InsetPlus 3.10 User Guide

Table of Contents

Chapter 1 — Introduction

Thank you for using InsetPlus plugin for structured FrameMaker. If you have any comments or questions regarding the software or its documentation, please send them to info@weststreetconsulting.com.

ATTENTION PRE-3.x USERS: The former “classic mode” is now deprecated. Only “conref mode” is now supported.

Licensing information

InsetPlus is distributed as freeware, which means you may use it in any setting or organization in an unlimited fashion, free of charge. Furthermore, you are granted an unrestricted license to bundle it with your own commercial solutions, as applicable.

Note that West Street believes this software to be reasonably robust, but no guarantees are made whatsoever. If it deletes content, crashes your hard drive, or brings a 30-year famine and pestilence upon the land, West Street cannot be held responsible. Use it at your own risk!

Having said all that, West Street is proud of its software and is committed to delivering reliable products that serve real needs. If you encounter problems with this software or its documentation, please send them to us and we will address them with all due diligence.

Important disclaimer

Again, when you use any West Street software, you agree to do so at your own risk. West Street Consulting and affiliates are not responsible for any damages or loss to software, hardware, or data, whether by user or software error, or due to errors in this documentation.

Tutorial information

InsetPlus includes a basic tutorial which may be your best resource for getting familiar with the software. If a normal install was performed, the tutorial PDF file is located in the same folder as this document, and uses the sample files that installed into the sample files folder.

What is InsetPlus?

InsetPlus is a text inset replacement tool for structured FrameMaker. The overall concept of functionality is identical to native text insets, with which you can store chunks of content as “modules” in separate files, and then inset those modules into other files. InsetPlus takes this paradigm and adapts it to the structured environment, leveraging the power of structural metadata to offer more features and flexibility.

The primary reason for using text insets is content reuse, such that you can author a piece of content one time, then reuse it in other files an unlimited number of times. Native text insets, while robust and reliable, have certain limitations that are well-known to active users. By leveraging the richness of structural metadata, InsetPlus takes the inset concept to a higher level and overcomes some of these limitations. Specifically, InsetPlus provides the following advantages, versus native insets:

- **Editable text** Unlike native insets, the inset text comes in “live.” This means it acts like any other text and can be edited at will. It can also be refreshed from the source at any time to override manual alterations.
- **Easy path to update a module source** InsetPlus has a reverse path to update the source of an inset. Once you inset a module, it can be edited in the new document. Afterwards, you can choose to “update the source,” which takes your changes and updates the source itself.
- **No limitations on source modules per file** With native insets, your source module must be an entire flow of some document. With InsetPlus, you can put as many source modules as you want in any flow, because the modules are identified by structural metadata. The contents of a module source, therefore, can range from an entire flow down to a single character. A module source can even be contained in the same document and flow as a reference to it.
- **Source module use tracking** InsetPlus includes a feature which allows you to track where your source modules are being used.
- **Use of books for module “repositories”** In addition to storing multiple module sources in a single file, you can also store a set of “source files” in a book and point your references to that book. In this way, an inset reference can search an entire book for a source module. This type of query functionality allows you to set up a virtual “database” of module sources, with the database being nothing more technically complex than structured FrameMaker files.
- **Robust nested inset support** Because of the cascading nature of InsetPlus inset updates, module sources can inset other modules, with virtually no limit to the level of nesting. Note, however, that module nesting can become a content management challenge that must still be addressed.
- **Better visibility and management of insets** Because inset containment is based on structural metadata, you can use all of structured FrameMaker’s native conveniences to navigate and manage the content. The Structure View makes it easy to see exactly what text belongs to what inset.
- **Auxiliary management tools** InsetPlus provides a number of tools to assist with inset management, including inset coloring and reporting.

InsetPlus is based on the author’s best judgment of most-needed features, gathered from personal usage and input from users like you. If you have suggestions for InsetPlus, please send them to West Street (info@weststreetconsulting.com).

Note: This document assumes a certain level of knowledge with structured FrameMaker. Furthermore, it assumes that you will have some access to EDD development expertise, such that your EDDs can be properly updated to allow the prerequisite text inset element(s) as applicable. In other words, for further details about structured FrameMaker and EDD development, you should consult the FrameMaker documentation from Adobe.

How it works overall

InsetPlus uses the traditional text inset concept, in that a module of content is stored in one location, and reused in one or more other locations. Unlike native text insets, though, InsetPlus leverages the markup of a structured document to enhance inset functionality and overcome limitations that are inherent to an unstructured environment.

For both an inset source and an inset reference, the content is wrapped in a designated text inset element and provided the proper attribute markup to identify it. An inset source element is tagged with some form of ID and each reference to a particular source provides a pointer to that ID and the source filename. When setting up reference elements, InsetPlus manages all attribute markup automatically, allowing you to work within convenient dialog boxes to set up reference elements.

In brief, the following might be a general start-to-finish process for creating an inset with InsetPlus:

1. Create the source module somewhere by inserting a source element and putting the module content within. Then, give the source element an ID, according to the attribute usage conventions specified in your InsetPlus settings.
2. In the document where you want to inset that module, insert a reference text element and point it to the source module element. To accomplish this, the “reference” element must indicate the name of the source and the file that contains it. All this can be completed with the inset element editor.
3. Update the reference created in the previous step. When this occurs, the entire contents of the source element replace the entire contents of the reference element.

Naturally, the final step can be repeated to refresh the inset reference any time it is necessary. As you may have noticed, the concept of a “source” versus “reference” inset element is important to fully understanding InsetPlus. Before using InsetPlus, you should read “Source elements versus reference elements” on page 7.

Source elements versus reference elements

In InsetPlus, all inset sources and references must be contained in some kind of text inset elements. Source and reference elements are defined as follows:

- **Source element** Any element with some form of ID assigned, according to the conventions defined in your InsetPlus settings. Source elements should be managed using standard FrameMaker attribute tools; that is, InsetPlus has no dedicated dialog box for editing source element IDs.
- **Reference element** A reference element resides in a document wherever you intend to inset an inset source. It must indicate the ID and filename of the source in a “conref” attribute, according to your InsetPlus settings. With these two pieces of information, InsetPlus will be able to locate the source material and inset the content as needed.

The functional model is very similar to the DITA conref mechanism. Note, however, InsetPlus is not advertised or intended as a DITA tool. The DITA mechanism is used simply because it is straightforward and well-designed. Neither West Street Consulting nor InsetPlus have any association with the OASIS DITA Technical Committee or any other efforts related to the development and maintenance of the DITA standard.

Important conref note

Unlike DITA, InsetPlus does not currently require hierarchical ID recognition. That is, when searching for a source element, it searches the specified file for the ID of the source element without consideration for the IDs of any ancestors. This means the following:

- Contextual unique IDs for conref links, as used in DITA, have no application within InsetPlus. The plugin can read them, but it will ignore any hierarchical context and simply look for the base ID specified.
- If you have your local settings set up as such, InsetPlus can write fully contextual IDs when creating the conref strings during reference element creation. This option does not affect InsetPlus behavior at all, and would generally only be chosen if the links must be read by some DITA-compliant process external to FrameMaker. The option is disabled by default, resulting in conref strings that include the source element ID only. For more information, see “Local settings” on page 21.

As an example, a DITA `conref` attribute might contain something like:

```
MyDoc.xml#topicID/sourceElemId
```

...which indicates that the ID of the source element is “sourceElemId”, but it is subordinate to some other element whose ID is “topicID”. For the purposes of updating the inset, InsetPlus sees only:

```
MyDoc.xml#sourceElemId
```

The important item to note is that InsetPlus will not require the specified hierarchy in order to locate the source and update the inset, and it will only write conref strings such as this if your local settings direct as such.

Requirements to use InsetPlus

InsetPlus usage requires structured FrameMaker, as the advanced features of InsetPlus depend on structural markup. Additionally, you must be using an EDD that provides at least one element with the following two attributes:

- **An ID attribute** Source elements are identified by some form of ID.
- **A “conref” attribute** On a reference element, the pointer to the desired source is contained in a “conref” attribute.

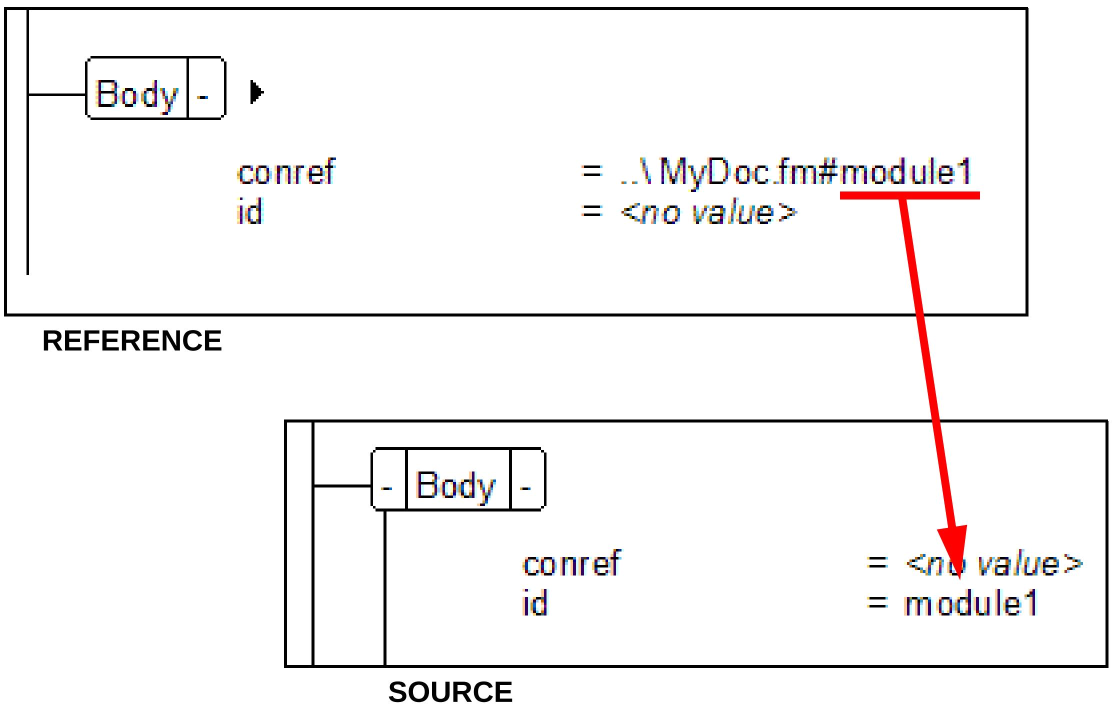
The names of these attributes is defined in your InsetPlus settings (see “Local settings” on page 21). Typically, the ID attribute is any attribute defined as a unique ID in the EDD and the conref attribute is simply “`conref`”.

More about inset elements and attributes

Typically, a source inset must have the same tag as any reference elements that point to it. The particular tag is not important, only that it is the same between source and reference elements. Therefore, in your structure definition, any element to be used for a source or reference element should have both a source name and a `conref` attribute.

The requirement for element tags to be the same is drawn from the DITA standard. InsetPlus provides an option to disable this requirement in your local settings. For more information, see “Local settings” on page 21.

A reference element and its respective source might appear as follows:



conref = ..\MyDoc.fm#module1
id = <no value>

conref = <no value>
id = module1

The following rules apply:

- The source file and the source element ID are separated by a pound sign (#).
- The source file path is normally relative, although InsetPlus does allow you to use absolute paths. If you use absolute paths, you should be aware that you are deviating from the normal conref standard.

If you want to use attributes other than `conref` and the unique ID, you can change this specification in your local settings. For more information, see “Local settings” on page 21.

Valid element types for inset elements

As defined in an EDD, the following element types are valid for use as inset elements:

- Container
- Table
- Table Row
- Table Cell

The remainder are either known to be unsupported or are currently untested. In all cases, a reference element and its corresponding source should be the same type. Note that for table rows, the source and reference elements must contain the same number of cells, otherwise an update will fail.

Translation of the InsetPlus interface

InsetPlus supports customizable translations of its menus, dialog boxes, and messaging, based on “lookup” files that you can create and edit. When a string is required for a dialog box control or a message, it looks for that string in one of these lookup files according to the currently active language. Note the following:

- West Street does not claim support for any foreign language, only that you may add your own translations as desired. You can use this feature to implement a real language or simply rename labels, etc. using text that you like better. The plugin installs with a sample “Bogusian” language intended to serve as a model for setting up another language.
- West Street does not guarantee that any particular feature will work correctly once you implement a new language. We intend for it to work and will address any problems you find; however, you should be aware that it is impossible to fully test a feature with a virtually infinite number of variations/permutations.
- West Street believes that translation features cover about 95% of the strings that are associated with active features. This means that a small percentage of strings remain fixed in English, especially as related to short prompts and other messaging. Additionally, note that none of the features scheduled for deprecation support translatable strings, such as the transformation features.
- West Street believes that the unicode range of character sets is fully supported for replacement text. The Bogusian sample provides an example of this.
- West Street believes that this feature is generally applicable for specialized use by select users only. For that reason, this documentation is brief. If you need assistance with translation features, please contact us and we will be happy to help.

Selecting a language

To select a language, select **InsetPlus > Language > Set Language**. Any languages that are properly configured will appear in the list (see “Language configuration” on page 10). A language change takes effect immediately.

You can also set a default language upon startup in your preferences file (see “Local settings” on page 9). This setting provides an option to default to the current language in use by the FrameMaker interface. Again, be aware that any setting in this file must represent a properly-configured language

Language configuration

For any new language, the plugin requires two lookup files, both of which much reside together in the plugin installation folder or the settings folder. These files are named as follows, where `language` is the case-insensitive language name that will appear in the **Set Language** dialog box:

- `InsetPlus_Strings_Dialogs_(language).fm` - The lookup file for strings that appear in the menus and major dialog boxes, such as the Node Wizard. This file consists of a set of tables with the English text in the left column and the replacement text in the right. For each dialog box string, the plugin effectively starts with the English text and attempts to look up the translation based on the contents of this file.
- Note that for these types of strings, the plugin is starting with “built-in” English versions. Therefore, when set to English, the plugin does not use this file. That is, a file named `InsetPlus_Strings_Dialogs_English.fm` will never be used. However, it is always used for any other language.

- `InsetPlus_Strings_General_(language).fm` - The lookup file for all other strings that strings that appear in error reports, short interactive prompts, and other places. The strings in this file are looked up based on an ID string, rather than the full English version. For this type of file, a `InsetPlus_Strings_General_English.fm` file does exist and is the source of all English strings that relate to prompts and messaging.

The two different files with their differing methodologies are required to accommodate how FrameMaker handles strings with respect to dialog boxes versus other functional areas, when programming to its API. Further explanation on this subject is beyond the scope of this document.

Once both of these files are properly-named and reside in the installation or settings folder, the respective language name automatically appears in the **Set Language** dialog box. Note the following:

- For new languages, the best approach is to copy the “Bogusian” examples and use them as templates. Each file contains additional instructions within.
- If you alter the file structure or otherwise make changes beyond that described in this document or within the files themselves, the results could be completely unpredictable. At worst, you may cause FrameMaker to crash.
- The strings files can also be stored in MIF format.

Additional language utilities

The plugin includes the following additional utilities in the **Language** menu that may be used rarely, if at all:

- **Create Dialog Strings File** - Creates a new dialog and menu strings file with English text only, ready for translation to a new language.
- **Update Dialog Strings File** - Attempts to update an active dialog and menu strings with the latest English strings used by the plugin. You must have a valid strings file currently open. Any new English strings are added as new rows to the respective tables. Any strings in the file that appear to be unused are colored red.

Note the following:

- These features were originally intended as a convenience for making updates, but may be deprecated. Again, it is recommended that you use the Bogusian files as templates instead.
- These features apply to the dialog strings file only. For the general strings file, you must always use an existing file as a template and all maintenance is done manually.

Limitations

InsetPlus is not a content management system. While it allows granular content reuse and basic source module tracking, it does not provide any other overhead or automation. For example, if you update a module source, there will be no automatic update to your documents that reference it. For best results with InsetPlus, you should always remember to manually update all inset references in a document or book before publishing.

Technical details

For those of you who like to know more about the internal workings, note the following:

- **Source file format** A source module must be in either structured FrameMaker, XML, or SGML format. InsetPlus cannot inset text from any file without XML or XML-like structural metadata. Note that this ability to inset text from XML may provide a convenient means of inserting XML content into your document without read/write rules or a structured application. Once content is inset into a complete template, that EDD will take over and format the text accordingly, provided that the element tags in the XML match the element tags in the EDD.
- **How content is transferred from source to reference** When updating a text inset, the actual internal process that transfers the content is no more complex than a programmatic copy and paste. This ensures that native conveniences associated with FrameMaker copy/paste are preserved, such as the automatic readjustment of referenced file and graphics links.
- **Applicable flows** Currently, InsetPlus processes the main flow only, for any reference or source element activity. This limitation can be removed if dictated by popular demand.

Trademarks

Adobe® and FrameMaker® are registered trademarks of Adobe Systems, Inc. InsetPlus is not a product of or endorsed by Adobe Systems, Inc., and West Street Consulting is a third-party entity not officially associated with Adobe Systems, Inc. Further legal information concerning Adobe and FrameMaker can be found at www.adobe.com.

Chapter 2 — Managing Inset Sources and References

This section contains information on creating and updating inset sources and references.

Right-click menu behavior

The majority of individual element functions and commands are found in the right-click popup menus in the Structure View and document window. All commands are based on invoking some function on a text inset element in the tree. If you have an entire text inset element selected, the function will apply to that element. If you do not, InsetPlus will walk up the tree from your current selection or insertion point and choose the first applicable element it finds; that is, the first element with a populated conref attribute.

Briefly, the right-click commands are as follows:

- **Edit Inset Element** Launches the inset element editor, with which you specify the parameters for an inset element. For more information, see “Editing a reference inset element” on page 13.
- **Update Inset Reference** Updates the selected or nearest ancestor reference element. It applies to inset references, not sources.
- **Jump To Inset Source** Jumps to the source document and element of the nearest inset reference. Applies to reference inset elements only.
- **Update Inset Reference And Nested Insets** Updates the selected or closest ancestor reference element, then performs a cascading update of any nested insets that came in with the update. For more information on nested insets, see “Nesting inset references within inset sources” on page 14.
- **Clear Inset Reference** Deletes the entire contents of the nearest inset reference element. This command does not affect inset source elements.
- **Select Inset Element** Selects the nearest inset element, reference or source.
- **Update Inset Source** Finds the nearest inset reference element, and updates the source element with its contents. This command effectively performs the reverse process of a reference update. For more information, see “Updating sources” on page 14.
- **Report Source Module Usage** Produces a source module tracking report for the selected source inset, or the source referenced by the selected reference inset. For more information, see “Source module tracking” on page 21.

Editing a reference inset element

Note: InsetPlus element tools apply to the configuration of reference insets only. Source elements are identified by a simple ID that may be configured using standard FrameMaker attribute tools.

Reference elements are edited through a dialog box that is produced when you right-click on the element and select **Edit Inset Element**. You can launch the dialog box through the

Structure View or the document window. Although inset element parameters are stored as attribute data, you should avoid editing the attributes directly through the attribute editor. InsetPlus expects certain syntaxes and configurations that could easily be violated by manual use, causing an inset element to become non-functional.

The following is some general information about the editor:

- **Source file specification** You can specify a source file by either selecting from an open files list, or browsing the file system.
- **Previews** The editor provides a basic preview function, but only if the source file(s) are open. If you've selected a book as the source file, the previews will only include those book files that are open.

  Tip: You can double-click a name in the previews list to automatically populate the Inset name box.
- **Inset history** The drop-down list associated with the inset name contains a basic history of insets you have used before. By selecting an item from the list, the inset name and source file will automatically be populated in the editor. The history is limited to 100 items, and you can rearrange and/or delete items by selecting **InsetPlus > Edit Inset History**.

Source module management

An inset source is a module of content intended for reuse somewhere else. It must be contained in a text inset element, and when referenced, the entire contents of that element are copied to the place where it is referenced.

Naming and storing inset sources

You can store an inset source in any FrameMaker document, with any ID. You should be sure to use a unique ID within that document for any source, and if you intend to reference the source at a book level, the name must be unique throughout the entire book.

Updating sources

You can automatically update a source from a place where it is referenced. That is, when you can make changes at the reference point and then refresh the source with your changes. This process must occur at the point of reference because the source itself does not know what reference element to update itself from. For more information on updating a source from a reference, see “Updating the source of an inset from the reference” on page 19.

Nesting inset references within inset sources

An inset source can contain any nature of content, including nested references to other sources. For more information, see “About nested insets” on page 18.

Inset reference management

An inset reference is a place where an inset source is referenced, or in other words, where it is inset. It is represented by a text inset element, whose attributes specify the file and name of the source inset element, and whether the path to the source file is absolute or relative. When the reference is updated, the entire contents of the source element are copied over the current contents of the reference element. The reference text inset element must remain at the reference point to preserve the link to the source, serving as a sort of anchor for the inset.

To create an inset reference

1. Insert a text inset element in the location where you want to inset content.
2. If the inset editor does not appear automatically (based on your InsetPlus settings), right-click on the element and select **Edit Inset Element**.
3. In the inset element editor, edit the following:

Inset ID	The name/ID of the source module to reference. This value should be exactly the same as the name specified when the source was created. ***Tip:*** If you have a source file specified and it is currently open, you can view the source names in the previews area. Also, you can automatically populate the **Inset ID** text box by double-clicking an item in the **Preview** area.
Source file select/ browse	Means by which to designate the file that contains the source element. You may select a FrameMaker document or a book.
Use relative file path	Causes InsetPlus to attempt a relative path specification. If it fails, it will produce a warning and automatically set the path type as “Absolute” when you click OK. The only reason it should fail is if you select a source file that is on a different drive than the document you are authoring.
Show inset previews	Allows a basic view of insets and respective content in the specified source file. ***Tip:*** You can double-click an inset name in the left pane to automatically populate the **Inset ID** text box above.
Show potential sources that currently have no ID assigned	Allows you to view previews for potential source elements that currently have no ID assigned. If you have chosen to identify source elements by unique ID attributes, this will allow you to view previews for all elements in the source file, whether or not the unique ID attribute is populated. In any case, you cannot reference a currently unnamed source.
Jump To Source	Closes the editor and jumps to the inset source that was specified in the editor. If the selection in the **Preview** area differs from the ID specified in the text box, the previews selection is used.

4. Click **OK**, and if prompted to update the inset now, choose **Yes** or **No**. An inset reference can be updated at any time the source file is accessible.

To remove an inset reference

Unwrap the inset element, if possible.

-or-

Use the FrameMaker attribute editor to delete the entire contents from the conref attribute.

Updating inset references

You can update references one at a time, a whole document at a time, or a whole book at a time. In either case, note the following important information about updating insets:

- When an inset updates, the contents of the reference element are replaced by the contents of the source element. Any manual edits within the reference element will be overwritten.
- If a particular source file is not open, InsetPlus cannot access the source element. InsetPlus can attempt to open files, if you have your local settings set as such. For more information, see “Local settings” on page 21.
- Updating references at a document or book level is just that - a reference update. Any source text inset elements found in the document or book are unaffected. For that matter, during any kind of reference update, your source content should never be affected, other than the replacement of any content within valid reference inset elements.
- InsetPlus does provide a “reverse path” for updating a source from a reference, on an element-by-element basis. For more information, see “Updating the source of an inset from the reference” on page 19.

  Note: If your local settings are set to allow it, you may also update all sources referenced by an entire book or document. For more information, see “Updating all sources referenced by a document” on page 20.
- When an inset is updated and content comes in, that content will respect the properties of the referencing document, such as conditional text and variable settings. If the content comes from a document with a different EDD, the content will assume the rules of the EDD of its new home, as applicable.

  Notes: When you update an inset reference, the existing content (if any) of the reference text inset element is replaced with the content from the source element. If you previously made any manual changes to that existing content, they will be overwritten.

To update a single inset reference

In the document window or Structure View, right-click on or within the reference inset element and select **Update Inset Reference**.

—or—

Select **More Inset Element Commands > Update Inset Reference And Nested Insets**.

If you do not include nested insets, the update will include the current inset only. If you include nested insets, any nested insets will also be updated in a cascading fashion. If you do not update nested insets, but the inset does include nested insets, those nested insets can be updated individually after the initial update.

To update all inset references in a document

With the document window active, select **InsetPlus > Update All Insets In Document**.

Note: The update launch dialog includes some general options for file opening and reporting. These options mirror those found in your local settings, and if you change them in the launch dialog, those changes will remain in effect until you change them again. Each time you start FrameMaker, though, your local settings will be restored. For more information on local settings, see “Local settings” on page 21.

For more information on the source module tracking option, see “About whole-document updates and source tracking” on page 24.

To update all inset references in a book

With the book window active and all chapter files open, select **InsetPlus > Update All Insets In Book**.

Note: The update launch dialog includes some general options for file opening and reporting. These options mirror those found in your local settings, and if you change them in the launch dialog, those changes will remain in effect until you change them again. Each time you start FrameMaker, though, your local settings will be restored. For more information on local settings, see “Local settings” on page 21.

For more information on the source module tracking option, see “About whole-document updates and source tracking” on page 24.

About attribute value transfer during updates

Optionally, InsetPlus can transfer attribute values from source elements to reference elements during updates. All settings related to this feature are found in the local settings file (see “Local settings” on page 21). Note the following:

- When attribute values are transferred, **the markup of your files is changed.** Be sure that this is what you intend before using the feature.
- By default, the feature is completely disabled.
- In addition to the overall enable/disable setting, the settings file contains a number of options to determine exactly which attributes qualify for the transfer. For example, you may or may not want existing values on a reference element to be overwritten. Again, review the local settings file carefully for an understanding of the options available.
- When a transfer occurs, the exact contents of an attribute are transferred, including multiple values as applicable.
- Values from the following attributes are never transferred, regardless of any settings: the conref/ID attributes used by InsetPlus, unique ID attributes, and IDReference (cross-reference) attributes.

About relative versus absolute file paths

When you specify a source file for a reference inset element, you can have InsetPlus attempt to use a relative file path. The advantage to relative paths is that you can transport your reference and source file to a new location, and if the folder structure is identical, all inset links will be preserved.

The choice whether to use relative or absolute paths is purely dependent on your workflow and network architecture. Relative paths are typically preferred, if feasible.

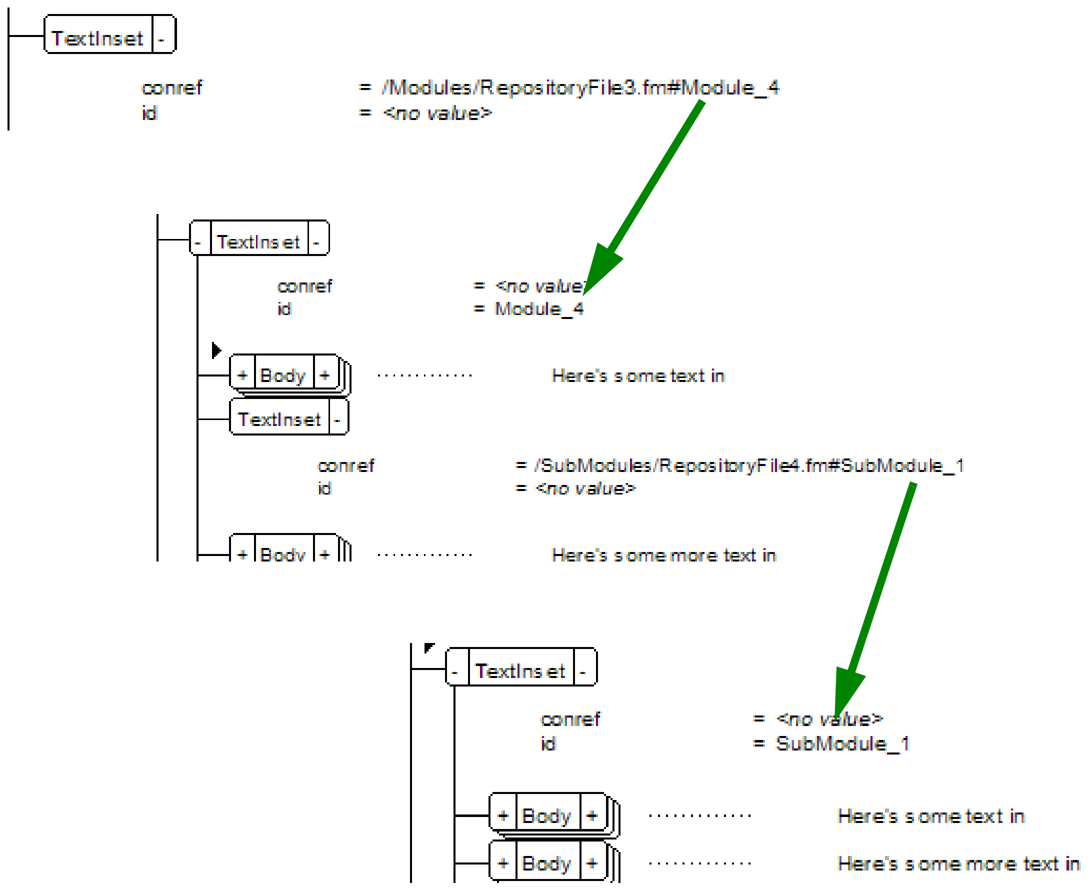
Note: InsetPlus makes all effort to readjust relative paths in the case of nested insets, if the content changes folders when its parent inset is updated. It also attempts to adjust relative paths of any reference elements if a file is saved to a new folder. If it cannot resolve an adjusted relative path, it will reset the path to absolute.

About nested insets

InsetPlus provides the ability to nest insets, with no physical limitation on the depth of nesting. An nested inset is normally represented as a reference inset element within a source inset element. This setup effectively represents an inset source that contains an inset itself, and when referenced in another document, the internal inset becomes a nested inset.

As an example, consider the following figure, which shows:

- A reference inset (to source “Module_4”)
- The actual source for Module_3, which references another source, “SubModule_1”
- The actual source for “SubModule_1”:



The middle diagram shows the source for Module_4, which contains a Body element *and* another inset reference to “SubModule_1,” effectively making SubModule_1 a nested inset, whose source is shown in the lower right.

When updating insets, InsetPlus works in a cascading fashion, and walks through the tree of any new content looking for more inset references. That is, whenever an inset is updated, InsetPlus copies in the specified content, and then walks down through that content looking for any more references to update, and so forth until all references are updated.

InsetPlus attempts to automatically adjust all relative paths when handling nested inset updates. For example, since the SubModule_1 reference points to a source in a different folder than the source for Module_4, an update process would attempt adjust that relative path when Module_4 is updated. Thus, under normal circumstances, relative paths can be used safely within nested insets.

It should be noted that nested insets should be used cautiously and judiciously, especially when combined with relative file paths. With extensive nesting, your documents and module repositories can become very complex and difficult to manage. In addition, the reliability of relative paths within nested insets is only valid when you perform whole-document updates, because the adjustment of relative paths depends on the cascading nature of the whole-document update. If you attempt to update individual nested insets in a random fashion, the automatic maintenance of relative paths may be disrupted and some reference links could become broken.

Updating the source of an inset from the reference

For any inset reference with a valid link to the source, you can right-click on it and select **More Inset Element Commands > Update Source**. This command has the reverse effect of a reference update, in that it replaces the contents of the source element with the current contents of the reference element. This function is provided as a convenience for making edits to your source modules. However, you should note the following:

- If you change a source module, any other place that references that module will change too, but only with the next reference update. If you are in a collaborative, heavy reuse environment, frequent usage of this command may not be the most practical means of managing your source modules.
- InsetPlus can only perform the update if the source document is currently open, or can be opened in an editable fashion.
- If InsetPlus opens the source and makes the update, it will not close it again without prompting you. Closure requires the document to be saved, and it is the intent of West Street that this plugin *never* makes permanent changes to your content without your consent.
- If the source file was already open, your update will not become permanent until you save that file. If you have your local settings set to use message boxes for single inset updates, you will receive a reminder following the update.
- When a source is updated, nothing happens automatically to those places that reference it. InsetPlus is not a content management system and does not provide this level of automation. However. if you are using the source module tracking feature, you can produce tracking reports that show all places where sources are reference. In any case, it is recommended that you make it a habit to always update all your references in a document or book before publishing. For more information on source module tracking, see “Source module tracking” on page 21.

Updating all sources referenced by a document

If you have your local settings set up as such, you can have the InsetPlus menu include the command to **Update All Referenced Sources**, which will locate all reference elements in the active file and update their respective sources. The command has the same effect as if you right-clicked on every reference element in the file and selected Update Inset Source. For more information on setting this preference, see “Local settings” on page 21.

YOU SHOULD MAKE VERY SURE that you want this command in your menus before enabling it. If something is amiss in your reference files, this could cause sweeping and possibly devastating alterations to your source library. Consider yourself warned. If you accidentally select it and mess up all your data, don’t call us.

Chapter 3 — Preferences and Utilities

InsetPlus provides a set of management utilities and customizable settings to assist with inset management.

Local settings

All local settings (general preferences, etc.) are stored in a text file that can be accessed by selecting **InsetPlus > Open Settings File**. Note the following:

- **The settings in this file are critically important to the behavior of InsetPlus.** For any new installation, you should review the entire file carefully to be sure that InsetPlus is properly configured for your usage.
- By default, the settings file is named `InsetPlus_Settings.txt` and is stored in the Windows “user profile” area. The following is a sample path on Windows XP for the user “rward”:

  ```
  C:\Documents and Settings\rward\Application
  Data\Adobe\FrameMaker\10\WestStreet\InsetPlus\InsetPlus_Settings.txt
  ```

- When you select the command mentioned above, InsetPlus attempts to open the file in Notepad. If the file fails to open, you may need to adjust the path to the Notepad EXE file in the settings file (after opening the file manually).
- All documentation for individual settings is contained within the file.
- Changes to settings in the file have no effect until you save the file and select **InsetPlus > Read Settings From Settings File**.

Source module tracking

InsetPlus provides a simple tracking mechanism that allows you to see where your source modules are being used. When enabled, it stores tracking data each time an inset source or reference is updated. At any point, you can produce a hyperlinked report on a source inset and easily see what document(s) reference it and where.

A demonstration of the tracking feature is included in the tutorial. While the information in this *User Guide* is important, it may be worth your time to do the tutorial first and see the feature in action.

Note: **InsetPlus is not a content management system and the tracking mechanism is not a comprehensive solution for managing your module library. It is a basic tool for reporting on where your source modules are currently referenced. It works well in an environment where source and reference files remain in static locations. However, if your content changes locations or you have redundant locations for your content, the tracking mechanism may require some study to understand how it works before it is useful to you.**

Requirements to use source module tracking

- Source tracking turned on in your local settings (see "Local settings" on page 21.
- A valid local or network folder specified for InsetPlus to store its tracking data. See "More about the tracking data folder" on page 24.
- All drives containing reference and source files mapped to a Windows drive letter, for example, `C:`.
- Source files that do not change their path locations, with the exception of being moved to another mapped drive within the same directory structure.

How source module tracking works - Overview

For every reference-to-source module link, the tracking mechanism evaluates how many references to that module are present in the reference document and stores that count. Then, when you produce a tracking report for a module, the report will contain a hyperlink to each of those references, labeled sequentially as #1, #2, etc. It does not actually record any explicit links between single reference and source elements.

For example, assume that you have a source "Module1" in `SourceFile.fm`. And, you have four references to Module1 in another file, `ReferenceFile.fm`. When you update one of these references, InsetPlus will count those four references and store the count in its tracking data. Then, if you produce a report on Module1 usage, you will see four links to the four reference elements, labeled something like:

C:\ReferenceFile.fm

(Relative path link)

- Reference #1
- Reference #2
- Reference #3
- Reference #4

Total references found for this module (at report generation): 4

The tracking data does not actually know what these specific reference elements are. It only knows that there are four contained in the file `ReferenceFile.fm`, and when you click a link, it finds the appropriate instance in document order. The reason for this behavior is that two reference elements in the same file to the same source do not inherently contain any unique quality that can distinguish them. They both have the same conref attribute markup and both contain the exact same text. Therefore, rather than place a requirement for all reference elements to be assigned some unique ID, this alternative methodology was implemented to keep the feature simple and useful out-of-the-box. The feature still allows you to find each individual reference element, only without any unique identifying characteristic for each.

Reporting source module usage

The tracking feature includes three ways to report on source usage:

- **Right-click menus** You can right-click on any source or reference element and select **More Inset Element Commands > Report Source Module Usage**. This command will produce a report on all reference elements that point to the selected element. If the selected element is a reference element, InsetPlus will first determine the source of the inset, then report on the source.

- **Whole-file report** You can generate a report on source modules within an entire document or book by selecting **InsetPlus > Source Module Tracking > Generate Audit Report (Whole File)**. This command produces a report for all referenced source elements found within the active document or book, according to the most recent tracking data. This command parses the active file for all potential sources, looks up the tracking data for each one, and reports its findings.
- **Whole-library report** You can generate a report of the entire source module library by selecting **InsetPlus > Source Module Tracking > Generate Audit Report (Whole Library)**. Note that this command uses the current tracking data only and does not parse any files. If you have a source element somewhere that is not accounted for in the tracking data, it will not appear in the report at all. Also, this command requires you to specify the default location for your source files. For more information on why this is required, see “About tracking data and absolute vs. relative paths” on page 24.

Starting to use the tracking feature

To start using the tracking feature, you need only turn it on and set the tracking data folder. For more information, see “Local settings” on page 21.

Populating your tracking data

Tracking data is recorded any time you:

- Update an inset. This includes whole-document update actions, if you chose the option to **Do source module tracking updates** in the launch dialog box.
- Open and/or close a file, if enabled (respectively) in your local settings (see “Local settings” on page 21).

If you are just starting to use the source tracking feature, a good way to initially populate your tracking data is to enable the option to update tracking data upon file open or close, then open and close all of your reference documents. Depending upon your usage module, it also may be useful to periodically delete all of your tracking data then repeat the steps above to refresh the data with clean information.

More about the source module tracking report

When you produce a tracking report, it is always created based on the current tracking data. For more information on ways to produce a report, see “Reporting source module usage” on page 22.

Because the report is based on tracking data without any direct analysis of your actual files, the report could contain errors if you have made structural alterations that are not reflected in the tracking data. For example, if you delete a reference element in a particular file but do not update insets in that file afterwards, the tracking mechanism may not know that this reference element no longer exists and will still report on it.

To help maintain accurate tracking data, a tracking report includes a data cleanup feature for non-existent reference elements. If you click a reference element hyperlink and InsetPlus is unable to find the element, it will warn you as such and ask you whether you would like to update your tracking data. If you click Yes, your tracking data (and the report) will be updated to reflect a revised number of occurrences for the respective reference element.

More about the tracking data folder

InsetPlus must store information about source module usage in some folder which you can specify in the source module tracking preferences. For more information, see "Local settings" on page 21.

This folder may be any folder on your computer or a network drive. If you want to use source module tracking in an enterprise fashion, you should reserve a folder on a network to contain this data and point all InsetPlus users to this folder.

Note that tracking data contained in this folder is generally oriented towards relative file associations between source and reference elements. For more information, see "About tracking data and absolute vs. relative paths" on page 24.

Within the folder, tracking data is stored within text files, which you can open with any text editor. While you may poke around in these files, it is highly recommended that you do not alter them in any way. InsetPlus depends upon the prescribed syntax and format of these files and any unexpected alterations may, at best, cause the tracking mechanism to malfunction, and, at worst, cause FrameMaker to crash.

About whole-document updates and source tracking

When you update insets in a whole file, the update dialog includes an option to perform source module updates as well. With this option checked, the source module tracking data is refreshed for each source module with each inset update, exactly as it would during an individual inset update.

Because tracking data updates slow down the overall update process, you may choose to disable this option if you know that your tracking data is current. However, if you can tolerate the slower update speeds, it is normally good practice to leave it enabled.

About tracking data and absolute vs. relative paths

InsetPlus is designed to support seamless usage of relative path designations to source modules. This feature allows a whole document set, including reference and source files, to be moved as a unit to any network location and still allow the links between references and sources to be preserved.

A disadvantage of this flexibility is that it adds a layer of complexity to source module tracking. For example, assume that you are working on drive `C:` and you have created a number of relative path references to other files on `C:`. Then, you move all your files as a set to drive `R:`. Because your paths are relative, all your insets should still update properly. However, if you produce a report on one of your source modules on drive `R:`, you would expect that the hyperlinked report would lead you to the respective reference file on drive `R:`, not the original drive `C:` where it was originally created.

For this reason, InsetPlus tracking data does not adhere to drive letters when storing information about relative path links. It stores the relative path only. As such, when generating a source module tracking report, it handles path designations as follows:

- If you are reporting on a single module or a whole file and the tracking data specifies a relative path, the respective hyperlink in the report will assume the same drive as the drive of the source file on which you are reporting.

- If you are reporting on the whole library and the tracking data specifies a relative path, the hyperlink in the report will assume the drive letter you specify in the dialog that launches the report. This step is necessary because the reporting mechanism would not otherwise know the location of the source file or the relative reference file, because in a relative path architecture, the location could be on any drive.
- If you are reporting on a single module, a whole file, or the whole library, and the tracking data specifies an absolute path, the hyperlink in the report will resolve to the absolute path.

In general, if you intend to use the source module tracking mechanism, it is recommended that you do not mix absolute and relative paths. If you do, you should take some time to get familiar with the behavior of the tracking mechanism and understand how it works.

About element tags and source tracking

For best results, it is highly recommended that you adhere to the convention of using the same element tags for source and reference elements. The use of differing tags could produce unexpected results and will result in InsetPlus disabling some source tracking features. You can configure InsetPlus to require identical tags with a setting in your local settings (see “Local settings” on page 21).

Tracking data maintenance and cleanup

Through normal InsetPlus usage, source module tracking data should maintain itself. However, in the event that you do make changes that require a major cleanup, such as the movement of files, you can delete all of your tracking data files and repeat the steps recommended for new users (see “Starting to use the tracking feature” on page 23).

General utilities

This section describes the general management utilities provided with InsetPlus.

Inset inventory report

With a book or document active, you can select **InsetPlus > Inset Management > Generate Inset Report** to produce a hyperlinked inventory of reference insets within the active file. The report is generated in read-only format such that the hyperlinking works automatically. To make the report editable, type the standard FrameMaker shortcut sequence Esc F l k.

Coloring insets

With a book or document active, you can select **InsetPlus > Inset Management > Color Insets** to color your reference inset elements and contents. This feature is provided as a simple convenience for quickly identifying insets within a document. When you launch the coloring dialog box, the color drop-down list is populated with the colors from the active document, or in the case of a book, the first open document found for the book. During processing, if InsetPlus attempts to apply a color that doesn’t exist in a particular document, nothing will happen. Also, if you are coloring a whole book, any files that are currently closed are skipped.

Note: Colors are applied as general format overrides, as if you had manually opened a format editor and selected a new color. To remove them, you should refresh your element definitions by selecting **InsetPlus > Inset Management > Unolor Insets (Refresh EDD)**.

Clearing insets

With a book or document active, you can select **InsetPlus > Inset Management > Clear Insets** to clear the content of all inset reference elements. It has the same effect document-wide as the right-click **Clear Inset** command. Like all plugin commands, this command cannot be undone. Remember that you can always close your files without saving changes to revert a command such as this. If you are clearing insets in a whole book, any files that are closed are skipped.

Source module collection

The source module collection utility (**InsetPlus > Inset Management > Collect Inset Source Files**) allows you to gather all referenced source files for a document or book and optionally redirect all reference insets to those files. It is intended as a general purpose consolidation or archival tool. Note the following:

- **Depending on the options you choose, this utility may alter your files!** Be sure that you are positive that the results are what you expected before saving any files!
- If your source files are in XML format, you may need to account for structure application behavior when the files are saved to the new location. For example, if your DTD or schema typically resides with your source files, you'll need to manually place the file(s) in the destination folder so that the FrameMaker save operation (as applicable) functions correctly.

Then the utility runs, it first parses your file(s) and gathers a list of all source files that are referenced. It then copies those files to the specified location and, if enabled, redirects all references to those copies. Note that the original source files in their original locations are not altered.

The dialog box contains the following options:

Option	Description
Folder in which to collect source files	Destination folder to which source files should be copied.
Redirect inset references to the new source locations	Enables/disables the option to redirect inset references. **IMPORTANT!** If you enable this option, InsetPlus will change the values of all applicable conref attributes as necessary to redirect reference elements. **Your files will be altered!**

Option	Description
Use FrameMaker to copy files	Causes the following behavior: • **If checked**, the utility copies files by opening them in FrameMaker and then doing a “save as” operation to the new location. This option allows you to preserve standard save-as conveniences, such as the automatic adjustment of paths to referenced graphics. • **If unchecked**, uses a simple system call to copy files, similar to a copy/paste action within Windows explorer. This method is faster; however, no automatic FrameMaker path management tools will be available.
Do warning and prompt messages	Enables/disables all message-box related prompting, such as file overwrite warnings. If disabled, InsetPlus assumes that you want to perform all specified and necessary operations to complete the collection, including overwriting files. It is recommended that you keep this option enabled, because the process may produce several important prompts and warnings.
Generate collection report	Enables/disables the generation of a report about the activities that occurred during collection.

Converting insets to conref mode

In previous releases, InsetPlus included a utility to convert classic mode insets to conref mode. Given the length of time since the introduction of conref mode and the deprecation of classic mode, this feature has been discontinued. If you still have need of this feature, please contact West Street.

Chapter 4 — External Calls to InsetPlus

Like many FrameMaker plugins, you can make external calls to InsetPlus from your own API clients or supported scripts. Specifically, you can call InsetPlus to:

- Launch the inset editor
- Update a single inset, a tree of insets, an entire document, or an entire book.

How to send an external call to the plugin

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 FrameScript or ExtendScript (FM10 or later)** FrameScript®, a scripting tool by Finite Matters, Ltd® and ExtendScript have a comparable function for calling FDK clients, `CallClient`(). When called from a script, the plugin behaves identically to a regular API call.
- **With FrameAC** FrameAC by Mekon® (www.mekon.com) is a plugin 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.

Tip: 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:

- Calls and returns sometimes involve document and element IDs, instead of names. Therefore, to use external calls effectively, you must be familiar with element and document IDs and how to convert them into the desired results.
- The default delimiter string between arguments in a call to the plugin is three dashes (---). In this document, the syntax of external calls use the default, which you should adjust accordingly if you decide to change the delimiter. For more information on changing the delimiter, see `ChangeCallDelimiter`.

- Due to the nature of `F_ApiCallClient()`, the plugin can only return a single integer after a call. No strings or other values can be returned. Therefore, all returns are in integer format and may represent items such as sequence numbers, element IDs, and error codes.
- Several calls to InsetPlus return zero (0) to indicate success, consistent with the behavior of other FDK functions. However, F_ApiCallClient() also returns zero if it fails to communicate at all with the specified API client. If you aren't sure whether your calls are reaching the plugin, you can call the special `Hello` command to verify that communications are getting through.
- Call strings are generally not case-sensitive.

Specifying document and book arguments

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

- **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", including just `f`.

Boolean arguments are not case-sensitive.

Call reference

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

ChangeCallDelimiter

Changes the delimiter for external call arguments. The default upon startup is three dashed ("`---`").

Syntax

```
F_ApiCallClient("InsetPlus", "ChangeCallDelimiterNewDelimiter");
```

Note: The new delimiter directly follows the `ChangeCallDelimiter` command. Do not separate them with the old delimiter. Anything following the command will be considered the new delimiter.

Returns

`F_ApiCallClient()` returns one of the following values:

Value	Meaning
0	A communication error occurred. Consider calling `Hello` to verify that InsetPlus is active.
1	The delimiter was successfully changed.
101	Unrecognized command. Make sure you spelled "`ChangeCallDelimiter`" correctly.
103	Incorrect number of arguments in the call string. Make sure you provided a new delimiter after `ChangeCallDelimiter`.

`ChangeCallDelimiter` syntax example

```
F_ApiCallClient("InsetPlus", "ChangeCallDelimiter++++");
```

ColorInsets

Applies coloring to a single reference inset, all insets in a specified branch, all insets in a document, or all insets in a book. The command has similar functionality to the comparable menu item, except with more precision as to where the coloring occurs.

Syntax

```
F_ApiCallClient("InsetPlus", "ColorInsets---File---ElemId---Color");
```

where:

`File`	Book or document in which to apply coloring. For more information, see *"Specifying document and book arguments"* on page 30.
`ElemId`	For document actions only, the object handle of the element at which to start coloring, in integer form (integer form of the FDK F_ObjHandleT type). Any inset represented by this element or descendant to this element will be colored. Note that: • You can specify zero (0) to default to the highest-level element. • This argument is ignored when `File` is a book, in which case all coloring begins at the highest-level element.
`Color`	Case-sensitive name of the color to apply, as a string. For document actions, if the color does not exist in the template, the command fails with an error. For book actions, no warning is given for a non-existent color.

Returns

`F_ApiCallClient()` returns one of the following values:

Value	Meaning
0	A communication error occurred. Consider calling `Hello` to verify that InsetPlus is active.
1	Coloring appears to have completed successfully.
101	Unrecognized command. Make sure you spelled “`ColorInsets`” correctly.
103	Incorrect number of arguments in the call string.
104	Invalid document argument.
105	Invalid element ID.
127	For document actions only, the specified color was not found in the template. Check the spelling and case-sensitivity.

ColorInsets syntax examples

```
F_ApiCallClient("InsetPlus", "ColorInsets---67108880---704950292---Red");
F_ApiCallClient("InsetPlus",
  "ColorInsets---C:/MyDocs/Myfile.fm---0---Green");
F_ApiCallClient("InsetPlus",
  "ColorInsets---C:/MyDocs/MyBook.book---0---Purple");
```

GetParm

Gets a parameter related to the plugin.

Syntax

```
F_ApiCallClient("InsetPlus", "GetParm---Parameter");
```

where *`Parameter`* may be as follows:

Parameter	Description
`InsetPlusVersionMajor`	Returns the major version number of the plugin, such as the “3” in v3.10.
`InsetPlusVersionMinor`	Returns the minor version number of the plugin, such as the “10” in v3.10.

Returns

`F_ApiCallClient()` returns one of the following values:

Value	Meaning
0	A communication error occurred. Consider calling `Hello` to verify that InsetPlus is active.
101	Unrecognized command. Make sure you spelled “`GetParm`” correctly.
103	Incorrect number of arguments in the call string.

Value	Meaning
109	Bad parameter argument.
Any other value	The requested parameter value

`GetParm` syntax examples

```
F_ApiCallClient("InsetPlus", "GetParm---InsetPlusVersionMajor");
```

`Hello`

Determines if InsetPlus is initialized and receiving external calls.

Syntax

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

Usage description

`Hello` is a simple call to ensure that InsetPlus is available and responding to external calls.

Returns

`F_ApiCallClient()` returns one of the following values after a `Hello` call:

Value	Meaning
0	Communication with InsetPlus failed. Check to make sure that InsetPlus is initialized and running. Also, make sure that InsetPlus is properly registered in the `maker.ini` file under the name “InsetPlus.”
1	InsetPlus is installed and ready.
101	Unrecognized command. Make sure you spelled “`Hello`” correctly.

`Hello` syntax example

```
. . .
IntT returnVal;

. . .
returnVal = F_ApiCallClient("InsetPlus", "Hello");

if(returnVal != 1)
  F_ApiAlert("Error. InsetPlus is not ready.", FF_ALERT_CONTINUE_WARN);
```

`LaunchInsetEditor`

Launches the inset editor for a specified reference element. Once the editor is launched, control is returned to the user and InsetPlus has no further interaction. The call will fail if you specify an element that is not valid as a reference inset element.

Syntax

```
F_ApiCallClient("InsetPlus", "LaunchInsetEditor---Document---ElemId");
```

where:

`Document`	See *“Specifying document and book arguments”* on page 30.
`ElemId`	The object handle of the inset element to edit, in integer form (integer form of the FDK F_ObjHandleT type).

Returns

`F_ApiCallClient()` returns one of the following values:

Value	Meaning
0	A communication error occurred. Consider calling `Hello` to verify that InsetPlus is active.
1	The editor was successfully launched. This value is returned whether or not the user clicks OK or Cancel in the editor.
101	Unrecognized command. Make sure you spelled “`LaunchInsetEditor`” correctly.
103	Incorrect number of arguments in the call string.
104	Invalid document argument.
105	Invalid element ID. This return may occur in some cases if the document ID is invalid.
120	General failure, cause unknown.
121	Element ID is valid, but element is not valid as a reference inset element.

LaunchInsetEditor syntax examples

```
F_ApiCallClient("InsetPlus", "LaunchInsetEditor---67108880---704950292");
F_ApiCallClient("InsetPlus",
  "LaunchInsetEditor---C:/MyDocs/Myfile.fm---704950292");
```

LaunchInsetEditor code sample

The following example launches the inset editor for the element that contains the current insertion point.

```
. . .
F_ObjHandleT docId, elemId;
UCharT arg[64];
IntT returnVal;
F_ElementRangeT er;

/* Find the active document and current element */
docId = F_ApiGetId(0, FV_SessionId, FP_ActiveDoc);
if(docId)
{
  er = F_ApiGetElementRange(FV_SessionId, docId, FP_ElementSelection);
  elemId = er.beg.parentId;
}
```

```
if(!docId || !elemId)
{
  F_ApiAlert("Invalid document and/or element", FF_ALERT_CONTINUE_WARN);
  return;
}

/* Form the argument for the LaunchInsetEditor call */
F_Sprintf(arg, "LaunchInsetEditor---%d---%d", docId, elemId);

/* Call InsetPlus to launch the editor */
returnVal = F_ApiCallClient("InsetPlus", (StringT)arg);

/* Report */
if(returnVal == 1)
  F_ApiAlert("It worked.", FF_ALERT_CONTINUE_WARN);
else
  F_ApiAlert("Something went wrong.", FF_ALERT_CONTINUE_WARN);
. . .
```

SetParm

Sets a functional parameter, as described under *“Syntax”* on page 35. If you require access to a parameter that is not supported, please contact West Street.

Syntax

```
F_ApiCallClient("InsetPlus", "SetParm---Parameter---Value");
```

where *Parameter* and *value* may be as follows:

Parameter	Valid values
SourceTrackingIsOn	**True** or **False** (see *“Specifying Boolean arguments”* on page 30). When source tracking is disabled, the source tracking data is not updated during inset updates, resulting in faster processing.

Returns

F_ApiCallClient() returns one of the following values:

Value	Meaning
0	A communication error occurred. Consider calling Hello to verify that InsetPlus is active.
1	The parameter was successfully changed.
101	Unrecognized command. Make sure you spelled “SetParm” correctly.
103	Incorrect number of arguments in the call string.
109	Bad parameter argument.

`SetParm` syntax examples

```
F_ApiCallClient("InsetPlus", "SetParm---SourceTrackingIsOn---False");
```

`UpdateInsets`

Updates a single inset, all insets within a branch or tree, or all insets within an entire document. This command supports both reference and source updates. Note the following:

- This command does not support whole-book updates. For book inset updates, you must run this command on each applicable chapter file.
- When a table or a table row element is updated, its ID is changed. Therefore, if you are updating a single inset of this type, the ID you send will not be valid following the operation. Therefore, if you need the ID of the updated inset following the operation, you must make provisions in your code to rediscover it afterwards.

Syntax

```
F_ApiCallClient("InsetPlus",
   "UpdateInsets---Document---ElemId---[UpdateType---OpenDocs---
   CloseDocs---DoWarnings---DoReporting]");
```

Note the following:

- The *`UpdateType`* through *`DoReporting`* arguments are optional. If unspecified, defaults are used. If you specify one, all arguments previous to it must be specified as well.
- A whole-document update is triggered by the *`ElemId`* argument. See the following table.
- If you have source module tracking turned on, the update process will be slower. Optionally, you can turn it off with `SetParm`.
- No documents are saved after any update action. To preserve changes, you must save them with your code.

Arguments:

`Document`	See *"Specifying document and book arguments"* on page 30.
`ElemId`	The object handle of the inset element to update, in integer form (integer form of the FDK F_ObjHandleT type). Note the following: • To update a whole document (main flow only), send a negative number • To update branch, tree, etc., send the parent element of the branch • To update a single inset element, send that element ID In summary, if you send an element ID, that element and any subordinate inset elements will be updated, as applicable.

`UpdateType`	Update type: • **1** - Reference inset update • **2** - Source inset update In either case, for `ElemId`, send the ID of a reference inset element. For source updates, the update occurs from the content of the reference element, much like using the menu commands in the FrameMaker GUI. If this argument is not specified, the default is 1 (reference update).
`OpenDocs`	Indicates whether to open source documents if they are currently closed (Boolean argument). If set to false, any updates to sources in closed documents will fail. See *“Specifying Boolean arguments”* on page 30.
`CloseDocs`	Indicates whether to close any documents that InsetPlus opened to complete an update, once all updates are finished (Boolean argument). See *“Specifying Boolean arguments”* on page 30.
`DoWarnings`	Indicates whether to display message boxes for prompts, warnings, and other messages (Boolean argument). If set to true, InsetPlus may display message boxes that will halt the update process until manually dismissed. See *“Specifying Boolean arguments”* on page 30.
`DoReporting`	Indicates whether to produce the hyperlinked report for warnings and errors (Boolean argument). See *“Specifying Boolean arguments”* on page 30.

Returns

`F_ApiCallClient()` returns one of the following values:

Note: If you are updating a branch or an entire tree which contains multiple insets, the process continues until finished even if one or more update actions produce an error. Therefore, if an error is returned, it does not mean that all updates failed, only that at least one failed. If the operation produces multiple failures, the error returned applies to the most recent failure event which may or may not be applicable to previous failures.

Value	Meaning
`0`	A communication error occurred. Consider calling `Hello` to verify that InsetPlus is active.
`1`	A single-element update action appears to have completed successfully. This value is only returned when the sent element is determined to be a valid reference element.

Value	Meaning
`2`	An update action appears to have completed successfully, but the element sent (if any) did not appear to be a reference inset element. This value is returned when you send a higher-level element or initiate a whole document update. This return does not indicate that all or any subordinate inset elements updated correctly. To know the success of individual updates that occurred, you must produce the error report. If you attempt to update a single reference element and receive this return, InsetPlus did not recognize the element as an inset reference. Make sure you are using the correct element ID.
`101`	Unrecognized command. Make sure you spelled “`UpdateInsets`” correctly.
`103`	Incorrect number of arguments in the call string.
`104`	Invalid document argument.
`105`	Invalid element ID.
`117`	The action was cancelled by the user (someone pressed Esc).
`120`	General failure, cause unknown.
`122`	No source file could be found to retrieve the source element text. This error may occur if the source file is closed and you have chosen not to open closed documents.
`123`	The source element could not be found. This value may be returned in some cases if the source file could not be accessed, in place of 122.
`124`	An update of a table row inset failed due to a mismatch in the number of cells (columns) between the reference and source.

`UpdateInsets` syntax examples

Update a single element, or an element and all subordinate elements:

```
F_ApiCallClient("InsetPlus",
   "UpdateInsets---67108880---704950292");
F_ApiCallClient("InsetPlus",
   "UpdateInsets---67108880---704950292---1");
F_ApiCallClient("InsetPlus",
   "UpdateInsets---67108880---704950292"---1---1---1---0---0);
```

Update a whole document (note that the third argument is a -1):

```
F_ApiCallClient("InsetPlus",
  "UpdateInsets---C:/MyDocs/Myfile.fm----1");
F_ApiCallClient("InsetPlus",
  "UpdateInsets---67108800----1");
F_ApiCallClient("InsetPlus",
  "UpdateInsets---67108800----1---1---1---1---0---0");
```