EASYCATALOG

FOR ADOBE® INDESIGN®

SCRIPTING REFERENCE

65bit Software Ltd

Revision History

Version	Date	Remarks
2.0.0	13 July 2005	First draft for InDesign CS2 modifications.
2.0.1	8 Nov. 2005	Synchronised English and French versions of manual
2.1.0	13 March 2006	Modifications for the release of EasyCatalog 2.1
2.1.1	17 August 2007	Re-formatted to be consistent with other manuals.
2.1.2	12 March 2008	Updated to include new methods and functions.
2.1.3	18 May 2010	Further updates with new methods.

CONTENTS

GETTING STARTED

OBJECT MODEL

SCRIPTING REFERENCE

CHAPTER 3

CHAPTER 1

GETTING STARTED

WELCOME

Thank you for downloading the EasyCatalog Scripting module for EasyCatalog.

65bit Software are committed to providing high quality software for Adobe InDesign, and appreciate the time you take to evaluate our products. All feedback is welcome, good or bad. Please email `support@65bit.com`. If you have any questions relating to this or any other product, or need any help, please use the support form on our website.

WHAT THE SCRIPTING MODULE?

The Scripting module is an add-on module for EasyCatalog that adds powerful scripting functionality to EasyCatalog. With this module EasyCatalog can be integrated tightly with your own custom-built solutions and even form part of a totally automated production process driven from Javascript, Applescript or applications such as Visual Basic.

INSTALLATION

The Scripting Module is an optional module for EasyCatalog, and is included in the main EasyCatalog installer. During installation, the Scripting Module will be available as an installable option.

ASSUMPTIONS

This manual assumes that you:

- Have a working knowledge of EasyCatalog.
- Have a working knowledge of your chosen scripting language - either JavaScript, VB Script (on Windows) or AppleScript (on the Macintosh).

NEED MORE?

We are constantly looking for ways to improve our software, so if you have any scripting requirements that are not covered by functionality of this module, please let us know. If you have any questions, please contact us using the support form at our web-site.

PURCHASING

If you've purchased or downloaded EasyCatalog from one of our partners, please obtain your serial number through them. Alternatively if you downloaded from the 65bit web site, serial numbers can be purchased through our web-store:

http://www.65bit.com/purchase/purchase.shtm

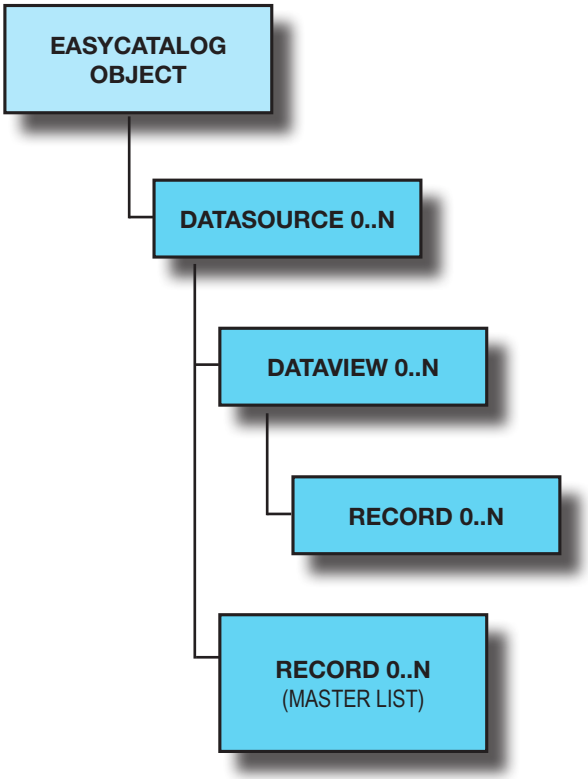
Serial numbers purchased via our web-store will be issued on completion of the credit card transaction, and will activate in around 5 - 10 minutes.

CHAPTER 2

OBJECT MODEL

OVERVIEW

All objects are accessed via the "EasyCatalog Object", which has a number of methods and gives access to a collection of DataSources. Each DataSource has a set of Records and a collection of DataViews. DataViews are represented as Panels in the desktop version of InDesign. Each DataView has a collection or records, which represent those currently held in the DataView.



DataViews, Records and DataSources can all be referenced by name. Records are identified by the value of their key field.

CHAPTER 3

SCRIPTING REFERENCE

EASYCATALOG OBJECT

CONTENT TRACKING

Get or set EasyCatalog's content tracking property.

AppleScript

```
tell application "Adobe InDesign CS5"
	tell EasyCatalog object
		set content tracking to false
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
myEasyCatalog.contentTracking = false;
alert (myEasyCatalog.contentTracking);
```

Visual Basic

```
Set myInDesign = CreateObject("Indesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
MsgBox (myEasyCatalog.Content Tracking)
```

EASYCATALOG OBJECT
(CONTINUED)

EXCHANGE TAGS

Exchange the key field of tags in a given document. The format of the input file is old key,new keyCR. A log file is produced by this function listing which keys were exchanged.

Parameters:

doc:	document object
file path:	path to input file
log path:	path to log file

AppleScript

```
tell application "Adobe InDesign CS5"
	set myDocument to active document
	tell EasyCatalog object
		exchange tags doc myDocument file path "Macintosh HD:tagfile.
		csv" log path "Macintosh HD:log.txt"
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDoc = app.activeDocument;
myEasyCatalog.exchangeTags(myDoc, "Macintosh HD:tagfile.csv", "Macintosh
    HD:log.txt");
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myDocument = myInDesign.ActiveDocument
Set myEasyCatalog = myInDesign.EasyCatalogObject
err = myEasyCatalog.ExchangeTags(myDocument, "c:\tagfile.csv", "c:\log.txt")
```

EASYCATALOG OBJECT (CONTINUED)

FIELD MARKER VISIBILITY

Used to get or set the setting of EasyCatalog's 'Field Marker Visibility' flag for a document.

AppleScript

```
tell application "Adobe InDesign CS5"
	tell active document
		tell EasyCatalog object
			set field marker visibility to false
		end tell
	end tell
end tell
```

JavaScript

```
var myDoc = app.activeDocument;
myDoc.easycatalogObject.fieldMarkerVisibility = false;
```

Visual Basic

```
Set myInDesign = CreateObject("Indesign.Application")
Set myDocument = myInDesign.ActiveDocument
MsgBox (myDocument.EasyCatalogObject.FieldMarkerVisibility)
```

EASYCATALOG OBJECT (CONTINUED)

LOAD FILE

Create a new data source from a delimited text file.

Parameters:

name:	the name of the new data source
file path:	path to the file to load
field separator:	character to use for the field separator
record separator:	character to use for the record separator
key field:	the name of the key field. Compount key fields can be separated by '||'

AppleScript

```
tell application "Adobe InDesign CS5"
	tell EasyCatalog object
		load file name "data source name" file path "file path here" field
		separator ";" record separator "\r" key field "key field name here"
	end tell
end tell
```

JavaScript

```
var myEC = app.easycatalogObject;
myEC.loadFile ("data source name", "file path here", ";", "\r", "key field name here")
```

Visual Basic

```
Set myInDesign = CreateObject("Indesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDS = myEasyCatalog.LoadFile("data source name", "file path here", ";", "\r",
    "key field name here")
```

EASYCATALOG OBJECT (CONTINUED)

PURGE DATA SOURCE

Remove a data source from the workspace folder, closing any open panels.

Parameters:

name: name of the data source to purge

AppleScript

```
tell application "Adobe InDesign CS5"
	tell EasyCatalog object
		purge data source name "Stock.csv"
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
myEasyCatalog.purgeDataSource("Stock.csv");
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
myEasyCatalog.PurgeDataSource("Stock.CSV")
```

EASYCATALOG OBJECT (CONTINUED)

REGISTER

Activate a module using the given serial number.

Parameters:

serial number: serial number for the module to register

AppleScript

```
tell application "Adobe InDesign CS5"
	tell EasyCatalog object
		register serial number "ECxxxxx"
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
myEasyCatalog.registerSerialNumber("ECxxxxx");
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
myEasyCatalog.Register("ECxxxxx")
```

EASYCATALOG OBJECT (CONTINUED)

REPORT TAGS

Create a file listing all tags in the specified document. Information written to this file includes page number, data source name, key value, field name and field content.

Parameters:

document:	document object
file path:	path to the output file to be generated

AppleScript

```
tell application "Adobe InDesign CS5"
	set myDocument to active document
	tell EasyCatalog object
		report tags doc myDocument file path "Macintosh HD:tags.txt"
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDoc = app.activeDocument;
myEasyCatalog.reportTags(myDoc, "Macintosh HD:Tags.txt");
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myDocument = myInDesign.ActiveDocument
Set myEasyCatalog = myInDesign.EasyCatalogObject
err = myEasyCatalog.ReportTags(myDocument, "c:\tags.txt")
```

EASYCATALOG OBJECT
(CONTINUED)

SELECTED DATA VIEW

Determine the selected Data View. Works on the desktop version of EasyCatalog only *(Read Only)*

AppleScript

```
tell application "Adobe InDesign CS5"
	tell EasyCatalog object
		tell (selected data view)
			set myName to data view name
		end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDV = myEasyCatalog.selectedDataView();
alert (myDV.name);
```

Visual Basic

```
Set myInDesign = CreateObject("Indesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDV = myEasyCatalog.SelectedDataView()
```

EASYCATALOG OBJECT (CONTINUED)

SERIAL NUMBER

Serial number of a registered copy of EasyCatalog *(Read Only)*.

AppleScript

```
tell application "Adobe InDesign CS5"
	tell EasyCatalog object
	set mySerialNumber to serial number
		display dialog mySerialNumber
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var mySerialNumber = myEasyCatalog.serialNumber;
alert (mySerialNumber);
```

Visual Basic

```
Set myInDesign = CreateObject("Indesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
MsgBox (myEasyCatalog.SerialNumber)
```

EASYCATALOG OBJECT (CONTINUED)

TAG PAGE ITEM

Tag the given page item

Parameters:

page item: page item to tag data source: data source name field: field name key: key value

AppleScript

```
tell application "Adobe InDesign CS5"
set myDocument to active document
set myDocument to active document
set myFrame to text frame 1 of myDocument
tell EasyCatalog object
        tag page item page item myFrame data source "test.csv" field "SKU" key
        "PRODUCTKEY"
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myFrame = app.documents[0].pages[0].pageItems[0]
myEasyCatalog.tagPageItem(myFrame, "test.csv", "SKU", "PRODUCTKEY")
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myFrame = myDoc.Pages.Item(1).TextFrames.Item(1)
myEasyCatalog.TagPageItem(myFrame, "test.csv", "SKU", "PRODUCTKEY")
```

EASYCATALOG OBJECT (CONTINUED)

TAG TEXT

Create a tag the specified text range

Parameters:

story offset: insertion point of a story length: length of the tag data source: data source name field: field name key: key value

AppleScript

```
tell application "Adobe InDesign CS5"
set myDocument to active document
set myDocument to active document
set myFrame to text frame 1 of myDocument
set myStory to parent story of myFrame
set myInsertionPoint to insertion point 1 of myStory
tell EasyCatalog object
        tag text range offset myInsertionPoint length 5 data source "test.csv" field
        "SKU" key "PRODUCTKEY"
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myInsertionPoint = app.documents[0].pages[0].textFrames[0].
    insertionPoints[0]
myEasyCatalog.tagPageItem(myInsertionPoint, "test.csv", "SKU", "PRODUCTKEY")
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myFrame = myDoc.Pages.Item(1).TextFrames.Item(1)
Set myInsertionPoint = myFrame.InsertionPoints.Item(1)
myEasyCatalog.TagPageItem(myInsertionPoint, "test.csv", "SKU", "PRODUCTKEY")
```

EASYCATALOG OBJECT (CONTINUED)

UPDATE FURNITURE

Update furniture items for the given document.

Parameters:

document: document object

AppleScript

```
tell application "Adobe InDesign CS5"
	set myDocument to active document
	tell EasyCatalog object
		update furniture doc myDocument
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDoc = app.activeDocument;
myEasyCatalog.updateFurniture(myDoc);
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myDocument = myInDesign.ActiveDocument
Set myEasyCatalog = myInDesign.EasyCatalogObject
myEasyCatalog.UpdateFurniture(myDocument)
```

EASYCATALOG OBJECT (CONTINUED)

UPDATE FURNITURE ON PAGE

Update furniture for the given document, starting at the given page

Parameters:

page item: page item object

AppleScript

```
tell application "Adobe InDesign CS5"
	set myDocument to active document
	tell EasyCatalog object
		update furniture on page doc myDocument page 1
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDoc = app.activeDocument;
myEasyCatalog.updateFurnitureOnPage(myDoc, 1);
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myDocument = myInDesign.ActiveDocument
Set myEasyCatalog = myInDesign.EasyCatalogObject
myEasyCatalog.UpdateFurnitureOnPage(myDocument, 1)
```

EASYCATALOG OBJECT (CONTINUED)

UPDATE TAGS

Update tags in the page given item. Tags are updated irrespective of which datasource they belong to, providing the datasource is available.

Parameters:

page item: page item object

AppleScript

```
tell application "Adobe InDesign CS5"
	set myDocument to active document
	set myFrame to text frame 1 of myDocument
	tell EasyCatalog object
		update tags page item myFrame
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDoc = app.activeDocument;
var myFrame = app.documents[0].pages[0].textFrames[0]
myEasyCatalog.updateTags(myFrame);
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myDocument = myInDesign.ActiveDocument
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myFrame = myDocument.Pages.Item(1).TextFrames.Item(1)
myEasyCatalog.UpdateTags(myFrame)
```

EASYCATALOG OBJECT
(CONTINUED)

WORKSPACE FOLDER

The active workspace folder location. The workspace folder stores all data source information, including field options, a snapshot of data and default view configurations.

AppleScript

```
tell application "Adobe InDesign CS5"
	tell EasyCatalog object
		set myWorkspace to workspace folder
		display dialog myWorkspace
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myWorkspace = myEasyCatalog.workspaceFolder;
alert (myWorkspace);
```

Visual Basic

```
Set myInDesign = CreateObject("Indesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
MsgBox (myEasyCatalog.WorkspaceFolder)
```

DATA SOURCE OBJECT

ADD CUSTOM FIELD

Adds a custom field to the data source

Parameters:

field name:	name of the field, must be unique
field value:	contents of the custom field

AppleScript

```
tell application "Adobe InDesign CS5"
	set myDocument to active document
	tell EasyCatalog object
		tell DataSource "Stock.csv"
			add custom field field name "Group Max" field value
			"GROUPMAX(Category,Price)"
		end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDoc = app.activeDocument;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
myDS.addCustomField("Group Max", "GROUPMAX(Category,Price)");
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myDoc = myInDesign.ActiveDocument
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDS = myEasyCatalog.DataSources.Item("Stock.csv")
myDS.AddCustomField("Group Max", "GROUPMAX(Category,Price)")
```

DATA SOURCE OBJECT (CONTINUED)

ADOPT

Adopt tags from the specified document. Adoption links tags in the target document to this datasource where field name match and the record can be found.

Parameters:

document: document to adopt

AppleScript

```
tell application "Adobe InDesign CS5"
	set myDocument to active Document
	tell EasyCatalog object
		tell DataSource "Stock.csv"
			adopt doc myDocument
		end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDoc = app.activeDocument;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
myDS.adopt(myDoc);
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDS = myEasyCatalog.DataSources.Item("Stock.csv")
myDS.Adopt(myDoc)
```

DATA SOURCE OBJECT
(CONTINUED)

COUNT DELETED

Returns the number of records that were marked as deleted during the last synchronize.

AppleScript

```
tell application "Adobe InDesign CS5"
	tell EasyCatalog object
	tell DataSource "Stock.csv"
		set countUpdated to count deleted
		display dialog countUpdated
	end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
alert(myDS.countDeleted);
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
MsgBox(myEasyCatalog.DataSources.Item(1).CountDeleted)
```

DATA SOURCE OBJECT
(CONTINUED)

COUNT ERRORS

Returns the number of records that contain fields marked in error.

AppleScript

```
tell application "Adobe InDesign CS5"
	tell EasyCatalog object
	tell DataSource "Stock.csv"
		set countErrors to count errors
		display dialog countErrors
	end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
alert(myDS.countErrors);
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
MsgBox(myEasyCatalog.DataSources.Item(1).CountErrors)
```

DATA SOURCE OBJECT
(CONTINUED)

COUNT INSERTED

Returns the number of records that were inserted during the last synchronize.

AppleScript

```
tell application "Adobe InDesign CS5"
	tell EasyCatalog object
		tell DataSource "Stock.csv"
		set countInserted to count inserted
		display dialog countInserted
	end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
alert(myDS.countInserted);
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
MsgBox(myEasyCatalog.DataSources.Item(1).CountInserted)
```

DATA SOURCE OBJECT
(CONTINUED)

COUNT UPDATED

Returns the number of records that were updated during the last synchronize operation.

AppleScript

```
tell application "Adobe InDesign CS5"
	tell EasyCatalog object
	tell DataSource "Stock.csv"
		set countUpdated to count updated
		display dialog countUpdated
	end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
alert(myDS.countUpdated);
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
MsgBox(myEasyCatalog.DataSources.Item(1).CountUpdated)
```

DATA SOURCE OBJECT
(CONTINUED)

DATA SOURCE NAME

The name of the data source *(Read Only)*.

AppleScript

```
tell application "Adobe InDesign CS5"
	tell EasyCatalog object
		tell DataSource "Stock.csv"
			set myName to data source name
			display dialog myName
		end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
alert(myDS.name);
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
MsgBox(myEasyCatalog.DataSources.Item(1).DataSourceName)
```

DATA SOURCE OBJECT (CONTINUED)

DATA SOURCE SPECIFIER

Access the data source specifier, typically a path to a file in the case of file or XML data sources or an ODBC connection string.

AppleScript

```
tell application "Adobe InDesign CS5"
	tell EasyCatalog object
		tell DataSource "Stock.csv"
			set myName to data source specifier
			display dialog myName
		end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
alert(myDS.dataSourceSpecifier);
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
MsgBox(myEasyCatalog.DataSources.Item(1).DataSourceSpecifier)
```

DATA SOURCE OBJECT
(CONTINUED)

LOAD FIELD DEFINITIONS

Load a previously defined set of field definitions.

Parameters:

file path: path to a field definition file

AppleScript

```
tell application "Adobe InDesign CS5"
	tell EasyCatalog object
		tell DataSource "Stock.csv"
			load field definitions file path "Macintosh HD:fields.xml"
		end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
myDS.loa//dFieldDefinitions("Macintosh HD:Fields.xml");
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDS = myEasyCatalog.DataSources.Item("Stock.csv")
myDS.LoadFieldDefinitions("c:\fields.xml")
```

DATA SOURCE OBJECT (CONTINUED)

ODBC CONNECTION

Deprecated - Use odbc statement.

AppleScript

```
tell application "Adobe InDesign CS5"
	tell EasyCatalog object
	tell DataSource "Stock Database"
				odbc connection statement "Select * from stock"
	end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
myDS.odbcConnection("Select * from stock");
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDS = myEasyCatalog.DataSources.Item(1)
myDS.OdbcConnection("Select * from stock")
```

DATA SOURCE OBJECT (CONTINUED)

ODBC STATEMENT

Reconfigure the SQL statement associated with and ODBC based data source. Applies only to ODBC data sources.

AppleScript

```
tell application "Adobe InDesign CS5"
	tell EasyCatalog object
		tell DataSource "MyDataBase"
			set myName to odbc statement
			display dialog myName
		end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
alert(myDS.odbcStatement);
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
MsgBox(myEasyCatalog.DataSources.Item(1).OdbcStatement)
```

DATA SOURCE OBJECT (CONTINUED)

PICKUP

Identify and tag fields in the selected text range.

Parameters:

prefix:	search prefix string
suffix:	search suffix string
create new records:	create records true/false
relink:	allow relinking true/false

AppleScript

```
tell application "Adobe InDesign CS5"
	tell EasyCatalog object
		tell DataSource "Stock.csv"
			pickup prefix "[key]^t[description]" suffix "^p" with
			create new records without relink
		end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
myDS.pickup("[key]^t[description]", "^p", true, true);
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDS = myEasyCatalog.DataSources.Item("Stock.csv")
retval = myDS.Pickup("[key]^t[description]", "^p", TRUE, TRUE)
```

DATA SOURCE OBJECT (CONTINUED)

PURGE DELETED

Remove all records from the data source that have been marked as deleted.

AppleScript

```
tell application "Adobe InDesign CS5"
	tell EasyCatalog object
		tell DataSource "Stock.csv"
			purge deleted
		end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
myDS.purgeDeleted();
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDS = myEasyCatalog.DataSources.Item("Stock.csv")
myDS.PurgeDeleted
```

DATA SOURCE OBJECT (CONTINUED)

REPLACE FIELDS

Performs a search-and-replace on EasyCatalog fields.

Parameters:

document:	document object
search for:	name of the field to search for
replace with:	name of the field to replace

AppleScript

```
tell application "Adobe InDesign CS5"
	set myDocument to active document
	tell EasyCatalog object
		tell DataSource "Stock.csv"
			replace fields doc myDocument search for "Field 1"
			replace with "Field 2"
		end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDoc = app.activeDocument;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
myDS.replaceFields(myDoc, "Field 1", "Field 2");
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myDoc = myInDesign.ActiveDocument
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDS = myEasyCatalog.DataSources.Item("Stock.csv")
myDS.ReplaceFields(myDoc, "Field 1", "Field 2")
```

DATA SOURCE OBJECT
(CONTINUED)

SET PAGINATION OPTION

Sets a pagination option using a key. Keys are the attributes of the XML in the the "Pagination.xml" configuration file.

Parameters:

option name:	name of the option
option value:	value for the option

AppleScript

```
tell application "Adobe InDesign CS5"
	set myDocument to active document
	tell EasyCatalog object
		tell DataSource "Stock.csv"
			set pagination option option name "paginationtype"
			option value "0"
		end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDoc = app.activeDocument;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
myDS.setPaginationOption("paginationtype", "0");
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myDoc = myInDesign.ActiveDocument
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDS = myEasyCatalog.DataSources.Item("Stock.csv")
myDS.SetPaginationOption("paginationtype", "0")
```

DATA SOURCE OBJECT
(CONTINUED)

SYNCHRONIZE DOCUMENT

Compares the given document content and flags fields which are placed or have errors.

Parameters:

document: document object

AppleScript

```
tell application "Adobe InDesign CS5"
	set myDocument to active document
	tell EasyCatalog object
		tell DataSource "Stock.csv"
			synchronize document doc myDocument
		end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDoc = app.activeDocument;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
myDS.synchronizeDocument(myDoc);
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myDoc = myInDesign.ActiveDocument
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDS = myEasyCatalog.DataSources.Item("Stock.csv")
myDS.SynchronizeWithDocument(myDoc)
```

DATA SOURCE OBJECT (CONTINUED)

SYNCHRONIZE WITH DATA SOURCE

Take a new snapshot of data from the datasource.

AppleScript

```
tell application "Adobe InDesign CS5"
	tell EasyCatalog object
		tell DataSource "Stock.csv"
			synchronize with data source
		end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
myDS.synchronizeWithDataSource();
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDS = myEasyCatalog.DataSources.Item("Stock.csv")
myDS.SynchronizeWithDataSource
```

DATA SOURCE OBJECT
(CONTINUED)

UPDATE DATA SOURCE

Commit changes in the snapshot to the data source. Remember to update the snapshot before calling if this hasn't already been done.

AppleScript

```
tell application "Adobe InDesign CS5"
	tell EasyCatalog object
		tell DataSource "Stock.csv"
			update data source
		end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
myDS.updateDataSource;
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDS = myEasyCatalog.DataSources.Item("Stock.csv")
myDS.UpdateDataSource
```

DATA SOURCE OBJECT (CONTINUED)

UPDATE DOCUMENT

Updates the given documents tags with updated content from the data source.

Parameters:

document: document object

AppleScript

```
tell application "Adobe InDesign CS5"
	set myDocument to active document
	tell EasyCatalog object
		tell DataSource "Stock.csv"
			update document doc myDocument
		end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDoc = app.activeDocument;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
myDS.updateDocument(myDoc);
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myDoc = myInDesign.ActiveDocument
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDS = myEasyCatalog.DataSources.Item("Stock.csv")
myDS.UpdateDocument(myDoc)
```

DATA SOURCE OBJECT
(CONTINUED)

UPDATE PAGE NUMBERS

Update a field in a data source with the page numbers of records in a doument

Parameters:

doc:	Document to inspect
instances:	Instance to update - first,last, all
field:	The name of the field to update
search field:	The field name to search for - optional

AppleScript

```
tell application "Adobe InDesign CS5"
	set myDocument to active document
	tell EasyCatalog object
		tell DataSource "Stock.csv"
			update page numbers doc myDocument instances "all"
			field "PageNum"
		end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDoc = app.activeDocument;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
myDS.updatePageNumbers(myDoc, "all", "PageNum")
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myDoc = myInDesign.ActiveDocument
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDS = myEasyCatalog.DataSources.Item("Stock.csv")
myDS.UpdatePageNumbers(myDoc, "all", "PageNum")
```

DATA SOURCE OBJECT (CONTINUED)

UPDATE SNAPSHOT

Update the snapshot with content from the given document.

Parameters:

document: document object

AppleScript

```
tell application "Adobe InDesign CS5"
	set myDocument to active document
	tell EasyCatalog object
		tell DataSource "Stock.csv"
			update snapshot doc myDocument
		end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDoc = app.activeDocument;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
myDS.updateSnapshot(myDoc);
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myDoc = myInDesign.ActiveDocument
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDS = myEasyCatalog.DataSources.Item("Stock.csv")
myDS.UpdateSnapshot(myDoc)
```

DATA SOURCE VIEW APPLY FILTER

Apply a previously saved filter to the data view.

Parameters:

filter name: filter to apply

AppleScript

```
tell application "Adobe InDesign CS5"
	tell EasyCatalog object
	tell DataSource "Stock.csv"
				tell DataView "Stock.csv"
					apply filter filter name "Sports"
				end tell
	end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
var myDV = myDS.dataviews.item("Stock.csv");
myDV.applyFilter("Sports");
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDS = myEasyCatalog.DataSources.Item("Stock.csv")
Set myDV = myDS.DataViews.Item("Stock.csv")
myDV.ApplyFilter("Sports")
```

DATA SOURCE VIEW (CONTINUED)

CLOSE DATA VIEW

Close the data view.

AppleScript

```
tell application "Adobe InDesign CS5"
	tell EasyCatalog object
		tell DataSource "Stock.csv"
			tell DataView "Stock.csv"
				close data view
			end tell
		end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
var myDV = myDS.dataviews.item("Stock.csv");
myDV.closeDataView();
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDS = myEasyCatalog.DataSources.Item("Stock.csv")
Set myDV = myDS.DataViews.Item("Stock.csv")
myDV.CloseDataView
```

DATA SOURCE VIEW (CONTINUED)

DATA VIEW NAME

The name of the data view (Read only).

AppleScript

```
tell application "Adobe InDesign CS5"
	tell EasyCatalog object
		tell DataSource "Stock.csv"
			tell DataView "Stock.csv"
				set myName to data view name
			end tell
		end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
var myDV = myDS.dataviews.item("Stock.csv");
alert(myDV.name);
```

Visual Basic

```
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDS = myEasyCatalog.DataSources.Item("Stock.csv")
Set myDV = myDS.DataViews.Item("Stock.csv")
MsgBox(myDV.DataViewName)
```

DATA SOURCE VIEW (CONTINUED)

EMPTY

Empty the data view.

AppleScript

```
tell application "Adobe InDesign CS5"
	tell EasyCatalog object
		tell DataSource "Stock.csv"
			tell DataView "Stock.csv"
				empty
			end tell
		end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
var myDV = myDS.dataviews.item("Stock.csv");
myDV.emptyDataView();
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDS = myEasyCatalog.DataSources.Item("Stock.csv")
Set myDV = myDS.DataViews.Item("Stock.csv")
myDV.EmptyDataView
```

DATA SOURCE VIEW (CONTINUED)

GROUP DATA VIEW

Group the data in the data view.

Parameters:

field name: field name to group by
ascending: group ascending - true/false
override sort field: override default sorting prior to grouping with this field - optional

AppleScript

```
tell application "Adobe InDesign CS5"
	tell EasyCatalog object
	tell DataSource "Stock.csv"
		tell DataView "Stock.csv"
			group data view field name "SKU" with
			ascending
		end tell
	end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
var myDV = myDS.dataviews.item("Stock.csv");
myDV.groupDataView("SKU", true);
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDS = myEasyCatalog.DataSources.Item("Stock.csv")
Set myDV = myDS.DataViews.Item("Stock.csv")
err = myDV.GroupDataView("SKU", TRUE)
```

DATA SOURCE VIEW (CONTINUED)

HIDE FIELD

Hide a field in the data view.

Parameters:

field name: field name to hide

AppleScript

```
tell application "Adobe InDesign CS5"
	tell EasyCatalog object
	tell DataSource "Stock.csv"
				tell DataView "Stock.csv"
					hide field field name "SKU"
				end tell
			end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
var myDV = myDS.dataviews.item("Stock.csv");
myDV.hideField("SKU");
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDS = myEasyCatalog.DataSources.Item("Stock.csv")
Set myDV = myDS.DataViews.Item("Stock.csv")
myDV.HideField("SKU")
```

DATA SOURCE VIEW (CONTINUED)

INSERT VIEW SELECTION

Insert the data view selection into a document at the given insertion point

Parameters:

start offset: start insertion point

AppleScript

```
tell application "Adobe InDesign CS5"
	set myDocument to active document
	set myFrame to text frame 1 of myDocument
	set myStory to parent story of myFrame
	set myInsertionPoint to insertion point 1 of myStory
	tell EasyCatalog object
		tell (selected data view)
			insert view selection story offset myInsertionPoint
		end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDoc = app.activeDocument;
var myDV = myEasyCatalog.selectedDataView();
var myInsertionPoint = app.documents[0].pages[0].textFrames[0].
    insertionPoints[0]
myDV.insertViewSelection(myInsertionPointStart);
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDoc = myInDesign.ActiveDocument
Set myDV = myEasyCatalog.SelectedDataView()
Set myFrame = myDoc.Pages.Item(1).TextFrames.Item(1)
Set myInsertionPoint = myFrame.InsertionPoints.Item(1)
err = myDV.InsertViewSelection(myInsertionPoint)
```

DATA SOURCE VIEW (CONTINUED)

MAKE SELECTION

Select the given row range in the view and optionally a specific field. If no field is supplied, all are selected by default.

Parameters:

from:	index of first row to select
to:	index of last row to select
field:	name of the field to select - optional

AppleScript

```
tell application "Adobe InDesign CS5"
	set myDocument to active document
	tell EasyCatalog object
		tell DataSource "Stock.csv"
			tell DataView "Stock.csv"
			make selection from 1 to 3 field "SKU"
			update selected document myDocument
			end tell
		end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
var myDV = myDS.dataviews.item("Stock.csv");
myDV.makeSelection(1,3, "Manufacturer");
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDS = myEasyCatalog.DataSources.Item("Stock.csv")
Set myDV = myDS.DataViews.Item("Stock.csv")
err = myDV.MakeSelection(1, 3, "Manufacturer")
```

DATA SOURCE VIEW (CONTINUED)

PAGINATE

Paginate the given page item with the current view selection.

Parameters:

page item: page item object

AppleScript

```
tell application "Adobe InDesign CS5"
	set myDocument to active document
	set myFrame to text frame 1 of myDocument
	tell EasyCatalog object
		tell DataSource "Stock.csv"
			tell DataView "Stock.csv"
			make selection from 1 to 3
				paginate page item myFrame
			end tell
	 end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDoc = app.activeDocument;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
var myDV = myDS.dataviews.item("Stock.csv");
myDV.makeSelection(1,3);
var myFrame = app.documents[0].pages[0].textFrames[0]
myDV.paginate(myFrame);
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDoc = myInDesign.ActiveDocument
Set myDS = myEasyCatalog.DataSources.Item("Stock.csv")
Set myDV = myDS.DataViews.Item("Stock.csv")
err = myDV.MakeSelection(1,3)
Set myFrame = myDoc.Pages.Item(1).TextFrames.Item(1)
err = myDV.Paginate(myFrame)
```

DATA SOURCE VIEW (CONTINUED)

PAGINATE INTO TEXT FLOW

Paginate the selected view items into the given insertion point using the specified library.

Parameters:

insertion point:	insertion point object
library path:	pagination elements library

AppleScript

```
tell application "Adobe InDesign CS5"
	set myDocument to active document
	set myFrame to text frame 1 of myDocument
	set myStory to parent story of myFrame
	set myInsertionPoint to insertion point 1 of myStory
	tell EasyCatalog object
		tell DataSource "Stock.csv"
			tell DataView "Stock.csv"
			make selection from 1 to 3
				paginate into text flow story offset
				myInsertionPoint library path "Macintosh
				HD:Library.indl"
		end tell
		end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDoc = app.activeDocument;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
var myDV = myDS.dataviews.item("Stock.csv");
myDV.makeSelection(1,3);
var myInsertionPoint = app.documents[0].pages[0].textFrames[0].
    insertionPoints[0]
myDV.paginateIntoTextFlow(myInsertionPoint, "Macintosh HD:Library.indl");
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDoc = myInDesign.ActiveDocument
Set myDS = myEasyCatalog.DataSources.Item("Stock.csv")
Set myDV = myDS.DataViews.Item("Stock.csv")
err = myDV.MakeSelection(1,3)
Set myFrame = myDoc.Pages.Item(1).TextFrames.Item(1)
Set myInsertionPoint = myFrame.InsertionPoints.Item(1)
err = myDV.PaginateIntoTextFlow(myInsertionPoint, "c:\library.indl")
```

DATA SOURCE VIEW (CONTINUED)

PAGINATE TEXT FLOW RANGE

Paginate the current selection into the specified text range.

Parameters:

start offset: start insertion point
end offset: end insertion point

AppleScript

```
tell application "Adobe InDesign CS5"
	set myDocument to active document
	set myFrame to text frame 1 of myDocument
	set myStory to parent story of myFrame
	set myInsertionPointStart to insertion point 1 of myStory
	set myInsertionPointEnd to insertion point 25 of myStory
	tell EasyCatalog object
		tell DataSource "Stock.csv"
			tell DataView "Stock.csv"
			make selection from 1 to 3
				paginate text flow range start offset
				myInsertionPointStart end offset
				myInsertionPointEnd
			end tell
		end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDoc = app.activeDocument;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
var myDV = myDS.dataviews.item("Stock.csv");
myDV.makeSelection(1,3);
var myInsertionPointStart = app.documents[0].pages[0].textFrames[0].
    insertionPoints[0]
var myInsertionPointEnd = app.documents[0].pages[0].textFrames[0].
    insertionPoints[24]
myDV.paginateIntoTextFlow(myInsertionPointStart, myInsertionPointEnd);
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDoc = myInDesign.ActiveDocument
Set myDS = myEasyCatalog.DataSources.Item("Stock.csv")
Set myDV = myDS.DataViews.Item("Stock.csv")
err = myDV.MakeSelection(1,3)
Set myInsertionPointStart = myDoc.Pages.Item(1).TextFrames.Item(1).
    InsertionPoints.Item(1)
Set myInsertionPointEnd = myDoc.Pages.Item(1).TextFrames.Item(1).
    InsertionPoints.Item(25)
err = myDV.PaginateTextFlowRange(myInsertionPointStart, myInsertionPointEnd)
```

DATA SOURCE VIEW (CONTINUED)

PAGINATE USING DEFAULTS

Paginate the selection using the default pagination settings for the data source. The active document is paginate

Parameters:

library path: path to the pagination elements library

AppleScript

```
tell application "Adobe InDesign CS5"
	set myDocument to active document
	tell EasyCatalog object
		tell (selected data view)
			paginate using defaults library path "Macintosh
			HD:Library.indl"
		end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDV = myEasyCatalog.selectedDataView();
var myDoc = app.activeDocument;
myDV.paginateUsingDefaults("Macintosh HD:Library.indl");
```

Visual Basic

```
Set myInDesign = CreateObject("Indesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDV = myEasyCatalog.SelectedDataView()
Set myDoc = myInDesign.ActiveDocument
myDV.PaginateUsingDefaults("c:\library.indl");
```

DATA SOURCE VIEW (CONTINUED)

PAGINATE USING GUIDES

Paginate the view selection using guide based pagination rules

Parameters:

doc: document to paginate into
library path: path to the pagination elements library
page index: page index to start paginating at (1 = first page in the document)

AppleScript

```
tell application "Adobe InDesign CS5"
	set myDocument to active document
	tell EasyCatalog object
		tell (selected data view)
			paginate using guides doc myDocument library path
			"Macintosh HD:Library.indl" page index 1
		end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDV = myEasyCatalog.selectedDataView();
var myDoc = app.activeDocument;
myDV.paginateUsingGuides(myDoc, "Macintosh HD:Library.indl", 1);
```

Visual Basic

```
Set myInDesign = CreateObject("Indesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDV = myEasyCatalog.SelectedDataView()
Set myDoc = myInDesign.ActiveDocument
myDV.PaginateUsingGuides(myDoc, "c:\library.indl", 1);
```

DATA SOURCE VIEW PAGINATE USING MASTERS

Paginate the view selection using master based pagination

Parameters:

doc:	document to paginate into
page index:	Page index to start paginating at (1 = first page in the document)
type:	type of pagination (page, spread)
break field:	optional - field name to break on
master field:	optional - field containing the master to apply

AppleScript

```
tell application "Adobe InDesign CS5"
	set myDocument to active document
	tell EasyCatalog object
		tell (selected data view)
			paginate using masters doc myDocument page index 1
			type page
		end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDV = myEasyCatalog.selectedDataView();
var myDoc = app.activeDocument;
myDV.paginateUsingMasters(myDoc, 1, "page");
```

Visual Basic

```
Set myInDesign = CreateObject("Indesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDV = myEasyCatalog.SelectedDataView()
Set myDoc = myInDesign.ActiveDocument
myDV.paginateUsingMasters(myDoc, 1, "page");
```

DATA SOURCE VIEW (CONTINUED)

REMOVE ROW

Remove a given row from the View. This only removes the row from view, not from the datasource.

Parameters:

no: index number (1..n)

AppleScript

```
tell application "Adobe InDesign CS5"
	tell EasyCatalog object
		tell DataSource "Stock.csv"
			tell DataView "Stock.csv"
				remove row no 1
			end tell
		end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
var myDV = myDS.dataviews.item("Stock.csv");
myDV.removeRow(1);
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDS = myEasyCatalog.DataSources.Item("Stock.csv")
Set myDV = myDS.DataViews.Item("Stock.csv")
err = myDV.RemoveRow(1)
```

DATA SOURCE VIEW (CONTINUED)

ROW COUNT

Returns the number of rows in the view.

AppleScript

```
set myDocument to active Document
tell application "Adobe InDesign CS5"
	tell EasyCatalog object
	tell DataSource "Stock.csv"
				tell DataView "Stock.csv"
					set myName to row count
				end tell
	end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
var myDV = myDS.dataviews.item("Stock.csv");
alert(myDV.rowCount);
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDS = myEasyCatalog.DataSources.Item("Stock.csv")
Set myDV = myDS.DataViews.Item("Stock.csv")
MsgBox(myDV.RowCount)
```

DATA SOURCE VIEW (CONTINUED)

SELECT GROUP

Select a given relational group. Applies to relational data sources only.

Parameters:

name: group name

AppleScript

```
tell application "Adobe InDesign CS5"
	set myDocument to active document
	set myFrame to text frame 1 of myDocument
	set myStory to parent story of myFrame
	set myInsertionPoint to insertion point 1 of myStory
	tell EasyCatalog object
		tell (selected data view)
			select group name "Attributes"
		end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDoc = app.activeDocument;
var myDV = myEasyCatalog.selectedDataView();
myDV.selectGroup("Attributes");
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDoc = myInDesign.ActiveDocument
Set myDV = myEasyCatalog.SelectedDataView()
myDV.SelectGroup("Attributes")
```

DATA SOURCE VIEW (CONTINUED)

SHOW ALL

Show all records in this view. Removes any filtering that is applied to the view.

AppleScript

```
tell application "Adobe InDesign CS5"
	tell EasyCatalog object
	tell DataSource "Stock.csv"
				tell DataView "Stock.csv"
					show all
				end tell
	end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
var myDV = myDS.dataviews.item("Stock.csv");
myDV.showAll();
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDS = myEasyCatalog.DataSources.Item("Stock.csv")
Set myDV = myDS.DataViews.Item("Stock.csv")
myDV.ShowAll
```

DATA SOURCE VIEW (CONTINUED)

SHOW ERRORS

Filter the view to only show records flagged with errors.

AppleScript

```
tell application "Adobe InDesign CS5"
	tell EasyCatalog object
	tell DataSource "Stock.csv"
				tell DataView "Stock.csv"
					show errors
				end tell
	end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
var myDV = myDS.dataviews.item("Stock.csv");
myDV.showErrors();
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDS = myEasyCatalog.DataSources.Item("Stock.csv")
Set myDV = myDS.DataViews.Item("Stock.csv")
myDV.ShowErrors
```

DATA SOURCE VIEW (CONTINUED)

SHOW FIELD

Show a field in the data view. (If it's not already visible).

Parameters:

field name: field name to show

AppleScript

```
tell application "Adobe InDesign CS5"
	tell EasyCatalog object
	tell DataSource "Stock.csv"
		tell DataView "Stock.csv"
			show field field name "SKU"
		end tell
	end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
var myDV = myDS.dataviews.item("Stock.csv");
myDV.showField("SKU");
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDS = myEasyCatalog.DataSources.Item("Stock.csv")
Set myDV = myDS.DataViews.Item("Stock.csv")
myDV.ShowField("SKU")
```

DATA SOURCE VIEW (CONTINUED)

SHOW SUBSET

Show a saved subset of data in the view.

Parameters:

subset name: name of the subset

AppleScript

```
tell application "Adobe InDesign CS5"
	tell EasyCatalog object
		tell DataSource "Stock.csv"
			tell DataView "Stock.csv"
				show subset subset name "sports"
			end tell
		end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
var myDV = myDS.dataviews.item("Stock.csv");
myDV.showSubset("sports");
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDS = myEasyCatalog.DataSources.Item("Stock.csv")
Set myDV = myDS.DataViews.Item("Stock.csv")
err = myDV.ShowSubset("sports")
```

DATA SOURCE VIEW (CONTINUED)

SORT DATA VIEW

Sort the view by the given field name. Clears any existing sort configuration.

Parameters:

field name:	field name to sort by
ascending:	sort ascending - true/false

AppleScript

```
tell application "Adobe InDesign CS5"
	tell EasyCatalog object
		tell DataSource "Stock.csv"
			tell DataView "Stock.csv"
				sort data view field name "SKU" with ascending
			end tell
	end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
var myDV = myDS.dataviews.item("Stock.csv");
myDV.sortDataView("SKU", true);
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDS = myEasyCatalog.DataSources.Item("Stock.csv")
Set myDV = myDS.DataViews.Item("Stock.csv")
err = myDV.SortDataView("SKU", TRUE)
```

DATA SOURCE VIEW (CONTINUED)

SUBGROUP DATA VIEW

Apply an additional level of grouping to the data view.

Parameters:

field name: field name to group by

ascending: group ascending - true/false override sort

field: override default sorting prior to grouping with this field - optional

AppleScript

```
tell application "Adobe InDesign CS5"
	tell EasyCatalog object
	tell DataSource "Stock.csv"
		tell DataView "Stock.csv"
			subgroup data view field name "SKU" with
			ascending
		end tell
	end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
var myDV = myDS.dataviews.item("Stock.csv");
myDV.subgroupDataView("Price", true);
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDS = myEasyCatalog.DataSources.Item("Stock.csv")
Set myDV = myDS.DataViews.Item("Stock.csv")
err = myDV.SubgroupDataView("Price", TRUE)
```

DATA SOURCE VIEW (CONTINUED)

SUBSET OF

Filter the view to contain only records matching the specified criteria.

Parameters:

field name:	field name
operand:	operand
value:	value to match again
search entire:	true to search the entire data set, false to only search those items shown

AppleScript

```
tell application "Adobe InDesign CS5"
	tell EasyCatalog object
		tell DataSource "Stock.csv"
			tell DataView "Stock.csv"
				subset of field name "Section" operand "=" value
				"Sports" with search entire
			end tell
		end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
var myDV = myDS.dataviews.item("Stock.csv");
myDV.subsetOf("Manufacturer", "=", "Sony", true);
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDS = myEasyCatalog.DataSources.Item("Stock.csv")
Set myDV = myDS.DataViews.Item("Stock.csv")
err = myDV.SubsetOf("Manufacturer", "=", "Sony", TRUE)
```

DATA SOURCE VIEW (CONTINUED)

SUBSORT DATA VIEW

Apply an additional level of sorting to the data view.

Parameters:

field name:	field name to sort by
ascending:	sort ascending - true/false

AppleScript

```
tell application "Adobe InDesign CS5"
	tell EasyCatalog object
	tell DataSource "Stock.csv"
		tell DataView "Stock.csv"
			subsort data view field name "SKU" with
			ascending
		end tell
	end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
var myDV = myDS.dataviews.item("Stock.csv");
myDV.subsortDataView("Price", true);
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDS = myEasyCatalog.DataSources.Item("Stock.csv")
Set myDV = myDS.DataViews.Item("Stock.csv")
err = myDV.SubSortDataView("Price", TRUE)
```

DATA SOURCE VIEW (CONTINUED)

UPDATE DOCUMENT

Update the specified document with data from the view.

Parameters:

AppleScript

```
tell application "Adobe InDesign CS5"
	set myDocument to active document
	set myFrame to text frame 1 of myDocument
	set myStory to parent story of myFrame
	set myInsertionPoint to insertion point 1 of myStory
	tell EasyCatalog object
		tell (selected data view)
			update document doc myDocument
		end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDoc = app.activeDocument;
var myDV = myEasyCatalog.selectedDataView();
myDV.updateDocument(myDoc);
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDoc = myInDesign.ActiveDocument
Set myDV = myEasyCatalog.SelectedDataView()
myDV.UpdateDocument(myDoc)
```

DATA SOURCE VIEW (CONTINUED)

UPDATE SELECTED

Update the given document with the views selected records.

Parameters:

document: document to update

AppleScript

```
tell application "Adobe InDesign CS5"
	set myDocument to activeDocument
	tell EasyCatalog object
		tell DataSource "Stock.csv"
			tell DataView "Stock.csv"
				make selection from 1 to 3
				update selected doc myDocument
			end tell
		end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDoc = app.activeDocument;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
var myDV = myDS.dataviews.item("Stock.csv");
myDV.makeSelection(1,3);
myDV.updateSelected(myDoc);
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myDoc = myInDesign.ActiveDocument
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDS = myEasyCatalog.DataSources.Item("Stock.csv")
Set myDV = myDS.DataViews.Item("Stock.csv")
err = myDV.MakeSelection(1,3)
myDV.UpdateSelected(myDoc)
```

DATA SOURCE VIEW (CONTINUED)

UPGROUP DATA VIEW

Remove any grouping applied to records in the view.

AppleScript

```
tell application "Adobe InDesign CS5"
	tell EasyCatalog object
	tell DataSource "Stock.csv"
				tell DataView "Stock.csv"
					ungroup data view
				end tell
	end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDS = app.easycatalogObject.datasources.item("Stock.csv");
var myDV = myDS.dataviews.item("Stock.csv");
myDV.ungroupDataView();
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDS = myEasyCatalog.DataSources.Item("Stock.csv")
Set myDV = myDS.DataViews.Item("Stock.csv")
myDV.UngroupDataView
```

RECORD MARKED AS PLACED

Returns true if any of the fields in the record are marked as placed.

AppleScript

```
tell application "Adobe InDesign CS3"
	tell EasyCatalog object
		tell DataSource "Stock.csv"
			tell record 1
				set myMarkedAsPlaced to markedasplaced
			end tell
		end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDS = app.easycatalogObject.datasources.item("STOCK.CSV");
var myDV = myDS.dataviews.item("STOCK.CSV");
var myRecord = myDV.records.item(1);
alert(myRecord.markedasplaced);
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDS = myEasyCatalog.DataSources.Item("Stock.csv")
Set myDV = myDS.DataViews.Item("Stock.csv")
Set myRecord = myDS.Records.Item(1)
MsgBox(myRecord.MarkedAsPlaced)
```

RECORD (CONTINUED)

MARKED IN ERROR

Returns true if any of the fields in the record are marked in error.

AppleScript

```
tell application "Adobe InDesign CS3"
	tell EasyCatalog object
		tell DataSource "Stock.csv"
			tell record 1
				set myMarkedInError to markedinerror
			end tell
		end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDS = app.easycatalogObject.datasources.item("STOCK.CSV");
var myDV = myDS.dataviews.item("STOCK.CSV");
var myRecord = myDV.records.item(1);
alert(myRecord.markedinerror);
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDS = myEasyCatalog.DataSources.Item("Stock.csv")
Set myDV = myDS.DataViews.Item("Stock.csv")
Set myRecord = myDS.Records.Item(1)
MsgBox(myRecord.MarkedInError)
```

RECORD
(CONTINUED)

PAGINATE RECORD

Use the contents of this record to paginate the given page item. Any field markers in the heirarchy of the page item will be populated with the contents of the records fields.

Parameters:

page item: index number (1..n)

AppleScript

```
tell application "Adobe InDesign CS3"
	tell EasyCatalog object
		tell DataSource "Stock.csv"
			tell record 1
				paginate record page item myTextFrame
			end tell
		end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDS = app.easycatalogObject.datasources.item("STOCK.CSV");
var myDV = myDS.dataviews.item("STOCK.CSV");
var myRecord = myDV.records.item(1);
var myFrame = app.documents[0].pages[0].textFrames[0]
myRecord.paginateRecord(myFrame);
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDS = myEasyCatalog.DataSources.Item("Stock.csv")
Set myDV = myDS.DataViews.Item("Stock.csv")
Set myFrame = myDoc.Pages.Item(1).TextFrames.Item(1)
Set myRecord = myDS.Records.Item(1)
myRecord.PaginateRecord(myFrame)
```

FIELD FIELD CONTENT

The contents of the field (Read only).

AppleScript

```
tell application "Adobe InDesign CS5"
	tell EasyCatalog object
		tell DataSource "Stock.csv"
			tell field "SKU" of record "SKU1234"
				set myContent to field content
			end tell
		end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDS = app.easycatalogObject.datasources.item("STOCK.CSV");
var myDV = myDS.dataviews.item("STOCK.CSV");
alert(myDV.records.item("SKU1234").fields.item("SKU").fieldContent);
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDS = myEasyCatalog.DataSources.Item("Stock.csv")
Set myDV = myDS.DataViews.Item("Stock.csv")
Set myFrame = myDoc.Pages.Item(1).TextFrames.Item(1)
Set myRecord = myDS.Records.Item("SKU1234")
Set myField = myRecord.Fields.Item("SKU1234")
MsgBox(myField.FieldContent)
```

FIELD (CONTINUED) INSERT TAGGED CONTENT

Insert the contents of the field as an EasyCatalog tag

Parameters:

story offset: Insertion point of a story

AppleScript

```
tell application "Adobe InDesign CS5"
	tell EasyCatalog object
		tell DataSource "Stock.csv"
			tell field "SKU" of record "SKU1234"
				insert tagged content story offset
				myInsertionPoint
			end tell
		end tell
	end tell
end tell
```

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDS = app.easycatalogObject.datasources.item("STOCK.CSV");
var myDV = myDS.dataviews.item("STOCK.CSV");
var myInsertionPoint = app.documents[0].pages[0].textFrames[0].
    insertionPoints[0]
myDV.records.item("SKU1234").fields.item("SKU").insertTaggedContent(myInserti
    onPoint);
```

Visual Basic

```
Set myInDesign = CreateObject("InDesign.Application")
Set myEasyCatalog = myInDesign.EasyCatalogObject
Set myDS = myEasyCatalog.DataSources.Item("Stock.csv")
Set myDV = myDS.DataViews.Item("Stock.csv")
Set myFrame = myDoc.Pages.Item(1).TextFrames.Item(1)
Set myRecord = myDS.Records.Item("SKU1234")
Set myField = myRecord.Fields.Item("SKU1234")
Set myFrame = myDoc.Pages.Item(1).TextFrames.Item(1)
Set myInsertionPoint = myFrame.InsertionPoints.Item(1)
myField.InsertTaggedContent(myInsertionPoint)
```

JAVASCRIPT SAMPLES

Open a data source, create a view for it and then output to the JavaScript console the contents of a named field.

```
var myEasyCatalog = app.easycatalogObject;
myEasyCatalog.workspaceFolder=”Macintosh HD:Workspace”;
var myDS = app.easycatalogObject.datasources.item(“Stock.csv”);
var myDV = myDS.dataviews.add();
$.writeln(myDV.records.count());
for (i = 0; i < myDV.records.count();++i)
{
    $.writeln (myDV.records.item(i).fields.item(“PaginaItemID”).fieldContent);
}
myDV.closeDataView()
```

Open a document, open a data source, update furniture, close data source, save document to a new name.

```
var myDoc = app.open(“c:\\Workspace Folder\\Template.indd”);
var myEasyCatalog = app.easycatalogObject;
myEasyCatalog.workspaceFolder=”c:\\Workspace Folder”;
var myDS = app.easycatalogObject.datasources.item(“Stock.csv”);
var myDV = myDS.dataviews.add();
$.writeln(myDV.records.count());
myEasyCatalog.updateFurniture(myDoc);
myDV.closeDataView()
myDoc = myDoc.save(“c:\\Workspace Folder\\output.indd”);
myDoc.close();
```

Open a data source, close all the panels for it, synchronize with new data, open a new panel, open a document, paginate at guide positions.

```
var myEC = app.easycatalogObject;
myDS = myEC.datasources.item(“STOCK.CSV”);

// close all views for this data source
var dvCount = myDS.dataviews.count();
for (index = dvCount-1; index >= 0; index--)
{
        myDS.dataviews.item(index).closeDataView();
}

// ensure that the datasource is pointing to the correct data file
myDS.dataSourceSpecifier = “Macintosh HD:users:somebody:Desktop:Stock.csv”;

//ensure that the most recent data has been synchronized
myDS.synchronizeWithDataSource();

//now use the dataview object to to make selection and do the pagination
var myDV = myDS.dataviews.add();
myDV.makeSelection(1,1);
myDoc = app.open(File(“Macintosh HD:Users:ianwhite:Desktop:Script.indd”), true);
myDV.paginateUsingGuides(myDoc, “Macintosh HD:Users:somebody:Desktop:script.indl”, 9999)
myDV.closeDataView()
```

Iterate the records of one data source, filtering another data source by on the contents of one of the original data sources fields.

```
var myEasyCatalog = app.easycatalogObject;
var bodyDS = app.easycatalogObject.datasources.item(“Stock.csv”);
var calloutDS = app.easycatalogObject.datasources.item(“DataSource_Callouts.txt”);
var bodyDV = bodyDS.dataviews.item(“DataSource_Body.txt”);
var calloutDV = calloutDS.dataviews.item(“DataSource_Callouts.txt”);
for (i = 0; i < bodyDV.records.count(); ++i)
{
        var record = bodyDV.records.item(i);
        $.writeln(record.fields.item(“ID”).fieldContent);
        calloutDV.subsetOf(“ProductNumber”, “=”, record.fields.item(“ID”).fieldContent, true);
}
```