



WS Utilities Plugin for FrameMaker®

v2.0 User Guide

Copyright May 2013

Contents

Overview and Feature Descriptions

Important notes	2
About the shareware (evaluation) version	2
About the plugin sample files	2
About the plugin settings file	2
Graphic utilities pod	3
Graphic profiles	5
Applying a profile	5
Saving and managing profiles	5
What data is saved and where it is saved	6
Referenced graphic tracking	7
When graphic reference data is collected	7
Where and how tracking data is stored	7
How to start over if tracking data becomes invalid	8
How the tracking pod works	8
Important disclaimer	9
Table utilities pod	9
Right-click open commands	9
Important keyboard shortcuts	10
Batch utilities	10
Trademarks and licensing information	11

External Calls to WS Utilities

How to send an external call	13
General information on external calls	14
Specifying document and book arguments	14
Specifying Boolean arguments	15
Call reference	15
ApplyProfile	15
Syntax	15
Examples	15
Returns	16
Hello	16
Syntax	16
Returns	16
SetCallDelimiter	16
Syntax	17
Example	17
Returns	17
Shrinkwrap	17
Syntax	17
Examples	17
Returns	18
SizeToFit	18
Syntax	18
Examples	18
Returns	19

1: Overview and Feature Descriptions

WS (West Street) Utilities is a general purpose plugin that adds important features and amenities to FrameMaker, including.

- A new graphic utilities pod, which provides quick controls for importing, sizing, rotation, shrinkwrapping, and more (see [Graphic utilities pod](#) on page 1-3).
- A graphic “profiles” feature, allowing you to store commonly-used graphic and/or frame characteristics for quick and automatic reconfiguration (see [Graphic profiles](#) on page 1-5).
- A referenced graphics tracking feature that allows you to quickly know where any particular graphic is being used (see [Referenced graphic tracking](#) on page 1-7).
- A table utilities pod with some handy column width management features.

...and more. If you have any ideas of additional features that the plugin should have, please contact us.

Important notes

- For graphic-related features, this plugin is generally intended for use with anchored frames that contain a single graphic or drawing object, especially in the case of a reference graphic. Some operations will work with “floating” graphics and with frames that contain multiple objects, but you should be sure that the respective operation is behaving as you expect it to before using it with confidence.
- Certain graphic-related operations may also be functional on other objects such as text frames, but again, you should be aware that West Street makes no claims of reliable functionality in those cases.
- All efforts have been made to keep functionality intuitive, with a special effort to reduce GUI overhead and documentation requirements. Therefore, this document is purposefully kept as minimal as possible. If you have trouble and need more information, please feel free to contact us at info@weststreetconsulting.com.

About the shareware (evaluation) version

The evaluation version of the plugin is intended to allow you a full view of plugin functionality, with the following restrictions:

- Graphic and table manipulations through the plugin periodically generate messages that “encourage” you to purchase the licensed version. When operating on the plugin sample files, no messages are produced.
- External calls to the plugin are not supported.

About the plugin sample files

The plugin includes sample files can be found in the installation area. Normally, you should be able to open the master book by selecting **WS Utilities > Open File > Sample Files** area. We encourage you to experiment with these files to get a full understanding of the plugin. Note that the shareware version of the plugin is fully-functional on the sample files, behaving as the licensed version would on any other file.

About the plugin settings file

The `WS_Utills_Settings.txt` file is the primary location for general settings associated with the plugin. Following a successful installation and initialization, it is normally located in the Windows “user profile” area, at a path similar to the following on Windows XP:

```
C:\Documents and Settings\(\your user name)\Application  
Data\Adobe\FrameMaker\11\WestStreet\WS_Utils\WS_Utils_Settings.txt
```

On later versions of Windows, it is normally found in the `C:\Users` area or similar. In either case, the best way to open this file is using the command **WS Utilities > Open Settings File**. This command attempts to open the file in Notepad, if it can find the Notepad EXE file. There is a setting in `WS_Utils_Settings.txt` for the Notepad EXE path itself, so if the command fails, you will need to track down the file manually and edit this setting as appropriate.

This file contains a large number of settings that have a significant impact on how the plugin operates. You should review it carefully to be sure that it is configured correctly for your personal needs. All documentation on these settings is contained within the file in the form of comments. You can add your own comments by proceeding them with a semicolon (;).

NOTE: If you make changes to the file, you must save it and then select **WS Utilities > Read Settings From File** to implement them. Some settings require a full FrameMaker restart to implement.

Graphic utilities pod

The graphic utilities pod, launched from the main **WS Utilities** menu, provides a variety of controls for configuring graphic objects and/or the anchored frames. The functionality should be intuitive, noting the following:

- Many aspects of the pod are highly customizable using settings in the local settings file. You should review these settings to ensure that it is configured properly for your needs. For more information, see [About the plugin settings file](#) on page 1-2.
- The top of the pod contains tools for importing graphics that may be used in conjunction with or in place of native FrameMaker tools (under **File > Import > File**). The tools in the pod offer a wide variety of convenience options that may make it more favorable to use than native tools. In particular, it is very easy to simply choose a file from the dropdown list or to shuffle through files in the current directory with the << and >> buttons. Additionally:
 - The list of files in the dropdown are populated based on the folder associated with the currently-selected referenced graphic, or, if no graphic is selected, the last folder used to populate the list.
 - You can limit the types of files that appear in the list; for example, perhaps you only want to see BMP and PNG files in the current folder. These and other important configuration settings are found in the local settings file (see [About the plugin settings file](#) on page 1-2).

- Some functions operate on a single object or frame, while others (like shrinkwrapping) require both. For those that require both, you can select either the object or the frame. For those that require a single object, only the object that is currently selected is affected. For example, if you have an anchored frame selected and click a rotation control, you will rotate the entire frame.
- If you invoke an action designed for an anchored frame that contains a single object, but the frame actually contains two or more objects, the plugin may apply the action based on the first object it finds. You will not be able to control which object that is. The plugin may also reject the action. Therefore, whenever working with frames that contain multiple objects, it is again recommended that you run some tests to ensure that the plugin behavior is acceptable before using it in a production setting.
- When a referenced graphic is selected (or an anchored frame that contains a referenced graphic), the **Referenced file path** box shows the graphic path. You can change this path manually with the **Set** button. When specifying a path, be sure to use forward slashes (/) as path delimiters, not backslashes (\).

NOTE: The **Set** button always does a new import, regardless of whether you made a change to the path or not, and always respects the options selected below the path text box. It may be useful for reimporting the same graphic, perhaps just to restore the original aspect ratio.

- **SW** = Shrinkwrap. This action causes an anchored frame to resize around the object it contains with a 1 pt. padding on all sides. This feature should be reliable for anchored frames that contain multiple objects. Note that you can also define a keyboard shortcut for this action in your local settings file. For more information, see [Important keyboard shortcuts](#) on page 1-10.
- **STF** = Size-to-fit. This action is effectively the opposite of a shrinkwrap, instead attempting to fit the object to its respective frame, without changing the frame. The aspect ratio is always maintained, so this action may cause empty space above or below the object if its frame is not the same aspect ratio as the object. If the frame contains multiple objects, only the selected object is resized. If multiple objects are selected, behavior of this function may be unpredictable. Note that you can also define a keyboard shortcut for this action in your local settings file. For more information, see [Important keyboard shortcuts](#) on page 1-10.
- **Lock graphic/a-frame** - With this option, a resize action on an object or its frame causes a proportional resize action on the corresponding frame or object (respectively), with an overall attempt to maintain the dimensional ratio between the two. It is not a shrinkwrap or size-to-fit action.
- By FrameMaker design, anchored frames support rotation in 90 degree increments only.
- For more information on graphic profiles, see [Graphic profiles](#) on page 1-5.

Graphic profiles

Graphic profiles, managed entirely through the graphic utilities pod (see [Graphic utilities pod](#) on page 1-3), are a simple convenience for storing the configuration of an object and/or anchored frames. When you save a profile, it stores the characteristics of the selected object, which can then be applied to other objects afterwards.

When a profile is saved, it will contain data for both an anchored frame and its nested object, if both can be derived from the current selection. If you attempt to save or apply a profile using an frame with multiple objects, the behavior will be unpredictable. For more information, see [Saving and managing profiles](#) on page 1-5.

Applying a profile

To apply a profile, select it under **Profiles** in the graphic utilities pod and click **Apply**. Note that a profile may contain data for an object, a parent anchored frame, or both. Naturally, only the data present will be applied, and only to the object available for application. For example, if a profile contains both object and anchored frame data but the currently-selected object has no anchored frame, only the graphic configuration data is applied.

You can also set up a profile to be applied automatically when graphics are imported. For more information, see [Saving and managing profiles](#) on page 1-5.

Note that this feature has some overlap with the object styles feature introduced in FrameMaker 11. It has somewhat different functionality, though, so it may still be of some use to you.

Saving and managing profiles

Profiles are created, updated, and otherwise managed with the **Manage** button in the graphic utilities pod. To create a new profile based on the current selection, type a profile name in the **Profiles** text box, then click **Manage**. Otherwise, select an existing profile and click **Manage**. Afterwards, the following management functions are available:

- **Update/save** - This option causes the configuration of the currently-selected object and/or frame to be saved under the currently-specified profile name. Note the following:
 - If no object or frame is selected, this option is not available.
 - If the specified profile currently exists, it will be overwritten.

- If you are attempting to create a new profile, enter the new profile name in the **Profiles** box before you click **Manage**.
- Only data available from the current selection will be written to a profile. If the current selection does not contain both an anchored frame and a nested object, data about the missing object type is simply omitted from the profile.
- **Rename** and **Delete** - Allows an existing profile to be renamed or deleted.
- **Make ... active for auto-configuration** - When selected, this option causes the plugin to automatically apply the specified profile to any referenced graphic that is inserted with normal FrameMaker procedures. Only a single profile may be active at any given time. If no profile is active, auto-configuration is effectively disabled.
- **Disable auto-configuration** - Deactivates the currently-active profile, effectively disabling auto-configuration

What data is saved and where it is saved

Graphic profile data is saved in a graphic configurations file, whose location is specified in the WS Utilities local settings file. By default, this file is in the same folder as the local settings file. For more information about the settings file, see [About the plugin settings file](#) on page 1-2.

The graphic configurations file can be any file format supported by FrameMaker, including FM binary, MIF, and XML. Although you can open it in FrameMaker, it is **HIGHLY RECOMMENDED** that you do not alter this file manually. If you do, you could make changes that cause the entire file to become unreadable to the plugin.

The following data is saved in a profile:

Parameter	Graphic objects	Anchored frames
Angle (rotation)	X	X
Border parameters (width, pattern, etc.)	X	X
Color	X	X
Fill pattern	X	X
Height and width	X	X
X and Y location	X	
NOTE: These parameters only apply to a graphic that has a parent anchored frame, referencing its location to that frame. If a graphic does not have a frame, these parameters are not stored during profile creation and/or applied during application.		
Runaround parameters	X	

Parameter	Graphic objects	Anchored frames
Cropped (true/false)		X
Floating (true/false)		X
Alignment		X
Anchor type (location)		X
Baseline and near-side offset, if applicable.		X

Referenced graphic tracking

Optionally, the plugin can track where referenced graphic files are used and report this information instantly using a special tracking pod. The following sections describe this functionality in more detail.

When graphic reference data is collected

To use the feature, it must be enabled in the local settings file (see [About the plugin settings file](#) on page 1-2). Also in this file, you should configure the settings determine when you want graphic tracking data to be collected, during one or more of the following events:

- **When a referenced graphic is imported** - Collects tracking data for that graphic only.
- **When files are opened and/or closed** - Collects tracking data for all referenced graphics in the file.

Tracking data collection during an import happens quickly, so this setting is typically enabled at all times. The other operations can cause some delay, so discretion should be used as necessary. If you are just getting started with the feature, you should normally enable collection during file opening. Then, you can open all applicable files at least one time to collect the data. Afterwards, you may (or may not) choose to disable this setting and write data during imports only.

Where and how tracking data is stored

All tracking data is stored in text files, in a folder specified in your local settings. You can configure this folder when enabling the feature or allow the plugin to create and use a default folder. It can be any folder accessible by FrameMaker.

In this folder, the plugin stores a single text file for each folder where one or more referenced graphics reside. Inside a tracking file is a simple list of graphics in the folder,

along with the files that reference them and how many times each is referenced by each file.

Like FrameMaker, the path to a referencing (FM) file is always stored as a relative path in a tracking file, if a relative path can be resolved. Therefore, if your graphics are referenced by relative paths, you should expect that you will not be able to move your source files to a different folder or drive without invalidating the tracking data.

Note that tracking data files should never be altered manually. Any unexpected formatting in a tracking data file may cause unexpected and/or unpredictable behavior.

How to start over if tracking data becomes invalid

If you have graphic tracking enabled and you want a “fresh start,” you can:

1. Manually delete all the tracking data files in the tracking data folder.
2. Enable the option to write tracking data during file opening.
3. Open all your files once to repopulate the data.

How the tracking pod works

The pod (**WS Utilities > Graphic Tracking Pod**) is mostly intended to present information on where graphics are currently referenced, a single graphic at a time. Under default settings, when you select a graphic, the **References** box lists all places where the graphic is used, if any. In this way, you have usage information available instantly as you work.

Each item in the **References** list is a hyperlink to the respective location. Note that hyperlink and file listing behaviors are somewhat configurable with the **Options** button.

With respect to listing references in the pod, there are three ways to select a graphic, which is subsequently displayed in the **Graphic** box at the top. You can:

- Select a graphic in the active document.
- Select a graphic from the file system using the . . . (browse) button next to the **Graphic** box.
- In the extended graphic only, select a graphic in the **Files** list at the bottom.

It is important to keep in mind that a selection from the file system or the **Files** list will override any graphic selection in the active document, at least temporarily, until you select the same graphic again or another graphic.

The extended pod (accessible with the **vv MORE vv** button) includes the following additional items:

- A **Current graphics folder** box which displays the folder where the currently-selected graphic resides.
- A **Files** list that displays all the contents of the **Current graphics folder**, subject to any filtering applied by local settings and/or options available by clicking **Options**. A single click in this list selects the respective file for the pod. You can also set your options to allow a double-click to open the file in its default application, according to the Windows file type associations.
- An **Open With** button and field that allows you to open the selected file with the selected application. The applications in this list are completely configurable in your local settings. For example, if you have Photoshop installed, you can put Photoshop in the list and then use **Open With** to attempt to open any selected file in Photoshop. See your local settings file for instructions on how to configure these applications.

Important disclaimer

The tracking feature is designed as a basic amenity for monitoring where your graphics are being referenced. It works well if you have it configured appropriately for your environment and do not move your files around. However, it is not intended as a replacement for a comprehensive content management system.

Table utilities pod

The table utilities pod contains a simple set of tools for managing column widths. In addition to absolute sizing, you can also copy and paste widths from one table to another. Note the following:

- When you paste column widths, you can select multiple tables or even an entire flow.
- Functionally-identical commands to copy and paste column widths also appear in the right-click table context menu.

Right-click open commands

In your local settings (see [About the plugin settings file](#) on page 1-2), you can configure one or more “open” commands to appear in the graphics right-click menu. When invoked, the plugin attempts to open the selected graphic in the application configured for the command.

You can configure any application for use with an open command. Additionally, you can optionally configure a command to perform a filename substitution before attempting

the action. For example, assume that you edit all graphics in Photoshop using PSD source files, but then save in PNG format for import to your FrameMaker documents. If the corresponding PSD files reside in the same folder, you can configure an open command to open those files instead.

For more information on how to configure these commands, see the instructions in your local settings file. Note that these commands always appear at the top of the right-click menu and a FrameMaker restart is required to implement any local settings changes.

Important keyboard shortcuts

In the local settings file (see [About the plugin settings file](#) on page 1-2), you can specify keyboard shortcuts for several functions provided by the graphic utilities pod. By default, these shortcuts are:

- **Shrinkwrap** - Esc m p

NOTE: When specified as such, this shortcut overrides the native FrameMaker shrinkwrap shortcut. Normally, this is desirable because the FrameMaker shortcut also changes the anchored frame alignment, which is often unwanted.

- **Size-to-fit** - Esc m o
- **Reimport** - Esc m i

NOTE: This shortcut has the same effect as clicking **Set** in the graphic utilities pod, without changing the path and with only the **Auto-STF** option selected.

Additionally, the following default shortcuts are provided for table utilities pod features:

- **Copy table widths** - Esc t 1
- **Paste table widths** - Esc t 2

Batch utilities

The main **WS Utilities** menu contains the following batch utilities:

- **Size-To-Fit All Graphics** - Performs a “size-to-fit” operation on all graphics in the document contained within anchored frames.
- **Shrinkwrap All Graphics** - Performs a shrinkwrap operation on all graphics in the document contained within anchored frames. Note that by default, the shrinkwrap operation follows the plugin methodology, rather than the native FrameMaker methodology that also configures the anchored frame positioning. You can change this to use the native FrameMaker methodology with a setting in your local settings file.
- **Reimport All Graphics** - Reimports all graphics in the document at their original aspect ratios, using a default DPI for raster images. No other operations are performed, such as resizing or repositioning. Note that any previous resizing may be overridden and you may consider running **Size-To-Fit All Graphics** afterwards.

Trademarks and licensing information

This software is provided both as a shareware version and a licensed version, as applicable. If you choose to use either version, be aware that YOU ARE FULLY RESPONSIBLE for any actions or consequences that result from its use, including hardware damage and data loss. West Street Consulting and associates will not be held responsible for any damage caused by its use. USE IT AT YOUR OWN RISK!

Redistribution of the software is generally prohibited. Please contact West Street for distribution information.

Adobe and FrameMaker are registered trademarks of Adobe Systems, Inc. Quadralay, ePublisher, and WebWorks are registered trademarks of Quadralay Corporation. FrameScript is a registered trademark of Finite Matters, Ltd. FrameAC is a product of Mekon Ltd. All other marks are trademarks of their respective owners. Adobe Systems has no official or legal association with the development or distribution of this software and provides no legally-binding approvals, endorsements, or guarantees.

2: External Calls to WS Utilities

Like many FrameMaker plugins, you can make external calls to WS Utilities to invoke certain plugin activities, normally for purposes of automation. Specifically, you can call this plugin to:

- Shrinkwrap and size-to-fit graphics ([Shrinkwrap](#) on page 2-17 and [SizeToFit](#) on page 2-18)
- Apply a graphic profile ([ApplyProfile](#) on page 2-15)

These functions are fully exposed through the FrameMaker API and allow you to programmatically mimic the behavior of the plugin as used interactively through the GUI. If you require API access to other plugin functionality that is not currently available, please contact West Street with your request.

How to send an external call

To call the plugin, you can use one of three methods:

- **With the FDK `F_ApiCallClient()` function, from another API client** If you are working on another FDK client, you can use `F_ApiCallClient()` to call the plugin. This function is part of the normal FDK library and does not require any changes to your normal project settings. For more information on the function itself, see the *FDK Developer's Reference* provided by Adobe with the FDK.
- **With ExtendScript (FrameMaker 10 or later)** The ExtendScript function `CallClient()` performs the identical task of its FDK counterpart. For more information, see the ExtendScript documentation.

- **With FrameScript** FrameScript®, a scripting tool by Finite Matters, Ltd®, has a comparable function for calling FDK clients, `CallClient`. When called from FrameScript, the plugin behaves identically to a regular API call.
- **With FrameAC** FrameAC by Mekon® (www.mekon.com) is a COM-based utility that enables developers to use Visual Basic to control FrameMaker. FrameAC also provides the ability to script calls to other API clients.

For any supported operation, you pass a string to the plugin which contains a command and any applicable parameters, and the plugin sends back a numeric code indicating the results. The syntax of these strings is the same for either API or scripting calls, and is explained in detail in this document.

NOTE: The call descriptions and examples in this document are written from an FDK/API perspective, using `F_ApiCallClient()`. If you are using FrameScript or FrameAC, the basic call syntax will be the same, sent using the mechanism supported by the respective tool.

General information on external calls

Before you attempt to call the plugin, note the following:

- Certain commands require that you specify a document or book, which can be done by one of three methods. For more information, see [Specifying document and book arguments](#) on page 2-14.
- The default delimiter string between arguments in a call is three dashes (---). You can change this delimiter with [SetCallDelimiter](#) on page 2-16.
- To effectively use the external interface to the plugin, you should be familiar with the functionality and workflow of the plugin through the GUI.

Specifying document and book arguments

When a document or book identifier is required, you may specify any of the following three items:

- **An object handle ID** - The integer form of the `F_ObjHandleT` object ID for the file.
- **A filename** - A non-qualified filename, such as `MyDocument.fm`.
- **A file path** - A fully-qualified file path, such as:

`C:\MyDocs\MyDocument.fm`

With this method, you may substitute forward-slashes for backslashes. For example:

`C:/MyDocs/MyDocument.fm`

In all cases, the file must be currently open. The plugin will not open any files.

Specifying Boolean arguments

When an argument requires a Boolean true or false, you can specify it as follows:

- For **true**, you can specify 1, `true`, or any word that begins with “t”, including just `t`.
- For **false**, you can specify 0, `false`, or any word that begins with “f” (yes, any word), including just `f`.

Boolean arguments are not case-sensitive.

Call reference

This section details the external calls you can make to the plugin.

ApplyProfile

Applies an existing graphic profile to an object and/or an anchored frame.

Syntax

```
F_ApiCallClient("WS_Utils", "ApplyProfile---DocId---ObjId---Profile");
```

where:

<i>DocID</i>	F_ObjHandleT object ID of the document that contains the object and anchor frame, see Specifying document and book arguments on page 2-14.
<i>ObjId</i>	F_ObjHandleT object ID of the target object, either a graphic or an anchored frame. To default to the currently-selected object or frame (if any), specify zero (0).
<i>Profile</i>	Name of an existing graphic profile to apply.

Examples

```
F_ApiCallClient("WS_Utils", "ApplyProfile---67108775---335823012---MyProfile");
```

```
F_ApiCallClient("WS_Utils", "ApplyProfile---67108775---0---MyProfile");
```

```
F_ApiCallClient("WS_Utils", "ApplyProfile---C:\\Temp\\MyDoc.fm---335823012---MyProfile");
```

Returns

Value	Meaning
0	Communication with the plugin appears to have failed. Use Hello to test connectivity.
1	The action appears to have completed successfully.
101	Unrecognized command. Check the syntax of the command itself.
103	Incorrect number of arguments sent with command.
104	Bad document argument. See Specifying document and book arguments on page 2-14.
109	Bad profile name.
120	The action failed for an unknown reason. An invalid object ID may be the cause.
125	Bad object or frame ID.

Hello

Tests whether the plugin is initialized and receiving external calls.

Syntax

```
F_ApiCallClient("WS_Utills", "Hello");
```

Returns

Value	Meaning
0	The plugin is not initialized and/or communication failed. Possible causes include: <ul style="list-style-type: none"> The plugin is not running at all. Check the FrameMaker interface for a plugin menu(s). Your call uses a syntax that differs from the plugin name in the <code>maker.ini</code> file. Using the plugin installation instructions, be sure that the plugin is registered under the same name to which you are making calls.
1	The plugin is ready.

SetCallDelimiter

Changes the delimiter for external call arguments. The default delimiter is ---.

Syntax

```
F_ApiCallClient("WS_Utills", "SetCallDelimiterNewDelimiter");
```

where:

<i>NewDelimiter</i>	New delimiter string. Note that this string must directly follow the SetCallDelimiter command. Do not use the existing delimiter. Any characters following the command become the new delimiter.
---------------------	--

Example

```
F_ApiCallClient("WS_Utills", "SetCallDelimiter$$");
```

Returns

Value	Meaning
0	Communication with the plugin appears to have failed. Use Hello to test connectivity.
1	Delimiter change was successful.
101	Unrecognized command or no delimiter supplied. Check the syntax of the command itself.

Shrinkwrap

Shrinkwraps an object and its anchored frame.

Syntax

```
F_ApiCallClient("WS_Utills", "Shrinkwrap---DocId---ObjId");
```

where:

<i>DocID</i>	F_ObjHandleT object ID of the document that contains the object and anchor frame, see Specifying document and book arguments on page 2-14.
<i>ObjId</i>	F_ObjHandleT object ID of the target object, either a graphic or an anchored frame. To default to the currently-selected object or frame (if any), specify zero (0).

Examples

```
F_ApiCallClient("WS_Utills", "Shrinkwrap---67108775---335823013");
```

```
F_ApiCallClient("WS_Utills", "Shrinkwrap---67108775---0");
```

```
F_ApiCallClient("WS_Utills", "Shrinkwrap---C:\\Temp\\MyDoc.fm---335823012");
```

Returns

Value	Meaning
0	Communication with the plugin appears to have failed. Use Hello to test connectivity.
1	The action appears to have completed successfully.
101	Unrecognized command. Check the syntax of the command itself.
103	Incorrect number of arguments sent with command.
104	Bad document argument. See Specifying document and book arguments on page 2-14.
120	The action failed for an unknown reason. Possible causes include invalid IDs, specifying an object that does not have an anchored frame, or vice-versa.
125	Bad object or frame ID.

SizeToFit

Sizes an object to fit its anchored frame.

Syntax

```
F_ApiCallClient("WS_Utills", "SizeToFit---DocId---ObjId");
```

where:

<i>DocID</i>	F_ObjHandleT object ID of the document that contains the object and anchor frame, see Specifying document and book arguments on page 2-14.
<i>ObjId</i>	F_ObjHandleT object ID of the target object, either a graphic or an anchored frame. To default to the currently-selected object or frame (if any), specify zero (0).

Examples

```
F_ApiCallClient("WS_Utills", "SizeToFit---67108775---335823013");
```

```
F_ApiCallClient("WS_Utills", "SizeToFit---67108775---0");
```

```
F_ApiCallClient("WS_Utills", "SizeToFit---C:\\Temp\\MyDoc.fm---335823012");
```

Returns

Value	Meaning
0	Communication with the plugin appears to have failed. Use Hello to test connectivity.
1	The action appears to have completed successfully.
101	Unrecognized command. Check the syntax of the command itself.
103	Incorrect number of arguments sent with command.
104	Bad document argument. See Specifying document and book arguments on page 2-14.
120	The action failed for an unknown reason. Possible causes include invalid IDs, specifying an object that does not have an anchored frame, or vice-versa.
125	Bad object or frame ID.

