



ORACLE® HYPERION SMART VIEW FOR OFFICE,  
FUSION EDITION

RELEASE 11.1.1.3

NEW FEATURES

ORACLE®  
ENTERPRISE PERFORMANCE  
MANAGEMENT SYSTEM

CONTENTS IN BRIEF

Overview .....	2
Drill-Through .....	2
Importing Smart View Metadata .....	2
New VBA Functions .....	4

## Overview

This document describes the new features introduced in Oracle Hyperion Smart View for Office, Fusion Edition Release 11.1.1.3.

## Drill-Through

With this release, Smart View's drill-through capabilities have been enhanced to include Oracle Hyperion Financial Data Quality Management ERP Integration Adapter for Oracle Applications, Oracle General Ledger, and Oracle Hyperion Financial Data Quality Management, Fusion Edition. If you are connected to Oracle Hyperion Planning, Fusion Edition or Oracle Hyperion Financial Management, Fusion Edition via Smart View, you can use the drill-through capabilities of Smart View to drill through your Planning or Financial Management application to detailed data in ERP Integrator or Oracle Hyperion Financial Data Quality Management, Fusion Edition data sources.

For applications created in Oracle Essbase Studio or Oracle Essbase Integration Services, you can continue to drill through to relational databases. For applications created in Oracle Essbase Studio, you can also drill through to administrator-configured URLs.

**Note:** You cannot use alias tables for drill-through; you must use member names.

**Note:** To enable drill-through, all data source providers are front-ended with a proxy server. Please see the *Oracle Hyperion Enterprise Performance Management System Installation and Configuration Guide* for more information.

## Importing Smart View Metadata

**Data Sources:** Oracle Essbase, Planning, Financial Management, Oracle's Hyperion Reporting and Analysis, Oracle's Hyperion® Enterprise®, Oracle Business Intelligence Enterprise Edition

Before this release, when you copied an Excel worksheet, the Smart View custom properties, or metadata, associated with the original sheet, such as connection information, POV selections, and alias tables, were not carried over. In 11.1.1.3, you can import this information from the original sheet to the newly copied sheet using the new **Import Metadata** feature.

You can use **Import Metadata** in the following:

- Ad hoc mode, including Smart Slices
- Data forms
- Functions
  - Query-bound functions in sheets created by Smart View copy and paste
  - Non-query-bound functions created by the Function Builder
- Worksheets that contain reports imported from Reporting and Analysis providers

Worksheets that contain Report Designer objects cannot use Import Metadata but can be replicated by cascading as described in the *Smart View User's Guide* or online help.

**Note:** On Microsoft Office 2003 systems, **Improved Metadata Storage** on the Options dialog must be checked.

- To import metadata to a copied worksheet (Note that this operation cannot be undone):
  - 1 Use Excel to copy a worksheet. This operation copies the visible contents of the source worksheet but not the metadata (connection information, POV selections, alias tables and the like) to the destination worksheet.
  - 2 With the destination worksheet active, select **Hyperion**, then **Import Metadata** to display a list of all open workbooks and their corresponding open worksheets.
  - 3 From the list, select the worksheet that contains the metadata that you want to import to the destination worksheet.
  - 4 Click **OK**. You will be asked to confirm your selection.
  - 5 Refresh.

You can also perform Import Metadata using the VBA function `HypCopyMetaData`.

## HypCopyMetaData

### Description

`HypCopyMetaData()` performs Export Metadata.

### Syntax

`HypCopyMetaData (vtSourceSheetName, vtDestinationSheetName)`

`ByVal vtSourceSheetName As Variant`

`ByVal vtDestinationSheetName As Variant`

### Parameters

**vtSourceSheetName:** Name of the source sheet where the Custom Properties should be copied from. (Required)

**vtDestinationSheetName:** Name of the destination sheet where the Custom Properties should be copied to. (Required)

### Return Value

Returns `SS_OK` if the function succeeded; otherwise, the appropriate error code.

## Example

```
Public Declare Function HypCopyMetaData Lib "HsAddin.dll" (ByVal  
vtSourceSheetName As Variant, ByVal vtDestinationSheetName As Variant) As  
Long  
  
Sub Sample_HypCopyMetaData()  
    Dim LRet As Long  
    LRet = HypCopyMetaData ("Sheet1", "Sheet1(2)")  
End Sub
```

## New VBA Functions

This release features several new VBA functions, which facilitate integration with Oracle Crystal Ball Enterprise Performance Management, Fusion Edition, but may be used more generally.

For more information on Smart View, Essbase, Planning, Financial Management, Oracle Hyperion Financial Data Quality Management ERP Integration Adapter for Oracle Applications, Oracle General Ledger, Oracle BI EE, Oracle's Hyperion Reporting and Analysis, and Oracle Crystal Ball Enterprise Performance Management, Fusion Edition, please see the product documentation.

### HypIsCellWritable

**Data Sources:** Essbase, Planning, Financial Management, Oracle BI EE

**Modes:** ad hoc, data forms

#### Description

HypIsCellWritable() checks to see whether a cell is writable.

#### Syntax

HypIsCellWritable (vtSheetName [in], vtCellRange [out])

ByVal vtSheetName As Variant

ByVal vtCellRange As Variant

#### Parameters

**vtSheetName:** Name of the worksheet. If this argument is NULL or empty, the default sheet is used.

**vtCellRange:** Range of the cell (one cell only) whose writability must be checked.

## Return Value

VARIANT\_TRUE if the cell is writable, otherwise VARIANT\_FALSE.

## Example

```
Public Declare Function HypIsCellWritable Lib "HsAddin.dll" (ByVal
vtSheetName As Variant, ByVal vtCellRange As Variant) As Boolean

Sub TestIsCellWritable()

    Dim oRet As Boolean
    Dim oSheetName As String
    Dim oSheetDisp As Worksheet

    oSheetName = "Sheet1"
    Set oSheetDisp = Worksheets(oSheetName)
    oRet = HypIsCellWritable (Sheet1, oSheetDisp.Range("G2"))

End Sub
```

## HypIsSmartViewContentPresent

**Data Sources:** Essbase, Planning, Financial Management, Oracle BI EE

**Modes:** ad hoc, data forms, reports created from the Report Designer

## Description

HypIsSmartViewContentPresent() checks to see whether the sheet contains Smart View content.

## Syntax

HypIsSmartViewContentPresent(vtSheetName [in], pContentType [out])

ByVal vtSheetName As Variant

ByRef vtTypeOfContentsInSheet

## Parameters

**vtSheetName:** Name of the worksheet. If this argument is NULL or empty, the default sheet is used.

**pContentType:** Function returns appropriate type of content on the sheet. Possible values are in the enum as defined below.

```
Enum TYPE_OF_CONTENTS_IN_SHEET
    EMPTY_SHEET
    ADHOC_SHEET
    FORM_SHEET
```

```
INTERACTIVE_REPORT_SHEET
End Enum
```

## Return Value

VARIANT\_TRUE if the worksheet contains Oracle Hyperion Smart View for Office, Fusion Edition content, otherwise VARIANT\_FALSE.

## Example

```
Public Declare Function HypIsSmartViewContentPresent Lib
"HsAddin.dll" (ByVal vtSheetName As Variant, _
ByRef vtTypeOfContentsInSheet As TYPE_OF_CONTENTS_IN_SHEET) As Boolean

Sub TestIsSVCCContentOnSheet()

    Dim oRet As Boolean
    Dim oContentType As TYPE_OF_CONTENTS_IN_SHEET
    Dim oSheetName As String
    Dim oSheetDisp As Worksheet

    oSheetName = "Sheet1"
    Set oSheetDisp = Worksheets(oSheetName)
    oRet = HypIsSmartViewContentPresent (Sheet1, oContentType)

End Sub
```

## HypGetDimMbrsForDataCell

**Data Sources:** Essbase, Planning, Financial Management, Oracle BI EE

**Modes:** ad hoc, data forms

## Description

HypGetDimMbrsForDataCell() retrieves the entire set of dimension members for a data cell.

## Syntax

HypGetDimMbrsForDataCell (vtSheetName [in], vtCellRange [in], vtServerName [out], vtAppName [out], vtCubeName [out], vtFormName [out], vtDimensionNames [out], vtMemberNames [out])

ByVal vtSheetName As Variant

ByVal vtCellRange As Variant

ByRef vtServerName As Variant

ByRef vtAppName As Variant

ByRef vtCubeName As Variant  
ByRef vtFormName As Variant  
ByRef vtDimensionNames As Variant  
ByRef vtMemberNames As Variant

## Parameters

**vtSheetName:** Name of the worksheet. If this argument is NULL or empty, the default sheet is used.

**vtCellRange:** Range of the cell (one cell only) whose writability must be checked.

**pvtServerName:** Name of the server the associated connection on the sheet is connected to

**pvtApplicationName:** Name of the application the associated connection on the sheet is connected to

**pvtCubeName:** Name of the cube /database (Plan Type in Planning) the associated connection on the sheet is connected to

**pvtFormName:** Name of the form the associated connection on the sheet is connected to (in ad hoc grids, this is returned as empty string)

**pvtDimensionNames:** Array of dimension names

**pvtMemberNames:** Array of member names

## Return Value

Returns SS\_OK if the function succeeded; otherwise, the appropriate error code.

## Example

```
Public Declare Function HypGetDimMbrsForDataCell Lib "HsAddin.dll" (ByVal vtSheetName As Variant, ByVal vtCellRange As Variant, _ ByRef vtServerName As Variant, ByRef vtAppName As Variant, _ ByRef vtCubeName As Variant, ByRef vtFormName As Variant, _ ByRef vtDimensionNames As Variant, ByRef vtMemberNames As Variant) As Long
```

```
Sub TestGetDimMbrsForDataCell()
```

```
Dim oRet As Long  
Dim oSheetName As String  
Dim oSheetDisp As Worksheet  
Dim vtDimNames As Variant  
Dim vtMbrNames As Variant  
Dim vtServerName As Variant  
Dim vtAppName As Variant  
Dim vtCubeName As Variant  
Dim vtFormName As Variant  
Dim lNumDims As Long  
Dim lNumMbrs As Long
```

```

Dim sPrintMsg As String

oSheetName = "Sheet1"
Set oSheetDisp = Worksheets(oSheetName$)
oRet = HypGetDimMbrsForDataCell("", oSheetDisp.Range("B2"), vtServerName,
vtAppName, vtCubeName, vtFormName, vtDimNames, vtMbrNames)

If (oRet = SS_OK) Then
    If IsArray(vtDimNames) Then
        lNumDims = UBound(vtDimNames) - LBound(vtDimNames) + 1
    End If

    If IsArray(vtMbrNames) Then
        lNumMbrs = UBound(vtMbrNames) - LBound(vtMbrNames) + 1
    End If

    sPrintMsg = "Number of Dimensions = " & lNumDims & " Number of Members
= " & lNumMbrs & " Cube Name - " & vtCubeName
    MsgBox (sPrintMsg)
End If

End Sub

```

## HypCaptureFormatting

**Data Sources:** Essbase, Planning, Financial Management, Oracle BI EE

**Modes:** ad hoc

### Description

HypCaptureFormatting() applies grid formatting to cells created by zooming in.

### Syntax

HypCaptureFormatting (vtSheetName [in], vtRange [in])

ByVal vtSheetName As Variant

ByVal vtSelectionRange As Variant

### Parameters

**vtSheetName:** Name of the worksheet. If this argument is NULL or empty, the default sheet is used.

**vtRange:** Range of the cell(s) for which formatting needs to be captured. (Multiple ranges are supported)

### Return Value

Returns SS\_OK if the function succeeded; otherwise, the appropriate error code.

## Example

```
Public Declare Function HypCaptureFormatting Lib "HsAddin.dll" (ByVal
vtSheetName As Variant, ByVal vtSelectionRange As Variant) As Long

Sub TestCaptureFormatting()

    Dim oRet As Long
    Dim oSheetName As String
    Dim oSheetDisp As Worksheet

    oSheetName = "Sheet1"
    Set oSheetDisp = Worksheets(oSheetName)
    oRet = HypCaptureFormatting ("", oSheetDisp.Range("B2"))

    MsgBox (oRet)

End Sub
```

## HypRemoveCapturedFormats

**Data Sources:** Essbase, Planning, Financial Management, Oracle BI EE

**Modes:** ad hoc

### Description

HypRemoveCapturedFormats() removes captured formats.

**Note:** Users must refresh the grid before the original formatting is applied.

### Syntax

HypRemoveCapturedFormats (vtSheetName [in], vtRemoveAllCapturedFormats [in], vtSelectionRange [in])

ByVal vtSheetName As Variant

ByVal vtRemoveAllCapturedFormats As Variant

ByVal vtSelectionRange As Variant

### Parameters

**vtSheetName:** Name of the worksheet. If this argument is NULL or empty, the default sheet is used.

**vtRemoveAllCapturedFormats:** Boolean to indicate whether all captured formats on the grid should be deleted. (If this parameter is true, the next parameter value is not used, so users can pass NULL for vtSelectionRange.)

**vtSelectionRange:** Range of the cell(s) for which formatting needs to be captured. (Multiple ranges are supported)

## Return Value

Returns SS\_OK if the function succeeded; otherwise, the appropriate error code.

## Example

```
Public Declare Function HypRemoveCapturedFormats Lib "HsAddin.dll" (ByVal vtSheetName As Variant, ByVal vtbRemoveAllCapturedFormats As Variant, ByVal vtSelectionRange As Variant) As Long
```

```
Sub TestRemoveCaptureFormatting()  
  
    Dim oRet As Long  
    Dim oSheetName As String  
    Dim oSheetDisp As Worksheet  
  
    oSheetName = "Sheet1"  
    Set oSheetDisp = Worksheets(oSheetName$)  
    'oRet = HypRemoveCapturedFormats("", False, oSheetDisp.Range("B2"))  
    oRet = HypRemoveCapturedFormats("", True, Null)  
    MsgBox (oRet)  
  
End Sub
```

## HypListCalcScriptsEx

**Data Sources:** Essbase, Planning

**Modes:** ad hoc, data forms

## Description

HypListCalcScriptsEx() lists all business rules.

**Note:** See [Usage](#) under HypListCalcScriptsEx for more information and example.

## Syntax

```
HypListCalcScriptsEx (vtSheetName [in], vtbRuleOnForm [in], pvtArCubeNames [out],  
pvtArBRNames [out], pvtArBRTypes [out], pvtArBRHasPrompts [out],  
pvtArBRNeedPageInfo [out], pvtArBRHidePrompts [out])
```

ByVal vtSheetName As Variant

ByVal vtbRuleOnForm As Variant

ByRef vtCubeNames As Variant

ByRef vtBRNames As Variant  
ByRef vtBRTypes As Variant  
ByRef vtBRHasPrompts As Variant  
ByRef vtBRNeedsPageInfo As Variant  
ByRef vtBRHidePrompts As Variant

## Parameters

**vtSheetName:** Name of the worksheet. If this argument is NULL or empty, the default sheet is used.

**vtbRuleOnForm:** Boolean to indicate whether the user wants to list business rules associated only with the form opened on the sheet. If this argument is false, all the business rules associated with the application will be returned.

**pvtArCubeNames:** Array of cube names (Plan Types in Planning) associated with the Business rules

**pvtArBRNames:** Array of Business Rule Names

**pvtArBRTypes:** Array of Business Rule Types

**pvtArBRHasPrompts:** – Array of booleans indicating whether the Business Rule has Run Time Prompts

**pvtArBRNeedPageInfo:** Array of booleans indicating whether the Business Rule needs Page Information on the sheet to be run

**pvtArBRHidePrompts:** Array of booleans indicating whether the RTPs for this Business Rule are hidden

## Return Value

Returns SS\_OK if the function succeeded; otherwise, the appropriate error code.

## HypExecuteCalcScriptEx

**Data Sources:** Essbase, Planning

**Modes:** ad hoc, data forms

## Description

HypExecuteCalcScriptEx() executes the selected business rule.

## Syntax

HypExecuteCalcScriptEx(vtSheetName [in], vtCubeName [in], vtBRName [in], vtBRType [in], vtbBRHasPrompts [in], vtbBRNeedPageInfo [in], vtRTPNames() [in], vtRTPValues() [in], vtbShowRTPDlg [in], vtbRuleOnForm [in], vtbBRRanSuccessfully [out], vtCubeName [out], vtBRName [out], vtBRType [out], vtbBRHasPrompts [out], vtbBRNeedPageInfo [out], vtbBRHidePrompts [out], vtRTPNamesUsed [out], vtRTPValuesUsed [out])

ByVal vtSheetName As Variant

ByVal vtCubeName As Variant

ByVal vtBRName As Variant

ByVal vtBRType As Variant

ByVal vtbBRHasPrompts As Variant

ByVal vtbBRNeedPageInfo As Variant

ByRef vtRTPNames() As Variant

ByRef vtRTPValues() As Variant

ByVal vtbShowRTPDlg As Variant

ByVal vtbRuleOnForm As Variant

ByRef vtbBRRanSuccessfully As Variant

ByRef vtCubeName As Variant

ByRef vtBRName As Variant

ByRef vtBRType As Variant

ByRef vtbBRHasPrompts As Variant

ByRef vtbBRNeedPageInfo As Variant

ByRef vtbBRHidePrompts As Variant

ByRef vtRTPNamesUsed As Variant

ByRef vtRTPValuesUsed As Variant

## Parameters

**vtSheetName:** Name of the worksheet. If this argument is NULL or empty, the default sheet is used.

**vtCubeName:** Cube Name (Plan type incase of Planning) Business Rule is associated with

**vtBRName:** Business Rule Name of the BR to be run

**vtBRType:** Business Rule Type for the BR to be run

**vtbBRHasPrompts:** Boolean indicating if the Business Rule has RTPs

**vtbNeedPageInfo:** Boolean indicating if the Business Rule needs Page Info to be run (Get this info either from HypListCalcScriptsEx or from prior run of HypExecuteCalcScriptEx)

**ppRTPNames:** Array of RTP Names associated with the Business Rule

**ppRTPValues:** Array of RTP Values corresponding to the RTP Names

**vtbShowBRDlg:** Boolean indicating whether to show the Business Rule dialog box and let the user select the Business Rule to run or of automating execution of BR. If this flag is true, all the input parameters related to the BR are ignored. Recommendation: This flag should be true when running the BR for the first time and then using the output parameters to automate the execution of the same BR from second time onwards. In this case, this flag should be false second time

**vtbRuleOnForm:** Boolean indicating if the Business Rule is associated to the form opened on active sheet

**pvtbBRRanSuccessfully:** Return boolean value indicating if the last Business Rule ran successfully

**pvtCubeNameUsed:** Cube name (Plan Types incase of Planning) associated with the last run Business Rule

**pvtBRNameUsed:** Business Rule Name of the last run Business Rule

**pvtBRTypeUsed:** Business Rule type of the last run Business Rule

**pvtbBRHasPrompts:** Boolean indicating if the last run Business Rule has RTPs

**pvtbBRNeedPageInfo:** Boolean indicating if the last run Business Rule requires Page information

**pvtbBRHidePrompts:** Boolean indicating if the last run Business Rule has hidden RTPs

**pvtRTPNamesUsed:** Array of RTP Names used to run last run Business Rule

**pvtRTPValuesUsed:** Array of RTP Values associated with RTP names used to run last run Business Rule

## Return Value

Returns SS\_OK if the function succeeded; otherwise, the appropriate error code.

## Example

```
Declare Function HypListCalcScriptsEx Lib "HsAddin.dll" (ByVal vtSheetName As Variant, ByVal vtbRuleOnForm As Variant, ByRef vtCubeNames As Variant, ByRef vtBRNames As Variant, ByRef vtBRTypes As Variant, ByRef vtbBRHasPrompts As Variant, ByRef vtBRNeedsPageInfo As Variant, ByRef vtbBRHidePrompts As Variant) As Long
```

```
Declare Function HypExecuteCalcScriptEx Lib "HsAddin.dll" (ByVal vtSheetName As Variant, ByVal vtCubeName As Variant, ByVal vtBRName As Variant, ByVal vtBRType As Variant, ByVal vtbBRHasPrompts As Variant, ByVal vtbBRNeedPageInfo As Variant, ByRef vtRTPNames() As Variant, ByRef vtRTPValues() As Variant, ByVal vtbShowRTPDlg As Variant, ByVal vtbRuleOnForm As Variant, ByRef vtbBRRanSuccessfully As Variant, ByRef vtCubeName As Variant, ByRef vtBRName As Variant, ByRef vtBRType As Variant, ByRef vtbBRHasPrompts As Variant, ByRef vtbBRNeedPageInfo As
```

```
Variant, ByRef vtBRHidePrompts As Variant, ByRef vtRTPNamesUsed As  
Variant, ByRef vtRTPValuesUsed As Variant) As Long
```

```
Sub TestListAndExecuteCalcScriptsEx()
```

```
Dim oRet As Long  
Dim oSheetName As String  
Dim oSheet As Worksheet  
Dim vtCubeNames As Variant  
Dim vtBRNames As Variant  
Dim vtBRTypes As Variant  
Dim vtBRHasPrompts As Variant  
Dim vtBRNeedsPageInfo As Variant  
Dim vtBRHidePrompts As Variant  
Dim sAllCalcs As String  
Dim sCalcName As String  
Dim bNeedPageInfo As Variant  
Dim vtInRTPNames() As Variant  
Dim vtInRTPValues() As Variant  
Dim vtOutRTPNames As Variant  
Dim vtOutRTPValues As Variant  
Dim vtbBRRanSuccessfully As Variant  
Dim vtbBRRanSuccessfully2 As Variant  
Dim vtOutCubeName As Variant  
Dim vtOutBRName As Variant  
Dim vtOutBRType As Variant  
Dim bBRHasPrompts As Variant  
Dim bBRNeedPageInfo As Variant  
Dim bBRHidePrompts As Variant  
Dim bShowDlg As Variant  
Dim bRuleOnForm As Variant
```

```
'Set oSheet = ActiveSheet  
'oSheetName = oSheet.Name  
oSheetName = "Sheet3"
```

```
oRet = HypListCalcScriptsEx (oSheetName, False, vtCubeNames, vtBRNames,  
vtBRTypes, vtBRHasPrompts, vtBRNeedsPageInfo, vtBRHidePrompts)  
If (oRet = 0) Then
```

```
    If IsArray(vtBRNames) Then  
        lNumMbrs = (UBound(vtBRNames) - LBound(vtBRNames) + 1)  
    End If
```

```
    sPrintMsg = "Number of Calc Scripts = " & lNumMbrs  
    MsgBox (sPrintMsg)
```

```
    'Start Executing the Calc Script
```

```
    bShowDlg = True  
    bRuleOnForm = False  
    iScript = 1
```

```
    oRet = HypExecuteCalcScriptEx (oSheetName, vtCubeNames(iScript),  
vtBRNames(iScript), vtBRTypes(iScript), vtBRHasPrompts(iScript),  
vtBRNeedsPageInfo(iScript), vtInRTPNames, vtInRTPValues, bShowDlg,  
bRuleOnForm, vtbBRRanSuccessfully, vtOutCubeName, vtOutBRName,  
vtOutBRType, bBRHasPrompts, bBRNeedPageInfo, bBRHidePrompts, vtOutRTPNames,
```

```

vtOutRTPValues)
    If (oRet = 0) Then
        MsgBox ("Last BR ran successfully - " & vtbBRRanSuccessfully)

        If (vtbBRRanSuccessfully = True) Then
            bShowDlg = False
            bRuleOnForm = False

            If IsArray(vtOutRTPNames) And IsArray(vtOutRTPValues) Then
                lNumRTPNames = (UBound(vtOutRTPNames) -
                    LBound(vtOutRTPNames) + 1)
                lNumRTPVals = (UBound(vtOutRTPValues) -
                    LBound(vtOutRTPValues) + 1)
            End If

            If (lNumRTPNames > 0) Then
                ReDim vtInRTPNames(lNumRTPNames - 1) As Variant
                ReDim vtInRTPValues(lNumRTPNames - 1) As Variant

                For iRTPs = 0 To lNumRTPNames - 1
                    sBRName = vtOutRTPNames(iRTPs)
                    sBRVal = vtOutRTPValues(iRTPs)

                    vtInRTPNames(iRTPs) = sBRName
                    vtInRTPValues(iRTPs) = sBRVal
                Next iRTPs
            End If

            oRet = HypExecuteCalcScriptEx (oSheetName, vtOutCubeName,
                vtOutBRName, vtOutBRType, bBRHasPrompts , bBRNeedPageInfo, vtInRTPNames,
                vtInRTPValues, bShowDlg, bRuleOnForm, vtbBRRanSuccessfully2,
                vtOutCubeName, vtOutBRName, vtOutBRType, bBRHasPrompts, bBRNeedPageInfo,
                bBRHidePrompts, vtOutRTPNames, vtOutRTPValues)
            MsgBox ("Automated BR ran successfully - " &
                vtbBRRanSuccessfully2)
        End If
    Else
        sPrintMsg = "Error - " & oRet
        MsgBox (sPrintMsg)
    End If
Else
    sPrintMsg = "Error - " & oRet
    MsgBox (sPrintMsg)
End If

End Sub

```

## Usage

You can use HypExecuteCalcScriptEx in four modes, depending on whether HypListCalcScriptsEx is called before HypExecuteCalcScriptEx.

If you do NOT call HypListCalcScriptsEx before HypExecuteCalcScriptEx, then the first time you call HypListCalcScriptsEx you should set the vtbShowBRDlg argument to true for the first usage and to false thereafter.

- When `vtbShowBRDlg` argument is true (mode 1):
  - In arguments: `vtSheetName`, `vtCubeName`, `vtbRuleOnForm` are used. `vtBRName`, `vtBRType`, `vtbBRHasPrompts`, `vtbNeedPageInfo`, `ppRTPNames`, `ppRTPValues` are ignored.
  - Behavior: The Business Rule dialog box displays all possible rules depending upon the `vtbRuleOnForm` value. When the user selects, runs and exits the Business Rule dialog box, the details of that Business Rule are filled in the out arguments and returned to the caller.
  - Out arguments: All out arguments are filled and returned to the caller so that they can be used in subsequent calls.
- When `vtbShowBRDlg` argument is false (mode 2):
  - In arguments: All in arguments are used.
  - Behavior: The business rule is run without displaying the Business Rule dialog box ,and the appropriate status is returned to the caller.
  - Out arguments: All out arguments are left unmodified as nothing needs to be passed on to the caller, who already has all the information to run this particular business rule.

If you DO call `HypListCalcScriptsEx` before `HypExecuteCalcScriptEx`, then when `HypListCalcScriptsEx` is called, users get information about all business rules and runtime prompts (RTP), if any.

If a user runs a business rule that has no RTP, `HypExecuteCalcScriptEx` can be called with `vtbShowBRDlg` argument as false and provides all other information as the in arguments.

If a user runs a business rule that has an RTP, `HypExecuteCalcScriptEx` must be called with `vtbShowBRDlg` as true so that the business rule and its RTPs can be displayed and the user can select the RTP values to run the business rule. (Note: in `inPlanning`, the RTP flag may be true for a business rule when there are no RTPs to be displayed.)

- If the cube name, business rule name and business rule type are passed as empty in `HypExecuteCalcScriptEx` (mode 3), the Business Rule dialog box is displayed and all business rules are shown depending upon `vtbRuleOnForm` argument. All else is the same as mode 1.
- If the cube name, business rule name and business rule type are passed with filled values in `HypExecuteCalcScriptEx` (mode 4), the Business Rule dialog box is displayed and only the passed business rule (business rule name for the provided cube name) is displayed along with its RTPs. All else is the same as mode 1.

## HypGetCellRangeForMbrCombination

**Data Sources:** Essbase, Planning, Financial Management, Oracle BI EE

**Modes:** ad hoc, data forms

## Description

HypGetCellRangeForMbrCombination() retrieves the cell range for the selected combination of members.

## Syntax

HypGetCellRangeForMbrCombination (vtSheetName [in], ppvtDimNames [in], ppvtMbrNames [in], pvtCellIntersectionRange [out])

By Val vtSheetName As Variant

ByRef vtDimNames As Variant

ByRef vtMbrNames As Variant

ByRef vtCellIntersectionRange As Variant

## Parameters

**vtSheetName:** Name of the worksheet. If this argument is NULL or empty, the default sheet is used.

**ppvtDimNames:** Array of dimension names

**ppvtMbrNames:** Array of member names corresponding to the dimensions (in the same order)

**pvtCellIntersectionRange:** Range of the cell(s) on the grid

## Return Value

Returns SS\_OK if the function succeeded; otherwise, the appropriate error code.

## Example

```
Public Declare Function HypGetCellRangeForMbrCombination Lib
"HsAddin.dll" (ByVal vtSheetName As Variant, ByRef vtDimNames() As Variant,
ByRef vtMbrNames() As Variant, ByRef vtCellIntersectionRange As Variant) As
Long
```

```
Sub GetCellRangeForMbrCombination()
```

```
    Dim oRet As Long
    Dim oSheetName As String
    Dim oSheetDisp As Worksheet
    Dim vtDimNames(3) As Variant
    Dim vtMbrNames(3) As Variant
    Dim vtReturnCellRange As Variant
    Dim oRange As Range
```

```
    'oSheetName = "Sheet1"
    'Set oSheetDisp = Worksheets(oSheetName$)
```

```

vtDimNames(0) = "Measures"
vtDimNames(1) = "Market"
vtDimNames(2) = "Year"
vtDimNames(3) = "Product"
'vtDimNames(4) = ""

vtMbrNames(0) = "Sales"
vtMbrNames(1) = "New York"
vtMbrNames(2) = "Year"
vtMbrNames(3) = " Product"
'vtMbrNames(4) = ""

```

```
oRet = HypGetCellRangeForMbrCombination ("", vtDimNames, vtMbrNames,
vtReturnCellRange)
```

```

If (oRet = 0) Then
    Set oRange = vtReturnCellRange
End If

```

## HypIsDataModified

**Data Sources:** Essbase, Planning, Financial Management, Oracle BI EE

**Modes:** ad hoc, data forms

### Description

HypIsDataModified() checks to see whether any data cells have been modified but not yet submitted.

### Syntax

```
HypIsDataModified (vtSheetName [in])
```

```
By Val vtSheetName As Variant
```

### Parameters

**vtSheetName:** Name of the worksheet. If this argument is NULL or empty, the default sheet is used.

### Return Value

VARIANT\_TRUE if the sheet contains any data cells that have been updated and not yet submitted, otherwise VARIANT\_FALSE.

### Example

```
Public Declare Function HypIsDataModified Lib "HsAddin.dll" (ByVal
```

```

vtSheetName As Variant) As Boolean

Sub TestIsSheetDirty()

    Dim oRet As Boolean

    oRet = HypIsDataModified("")
    MsgBox (oRet)

End Sub

```

## HypIsFreeForm

**Data Sources:** Oracle Essbase, Oracle Hyperion Planning, Fusion Edition, Financial Management, Oracle Business Intelligence Enterprise Edition

**Modes:** ad hoc, data forms (ad hoc only in Oracle Hyperion Financial Management, Fusion Edition)

### Description

HypIsFreeForm() checks to see whether the worksheet is in free form.

### Syntax

HypIsFreeForm (vtSheetName [in])

By Val vtSheetName As Variant

### Parameters

**vtSheetName:** Name of the worksheet. If this argument is NULL or empty, the default sheet is used.

### Return Value

VARIANT\_TRUE if the cell is in free form state, i.e., either member cells or comment cells have been modified and the sheet has not been refreshed, otherwise VARIANT\_FALSE.

### Example

```

Public Declare Function HypIsFreeForm Lib "HsAddin.dll" (ByVal vtSheetName
As Variant) As Boolean
Sub TestIsSheetFreeForm()

Sub HypIsFreeForm()
    Dim oRet As Boolean

    oRet = HypIsFreeForm("")
    MsgBox (oRet)

```

End Sub



## COPYRIGHT NOTICE

Smart View New Features, 11.1.1.3

Copyright © 2004, 2009, Oracle and/or its affiliates. All rights reserved.

Authors: EPM Information Development Team

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable: U.S. GOVERNMENT RIGHTS: Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

This software and documentation may provide access to or information on content, products and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third party content, products and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third party content, products or services.