Return	String, WebColor value (In case of Get Method)		
Parameter	Type	Required Y/N	Description

➢ **Example**

```
//Check background color of sum row

mySheet.GetSumBackColor();
```

➢ **Syntax**

Syntax	Set	ObjId.**SetSumBackColor**(Color)

➢ **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Color	String	Required	WebColor value

➢ **Example**

```
//Set background color of sum row as green.

mySheet.SetSumBackColor("#00FF00");
```

SumFontBold Method

➢ **Purpose**

Check or configure whether to bolden font in sum row.

If this property is set as 1, sum row font is boldened. If 0, they will remain as general font. If sum row font is boldened, the downside is that they do not align well with other data. The default setting is 0.

➢ **Syntax**

Syntax	Get	ObjId.**GetSumFontBold**()

➢ **Info**

Return	Boolean, bold status of the current font (In case of Get Method)		
Parameter	Type	Required Y/N	Description

➢ **Example**

```
//Check font bold status of sum row.

mySheet.GetSumFontBold();
```

➢ **Syntax**

Syntax	Set	ObjId.**SetSumFontBold**(Bold)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Bold	Boolean	Required	Font bold status

- **Example**

```
//Bolden the font in sum row.

mySheet.SetSumFontBold(1);
```

SumFontColor Method

- **Purpose**

  Check or configure font color of sum row.

  Select the color using WebColor.

- **Syntax**

Syntax	Get	ObjId.**GetSumFontColor**()

- **Info**

Return	String, current font color of sum row (in case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Check for font color of sum row

mySheet.GetSumFontColor();
```

➢ **Syntax**

Syntax	Set	ObjId.**SetSumFontColor**(Color)

➢ **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Color	String	Required	WebColor value

➢ **Example**

```
//Set the font color of sum row as green.

mySheet.SetSumFontColor("#00FF00");
```

SumRowHidden Method

➢ **Purpose**

Check or configure hiding status of sum row.

➢ **Syntax**

Syntax	Get	ObjId.**GetSumRowHidden**()

➢ **Info**

Return	Boolean, hiding setting value (in case of Get Method)		
Parameter	Type	Required Y/N	Description

➢ **Example**

```
//Check hiding status of sum row and unhide if hidden.

if(mySheet.GetSumRowHidden() == 1) {
        mySheet.SetSumRowHidden(0);

}
```

- **Syntax**

Syntax	Set	ObjId.**SetSumRowHidden**(Hidden)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Hidden	Boolean	Required	Hidden or not

- **Example**

```
//Check hiding status of sum row and unhide if hidden.

if(mySheet.GetSumRowHidden() == 1) {

        mySheet.SetSumRowHidden(0);

}
```

SumValue Method

- **Purpose**

Check or configure sum cell values without formatting.

- **Syntax**

Syntax	Get	ObjId.**GetSumValue**(Col)

- **Info**

Return	String, value of the sum cell (in case of Get Method)		
Parameter	Type	Required	Description

		Y/N	
Col	Long / String	Required	Column index of sum cell or SaveName

➢ **Example**

```
//Read value of column 2 in the first sum row. If the value is 1,234.56,

//1234.56 will be returned.

alert("The sum is " + mySheet.GetSumValue(2) + ".");
```

➢ **Syntax**

Syntax	Set	ObjId.**SetSumValue**(Col,Value)

➢ **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Col	Long / String	Required	Column index of sum cell or SaveName
Value	Long	Required	CellValue without formatting

➢ **Example**

```
//Change the value of column 2 in the sum row to 5432.12

mySheet.SetSumValue(2,5432.12);
```

Theme Method

➢ **Purpose**

Check or configure theme design of IBSheet.

To set a theme, you need a defined theme design. (See Appendix 3. Creating a Theme at the end of this document for more details)

- **Syntax**

Syntax	Get	ObjId.**GetTheme**()

- **Info**

Return	String, prefix of the current theme (in case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Check the currently set theme

var Prefix = mySheet.GetTheme();
```

- **Syntax**

Syntax	Set	ObjId.**SetTheme**(Prefix, Folder)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Prefix	String	Required	Prefix of the theme
Folder	String	Required	Theme folder name

- **Example**

```
//Apply Gray theme.

mySheet.SetTheme(“GG”, “Gray”);
```

ToolTipText Method

- **Purpose**

  Check or configure cell tooltip.

  You can check or configure tooltips for each cell in header or data row. Hover help appears when mouse hovers over the cell.

- **Syntax**

Syntax	Get	ObjId.**GetToolTipText**(Row, Col)

- **Info**

Return	String, currently set value (In case of Get Method)		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row index of the cell
Col	Long / String	Required	Column index of the cell or SaveName

- **Example**

```
//Check tooltip configuration

alert(mySheet.GetToolTipText(1, 1));
```

- **Syntax**

Syntax	Set	ObjId.**SetToolTipText**(Row, Col, ToolTip)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description

Row	Long	Required	Row index of the cell
Col	Long / String	Required	Column index of the cell or SaveName
ToolTip	String	Required	Tooltip configuration

- **Example**

```
//Set to display tooltip for a cell

mySheet.SetToolTipText(1,1, "The cell amount is " + mySheet.GetCellText(1,1)
+ ". ");
```

TopRow Method

- **Purpose**

Check or configure the topmost row index.

The row set in this method is the topmost row of the data area, except for the header area. The row set in this method does not become selected.

Even though a wrong row index is used, the closest row will be used as the topmost row without an error message. This only works when there are enough number of rows to display below the topmost row.

For example, let's assume that the total row count is 103 and 20 rows display in one page. Even though you set the TopRow property as 100, at least 18 rows should display, so TopRow will be changed to 85. The number of rows to display in the display area is deducted first, and then the TopRow is determined.

- **Syntax**

Syntax	Get	ObjId.**GetTopRow**()

- **Info**

Return	Long, the topmost row index (In case of Get Method)		
Parameter	Type	Required Y/N	Description

➢ **Example**

```
//Check for the topmost row index

mySheet.GetTopRow();
```

➢ **Syntax**

Syntax	Set	ObjId.**SetTopRow**(Row)

➢ **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Row	Long	Required	Row Index

➢ **Example**

```
//Configure the topmost row index

mySheet.SetTopRow(100);
```

TotalRows Method

➢ **Purpose**

Check or configure the DB record count to search depending on the search parameters.

➢ **Syntax**

Syntax	Get	ObjId.**GetTotalRows**()

➢ **Info**

Return	String, currently set value (In case of Get Method)

Parameter	Type	Required Y/N	Description

➢ **Example**

```
Check the total data count.

alert("Total data count is " + mySheet.GetTotalRows() + ". ");
```

➢ **Syntax**

Syntax	Set	ObjId.**SetTotalRows**(Count)

➢ **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Count	Long	Required	Total count configuration

➢ **Example**

```
//Set the total data count as 1000.

mySheet.SetTotalRows(1000);
```

TreeActionMode Method

➢ **Purpose**

Check or configure various properties related to functional processes in tree structure.

Property options are as follows:

Value	Purpose
0	Do not allow deletion if there is child level.

	If the parent is deleted, deletion cannot be reversed. (Default)
1	If deletion is checked for a parent, child will be also deleted.

- **Syntax**

Syntax	Get	ObjId.**GetTreeActionMode**()

- **Info**

Return	Integer, set value (In case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Check the currently set TreeActionMode value.

alert(mySheet.GetTreeActionMode());
```

- **Syntax**

Syntax	Set	ObjId.**SetTreeActionMode**(Value)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Value	Integer	Required	TreeActionMode value

- **Example**

```
//Delete child levels when delete is checked for a parent row.

mySheet.SetTreeActionMode(1);
```

UnicodeByte Method

- **Purpose**

  Check or configure bytes of Asian charaters.

  In Java script, all characters are recognized as 1 byte.

  However, depending on the DB language configurations, Korean, Japanese or Chinese letter may need to be recognized as 2 byte or larger.

  In this case, you may adjust byte count using this method.

- **Syntax**

Syntax	Get	ObjId.**GetUnicodeByte**()

- **Info**

Return	Integer,current set value (in case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
// Check for the currently set byte

mySheet. GetUnicodeByte();
```

- **Syntax**

Syntax	Set	ObjId.**SetUnicodeByte**(byte)

- **Info**

Return	None		
Parameter	Type	Required	Description

		Y/N	
byte	Integer	Required	Byte to set (Default=1)

- **Example**

```
//Change the byte count for one Asian character to 3.

mySheet. SetUnicodeByte(3);
```

UseDefaultTime Method

- **Purpose**

  When a cell format is "Hms" or "Hm", the current system time will display when the cell value is null and editing starts. When this property is set as 0, the cell will display nothing instead of current system time. The default value of this property is 1.

- **Syntax**

Syntax	Get	ObjId.**GetUseDefaultTime**()

- **Info**

Return	Boolean,current set value (in case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Check the current time display setting

mySheet.GetUseDefaultTime();
```

- **Syntax**

Syntax	Set	ObjId.**SetUseDefaultTime**(Value)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Value	Boolean	Required	Whether to display current time (Default=1)

- **Example**

```
//Set not to display the current time

mySheet.SetUseDefaultTime(0);
```

UserAgent Method

- **Purpose**

  Check or configure the IBUserAgent value which is sent to the server as part of header information during search or saving.

- **Syntax**

Syntax	Get	ObjId.**GetUserAgent**()

- **Info**

Return	String, currently set value (In case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
// Check the IBUserAgent value.

mySheet.GetUserAgent();
```

- **Syntax**

Syntax	Set	ObjId.**SetUserAgent**(Value)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Value	String	Required	User-Agent configuration

- **Example**

```
// Send the IBUserAgent value in HTTP header information as My Agent Name

mySheet.SetUserAgent("My Agent Name");
```

UploadingImage Method

- **Purpose**

  Check or configure the waiting image file path for file upload.

  This method can be used to customize waiting image as the user want which displays during file upload.

- **Syntax**

Syntax	Get	ObjId.**GetUploadingImage**()

- **Info**

Return	String, currently set value (In case of Get Method)		
Parameter	Type	Required Y/N	Description

➢ **Example**

```
//Check the currently set image path of the upload waiting image.

alert(mySheet.GetUploadingImage());
```

➢ **Syntax**

Syntax	Set	ObjId.**SetUploadingImage**(Url)

➢ **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Url	String	Required	Image URL

➢ **Example**

```
//Change the upload waiting image.

mySheet.SetUploadingImage( "/sheet/imgUpload.gif");
```

ValidateFail Method

➢ **Purpose**

If you want to stop saving as an invalid data piece is found during pre-saving data validation using OnValidation event, use this method to abort saving.

➢ **Syntax**

Syntax	ObjId.**ValidateFail**(Flag)

➢ **Info**

Return	None		
Parameter	Type	Required	Description

		Y/N	
Flag	Boolean	Required	Whether to abort saving

- **Example**

```
function mySheet_OnValidation(Row, Col, Value) {
  if(Col == 2)     {
    if(Value=="Korean won" && mySheet.GetCellValue(Row,Col+1) >= 10000000) {
      alert("When the currency is Korean won, the amount cannot exceed ten million Korean won."); ");
      mySheet.ValidateFail(1);
      mySheet.SetSelectCell(Row, Col+1);
    } else if(Value=="Foreign currency" && mySheet.GetCellValue(Row,Col+1) < 10000000) {
      alert("When the currency is a foreign one, the amount must be ten million or greater."); ");
      mySheet.ValidateFail(1);
      mySheet.SetSelectCell(Row, Col+1);
    }
  }
}
```

Visible Method

- **Purpose**

  Check or configure whether to display IBSheet.

  When this property is set as 0, everything including count information will be hidden. When set as 1, all information will become visible.

- **Syntax**

Syntax	Get	ObjId.**GetVisible**()

- **Info**

Return	Boolean,current set value (in case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
// Check whether to display IBSheet

if(mySheet.GetVisible()){

      alert("Display IBSheet");

}else{

alert("IBSheet is hidden");

}
```

- **Syntax**

Syntax	Set	ObjId.**SetVisible**(Visible)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Visible	Boolean	Required	Whether to display sheet on the screen

- **Example**

```
// Hide IBSheet

mySheet.SetVisible(0);
```

WaitImage Method

- **Purpose**

  Check or configure waiting image file path.

  This method can be used to customize waiting image as the user want which displays during processing. Waiting image for processing includes both the search waiting image and save waiting image.

- **Syntax**

Syntax	Get	ObjId.**GetWaitImage**()

- **Info**

Return	String, currently set URL (In case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Check the processing waiting image path.

mySheet.GetWaitImage();
```

- **Syntax**

Syntax	Set	ObjId.**SetWaitImage**(Url)

- **Info**

Return	None

Parameter	Type	Required Y/N	Description
Url	String	Required	Image URL

- **Example**

```
//Change waiting image for processing.

mySheet.SetWaitImage("/sheet/imgWait.gif");
```

WaitImageVisible Method

- **Purpose**

  Check or configure whether to display waiting image during processing.

  When a search or saving method is called where the end user waiting time is required, waiting image displays as default.

  If you do not want to use the waiting image for any reason, set this property as 0 to remove waiting image.

  Depending on the property setting, waiting images may or may not display for different processing.

- **Syntax**

Syntax	Get	ObjId.**GetWaitImageVisible**()

- **Info**

Return	Boolean,current set value (in case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
//Check whether to display waiting image for processing.

alert(mySheet.GetWaitImageVisible());
```

- **Syntax**

Syntax	Set	ObjId.**SetWaitImageVisible**(Visible)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Visible	Boolean	Required	Whether to display various waiting images

- **Example**

```
//Set not to display waiting image for processing.

mySheet.SetWaitImageVisible(0);
```

WaitTimeOut Method

- **Purpose**

Check or configure timeout for server response. (Unit: second)

- **Syntax**

Syntax	Get	ObjId.**GetWaitTimeOut**()

- **Info**

Return	Integer,current set value (in case of Get Method)		
Parameter	Type	Required Y/N	Description

- **Example**

```
// Check for the current timeout for server response

alert(mySheet.GetWaitTimeOut());
```

- **Syntax**

Syntax	Set	ObjId.**SetWaitTimeOut**(time)

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
time	Integer	Required	Timeout to set in seconds (Default=60)

- **Example**

```
// Check or configure timeout for server response

mySheet.SetWaitTimeOut(120);
```

Chapter 10. IBSheet Global Function

10. IBSheet Global Function

10.1 Use Global Function

To use Global functions supported by IBSheet, you must include ibsheet.js file within the corresesonding page.

IBSheet offers the following global functions:

10.2 Function List

IBCalendarSetTheme Method

- **Purpose**

  Set a theme design for calendar pop-up to use for external control.

  To set a theme, you need a defined theme design.

  (See **Appendix** 3. Creating a Theme at the end of this document for more details)

- **Syntax**

Syntax	**IBCalendarSetTheme**(Prefix, Folder);

- **Info**

Return	None		
Parameter	Type	Required Y/N	Description
Prefix	String	Required	Prefix of the theme
Folder	String	Required	Theme folder name

- **Example**

```
//Apply Gray theme.

IBCalendarSetTheme(“GG”, “Gray”);
```

IBShowCalendar Method

- **Purpose**

  Set to allow use of calendar pop-up by external control.

  When a CallBack method is configured, the first parameter is usually the selected date string. If there are other parameters to send by CallBack method, set it as CallBackParam. Object set in CallBackParam will be sent as the second parameter of CallBack method.

- **Syntax**

Syntax	**IBShowCalendar**(val, obj);

- **Info**

Return	String, (Date value set in calendar pop-up dialog)		
Parameter	Type	Required Y/N	Description
val	String	Required	Date data value (Defaul=today's date)
obj	Object	Optional	Configure function into JSON format.

Details

Properties configurable by column are as follows:

Property	Type	Description
CalButtonAlign	String	Configure how to align buttons to use in calendar pop-up Value / Details: Left: Align left Center: Align center (Default) Right: Align right
CalButtons	String	Configure the button to use for calendar popup Adjoin buttons to use with "|". Value / Details: Close: Cancel button Today: Enter today's date button Yesterday: Enter yesterday's date button

CalButtonAlign values:

Value	Details
Left	Align left
Center	Align center (Default)
Right	Align right

CalButtons values:

Value	Details
Close	Cancel button
Today	Enter today's date button
Yesterday	Enter yesterday's date button

CallBack	String	Function Name Method name to use upon return
CallBackParam	Object	Parameter object to receive from CallBack method
Format	String	Date format pattern (Default="yyyy/MM/dd")
Holiday	String	Set custom holidays to show on calendar You can set custom holidays using a string of values connected with a pipe operator. The date format should be yyyyMMdd format. Use asterisk (*) for recurring holidays on a monthly or yearly basis. **Sample)** Holiday : "20120725|*0703|2012*27|**17" *0703 : July 3th, every year매년 7월 3일 2012*27 : 27th of each month for the year 2012 **17 : 17th of every month, every year
Result	Object	Object where to include selected values (Input object)
Target	String, Object	"Mouse" (When last mouse location is used, Default) or calendar button Object (when calendar button location is used)
X	Integer	(When coordinate values are used), X coordinate
Y	Integer	(When coordinate values are used), Y coordinate

➢ **Example**

```
// Enter date data value
var val = document.getElementById(“DateText”).value;


// Calendar pop-up dialog location: When X and Y coordinates are used
   var obj = { Format:”yyyy/MM/dd”, X:300, Y:600, CallBack:”Test” };


// Calendar pop-up dialog location: When last mouse location is used
   var obj = { Format:”yyyy/MM/dd”, Target:”Mouse”, CallBack: “Test” };


// Calendar pop-up dialog location: When calendar button location is used
   var obj = { Target:document.getElementById(“DateBtn”), CallBack: “Test” };


// CalButtons property: when closing calendar button option setting is used
   var obj = { Format:”Ymd”, Target:document.getElementById(“DateBtn”),
   CallBack: “Test” , CalButtons : “Close”};


// Alignment setting when closing calendar button option setting is used:
Align left
   var obj = { Format:”Ymd”, Target:document.getElementById(“DateBtn”),
   CallBack: “Test”, CalButtons : “Close”, CalButtonAlign : “Left” };


// Custom holiday setting
   var obj = { Format:”Ymd”, Target:document.getElementById(“DateBtn”),
   CallBack: “Test”, Holiday: " 20101215|*1203"};


// fnName: create a function using the method name
function Test (date){
   // Return value when a date is selected from calendar pop-up dialogue

   document.getElementById("DateText").value = date;

}
```

IBCloseCalendar Method

➢ **Purpose**

Set to close calendar pop-up used by external control.

➢ **Syntax**

Syntax	**IBCloseCalendar**();

➢ **Info**

Return	None		
Parameter	Type	Required Y/N	Description
N/A			

```
// Open calendar pop-up.

IBShowCalendar("20121116", {Format:"yyyy/MM/dd", X:300, Y:600});



// Close calendar pop-up dialog.

IBCloseCalendar();
```

Appendix

1. Configurable properties by column type

In InitColumns method, you can configure the following properties for each type as in the table below:

1.1. Text format type

Property	Text	Popup	PopupEdit
Type	O	O	O
AcceptKeys	O	X	O
Align	O	O	O
ApproximateType	X	X	X
BackColor	O	O	O
ButtonUrl	X	O	O
CalcLogic	X	X	X
CaseSensitive	O	X	O
ColMerge	O	O	O
ColSpan	O	O	O
ComboCode	X	X	X
ComboText	X	X	X
Cursor	O	O	O
DefaultValue	O	O	O
Edit	O	X	O
EditLen	O	X	O
Ellipsis	O	O	O
ExcludeEmpty	X	X	X
FalseValue	X	X	X
FontColor	O	O	O

Format	O	O	O
FormatFix	O	O	O
FullInput	O	X	O
HeaderCheck	X	X	X
Hidden	O	O	O
HoverUnderline	O	O	O
Image	O	X	X
ImgAlign	O	X	X
ImgHeight	O	X	X
ImgWidth	O	X	X
InsertEdit	O	O	O
KeyField	O	O	O
LevelSaveName	O	O	O
MaximumValue	X	X	X
MinimumValue	X	X	X
MultiLineText	O	X	X
PointCount	X	X	X
PopupCheckEdit	O	O	O
PopupCode	O	O	O
PopupText	O	O	O
RadioIcon	X	X	X
RowSpan	O	O	O
SaveName	O	O	O
Sort	O	O	O
ToolTipText	O	O	O

TreeCol	O	O	O
TrueValue	X	X	X
UpdateEdit	O	O	O
VAlign	O	O	O
Width	O	O	O
Wrap	O	O	O

1.2. Date format type

Property	Date
Type	O
AcceptKeys	X
Align	O
ApproximateType	X
BackColor	O
ButtonUrl	X
CalcLogic	X
CaseSensitive	X
ColMerge	O
ColSpan	O
ComboCode	X
ComboText	X
Cursor	O
DefaultValue	O
Edit	O
EditLen	O
Ellipsis	O

ExcludeEmpty	X
FalseValue	X
FontColor	O
Format	O
FormatFix	O
FullInput	O
HeaderCheck	X
Hidden	O
HoverUnderline	O
Image	X
ImgAlign	X
ImgHeight	X
ImgWidth	X
InsertEdit	O
KeyField	O
LevelSaveName	O
MaximumValue	X
MinimumValue	X
MultiLineText	X
PointCount	X
PopupCheckEdit	O
PopupCode	O
PopupText	O
RadioIcon	X
RowSpan	O

SaveName	O
Sort	O
ToolTipText	O
TreeCol	O
TrueValue	X
UpdateEdit	O
VAlign	O
Width	O
Wrap	X

1.3. Number format type

Property	Int	Float	AutoSum	AutoAvg
Type	O	O	O	O
AcceptKeys	X	X	X	X
ApproximateType	O	O	O	O
Align	O	O	O	O
BackColor	O	O	O	O
ButtonUrl	X	X	X	X
CalcLogic	O	O	O	O
CaseSensitive	X	X	X	X
ColMerge	O	O	O	O
ColSpan	O	O	O	O
ComboCode	X	X	X	X
ComboText	X	X	X	X
Cursor	O	O	O	O
DefaultValue	O	O	O	O

Edit	O	O	O	O
EditLen	O	O	O	O
Ellipsis	O	O	O	O
ExcludeEmpty	X	X	X	O
FalseValue	X	X	X	X
FontColor	O	O	O	O
Format	O	O	O	O
FormatFix	O	O	O	O
FullInput	O	O	O	O
HeaderCheck	X	X	X	X
Hidden	O	O	O	O
HoverUnderline	O	O	O	O
Image	O	O	O	O
ImgAlign	O	O	O	O
ImgHeight	O	O	O	O
ImgWidth	O	O	O	O
InsertEdit	O	O	O	O
KeyField	O	O	O	O
LevelSaveName	O	O	O	O
MaximumValue	O	O	O	O
MinimumValue	O	O	O	O
MultiLineText	X	X	X	X
PointCount	X	O	O	O
PopupCheckEdit	O	O	O	O
PopupCode	O	O	O	O

PopupText	O	O	O	O
RadioIcon	X	X	X	X
RowSpan	O	O	O	O
SaveName	O	O	O	O
Sort	O	O	O	O
ToolTipText	O	O	O	O
TreeCol	O	O	O	O
TrueValue	X	X	X	X
UpdateEdit	O	O	O	O
VAlign	O	O	O	O
Width	O	O	O	O
Wrap	X	X	X	X

1.4. Checkbox format type

Property	DelCheck	CheckBox	DummyCheck	Radio
Type	O	O	O	O
AcceptKeys	X	X	X	X
Align	X	X	X	X
ApproximateType	X	X	X	X
BackColor	O	O	O	O
ButtonUrl	X	X	X	X
CalcLogic	X	X	X	X
CaseSensitive	X	X	X	X
ColMerge	O	O	O	O
ColSpan	O	O	O	O
ComboCode	X	X	X	X

ComboText	X	X	X	X
Cursor	X	X	X	X
DefaultValue	X	O	O	X
Edit	O	O	O	O
EditLen	X	X	X	X
Ellipsis	O	O	O	O
ExcludeEmpty	X	X	X	X
FalseValue	O	O	O	O
FontColor	X	X	X	X
Format	X	X	X	X
FormatFix	O	O	O	O
FullInput	X	X	X	X
HeaderCheck	O	O	O	X
Hidden	O	O	O	O
HoverUnderline	X	X	X	X
Image	X	X	X	X
ImgAlign	X	X	X	X
ImgHeight	X	X	X	X
ImgWidth	X	X	X	X
InsertEdit	X	O	O	O
KeyField	O	O	O	O
LevelSaveName	O	O	O	O
MaximumValue	X	X	X	X
MinimumValue	X	X	X	X
MultiLineText	X	X	X	X

PointCount	X	X	X	X
PopupCheckEdit	O	O	O	O
PopupCode	O	O	O	O
PopupText	O	O	O	O
RadioIcon	O	O	O	O
RowSpan	O	O	O	O
SaveName	O	O	O	O
Sort	O	O	O	O
ToolTipText	O	O	O	O
TreeCol	O	O	O	O
TrueValue	O	O	O	O
UpdateEdit	X	O	O	O
VAlign	X	X	X	X
Width	O	O	O	O
Wrap	X	X	X	X

1.5. Combo format type

Property	Combo	ComboEdit
Type	O	O
AcceptKeys	X	O
Align	O	O
ApproximateType	X	X
BackColor	O	O
ButtonUrl	X	X
CalcLogic	X	X
CaseSensitive	X	O

ColMerge	O	O
ColSpan	O	O
ComboCode	O	O
ComboText	O	O
Cursor	X	O
DefaultValue	O	O
Edit	X	O
EditLen	X	O
Ellipsis	O	O
ExcludeEmpty	X	X
FalseValue	X	X
FontColor	O	O
Format	X	X
FormatFix	O	O
FullInput	X	O
HeaderCheck	X	X
Hidden	O	O
HoverUnderline	O	O
Image	X	X
ImgAlign	X	X
ImgHeight	X	X
ImgWidth	X	X
InsertEdit	O	O
KeyField	O	O
LevelSaveName	O	O

MaximumValue	X	X
MinimumValue	X	X
MultiLineText	X	X
PointCount	X	X
PopupCheckEdit	O	O
PopupCode	O	O
PopupText	O	O
RadioIcon	X	X
RowSpan	O	O
SaveName	O	O
Sort	O	O
ToolTipText	O	O
TreeCol	O	O
TrueValue	X	X
UpdateEdit	O	O
VAlign	O	O
Width	O	O
Wrap	X	O

1.6. Other format types

Property	Result	Img	Pass	Status	Seq
Type	O	O	O	O	O
AcceptKeys	X	X	X	X	X
Align	O	O	O	O	O
ApproximateType	X	X	X	X	X
BackColor	O	O	O	O	O

ButtonUrl	X	X	X	X	X
CalcLogic	X	X	X	X	X
CaseSensitive	O	X	X	X	X
ColMerge	O	O	O	O	X
ColSpan	O	O	O	O	O
ComboCode	X	X	X	X	X
ComboText	X	X	X	X	X
Cursor	O	O	O	O	O
DefaultValue	O	X	X	X	X
Edit	X	X	O	X	X
EditLen	X	X	O	X	X
Ellipsis	O	X	O	O	O
ExcludeEmpty	X	X	X	X	X
FalseValue	X	X	X	X	X
FontColor	O	X	O	O	O
Format	X	X	X	X	X
FormatFix	O	O	O	O	O
FullInput	X	X	O	X	X
HeaderCheck	X	X	X	X	X
Hidden	O	O	O	O	O
HoverUnderline	O	X	O	O	O
Image	X	X	X	X	X
ImgAlign	X	X	X	X	X
ImgHeight	X	O	X	X	X
ImgWidth	X	O	X	X	X

InsertEdit	O	O	O	O	O
KeyField	O	O	O	O	O
LevelSaveName	O	O	O	O	O
MaximumValue	X	X	X	X	X
MinimumValue	X	X	X	X	X
MultiLineText	X	X	X	X	X
PointCount	X	X	X	X	X
PopupCheckEdit	O	O	O	O	O
PopupCode	O	O	O	O	O
PopupText	O	O	O	O	O
RadioIcon	X	X	X	X	X
RowSpan	O	O	O	O	O
SaveName	O	O	O	O	O
Sort	O	O	O	O	O
ToolTipText	O	O	O	O	O
TreeCol	O	O	O	O	O
TrueValue	X	X	X	X	X
UpdateEdit	O	O	O	O	O
VAlign	O	O	O	O	O
Width	O	O	O	O	O
Wrap	O	X	O	O	X

2. Data format allowed for Fx (Formatted) search mode

When Fx option is used for DoSearch or LoadSearchData methods, formats allowed for each type are as follows:

Column type	Format	Supported values	Supported value sample	Unacceptable value sample	Remarks
Text	General	String			
	User Format	String without formatting	7912121022345	791212-1022345	
	Date	See Date type			
Popup	General	String			
	Date	See Date type			
PopupEdit	General	String			
	Date	See Date type			
Pass		String			
Date	Date	Date format string	2012-02-16	20120216	Year, month and date separator does not need to be the same as in the set format
	Time	Date format string	13:15:16	131516	
	Date+time	Date and time format string	2012-02-16 13:15:16	20120216131516	Put a space between date and time
Int		Number values without formatting	1234567	1,234,567	
Float		Number values without formatting	123456.789	123,456.789	
AutoSum	Int	Number values without formatting	1234567	1,234,567	
	Float	Number values without	123456.789	123,456.789	

		formatting			
AutoAvg	Int	Number values without formatting	1234567	1,234,567	
	Float	Number values without formatting	123456.789	123,456.789	
CheckBox		0 or 1		String except for 0, 1	
DelCheck		0 or 1		String except for 0, 1	
DummyCheck		0 or 1		String except for 0, 1	
Radio		0 or 1		String except for 0, 1	
Combo		Set ComboCode value			InitCombo NoMatchText unacceptable
ComboEdit		Set ComboCode value			InitCombo NoMatchText unacceptable
Image		Image Url			
Seq		Space		String	
Status		I, D		String without a space, I or D	

3. Creating a theme

In IBSheet, a design theme is composed of image files and ibsheet.css file within the "Main" folder where ibsheet.js file is located.

To create a new design theme or update an existing theme, go through the following steps:

1. Copy the Main folder and save it in another name. (Let's assume that we copied the folder and saved in "DeepBlue" folder.)
2. Change the various button images within the folder to suit the theme of the current development project.
3. See ibsheet.css file for sheet color. When you open the file, you can see that all css class names start with .GM. You must replace this name with another to use. (Let's assume that we replaced .GM with .DB.)
4. As the last step, you can apply the new theme as follows in IBSheet initialization statement.
   mySheet.SetTheme("DB","DeepBlue");

4. Apply Merge setting to excel download

	DownCols	
	Same column setting as in the screen	Different column setting from the screen
Use DownRows	X	X
Not use DownRows	○	△

X : Do not apply merge setting to excel file

△ : Merge is applied, but not necessarily the same way as the merging on the screen

○ : Merge is applied the same way as it displays on the screen

1. When merge is not applied

   - When you use DownRows, merge will not apply to excel file even when Merge is set as 1.

2. When the excel file cells are merged the same way as the screen

   - Download columns as they appear on the screen

3. When merge is applied, but not necessarily the same way as the merging on the screen

   - Columns that appear on the screen may be different from the columns downloaded.