

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.
2.1.4	15 May 2012	Updated with latest methods.
2.1.5	20 July 2012	Updated
2.1.6	11 April 2013	Updated

© Copyright 2005 - 2013 65bit Software Limited. All Rights reserved. Reproduction or copying prohibited.

Adobe and InDesign are either trademarks or registered trademarks of Adobe Systems Incorporated in the United States and/or other countries.

All other trademarks and copyrights are the property of their respective owners.

CONTENTS

GETTING STARTED

Welcome.....	6
What The Scripting Module?.....	6
Installation	6
Assumptions.....	6
Need more?.....	6
Purchasing.....	6

OBJECT MODEL

Overview.....	7
---------------	---

SCRIPTING REFERENCE

EasyCatalog Object.....	8
content tracking	8
exchange tags.....	9
field marker visibility.....	10
get links	11
get selected links	12
justify rows	13
load file.....	14
purge data source	15
register	16
remove field markers.....	17
remove selected field markers	18
report tags.....	19
selected data view	20
serial number.....	21
tag page item	22
tag text range.....	23
update furniture.....	24
update furniture on page.....	25
update tags	26
workspace folder.....	27
Data Source Object.....	28
add custom field	28
add field	29
adopt.....	30
change key	31
count deleted	32
count errors.....	33
count inserted	34

CHAPTER 3

count updated.....	35
data source name	36
data source odbc statement.....	37
data source specifier.....	38
document Report	39
get field names.....	40
load field definitions	41
load formatting tags	42
odbc connection	43
pickup	44
purge deleted	45
remove field.....	46
replace fields	47
save.....	48
set field option.....	49
set pagination option	50
set parameters	51
synchronize document.....	52
synchronize with data source.....	53
update custom fields.....	54
update data source	55
update document.....	56
update page numbers.....	57
update snapshot	58
Data Source View	59
apply configuration	59
apply filter.....	60
close data view	61
data view name.....	62
empty	63
group data view.....	64
group drill down	65
group up.....	66
hide field.....	67
insert view selection.....	68
make selection	69
paginate	70
paginate into text flow.....	71
paginate text flow range.....	72
paginate using defaults.....	73
paginate using guides.....	74
paginate using masters.....	75
remove row	76
row count	77
select group	78
selected records.....	79

selected relational group.....	80
show all	81
show errors	82
show field	83
show records.....	84
show subset	85
sort data view.....	86
subgroup data view.....	87
subset of	88
subsort data view.....	89
update document.....	90
update selected	91
update selected snapshot.....	92
upgroup data view	93
visible	94
Record.....	95
marked as placed.....	95
marked as updated locally	96
marked in error.....	97
paginate record	98
Field.....	99
field content	99
insert tagged content	100
markedasupdatedlocally.....	101

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:

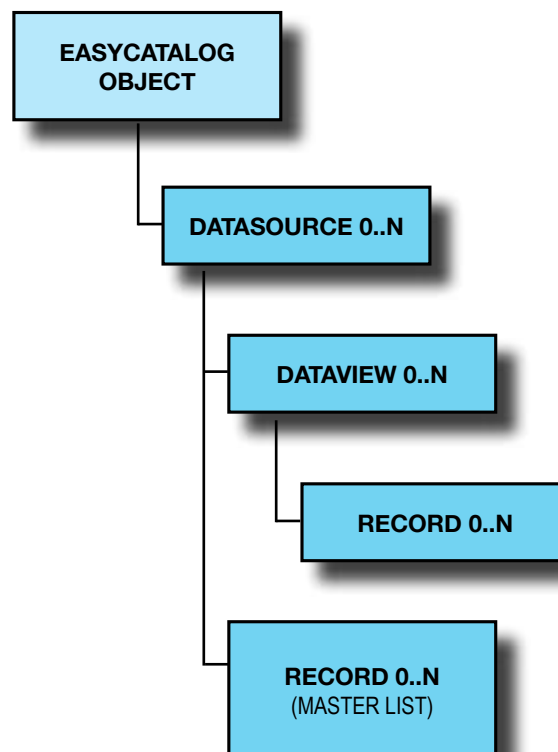
<http://www.65bit.com/purchase/purchase.shtml>

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 CONTENT TRACKING
OBJECT**

Get or set EasyCatalog's content tracking property.

AppleScript

```
tell application "Adobe InDesign CS6"  
  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)
```


EASYPATH OBJECT**OBJECT
(CONTINUED)**

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

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" 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</pre>
--------------------	--

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDoc = app.activeDocument; myEasyCatalog.exchangeTags(myDoc, "Macintosh HD:tagfile.csv", "Macintosh HD:log.txt");</pre>
-------------------	--

<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("InDesign.Application") Set myDocument = myInDesign.ActiveDocument Set myEasyCatalog = myInDesign.EasyCatalogObject err = myEasyCatalog.ExchangeTags(myDocument, "c:\tagfile.csv", "c:\log.txt")</pre>
---------------------	---

EASYCATALOG FIELD MARKER VISIBILITY**OBJECT
(CONTINUED)**

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

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" tell active document tell EasyCatalog object set field marker visibility to false end tell end tell end tell</pre>
--------------------	---

<i>JavaScript</i>	<pre>var myDoc = app.activeDocument; myDoc.easycatalogObject.fieldMarkerVisibility = false;</pre>
-------------------	---

<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("Indesign.Application") Set myDocument = myInDesign.ActiveDocument MsgBox (myDocument.EasyCatalogObject.FieldMarkerVisibility)</pre>
---------------------	---

EASYPAGELIST GET LINKS**OBJECT
(CONTINUED)**

Returns details of the links in the given page item selection. The array consists of sets of data source, key field value, field name.

Parameters:

```
JavaScript
var myEasyCatalog = app.easycatalogObject;
var myPageItem = myPageItem = app.selection[0];
var mySelection = myEasyCatalog.getLinks(myPageItem);
for (index = 0; index < mySelection.length; index += 3) {
    alert(mySelection[index + 0] + "-" + mySelection[index + 1] + "-" +
        mySelection[index + 2]);
}
```

EASYCATALOG GET SELECTED LINKS OBJECT (CONTINUED)

Returns details of the links in the current selection. The array consists of sets of: "data source, key field value, field name"

Parameters:

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" tell EasyCatalog object Set myArray to get selected links end tell end tell</pre>
<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var mySelection = myEasyCatalog.getSelectedLinks(); for (index = 0; index < mySelection.length; index += 3) { alert(mySelection[index + 0] + "-" + mySelection[index + 1] + "-" + mySelection[index + 2]); }</pre>
<i>Visual Basic</i>	<pre>Set myEasyCatalog = myInDesign.EasyCatalogObject Set mySelection = myEasyCatalog.GetSelectedLinks();</pre>

EASYCATALOG JUSTIFY ROWS**OBJECT
(CONTINUED)**

Justify the tables in the given page item to the bottom of the text frame. The rows that can be adjusted must have a script label attached.

Parameters:

page item: page item object justify to end: also justify
the frames after the given frame justify final frame: justify the
final table page in the chain

JavaScript

```
var myEasyCatalog = app.easycatalogObject;  
var myPageItem = app.selection[0];  
myEasyCatalog.justifyRows(myPageItem,true,true);
```

EASYCATALOG LOAD FILE OBJECT (CONTINUED)

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. Compound key fields can be separated by '|' encoding type: encoding type of file, optional UTF8, UTF16. By default an attempt to determine the correct type will be made

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" 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" encoding "UTF8" end tell end tell</pre>
--------------------	---

<i>JavaScript</i>	<pre>var myEC = app.easycatalogObject; myEC.loadFile ("data source name", "file path here", ";", "\r", "key field name here","UTF8")</pre>
-------------------	--

<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("Indesign.Application") Set myEasyCatalog = myInDesign.EasyCatalogObject Set myDS = myEasyCatalog.LoadFile("data source name", "file path here", ";", "\r", "key field name here","UTF8")</pre>
---------------------	--

EASYPURGE DATA SOURCE**OBJECT
(CONTINUED)**

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

Parameters:

name: name of the data source to purge

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" tell EasyCatalog object purge data source name "Stock.csv" end tell end tell</pre>
--------------------	---

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; myEasyCatalog.purgeDataSource("Stock.csv");</pre>
-------------------	---

<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("InDesign.Application") Set myEasyCatalog = myInDesign.EasyCatalogObject myEasyCatalog.PurgeDataSource("Stock.CSV")</pre>
---------------------	--

EASYPATALOG REGISTER**OBJECT
(CONTINUED)**

Activate a module using the given serial number.

Parameters:

serial number: serial number for the module to register

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" tell EasyCatalog object register serial number "ECxxxxx" end tell end tell</pre>
--------------------	---

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; myEasyCatalog.register("ECxxxxx");</pre>
-------------------	--

<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("InDesign.Application") Set myEasyCatalog = myInDesign.EasyCatalogObject myEasyCatalog.Register("ECxxxxx")</pre>
---------------------	---

EASYPATALOG REMOVE FIELD MARKERS**OBJECT
(CONTINUED)**

remove field markers from the given document

Parameters:

doc: document

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" set myDocument to active document tell EasyCatalog object remove field markers doc myDocument end tell end tell</pre>
--------------------	--

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDoc = app.activeDocument; myEasyCatalog.removeFieldMarkers(myDoc);</pre>
-------------------	--

<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("InDesign.Application") Set myDocument = myInDesign.ActiveDocument Set myEasyCatalog = myInDesign.EasyCatalogObject myEasyCatalog.RemoveFieldMarkers(myDocument)</pre>
---------------------	---

EASYPATALOG REMOVE SELECTED FIELD MARKERS
OBJECT remove field markers from selection
(CONTINUED)*Parameters:*

doc: document

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" set myDocument to active document tell EasyCatalog object remove selected field markers end tell end tell</pre>
--------------------	--

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDoc = app.activeDocument; myEasyCatalog.removeSelectedFieldMarkers();</pre>
-------------------	---

<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("InDesign.Application") Set myDocument = myInDesign.ActiveDocument Set myEasyCatalog = myInDesign.EasyCatalogObject myEasyCatalog.RemoveSelectedFieldMarkers()</pre>
---------------------	---

EASYPATALOG REPORT TAGS**OBJECT
(CONTINUED)**

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

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" set myDocument to active document tell EasyCatalog object report tags doc myDocument file path "Macintosh HD:tags.txt" end tell end tell</pre>
--------------------	---

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDoc = app.activeDocument; myEasyCatalog.reportTags(myDoc, "Macintosh HD:Tags.txt");</pre>
-------------------	---

<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("InDesign.Application") Set myDocument = myInDesign.ActiveDocument Set myEasyCatalog = myInDesign.EasyCatalogObject err = myEasyCatalog.ReportTags(myDocument, "c:\tags.txt")</pre>
---------------------	--

EASYPATALOG SELECTED DATA VIEW**OBJECT
(CONTINUED)**

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

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" tell EasyCatalog object tell (selected data view) set myName to data view name end tell end tell end tell</pre>
--------------------	--

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDV = myEasyCatalog.selectedDataView(); alert (myDV.name);</pre>
-------------------	---

<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("Indesign.Application") Set myEasyCatalog = myInDesign.EasyCatalogObject Set myDV = myEasyCatalog.SelectedDataView()</pre>
---------------------	---

EASYPATALOG SERIAL NUMBER**OBJECT
(CONTINUED)**

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

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" tell EasyCatalog object set mySerialNumber to serial number display dialog mySerialNumber end tell end tell</pre>
<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var mySerialNumber = myEasyCatalog.serialNumber; alert (mySerialNumber);</pre>
<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("Indesign.Application") Set myEasyCatalog = myInDesign.EasyCatalogObject MsgBox (myEasyCatalog.SerialNumber)</pre>

EASYPAGE 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

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" 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</pre>
--------------------	---

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myFrame = app.documents[0].pages[0].pageltems[0] myEasyCatalog.tagPageItem(myFrame, "test.csv", "SKU", "PRODUCTKEY")</pre>
-------------------	--

<i>Visual Basic</i>	<pre>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")</pre>
---------------------	--

EASYCATALOG TAG TEXT RANGE OBJECT (CONTINUED)

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

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" 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</pre>
--------------------	---

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myInsertionPoint = app.documents[0].pages[0].textFrames[0]. insertionPoints[0] myEasyCatalog.tagTextRange(myInsertionPoint, 5, "test.csv", "SKU", "PRODUCTKEY")</pre>
-------------------	---

<i>Visual Basic</i>	<pre>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.TagTextRange(myInsertionPoint, 5, "test.csv", "SKU", "PRODUCTKEY")</pre>
---------------------	--

EASYPATALOG UPDATE FURNITURE**OBJECT
(CONTINUED)**

Update furniture items for the given document.

Parameters:

document: document object

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" set myDocument to active document tell EasyCatalog object update furniture doc myDocument end tell end tell</pre>
--------------------	--

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDoc = app.activeDocument; myEasyCatalog.updateFurniture(myDoc);</pre>
-------------------	---

<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("InDesign.Application") Set myDocument = myInDesign.ActiveDocument Set myEasyCatalog = myInDesign.EasyCatalogObject myEasyCatalog.UpdateFurniture(myDocument)</pre>
---------------------	--

EASYPAGE OBJECT **UPDATE FURNITURE ON PAGE****OBJECT
(CONTINUED)**

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

Parameters:

page item: page item object

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" set myDocument to active document tell EasyCatalog object update furniture on page doc myDocument page 1 end tell end tell</pre>
--------------------	---

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDoc = app.activeDocument; myEasyCatalog.updateFurnitureOnPage(myDoc, 1);</pre>
-------------------	--

<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("InDesign.Application") Set myDocument = myInDesign.ActiveDocument Set myEasyCatalog = myInDesign.EasyCatalogObject myEasyCatalog.UpdateFurnitureOnPage(myDocument, 1)</pre>
---------------------	---

EASYPAGE OBJECT**OBJECT
(CONTINUED)**

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

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" 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</pre>
--------------------	--

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDoc = app.activeDocument; var myFrame = app.documents[0].pages[0].textFrames[0] myEasyCatalog.updateTags(myFrame);</pre>
-------------------	--

<i>Visual Basic</i>	<pre>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)</pre>
---------------------	--

EASYPATALOG WORKSPACE FOLDER**OBJECT
(CONTINUED)**

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

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" tell EasyCatalog object set myWorkspace to workspace folder display dialog myWorkspace end tell end tell</pre>
--------------------	---

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myWorkspace = myEasyCatalog.workspaceFolder; alert (myWorkspace);</pre>
-------------------	---

<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("Indesign.Application") Set myEasyCatalog = myInDesign.EasyCatalogObject MsgBox (myEasyCatalog.WorkspaceFolder)</pre>
---------------------	--

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

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" 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</pre>
--------------------	--

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDoc = app.activeDocument; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); myDS.addCustomField("Group Max", "GROUPMAX(Category,Price)");</pre>
-------------------	---

<i>Visual Basic</i>	<pre>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)")</pre>
---------------------	---

DATA SOURCE OBJECT (CONTINUED) ADD FIELD

Adds a field to the data source. Only applies to XML data sources

Parameters:

- field name: name of the field, must be unique
- field value: field description - xpath

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" set myDocument to active document tell EasyCatalog object tell DataSource "Stock.csv" add field field name "value" field value "/value/text()" end tell end tell end tell</pre>
--------------------	--

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDoc = app.activeDocument; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); myDS.addField("value", "/value/text()");</pre>
-------------------	--

<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("InDesign.Application") Set myDoc = myInDesign.ActiveDocument Set myEasyCatalog = myInDesign.EasyCatalogObject Set myDS = myEasyCatalog.DataSources.Item("Stock.csv") myDS.AddField("value", "/value/text()")</pre>
---------------------	--

DATA SOURCE ADOPT
OBJECT
(CONTINUED)

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

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" set myDocument to active Document tell EasyCatalog object tell DataSource "Stock.csv" adopt doc myDocument end tell end tell end tell</pre>
--------------------	--

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDoc = app.activeDocument; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); myDS.adopt(myDoc);</pre>
-------------------	--

<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("InDesign.Application") Set myEasyCatalog = myInDesign.EasyCatalogObject Set myDS = myEasyCatalog.DataSources.Item("Stock.csv") myDS.Adopt(myDoc)</pre>
---------------------	--

DATA SOURCE CHANGE KEY**OBJECT
(CONTINUED)**

Changes the nomination of key fields for the data source. This will close all existing open DSVs

Parameters:

field name: name of the new key field

JavaScript

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

DATA SOURCE COUNT DELETED**OBJECT
(CONTINUED)**

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

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" tell EasyCatalog object tell DataSource "Stock.csv" set countUpdated to count deleted display dialog countUpdated end tell end tell end tell</pre>
--------------------	---

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); alert(myDS.countDeleted);</pre>
-------------------	---

<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("InDesign.Application") Set myEasyCatalog = myInDesign.EasyCatalogObject MsgBox(myEasyCatalog.DataSources.Item(1).CountDeleted)</pre>
---------------------	--

**DATA SOURCE COUNT ERRORS
OBJECT
(CONTINUED)**

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

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" tell EasyCatalog object tell DataSource "Stock.csv" set countErrors to count errors display dialog countErrors end tell end tell end tell</pre>
<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); alert(myDS.countErrors);</pre>
<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("InDesign.Application") Set myEasyCatalog = myInDesign.EasyCatalogObject MsgBox(myEasyCatalog.DataSources.Item(1).CountErrors)</pre>

DATA SOURCE COUNT INSERTED**OBJECT
(CONTINUED)**

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

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" tell EasyCatalog object tell DataSource "Stock.csv" set countInserted to count inserted display dialog countInserted end tell end tell end tell</pre>
<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); alert(myDS.countInserted);</pre>
<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("InDesign.Application") Set myEasyCatalog = myInDesign.EasyCatalogObject MsgBox(myEasyCatalog.DataSources.Item(1).CountInserted)</pre>

DATA SOURCE COUNT UPDATED**OBJECT
(CONTINUED)**

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

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" tell EasyCatalog object tell DataSource "Stock.csv" set countUpdated to count updated display dialog countUpdated end tell end tell end tell</pre>
--------------------	---

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); alert(myDS.countUpdated);</pre>
-------------------	---

<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("InDesign.Application") Set myEasyCatalog = myInDesign.EasyCatalogObject MsgBox(myEasyCatalog.DataSources.Item(1).CountUpdated)</pre>
---------------------	--

**DATA SOURCE OBJECT
(CONTINUED)****DATA SOURCE NAME**

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

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" tell EasyCatalog object tell DataSource "Stock.csv" set myName to data source name display dialog myName end tell end tell end tell</pre>
<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); alert(myDS.name);</pre>
<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("InDesign.Application") Set myEasyCatalog = myInDesign.EasyCatalogObject MsgBox(myEasyCatalog.DataSources.Item(1).DataSourceName)</pre>

DATA SOURCE OBJECT (CONTINUED) **DATA SOURCE ODBC STATEMENT**

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

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" tell EasyCatalog object tell DataSource "MyDataBase" set myName to data source odbc statement display dialog myName end tell end tell end tell</pre>
--------------------	---

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); alert(myDS.dataSourceOdbcStatement);</pre>
-------------------	--

<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("InDesign.Application") Set myEasyCatalog = myInDesign.EasyCatalogObject MsgBox(myEasyCatalog.DataSources.Item(1).DataSourceOdbcStatement)</pre>
---------------------	---

**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.

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" tell EasyCatalog object tell DataSource "Stock.csv" set myName to data source specifier display dialog myName end tell end tell end tell</pre>
<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); alert(myDS.dataSourceSpecifier);</pre>
<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("InDesign.Application") Set myEasyCatalog = myInDesign.EasyCatalogObject MsgBox(myEasyCatalog.DataSources.Item(1).DataSourceSpecifier)</pre>

DATA SOURCE DOCUMENT REPORT
OBJECT
(CONTINUED)

Report the links for the given document as an XML file.

Parameters:

document: The document
file path: Path to the file"

JavaScript

```
var myEasyCatalog = app.easycatalogObject;  
var myDoc = app.activeDocument;  
var myDS = app.easycatalogObject.datasources.item("Stock.csv");  
myDS.setFieldOption(myDoc,    "Macintosh HD:report.xml");
```

DATA SOURCE GET FIELD NAMES**OBJECT
(CONTINUED)**

Returns an array of the data sources field names

Parameters:

<i>JavaScript</i>	<pre>var myEC = app.easycatalogObject; var myDS = myEC.datasources.item("stock.csv"); var myDSFields = myDS.getFieldNames(); for (index = 0; index < myDSFields.length; index++) { alert(myDSFields[index]); }</pre>
-------------------	---

DATA SOURCE OBJECT (CONTINUED)

LOAD FIELD DEFINITIONS

Load a previously defined set of field definitions.

Parameters:

file path: path to a field definition file

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" tell EasyCatalog object tell DataSource "Stock.csv" load field definitions file path "Macintosh HD:fields.xml" end tell end tell end tell</pre>
--------------------	--

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); myDS.loadFieldDefinitions("Macintosh HD:Fields.xml");</pre>
-------------------	---

<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("InDesign.Application") Set myEasyCatalog = myInDesign.EasyCatalogObject Set myDS = myEasyCatalog.DataSources.Item("Stock.csv") myDS.LoadFieldDefinitions("c:\fields.xml")</pre>
---------------------	---

DATA SOURCE OBJECT (CONTINUED)

LOAD FORMATTING TAGS

Load a previously defined set of formatting tags.

Parameters:

file path: path to a field definition file

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" tell EasyCatalog object tell DataSource "Stock.csv" load formatting tags file path "Macintosh HD:tags.xml" end tell end tell end tell</pre>
--------------------	--

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); myDS.loadFormattingTags("Macintosh HD:Tags.xml");</pre>
-------------------	---

<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("InDesign.Application") Set myEasyCatalog = myInDesign.EasyCatalogObject Set myDS = myEasyCatalog.DataSources.Item("Stock.csv") myDS.LoadFormattingTags("c:\tags.xml")</pre>
---------------------	---

**DATA SOURCE OBJECT
(CONTINUED)****ODBC CONNECTION**

Deprecated - Use odbc statement.

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" tell EasyCatalog object tell DataSource "Stock Database" odbc connection statement "Select * from stock" end tell end tell end tell</pre>
<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); myDS.odbcConnection("Select * from stock");</pre>
<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("InDesign.Application") Set myEasyCatalog = myInDesign.EasyCatalogObject Set myDS = myEasyCatalog.DataSources.Item(1) myDS.OdbcConnection("Select * from stock")</pre>

DATA SOURCE PICKUP OBJECT (CONTINUED)

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

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" 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</pre>
--------------------	---

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); myDS.pickup("[key]^t[description]", "^p", true, true);</pre>
-------------------	--

<i>Visual Basic</i>	<pre>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)</pre>
---------------------	---

**DATA SOURCE OBJECT
(CONTINUED)****PURGE DELETED**

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

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" tell EasyCatalog object tell DataSource "Stock.csv" purge deleted end tell end tell end tell</pre>
--------------------	---

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); myDS.purgeDeleted();</pre>
-------------------	--

<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("InDesign.Application") Set myEasyCatalog = myInDesign.EasyCatalogObject Set myDS = myEasyCatalog.DataSources.Item("Stock.csv") myDS.PurgeDeleted</pre>
---------------------	--

DATA SOURCE REMOVE FIELD OBJECT (CONTINUED)

Removes a field to the data source. Only applies to XML data sources where fields are defined using xpath

Parameters:

field name: name of the field

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" set myDocument to active document tell EasyCatalog object tell DataSource "Stock.csv" remove field field name "value" end tell end tell end tell</pre>
<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDoc = app.activeDocument; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); myDS.removeField("value");</pre>
<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("InDesign.Application") Set myDoc = myInDesign.ActiveDocument Set myEasyCatalog = myInDesign.EasyCatalogObject Set myDS = myEasyCatalog.DataSources.Item("Stock.csv") myDS.RemoveField("value")</pre>

DATA SOURCE REPLACE FIELDS**OBJECT
(CONTINUED)**

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

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" 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</pre>
<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDoc = app.activeDocument; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); myDS.replaceFields(myDoc, "Field 1", "Field 2");</pre>
<i>Visual Basic</i>	<pre>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")</pre>

**DATA SOURCE OBJECT
(CONTINUED)**

Save the data source's data snapshot to the workspace folder. This should be called, for example, after changing the content of one of the fields via scripting to ensure the snapshot is up-to-date.

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" tell EasyCatalog object tell DataSource "Stock Database" save end tell end tell end tell</pre>
--------------------	---

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); myDS.save();</pre>
-------------------	--

<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("InDesign.Application") Set myEasyCatalog = myInDesign.EasyCatalogObject Set myDS = myEasyCatalog.DataSources.Item(1) myDS.Save()</pre>
---------------------	--

DATA SOURCE SET FIELD OPTION**OBJECT
(CONTINUED)**

Set a specific field option using the fields.xml attribute name.

Parameters:

field name:	name of the new key field
option name:	name of the new option
option value:	value of the option

JavaScript

```
var myEasyCatalog = app.easycatalogObject;  
var myDS = app.easycatalogObject.datasources.item("Stock.csv");  
myDS.setFieldOption("PageNbr", "prefix", "<b>");
```

DATA SOURCE SET PAGINATION OPTION OBJECT (CONTINUED)

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

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" set myDocument to active document tell EasyCatalog object tell DataSource "Stock.csv" set pagination option option name "paginaintype" option value "0" end tell end tell end tell</pre>
--------------------	---

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDoc = app.activeDocument; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); myDS.setPaginationOption("paginaintype", "0");</pre>
-------------------	--

<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("InDesign.Application") Set myDoc = myInDesign.ActiveDocument Set myEasyCatalog = myInDesign.EasyCatalogObject Set myDS = myEasyCatalog.DataSources.Item("Stock.csv") myDS.SetPaginationOption("paginaintype", "0")</pre>
---------------------	--

DATA SOURCE SET PARAMETERS**OBJECT
(CONTINUED)**

Sets the user definable parameters of the data source.

Parameters:

parameter pair array: List of parameter key value pairs

JavaScript

```
var myEC = app.easycatalogObject;
myDS = myEC.datasources.item(dataSourceName);
var myParams = new Array;
var param1 = new Array;
param1.push("id");
param1.push("12");
myParams.push(param1);
myDS.setParameters(myParams);
var myDV = myDS.dataviews.add();
```

DATA SOURCE SYNCHRONIZE DOCUMENT**OBJECT
(CONTINUED)**

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

Parameters:

document: document object

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" set myDocument to active document tell EasyCatalog object tell DataSource "Stock.csv" synchronize document doc myDocument end tell end tell end tell</pre>
<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDoc = app.activeDocument; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); myDS.synchronizeDocument(myDoc);</pre>
<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("InDesign.Application") Set myDoc = myInDesign.ActiveDocument Set myEasyCatalog = myInDesign.EasyCatalogObject Set myDS = myEasyCatalog.DataSources.Item("Stock.csv") myDS.SynchronizeWithDocument(myDoc)</pre>

DATA SOURCE OBJECT (CONTINUED)

SYNCHRONIZE WITH DATA SOURCE

Take a new snapshot of data from the datasource.

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" tell EasyCatalog object tell DataSource "Stock.csv" synchronize with data source end tell end tell end tell</pre>
<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); myDS.synchronizeWithDataSource();</pre>
<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("InDesign.Application") Set myEasyCatalog = myInDesign.EasyCatalogObject Set myDS = myEasyCatalog.DataSources.Item("Stock.csv") myDS.SynchronizeWithDataSource</pre>

**DATA SOURCE OBJECT
(CONTINUED)**

UPDATE CUSTOM FIELDS

Force the custom fields in the data source to be update. This should be called, for example, after changing the content of one of the fields via scripting to ensure all custom fields have been updated.

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" tell EasyCatalog object tell DataSource "Stock Database" update custom fields end tell end tell end tell</pre>
<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); myDS.updateCustomFields();</pre>
<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("InDesign.Application") Set myEasyCatalog = myInDesign.EasyCatalogObject Set myDS = myEasyCatalog.DataSources.Item(1) myDS.UpdateCustomFields()</pre>

**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.

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" tell EasyCatalog object tell DataSource "Stock.csv" update data source end tell end tell end tell</pre>
--------------------	--

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); myDS.updateDataSource;</pre>
-------------------	--

<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("InDesign.Application") Set myEasyCatalog = myInDesign.EasyCatalogObject Set myDS = myEasyCatalog.DataSources.Item("Stock.csv") myDS.UpdateDataSource</pre>
---------------------	--

DATA SOURCE UPDATE DOCUMENT OBJECT (CONTINUED)

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

Parameters:

document: document object

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" set myDocument to active document tell EasyCatalog object tell DataSource "Stock.csv" update document doc myDocument end tell end tell end tell</pre>
<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDoc = app.activeDocument; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); myDS.updateDocument(myDoc);</pre>
<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("InDesign.Application") Set myDoc = myInDesign.ActiveDocument Set myEasyCatalog = myInDesign.EasyCatalogObject Set myDS = myEasyCatalog.DataSources.Item("Stock.csv") myDS.UpdateDocument(myDoc)</pre>

DATA SOURCE UPDATE PAGE NUMBERS

**OBJECT
(CONTINUED)**

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

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

<i>AppleScript</i>	<pre> tell application "Adobe InDesign CS6" 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 </pre>
--------------------	--

<i>JavaScript</i>	<pre> var myEasyCatalog = app.easycatalogObject; var myDoc = app.activeDocument; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); myDS.updatePageNumbers(myDoc, "all", "PageNum") </pre>
-------------------	---

<i>Visual Basic</i>	<pre> 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") </pre>
---------------------	--

DATA SOURCE OBJECT
UPDATE SNAPSHOT
(CONTINUED)

Update the snapshot with content from the given document.

Parameters:

document: document object

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" set myDocument to active document tell EasyCatalog object tell DataSource "Stock.csv" update snapshot doc myDocument end tell end tell end tell</pre>
--------------------	--

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDoc = app.activeDocument; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); myDS.updateSnapshot(myDoc);</pre>
-------------------	---

<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("InDesign.Application") Set myDoc = myInDesign.ActiveDocument Set myEasyCatalog = myInDesign.EasyCatalogObject Set myDS = myEasyCatalog.DataSources.Item("Stock.csv") myDS.UpdateSnapshot(myDoc)</pre>
---------------------	---

DATA SOURCE VIEW APPLY CONFIGURATION

Apply a previously saved configuration to the data view.

Parameters:

configuration name: configuration to apply

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" tell EasyCatalog object tell DataSource "Stock.csv" tell DataView "Stock.csv" apply configuration configuration name "Sports" end tell end tell end tell end tell</pre>
--------------------	--

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); var myDV = myDS.dataviews.item("Stock.csv"); myDV.applyConfiguration("Sports");</pre>
-------------------	---

<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("InDesign.Application") Set myEasyCatalog = myInDesign.EasyCatalogObject Set myDS = myEasyCatalog.DataSources.Item("Stock.csv") Set myDV = myDS.DataViews.Item("Stock.csv") myDV.applyConfiguration("Sports")</pre>
---------------------	--

DATA SOURCE VIEW APPLY FILTER (CONTINUED)

Apply a previously saved filter to the data view.

Parameters:

filter name: filter to apply

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" tell EasyCatalog object tell DataSource "Stock.csv" tell DataView "Stock.csv" apply filter filter name "Sports" end tell end tell end tell end tell</pre>
--------------------	--

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); var myDV = myDS.dataviews.item("Stock.csv"); myDV.applyFilter("Sports");</pre>
-------------------	--

<i>Visual Basic</i>	<pre>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")</pre>
---------------------	---

DATA SOURCE VIEW CLOSE DATA VIEW
(CONTINUED)

Close the data view.

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" tell EasyCatalog object tell DataSource "Stock.csv" tell DataView "Stock.csv" close data view end tell end tell end tell end tell</pre>
--------------------	--

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); var myDV = myDS.dataviews.item("Stock.csv"); myDV.closeDataView();</pre>
-------------------	--

<i>Visual Basic</i>	<pre>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</pre>
---------------------	---

DATA SOURCE VIEW DATA VIEW NAME (CONTINUED)

The name of the data view (Read only).

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" 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</pre>
<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); var myDV = myDS.dataviews.item("Stock.csv"); alert(myDV.name);</pre>
<i>Visual Basic</i>	<pre>Set myEasyCatalog = myInDesign.EasyCatalogObject Set myDS = myEasyCatalog.DataSources.Item("Stock.csv") Set myDV = myDS.DataViews.Item("Stock.csv") MsgBox(myDV.DataViewName)</pre>

DATA SOURCE VIEW EMPTY
(CONTINUED)

Empty the data view.

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" tell EasyCatalog object tell DataSource "Stock.csv" tell DataView "Stock.csv" empty end tell end tell end tell end tell</pre>
<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); var myDV = myDS.dataviews.item("Stock.csv"); myDV.emptyDataView();</pre>
<i>Visual Basic</i>	<pre>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</pre>

DATA SOURCE VIEW GROUP DATA VIEW (CONTINUED)

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

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" 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 end tell</pre>
--------------------	---

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); var myDV = myDS.dataviews.item("Stock.csv"); myDV.groupDataView("SKU", true);</pre>
-------------------	---

<i>Visual Basic</i>	<pre>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)</pre>
---------------------	--

DATA SOURCE VIEW GROUP DRILL DOWN
(CONTINUED)

Drill down the selected relational record.

Parameters:

<i>JavaScript</i>	<pre>var myEC = app.easycatalogObject; myDS = myEC.datasources.item("Relational"); var myDV = myDS.dataviews.add(); myDV.makeSelection(1,1); myDV.groupDrillDown(); myDV.makeSelection(1,1); myDV.groupDrillDown(); myDV.groupUp();</pre>
-------------------	---

DATA SOURCE VIEW GROUP UP
(CONTINUED)

Move the relational DSV view to the parent group.

JavaScript

```
var myEC = app.easycatalogObject;
myDS = myEC.datasources.item("Relational");
var myDV = myDS.dataviews.add(); myDV.makeSelection(1,1);
myDV.groupDrillDown();
myDV.makeSelection(1,1);
myDV.groupDrillDown();
myDV.groupUp();
```

DATA SOURCE VIEW HIDE FIELD
(CONTINUED)

Hide a field in the data view.

Parameters:

field name: field name to hide

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" tell EasyCatalog object tell DataSource "Stock.csv" tell DataView "Stock.csv" hide field field name "SKU" end tell end tell end tell end tell</pre>
--------------------	--

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); var myDV = myDS.dataviews.item("Stock.csv"); myDV.hideField("SKU");</pre>
-------------------	---

<i>Visual Basic</i>	<pre>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")</pre>
---------------------	--

DATA SOURCE VIEW INSERT VIEW SELECTION (CONTINUED)

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

Parameters:

start offset: start insertion point

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" 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</pre>
--------------------	---

<i>JavaScript</i>	<pre>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);</pre>
-------------------	--

<i>Visual Basic</i>	<pre>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)</pre>
---------------------	--

DATA SOURCE VIEW MAKE SELECTION (CONTINUED)

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

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" 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</pre>
--------------------	---

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); var myDV = myDS.dataviews.item("Stock.csv"); myDV.makeSelection(1,3, "Manufacturer");</pre>
-------------------	---

<i>Visual Basic</i>	<pre>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")</pre>
---------------------	---

DATA SOURCE VIEW PAGINATE (CONTINUED)

Paginate the given page item with the current view selection.

Parameters:

page item: page item object

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" 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 end tell</pre>
--------------------	--

<i>JavaScript</i>	<pre>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);</pre>
-------------------	---

<i>Visual Basic</i>	<pre>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)</pre>
---------------------	--

DATA SOURCE VIEW PAGINATE INTO TEXT FLOW (CONTINUED)

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

<i>AppleScript</i>	<pre> tell application "Adobe InDesign CS6" 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 </pre>
--------------------	--

<i>JavaScript</i>	<pre> 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"); </pre>
-------------------	--

<i>Visual Basic</i>	<pre> 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") </pre>
---------------------	---

DATA SOURCE VIEW PAGINATE TEXT FLOW RANGE (CONTINUED)

Paginate the current selection into the specified text range.

Parameters:

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

<i>AppleScript</i>	<pre> tell application "Adobe InDesign CS6" 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 </pre>
<i>JavaScript</i>	<pre> 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); </pre>
<i>Visual Basic</i>	<pre> 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) </pre>

DATA SOURCE VIEW PAGINATE USING DEFAULTS (CONTINUED)

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

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" 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</pre>
--------------------	--

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDV = myEasyCatalog.selectedDataView(); var myDoc = app.activeDocument; myDV.paginateUsingDefaults("Macintosh HD:Library.indl", myDoc);</pre>
-------------------	--

<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("Indesign.Application") Set myEasyCatalog = myInDesign.EasyCatalogObject Set myDV = myEasyCatalog.SelectedDataView() Set myDoc = myInDesign.ActiveDocument myDV.PaginateUsingDefaults("c:\library.indl", myDoc);</pre>
---------------------	---

DATA SOURCE VIEW PAGINATE USING GUIDES (CONTINUED)

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)

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" 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</pre>
--------------------	--

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDV = myEasyCatalog.selectedDataView(); var myDoc = app.activeDocument; myDV.paginateUsingGuides(myDoc, "Macintosh HD:Library.indl", 1);</pre>
-------------------	---

<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("InDesign.Application") Set myEasyCatalog = myInDesign.EasyCatalogObject Set myDV = myEasyCatalog.SelectedDataView() Set myDoc = myInDesign.ActiveDocument myDV.PaginateUsingGuides(myDoc, "c:\library.indl", 1);</pre>
---------------------	--

DATA SOURCE VIEW PAGINATE USING MASTERS (CONTINUED)

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

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" 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</pre>
--------------------	--

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDV = myEasyCatalog.selectedDataView(); var myDoc = app.activeDocument; myDV.paginateUsingMasters(myDoc, 1, "page");</pre>
-------------------	---

<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("Indesign.Application") Set myEasyCatalog = myInDesign.EasyCatalogObject Set myDV = myEasyCatalog.SelectedDataView() Set myDoc = myInDesign.ActiveDocument myDV.paginateUsingMasters(myDoc, 1, "page");</pre>
---------------------	--

DATA SOURCE VIEW REMOVE ROW (CONTINUED)

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

Parameters:

no: index number (1..n)

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" tell EasyCatalog object tell DataSource "Stock.csv" tell DataView "Stock.csv" remove row no 1 end tell end tell end tell end tell</pre>
--------------------	--

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); var myDV = myDS.dataviews.item("Stock.csv"); myDV.removeRow(1);</pre>
-------------------	---

<i>Visual Basic</i>	<pre>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)</pre>
---------------------	--

DATA SOURCE VIEW ROW COUNT
(CONTINUED)

Returns the number of rows in the view.

<i>AppleScript</i>	<pre>set myDocument to active Document tell application "Adobe InDesign CS6" tell EasyCatalog object tell DataSource "Stock.csv" tell DataView "Stock.csv" set myName to row count end tell end tell end tell end tell</pre>
<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); var myDV = myDS.dataviews.item("Stock.csv"); alert(myDV.rowCount);</pre>
<i>Visual Basic</i>	<pre>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)</pre>

DATA SOURCE VIEW SELECT GROUP (CONTINUED)

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

Parameters:

name: group name

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" 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</pre>
<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDoc = app.activeDocument; var myDV = myEasyCatalog.selectedDataView(); myDV.selectGroup("Attributes");</pre>
<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("InDesign.Application") Set myEasyCatalog = myInDesign.EasyCatalogObject Set myDoc = myInDesign.ActiveDocument Set myDV = myEasyCatalog.SelectedDataView() myDV.SelectGroup("Attributes")</pre>

DATA SOURCE VIEW SELECTED RECORDS
(CONTINUED)

Creates a new data view of the selected records

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" tell EasyCatalog object tell (selected data view) set myView to selected Records end tell end tell end tell</pre>
--------------------	--

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDV = myEasyCatalog.selectedDataView(); var myDVSelected = myDV.selectedRecords();</pre>
-------------------	---

<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("Indesign.Application") Set myEasyCatalog = myInDesign.EasyCatalogObject Set myDV = myEasyCatalog.SelectedDataView() Set myDVSelected = myDV.SelectedRecords()</pre>
---------------------	---

DATA SOURCE VIEW SELECTED RELATIONAL GROUP
(CONTINUED)

Returns a name of the selected relational group.

JavaScript

```
var myEC = app.easycatalogObject;  
var myDS = myEC.datasources.item("Relational");  
var myDV = myDS.dataviews.item("Relational");  
var levelName = myTempDV.selectedRelationalGroup();
```


DATA SOURCE VIEW SHOW ALL
(CONTINUED)

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

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" tell EasyCatalog object tell DataSource "Stock.csv" tell DataView "Stock.csv" show all end tell end tell end tell end tell</pre>
--------------------	---

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); var myDV = myDS.dataviews.item("Stock.csv"); myDV.showAll();</pre>
-------------------	--

<i>Visual Basic</i>	<pre>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</pre>
---------------------	---

DATA SOURCE VIEW SHOW ERRORS
(CONTINUED)

Filter the view to only show records flagged with errors.

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" tell EasyCatalog object tell DataSource "Stock.csv" tell DataView "Stock.csv" show errors end tell end tell end tell end tell</pre>
--------------------	--

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); var myDV = myDS.dataviews.item("Stock.csv"); myDV.showErrors();</pre>
-------------------	---

<i>Visual Basic</i>	<pre>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</pre>
---------------------	--

DATA SOURCE VIEW SHOW FIELD (CONTINUED)

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

Parameters:

field name: field name to show

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" tell EasyCatalog object tell DataSource "Stock.csv" tell DataView "Stock.csv" show field field name "SKU" end tell end tell end tell end tell</pre>
--------------------	--

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); var myDV = myDS.dataviews.item("Stock.csv"); myDV.showField("SKU");</pre>
-------------------	---

<i>Visual Basic</i>	<pre>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")</pre>
---------------------	--

DATA SOURCE VIEW SHOW RECORDS
(CONTINUED)

Assigns the view with a set of records.

Parameters:

record list: List of record key values

JavaScript

```
var myEasyCatalog = app.easycatalogObject;
var myDS = myEasyCatalog.datasources.item('mys');
var myDV = myDS.dataviews.item('mys');
var myKeys = new Array();
myKeys[0] = '1';
myKeys[1] = '2';
myDV.showRecords(myKeys);
```

DATA SOURCE VIEW SHOW SUBSET
(CONTINUED)

Show a saved subset of data in the view.

Parameters:

subset name: name of the subset

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" tell EasyCatalog object tell DataSource "Stock.csv" tell DataView "Stock.csv" show subset subset name "sports" end tell end tell end tell end tell</pre>
--------------------	---

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); var myDV = myDS.dataviews.item("Stock.csv"); myDV.showSubset("sports");</pre>
-------------------	---

<i>Visual Basic</i>	<pre>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")</pre>
---------------------	--

DATA SOURCE VIEW SORT DATA VIEW (CONTINUED)

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

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" 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</pre>
--------------------	---

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); var myDV = myDS.dataviews.item("Stock.csv"); myDV.sortDataView("SKU", true);</pre>
-------------------	--

<i>Visual Basic</i>	<pre>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)</pre>
---------------------	---

DATA SOURCE VIEW SUBGROUP DATA VIEW (CONTINUED)

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

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" 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 end tell</pre>
--------------------	--

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); var myDV = myDS.dataviews.item("Stock.csv"); myDV.subgroupDataView("Price", true);</pre>
-------------------	--

<i>Visual Basic</i>	<pre>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)</pre>
---------------------	---

DATA SOURCE VIEW SUBSET OF (CONTINUED)

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

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" 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</pre>
--------------------	---

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); var myDV = myDS.dataviews.item("Stock.csv"); myDV.subsetOf("Manufacturer", "=", "Sony", true);</pre>
-------------------	--

<i>Visual Basic</i>	<pre>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)</pre>
---------------------	---

DATA SOURCE VIEW SUBSORT DATA VIEW (CONTINUED)

Apply an additional level of sorting to the data view.

Parameters:

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

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" 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</pre>
<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); var myDV = myDS.dataviews.item("Stock.csv"); myDV.subsortDataView("Price", true);</pre>
<i>Visual Basic</i>	<pre>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)</pre>

DATA SOURCE VIEW UPDATE DOCUMENT (CONTINUED)

Update the specified document with data from the view.

Parameters:

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" 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</pre>
--------------------	--

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDoc = app.activeDocument; var myDV = myEasyCatalog.selectedDataView(); myDV.updateDocument(myDoc);</pre>
-------------------	--

<i>Visual Basic</i>	<pre>Set myInDesign = CreateObject("InDesign.Application") Set myEasyCatalog = myInDesign.EasyCatalogObject Set myDoc = myInDesign.ActiveDocument Set myDV = myEasyCatalog.SelectedDataView() myDV.UpdateDocument(myDoc)</pre>
---------------------	--

DATA SOURCE VIEW UPDATE SELECTED (CONTINUED)

Update the given document with the views selected records.

Parameters:

document: document to update

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" 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</pre>
--------------------	---

<i>JavaScript</i>	<pre>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);</pre>
-------------------	---

<i>Visual Basic</i>	<pre>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)</pre>
---------------------	---

DATA SOURCE VIEW UPDATE SELECTED SNAPSHOT
(CONTINUED)

Update the selected records in the given document.

JavaScript

```
var myDV = myEC.selectedDataView();  
var myDoc = app.activeDocument;  
myDV.updateSelectedSnapshot(myDoc);
```

DATA SOURCE VIEW UPGROUP DATA VIEW
(CONTINUED)

Remove any grouping applied to records in the view.

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" tell EasyCatalog object tell DataSource "Stock.csv" tell DataView "Stock.csv" ungroup data view end tell end tell end tell end tell</pre>
--------------------	--

<i>JavaScript</i>	<pre>var myEasyCatalog = app.easycatalogObject; var myDS = app.easycatalogObject.datasources.item("Stock.csv"); var myDV = myDS.dataviews.item("Stock.csv"); myDV.ungroupDataView();</pre>
-------------------	--

<i>Visual Basic</i>	<pre>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</pre>
---------------------	---

DATA SOURCE VIEW VISIBLE
(CONTINUED)

Determine if the DSV is visible.

Parameters:

<i>JavaScript</i>	<pre>var myDV = myEC.selectedDataView(); var IsVisible = myDV.visible;</pre>
-------------------	--

RECORD MARKED AS PLACED

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

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" tell EasyCatalog object tell DataSource "Stock.csv" tell record 1 set myMarkedAsPlaced to markedasplaced end tell end tell end tell end tell</pre>
--------------------	---

<i>JavaScript</i>	<pre>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);</pre>
-------------------	---

<i>Visual Basic</i>	<pre>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)</pre>
---------------------	--

RECORD MARKED AS UPDATED LOCALLY (CONTINUED)

Returns true if any of the fields in the record have been locally updated.

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" tell EasyCatalog object tell DataSource "Stock.csv" tell record 1 set myMarkedAUpdatedFromDoc to markedasupdatedlocally end tell end tell end tell end tell</pre>
<i>JavaScript</i>	<pre>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.markedasupdatedlocally);</pre>
<i>Visual Basic</i>	<pre>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.Markedasupdatedlocally)</pre>

RECORD MARKED IN ERROR
(CONTINUED)

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

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" tell EasyCatalog object tell DataSource "Stock.csv" tell record 1 set myMarkedInError to markedinerror end tell end tell end tell end tell</pre>
<i>JavaScript</i>	<pre>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);</pre>
<i>Visual Basic</i>	<pre>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)</pre>

RECORD PAGINATE RECORD (CONTINUED)

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)

<i>AppleScript</i>	<pre>tell application "Adobe InDesign CS6" tell EasyCatalog object tell DataSource "Stock.csv" tell record 1 paginate record page item myTextFrame end tell end tell end tell end tell</pre>
--------------------	--

<i>JavaScript</i>	<pre>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);</pre>
-------------------	---

<i>Visual Basic</i>	<pre>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)</pre>
---------------------	--

FIELD FIELD CONTENT

The contents of the field (Read only).

<i>AppleScript</i>	<pre> 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 </pre>
<i>JavaScript</i>	<pre> 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); </pre>
<i>Visual Basic</i>	<pre> 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) </pre>

FIELD INSERT TAGGED CONTENT (CONTINUED)

Insert the contents of the field as an EasyCatalog tag

Parameters:

story offset: Insertion point of a story

<i>AppleScript</i>	<pre>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</pre>
<i>JavaScript</i>	<pre>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);</pre>
<i>Visual Basic</i>	<pre>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)</pre>

FIELD MARKEDASUPDATEDLOCALLY
(CONTINUED)

Property to indicate if the field has been updated locally

JavaScript

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

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);
}
```