



MadCap Software

# Context-sensitive Help Guide

Flare 10



Copyright 2014 MadCap Software. All rights reserved.

Information in this document is subject to change without notice. The software described in this document is furnished under a license agreement or nondisclosure agreement. The software may be used or copied only in accordance with the terms of those agreements. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or any means electronic or mechanical, including photocopying and recording for any purpose other than the purchaser's personal use without the written permission of MadCap Software.

MadCap Software  
7777 Fay Avenue  
La Jolla, California 92037  
858-320-0387  
[www.madcapsoftware.com](http://www.madcapsoftware.com)

# CONTENTS

<b>CHAPTER 1 About Context-Sensitive Help .....</b>	<b>5</b>
Who is Involved? .....	6
Planning the CSH .....	7
What Needs to be Done and Who Does What? .....	8
<b>CHAPTER 2 Header Files .....</b>	<b>11</b>
Who Develops the Header File and How? .....	12
What is Contained in a Header File? .....	13
Adding Header Files .....	15
Importing Header Files .....	17
Opening Header Files .....	18
Providing a Developer with a Header File .....	19
<b>CHAPTER 3 Alias Files .....</b>	<b>21</b>
Adding Alias Files .....	22
Setting Identifier Options .....	23
Creating and Assigning Identifiers .....	32
Moving Identifiers to Different Headers .....	55
Importing Alias Files .....	56
Opening Alias Files .....	57
Associating an Alias File with a Target .....	58
<b>CHAPTER 4 Additional Tasks for CSH .....</b>	<b>59</b>
Testing Context-Sensitive Help .....	60
Creating Topic Alias Markers in FrameMaker for CSH .....	63
CSH Calls for DotNet Help .....	65
CSH Calls for HTML Help .....	82
CSH Calls for HTML5 Output .....	85
CSH Calls for WebHelp .....	91



CSH Calls for WebHelp Mobile .....	98
<b>APPENDIX PDF Guides .....</b>	<b>101</b>

## About Context-Sensitive Help

This feature is supported in DotNet Help, Microsoft HTML Help, HTML5, WebHelp, WebHelp Mobile, and WebHelp Plus output.

Context-sensitive Help (CSH) is a way to tie your existing topics with specific dialogs or windows in a software application, or with simple web links created somewhere (e.g., on a website). When users open a particular dialog or window in a software application, or click a web link, they can quickly open a topic pertaining to it.

### EXAMPLE

Let's say you are creating an online Help system for your company's software application. This application contains a dialog called "Properties" that users open to specify settings for a particular element. In your Help project, you have written a topic to explain this Properties dialog. By creating CSH, users will be able to open that specific topic by clicking a Help button on the Properties dialog (or by pressing F1 when it is open).

This chapter discusses the following:

Who is Involved? .....	6
Planning the CSH .....	7
What Needs to be Done and Who Does What? .....	8



## Who is Involved?

Creating CSH is mostly a joint effort between *you* and the *software developer*. There are tasks that you must perform and tasks that the developer must perform in order for CSH to be implemented successfully. For this reason, it is essential that you communicate clearly with the developer when planning, creating, and implementing CSH. Other individuals (managers, other Help authors, etc.) may also be involved as well, particularly in the early planning stages.

However, there might be times when you function as both the author and the developer. For example, this might be the case if you are generating WebHelp output and simply want to create links on a website that open specific parts of your output. In that situation, you might first generate the online output with the CSH information. Then you might serve as the developer, modifying pages on a website to include CSH links pointing to your documentation.

## Planning the CSH

For CSH to be successful, some initial planning is necessary. This includes making decisions such as:

- » Which type of output is being created?
- » Which dialogs and windows will be included in the CSH (if connecting to an application)?
- » Where will the links be located (if connecting to simple web links)?
- » What will the Help buttons look like on the dialogs or windows?
- » Will the CSH Help topics open in a different window than the other topics in the Help system (different size, position, and appearance)?

Depending on how your company operates, questions such as these may be decided independently by you or the developer. Or they may be decided jointly by you, the software developer, and others (managers, other authors).

Another major decision that needs to be made at the beginning of the process is whether you or the developer will be responsible for providing the header file that is necessary for CSH. This decision is typically made jointly by you and the software developer. See "Header Files" on page 11.

## What Needs to be Done and Who Does What?

Following are two sets of general guidelines containing the necessary steps for creating CSH. Each set of guidelines is slightly different, based on whether you or the software developer provides the header file.

### CSH Steps—If You Provide the Header File

1. **You** Add a header file to the project.  
See "Header Files" on page 11 and "Adding Header Files" on page 15.
2. **You** Add an alias file to the project. The header file is the vital piece in this process. The alias file simply lets you edit the header file in a user-friendly interface.  
See "Alias Files" on page 21 and "Adding Alias Files" on page 22.
3. **You** Create and assign identifiers.  
See "Creating and Assigning Identifiers" on page 32.
4. **You** Associate the alias file with the target.  
This is necessary only if you have created more than one alias file. See "Associating an Alias File with a Target" on page 58.
5. **You** Provide the developer with a copy of the header file.  
For example, you can do this by exporting the file to a shared drive. See "Providing a Developer with a Header File" on page 19.
6. **You** Generate output.
7. **You** Provide the developer with a copy of the Help output files.
8. **Software Developer** "Hooks" application interface or web links to online topics.  
The developer connects the dialogs, windows, or web links to the appropriate online topics using the information in the header file. See the following for more details about the information that you might need to provide for the developer (depending on the output type you generated).
  - » "CSH Calls for DotNet Help" on page 65
  - » "CSH Calls for HTML Help" on page 82
  - » "CSH Calls for HTML5 Output" on page 85
  - » "CSH Calls for WebHelp" on page 91
  - » "CSH Calls for WebHelp Mobile" on page 98
9. **Software Developer** Creates a new build of the software application or publishes updated web links.



10. **You** Install the new software build or view updated website. Test the CSH.  
See "Testing Context-Sensitive Help" on page 60.

## CSH Steps—If Developer Provides the Header File

1. **Software Developer** Creates the header file.
2. **Software Developer** Provides you with a copy of the header file.
3. **You** Add the developer's header file to the project.  
See "Header Files" on page 11 and "Importing Header Files" on page 17.
4. **You** Add an alias file to the project. The header file is the vital piece in this process. The alias file simply lets you edit the header file in a user-friendly interface.  
See "Alias Files" on page 21 and "Adding Alias Files" on page 22.
5. **You** Assign identifiers for the header file.  
See "Creating and Assigning Identifiers" on page 32.
6. **You** Associate the alias file with the target.  
This is necessary only if you have created more than one alias file. See "Associating an Alias File with a Target" on page 58.
7. **You** Generate output.
8. **You** Provide the developer with a copy of the Help output files.
9. **Software Developer** "Hooks" application interface or web links to online topics.  
The developer connects the dialogs, windows, or web links to the appropriate online topics using the information in the header file. See the following for more details about the information that you might need to provide for the developer (depending on the output type you generated).
  - » "CSH Calls for DotNet Help" on page 65
  - » "CSH Calls for HTML Help" on page 82
  - » "CSH Calls for HTML5 Output" on page 85
  - » "CSH Calls for WebHelp" on page 91
  - » "CSH Calls for WebHelp Mobile" on page 98
10. **Software Developer** Creates a new build of the software application or publishes updated web links.
11. **You** Install the new software build or view updated web links. Test the CSH.  
See "Testing Context-Sensitive Help" on page 60.



**Note:** If you are creating CSH calls for WebHelp outputs (i.e., WebHelp, HTML5, WebHelp Plus, WebHelp Mobile) using the URL method, you do not need to create a header file, alias file, and IDs. Those are optional steps. See "CSH Calls for WebHelp and WebHelp Plus—Developers" on page 92, "CSH Calls for HTML5 Output—Developers" on page 86, and "CSH Calls for WebHelp Mobile—Developers" on page 99.

## Header Files

A header file is a simple text file that contains basic information about connecting the dialogs or windows in a software application to the corresponding topics in the Help system. Both you and the software developer need access to this file.

A header file has an .h extension and is stored in the Project Organizer under the Advanced folder. You can export the Flare header file into other file formats (e.g., .bas, .properties, .inc, etc.) if necessary.

This chapter discusses the following:

Who Develops the Header File and How? .....	12
What is Contained in a Header File? .....	13
Adding Header Files .....	15
Importing Header Files .....	17
Opening Header Files .....	18
Providing a Developer with a Header File .....	19



## Who Develops the Header File and How?

Either you or the software developer is responsible for creating the header file. That is something you must decide with the developer.

If it is decided that you are responsible for creating the header file, you can do so by adding a header file to the project, adding an alias file to the project, and then creating and assigning identifiers.

## What is Contained in a Header File?

A completed header file contains one or more lines of text ("identifiers"). Each identifier refers to a specific dialog or window that is linked to a corresponding topic in the Help system. Here is part of a header file, showing three identifiers:

```
#define Bookmarks_dialog 1
#define Browse_Sequences_dialog 2
#define Concept_Entries_dialog 3
```

The following images provide a breakdown of what each part of an identifier means.

```
#define Concept_Entries_dialog 3
```

This is preliminary text that needs to be included at the beginning of each identifier (before the header ID) in the header file. Flare will add this text if you create identifiers from within Flare. If you create the identifier by using another text editor (such as Word or Notepad), you need to type this text manually.

```
#define Concept_Entries_dialog 3
```

This portion of the identifier is the topic ID. It simply tells you and the developer which dialog or window this line refers to. The topic ID name is determined by you or the developer – whatever helps you to identify the dialog or window. The underscores in this example are necessary because spaces are not allowed in the topic ID.

```
#define Concept_Entries_dialog 3
```

This is the numerical value of the identifier. Each identifier must have a unique number assigned to it. The developer can "hook" the actual dialog or window by assigning it to this number. This way, the software application and your Help system can communicate with each other.



**Note:** A header file is sometimes referred to as a "map file."



**Note:** If you are importing FrameMaker documents and you create topic alias markers in the source files, this file will be created automatically when you perform the import.



**Note:** If you have multiple header files in your project, their contents are merged when you generate output.

## Adding Header Files

Aside from planning the context-sensitive Help (CSH), the first step in this process is to add the header file to the Flare project. Exactly how you do this depends on whether you or the software developer are responsible for creating the header file.


### HOW TO ADD A HEADER FILE (IF YOU ARE RESPONSIBLE FOR CREATING IT)


1. Do one of the following, depending on the part of the user interface you are using:
  - » **Ribbon** Select the **Project** ribbon. In the **Content** section select **New>Advanced>Header File**.
  - » **Menu** Select **Project>Advanced>Add Header File**.
  - » **Right-Click** In the Project Organizer, right-click on the **Advanced** folder and from the context menu select **Add Header File**.

The Add File dialog opens.

2. In the **File Type** field at the top, make sure **Header File** is selected.
3. In the **Source** area select one of the following:
  - » **New from template** This lets you choose either the factory template file or one of your own customized template files as a starting point. The new file will take on all of the settings contained in the template. If you want to use the factory template provided by Flare, expand the **Factory Templates** folder and click on a template file. If you want to use your own customized template file, expand the appropriate folder and click on a file. For more information about templates, see the online Help.



**Note:** In some dialogs and wizards you can click the **Manage Templates** button  if you want to open the Template Manager. This lets you manage any of your template files (e.g., add new templates, enter descriptions for templates). For more information see the online Help.


- » **New from existing** This lets you choose an existing file of the same type as a starting point for your new file. As with template files, your new file will take on all of the settings contained in the file you select. To use this option, click the browse button, use the Open File dialog to find a file, and double-click it.
4. (Optional) If you want to place the file into a subfolder that you previously created in the Project Organizer, in the **Folder** field click  and select the subfolder. Otherwise, keep the default location.
  5. In the **File Name** field, type a new name for the header file.
  6. Click **Add**. The header file is added to the Advanced folder in the Project Organizer. The Text Editor opens to the right, with the page for the new header file (including an initial identifier) shown.
  7. You can close the Text Editor by clicking the **x** at the top-right corner of the tab.

You will not be entering content to this editor directly. It will be added automatically when you work in the alias file. See "Creating and Assigning Identifiers" on page 32.

#### HOW TO ADD A HEADER FILE (IF THE DEVELOPER IS RESPONSIBLE FOR CREATING IT)

1. Do one of the following, depending on the part of the user interface you are using:
  - » **Ribbon** Select the **Project** ribbon. In the **Content** section select **New>Advanced>Header File**.
  - » **Menu** Select **Project>Advanced>Add Header File**.
  - » **Right-Click** In the Project Organizer, right-click on the **Advanced** folder and from the context menu select **Add Header File**.

The Add File dialog opens.

2. In the **File Type** field at the top, make sure **Header File** is selected.
3. Select **New from existing** and click .
4. Find and select the header file that you want to import.
5. Click **Open**. The Source File field now contains the path to the file that you are importing. Also, the name of the file is displayed in the File Name field.
6. If you want to give the header file a different name than that for the imported file, click in the **File name** field and replace the text.
7. Click **Add**. The header file is added.



**Note:** Whether you add a header file from inside Flare or "import" one from outside the program, you need to make sure that it uses the following format on each line: `#define IdentifierName IdentifierValue`. Make sure that there are no spaces between the words in the "IdentifierName." For example, use `#define MyDialog 17` or `#define My_Dialog 17,` rather than `#define My Dialog 17.`




## Importing Header Files

Not only can you add new header files to Flare, but you can also import existing header files (e.g., files with .h or .hh extensions).

### HOW TO IMPORT A HEADER FILE

1. Do one of the following, depending on the part of the user interface you are using:
  - » **Ribbon** Select the **Project** ribbon. In the **Content** section select **New>Advanced>Header File**.
  - » **Menu** Select **Project>Advanced>Add Header File**.
  - » **Right-Click** In the Project Organizer, right-click on the **Advanced** folder and from the context menu select **Add Header File**.

The Add File dialog opens.

2. In the **File Type** field at the top, make sure **Header File** is selected.
3. Select **New from existing** and click .
4. Find and select the header file that you want to import.
5. Click **Open**. The Source File field now contains the path to the file that you are importing. Also, the name of the file is displayed in the File Name field.
6. If you want to give the header file a different name than that for the imported file, click in the **File name** field and replace the text.
7. Click **Add**. The header file is added.

## Opening Header Files

When you want to work on an existing header file directly (as opposed to populating it by using the alias file), use the following steps to open it.

### *HOW TO OPEN AN EXISTING HEADER FILE*

1. Make sure the Project Organizer is open.
2. Double-click the **Advanced** folder. The header file(s) in your project are displayed (next to any other files that you have created, such as browse sequences, search filter sets, or alias files).
3. Double-click the header file that you want to open. The Text Editor opens to the right, with the header file page shown.

## Providing a Developer with a Header File

If it is decided that you are responsible for creating the header file (i.e., the file containing an .h extension in your project), you must provide a copy of it to the software developer. The developer will then use the header file to "hook" the numerical values in the header file to the appropriate dialogs or windows in the software.

When you are finished creating the header file, you can export it using the steps below so that the developer has access to it. If you make further changes to the header file, you need to ensure that the developer receives a new copy of it.

### HOW TO EXPORT HEADER FILES

1. Do one of the following, depending on the part of the user interface you are using:
  - » **Ribbon** Select the **Tools** ribbon. In the **Context Sensitive Help** section select **Export Header File(s)**.
  - » **Menu** Select **Build>Export Header File(s)**.
  - » **Right-Click** In the Project Organizer, right-click the **Advanced** folder and select **Export Header File(s)**.

The Export Header Files dialog opens.

2. On the left side of the dialog, select the format(s) that you want to use for exporting the header file(s). Work with your developer to determine the appropriate type of format:
  - » **C/C++ (.h)** If this option is selected, a copy of the header file will be created with an .h file extension.
  - » **Visual Basic (.bas)** If this option is selected, a copy of the header file will be created with a .bas file extension.
  - » **Java (.properties)** If this option is selected, a copy of the header file will be created with a .properties file extension.
  - » **Delphi Pascal (.inc)** If this option is selected, a copy of the header file will be created with an .inc file extension.
3. On the right side of the dialog, select the header file(s) to be exported.
4. Click the **Browse** button and select the location where the exported file(s) will be sent.
5. Click **Export**.
6. (Optional) If the file(s) are not in a shared location where the developer can retrieve them, you need to copy the exported files(s) from that location and send them to the developer.



## Alias Files

An alias file is used to populate a header file with the information necessary for producing context-sensitive Help (CSH). In Flare, you can open an alias file and use the Alias Editor to create and assign identifiers for the header file. You can use a single alias file in a project for multiple header files, or you can create a separate alias file to go with each header file.

An alias file has an .flali extension and is stored in the Project Organizer under the Advanced folder.

This chapter discusses the following:

Adding Alias Files .....	22
Setting Identifier Options .....	23
Creating and Assigning Identifiers .....	32
Moving Identifiers to Different Headers .....	55
Importing Alias Files .....	56
Opening Alias Files .....	57
Associating an Alias File with a Target .....	58




## Adding Alias Files




After you add a header file to your project, the next step in creating context-sensitive Help (CSH) is to add an alias file. The alias file will help you to add content to the header file.

### HOW TO ADD AN ALIAS FILE

1. Do one of the following, depending on the part of the user interface you are using:
  - » **Ribbon** Select the **Project** ribbon. In the **Content** section select **New>Advanced>Alias File**.
  - » **Menu** Select **Project>Advanced>Add Alias File**.
  - » **Right-Click** In the Project Organizer, right-click on the **Advanced** folder and from the context menu select **Add Alias File**.

The Add File dialog opens.


2. In the **File Type** field at the top, make sure **Alias File** is selected.
  3. In the **Source** area select one of the following:
    - » **New from template** This lets you choose either the factory template file or one of your own customized template files as a starting point. The new file will take on all of the settings contained in the template. If you want to use the factory template provided by Flare, expand the **Factory Templates** folder and click on a template file. If you want to use your own customized template file, expand the appropriate folder and click on a file. For more information about templates, see the online Help.
- 

**Note:** In some dialogs and wizards you can click the **Manage Templates** button  if you want to open the Template Manager. This lets you manage any of your template files (e.g., add new templates, enter descriptions for templates). For more information see the online Help.
- » **New from existing** This lets you choose an existing file of the same type as a starting point for your new file. As with template files, your new file will take on all of the settings contained in the file you select. To use this option, click the browse button , use the Open File dialog to find a file, and double-click it.
  4. (Optional) If you want to place the file into a subfolder that you previously created in the Project Organizer, in the **Folder** field click  and select the subfolder. Otherwise, keep the default location.
  5. In the **File Name** field, type a new name for the alias file.
  6. Click **Add**. The alias file is added to the Advanced folder in the Project Organizer. The Alias Editor opens to the right, with the page for the new alias file shown. The file includes an initial identifier for the header file that you created previously.

## Setting Identifier Options

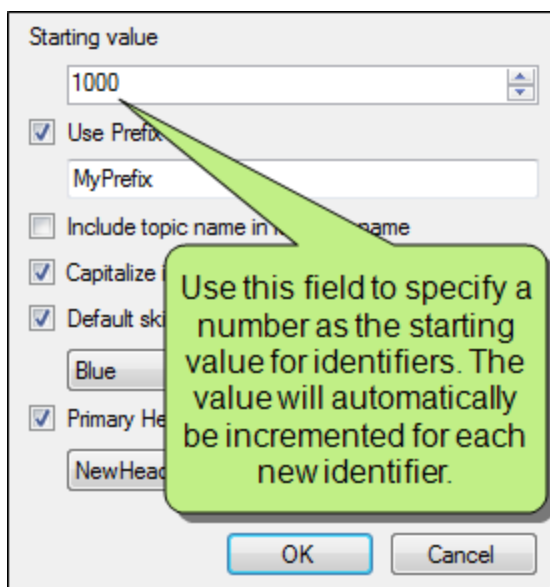
When creating context-sensitive Help (CSH), you can set options for new identifiers in advance. Doing this will supply some of the information (e.g., starting value, prefix, include topic in ID name, assign skin) for you automatically as you create new identifiers. See "Creating and Assigning Identifiers" on page 32.

### HOW TO SET IDENTIFIER OPTIONS

1. Open an alias file.
2. In the local toolbar of the Alias Editor click . The Identifier Options dialog opens.
3. Complete any of the options as necessary:

#### STARTING VALUE

Enter the starting value for identifiers that you create. Additional identifiers that are created will be incremented automatically based on that starting value.



MyAliasFile

Alias Editor (all identifiers) Unassigned Identifiers: 3

Topics

- Content
  - Contading
  - Getting\_Started
  - Print Topics
  - Resources
  - Using
  - Welcome.htm
  - Whats new.htm

Identifiers

Select skin

Identifier	Topic	Path	Skin	Value
MyPrefixNEW			Blue	1000
MyPrefixNEW1			Blue	1001
MyPrefixNEW2			Blue	1002
MyPrefixNEW3		/Content/	Blue	1003

In this example the starting value is 1000. Therefore, the first new identifier has that number, with the next three new identifiers using values 1001, 1002, and 1003.



### USE PREFIX

Select this check box and enter a prefix to be added at the beginning of each new identifier that you create (e.g., ID\_, Dialog, Module1).

The 'Starting value' dialog box contains the following settings:

- Starting value: 1000
- ☒ Use Prefix: MyPrefix
- ☐ Include topic name in identifier name
- ☐ Capitalize identifier names
- ☒ Default skin: Blue
- ☒ Primary Header: applicationhelp.h

Buttons: OK, Cancel

**Callout:** Use this check box and field to specify any prefix that you want. In this example we have entered "MyPrefix."

MyAliasFile Alias Editor (all identifiers) Unassigned Identifiers: 3

Topics: Content, Contending, Getting\_Started, Print Topics, Resources, Using, Welcome.htm, Whats new.htm

Select skin

Identifier	Topic	Path	Skin	Value
MyPrefixNEW			Blue	1000
MyPrefixNEW1			Blue	1001
MyPrefixNEW2			Blue	1002

**Callout:** In this example "MyPrefix" is being added automatically to the beginning of each new identifier.

### INCLUDE TOPIC NAME IN IDENTIFIER NAME

Select this check box if you want the names of the assigned topics to be included automatically in the names of the new identifiers.

Starting value: 1000

☒ Use Prefix: MY\_PREFIX

☒ Include topic name in identifier name

☒ Capitalize identifier names

☒ Default skin: Blue

☒ Primary Header: applicationhelp.h

OK Cancel

Use this check box to automatically include the topic name in the new identifier name.

In this example we first selected a topic called "Welcome" from the list of topics on the left.

Second, we clicked this button to create a new identifier and assign it to the selected topic at the same time.

The result is the the new identifier name includes the prefix that we specified in the Identifier Options dialog (MYPREFIX), followed by the name of the topic (WELCOME).

MyAtlasFile (all identifiers) Unassigned Identifiers: 3

Identifiers

Select skin

Identifier	Topic	Path	Skin	Value
MYPREFIXNEW			Blue	1001
MYPREFIXNEW1			Blue	1002
MYPREFIXNEW2			Blue	1003
MYPREFIXWELCOME	Welcome.htm	/Content/	Blue	1004

Content: Contating, Getting\_Started, Print Topics, resources, Using, Welcome.htm, Whats new.htm

### CAPITALIZE IDENTIFIER NAMES

Select this check box if you want the name that is automatically added for new identifiers to use all caps.

Starting value

1000

☒ Use Prefix

MYPREFIX

☒ Include topic name in identifier

☒ Capitalize identifier names

☒ Default skin

Blue

☐ Primary Header

OK Cancel

MyAliasFile

Alias Editor (all identifiers) Unassigned Identifiers: 3

Topics

- Content
  - Contating
  - Getting\_Started
  - Print Topics
  - Resources
  - Using
  - Welcome.htm
  - Whats new.htm

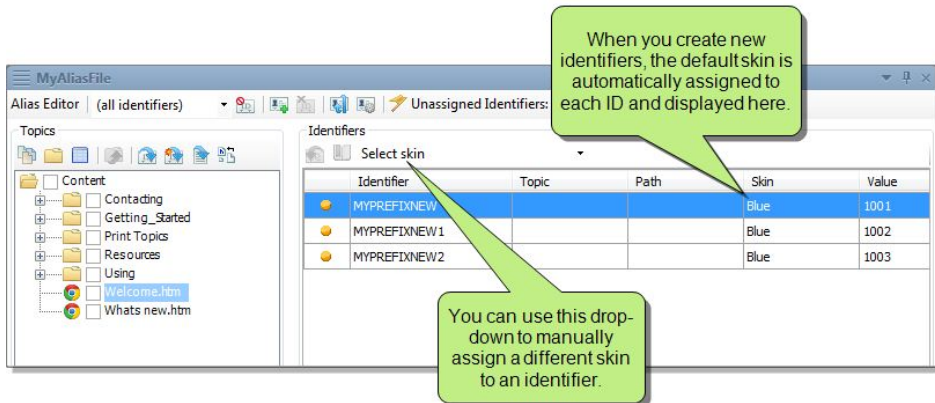
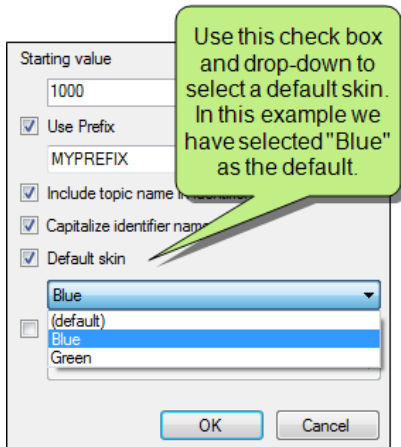
Identifiers

Select skin

Identifier	Topic	Path	Skin	Value
MYPREFIXWELCOME	Welcome.htm	/Content/	Blue	1000
MYPREFIXNEW			Blue	100
MYPREFIXNEW 1			Blue	100
MYPREFIXNEW 2			Blue	100

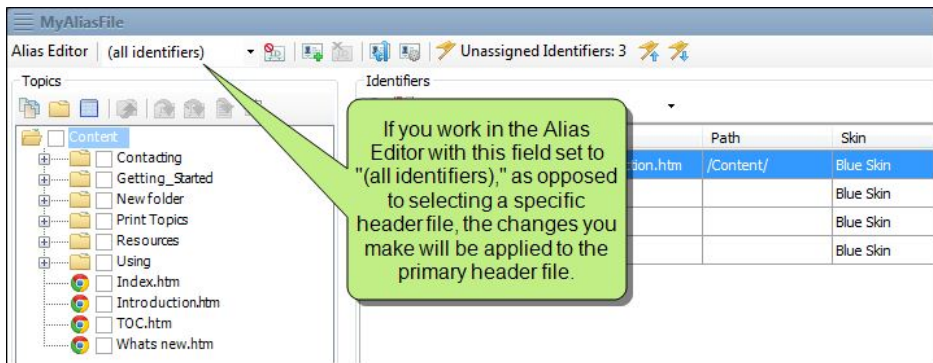
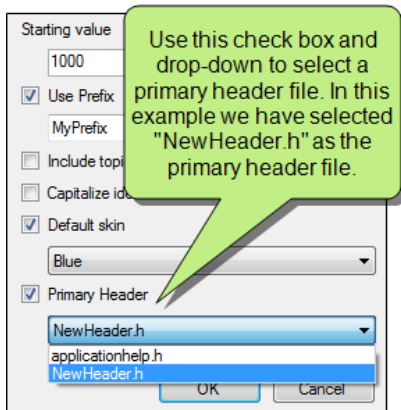
## DEFAULT SKIN

Select this check box and from the drop-down field choose the skin that you want to be assigned by default to new identifiers that are created. Of course, you can always manually select a different skin for any identifier, but when you first create a new identifier, it will initially be assigned to the default skin that you specified.



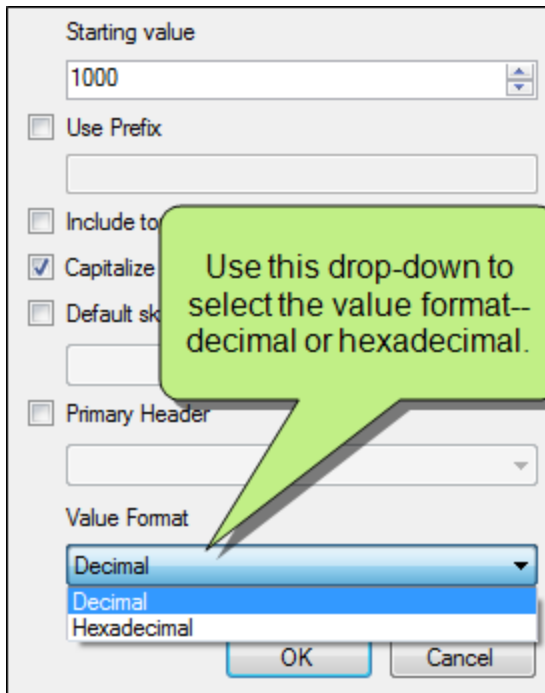
### PRIMARY HEADER

Select this check box and from the drop-down field choose a specific header file, if you have more than one in your project. When you are working in the Alias Editor, you can select a specific header file in the local toolbar. But what if you do not select a header file and "(all identifiers)" is shown in the drop-down field in the Alias Editor? In that case, the changes you make in the Alias Editor are applied to the primary header file that you have selected in the Identifier Options dialog.

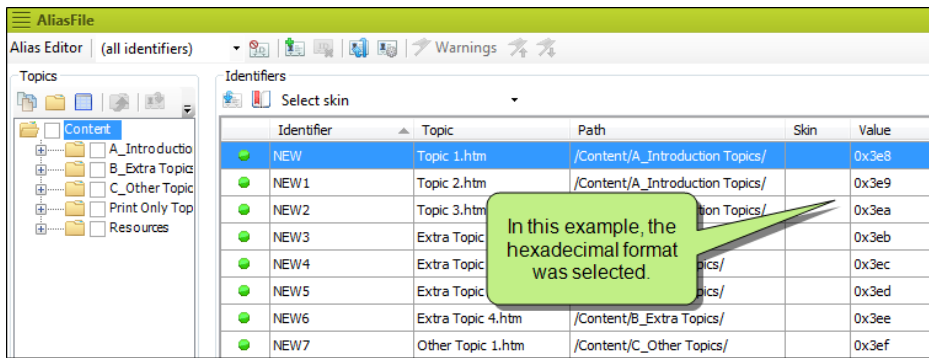
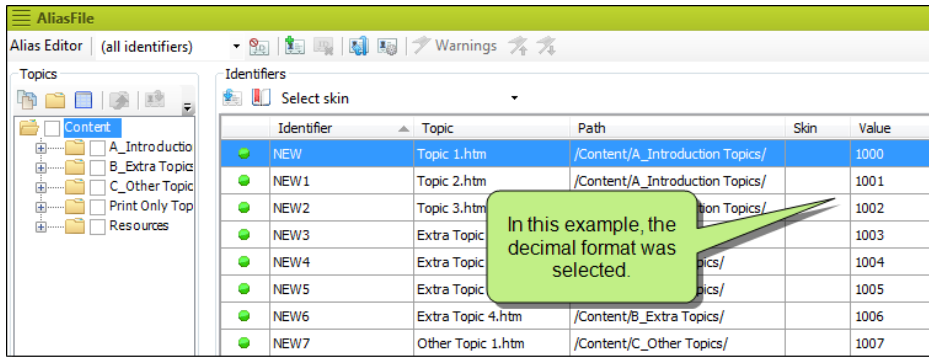


#### VALUE FORMAT

You can specify whether the identifier values should use a decimal or hexadecimal format. Using hexadecimal values does not affect your CSH in a different way; it's simply another option in case your developers prefer that format.



The image shows a dialog box titled "Starting value" with a text field containing "1000". Below this are several checkboxes: "Use Prefix" (unchecked), "Include to" (unchecked), "Capitalize" (checked), "Default sk" (unchecked), and "Primary Header" (unchecked). At the bottom, there is a "Value Format" section with a drop-down menu currently set to "Decimal". The drop-down menu is open, showing "Decimal" and "Hexadecimal" as options. A green callout bubble points to the drop-down menu with the text: "Use this drop-down to select the value format-- decimal or hexadecimal." At the bottom right of the dialog are "OK" and "Cancel" buttons.




4. Click **OK**.

## Creating and Assigning Identifiers

The first steps in developing context-sensitive Help (CSH) for your project are to add a header file and add an alias file. After this, you need to work on the identifiers for the header file.

- » If it is decided that you are responsible for creating the header file in CSH, you need to create *and* assign identifiers in the Alias Editor.
- » If it is decided that the software developer is responsible for creating the header file in CSH, you only need to assign the identifiers, which should already be contained in the header file that the developer provides for you. Before assigning the identifiers in the alias file, you must first import the header file from the developer so that it is located in the Advanced folder in the Project Organizer. See "Importing Header Files" on page 17.

Creating and assigning identifiers means:


- » **Creating Topic IDs and Unique Numerical Values That Correspond to the Different CSH Help Topics That You Want to Include** For example, if the software application that you are documenting contains a dialog called the "Properties dialog," you might have written a topic for it called "Using the Properties dialog." To connect the actual dialog with your topic, you might create a topic ID in the Alias Editor, naming it "Properties\_Dialog." Furthermore, let's say that you have already created similar topic IDs for 157 other topics and dialogs. You have given each of those topic IDs a unique value from 1 to 157. So for "Properties\_Dialog," you assign 158. Therefore, you end up with a topic ID and numerical value that looks like this: Properties\_Dialog 158.
- » **Assigning a Topic (or Even a Bookmark Within a Topic) to a Topic ID That You Created** For example, if you have created a topic ID called "Properties\_Dialog," you need to somehow link it to the topic that you want users to see when they click the Help button in that dialog. Let's say you want to link that topic ID with your topic named "Using the Properties dialog." Therefore, in the Alias Editor you could select the topic ID and then double-click the topic. This "ties" the topic ID to that specific topic.
- » **(Optional) Assigning a Skin to a Topic ID That You Create** For example, let's say that you want most of the topics in your Help system to open in a window that is 5 inches wide and 7 inches high (as well as other characteristics). Therefore, you create a skin that contains those specifications and name it "Main." However, for CSH topics that are opened from individual dialogs or windows, you want the Help window to be only 4 inches wide and 6 inches high (as well as other characteristics). So you create an additional skin containing those specifications and name it "Dialogs." In the Alias Editor, you can click  and select the Dialogs skin in the Identifier Options dialog. Then, whenever you create a new identifier, it will automatically be associated with that skin. You can also assign skins manually to each identifier in the Alias Editor. If you are generating HTML5 output, make sure that all of your CSH skins are either enabled with responsive output or not enabled with it; you should not have some skins that have it enabled and others that do not.




The following steps show you how to complete these tasks. Use whichever set of steps best applies to your situation:

- » Automatically generate identifiers for all topics
- » Create and assign identifiers to topics at the same time
- » Create identifiers only
- » Assign identifiers to topics only


#### HOW TO AUTOMATICALLY GENERATE IDENTIFIERS FOR ALL TOPICS

1. Open the alias file that you created.
2. (Optional) You can set options for new identifiers in advance. This will supply some of the information (e.g., starting value, prefix, include topic in ID name, assign skin) for you automatically as you create new identifiers. See "Setting Identifier Options" on page 23.
3. Do one of the following:
  - » In the local toolbar of the Alias Editor click .
  - » Right-click in the **Identifiers** side of the editor, and from the context menu select **Auto Generate**.

The Generate Identifiers dialog opens.

4. In the **Header File** area select one of the following, depending on whether you need to create a new header file or have an existing one you can select.
  - » **New from template** This lets you choose either a factory template file or one of your own customized template files as a starting point. The new file will take on all of the settings contained in the template. For more information about templates, see the online Help.
    - a. If you want to use a factory template provided by Flare, expand the **Factory Templates** folder and click on a template file. If you want to use your own customized template file, expand the appropriate folder and click on a file.
    - b. After selecting a template, use the **File Name** field to enter a name for the new header file.
    - c. If you want the new header file to be placed at the root level (i.e., directly under the Advanced folder in the Project Organizer), leave the **Folder** field set to **(root folder)**. If instead you have created a subfolder within the Advanced folder and want to place the new header file there, click  and select that subfolder.



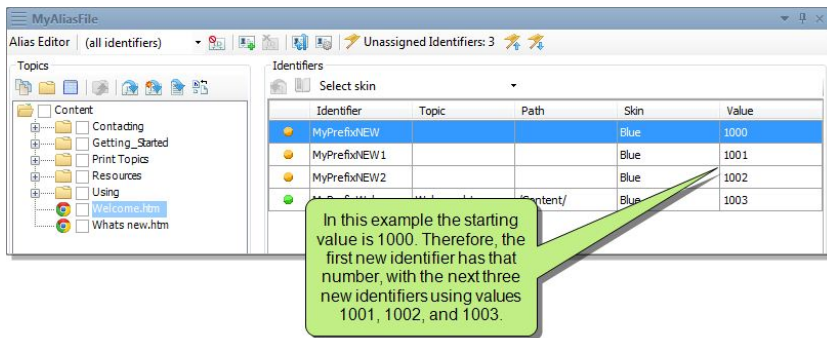
**Note:** In some dialogs and wizards you can click the **Manage Templates** button  if you want to open the Template Manager. This lets you manage any of your template files (e.g., add new templates, enter descriptions for templates). For more information see the online Help.

» **Choose Existing** This lets you choose an existing header file in your project. To use this option, click the drop-down field and select a header file.

- (Optional) In the **Identifier Options** area, you can override any of the values that are already set in the Identifier Options dialog (see Step 2).

### STARTING VALUE

You can specify the starting value for identifiers that you create, with the value for additional identifiers being incremented automatically based on that starting value.

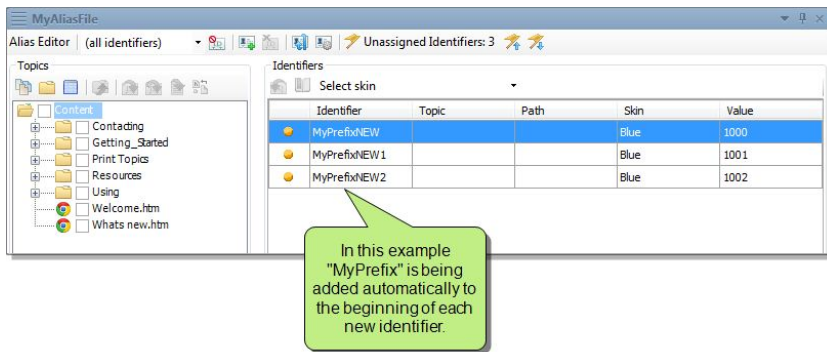


The screenshot shows the 'MyAliasFile' Alias Editor window. On the left is a 'Topics' tree with folders like 'Contating', 'Getting\_Started', 'Print Topics', 'Resources', 'Using', and files like 'Welcome.htm' and 'Whats new.htm'. The main area is titled 'Identifiers' and contains a table with columns: Identifier, Topic, Path, Skin, and Value. The table has four rows: 'MyPrefixNEW' with Value 1000, 'MyPrefixNEW1' with Value 1001, 'MyPrefixNEW2' with Value 1002, and a fourth row with Value 1003. A green callout bubble points to the 'Value' column, stating: 'In this example the starting value is 1000. Therefore, the first new identifier has that number, with the next three new identifiers using values 1001, 1002, and 1003.'

Identifier	Topic	Path	Skin	Value
MyPrefixNEW			Blue	1000
MyPrefixNEW1			Blue	1001
MyPrefixNEW2			Blue	1002
			Blue	1003

### PREFIX

You can specify a prefix to be added at the beginning of each new identifier that you create (e.g., ID\_, Dialog, Module1).

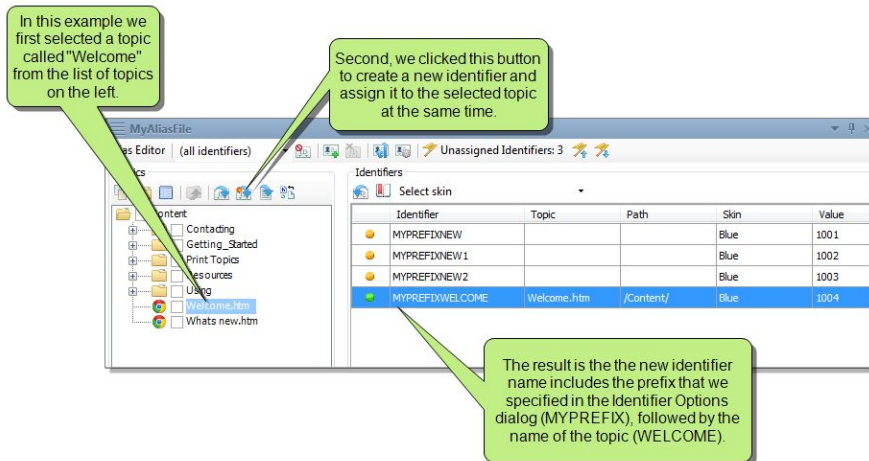


The screenshot shows the 'MyAliasFile' Alias Editor window, similar to the previous one. The 'Identifiers' table now has three rows: 'MyPrefixNEW' with Value 1000, 'MyPrefixNEW1' with Value 1001, and 'MyPrefixNEW2' with Value 1002. A green callout bubble points to the 'Identifier' column, stating: 'In this example "MyPrefix" is being added automatically to the beginning of each new identifier.'

Identifier	Topic	Path	Skin	Value
MyPrefixNEW			Blue	1000
MyPrefixNEW1			Blue	1001
MyPrefixNEW2			Blue	1002

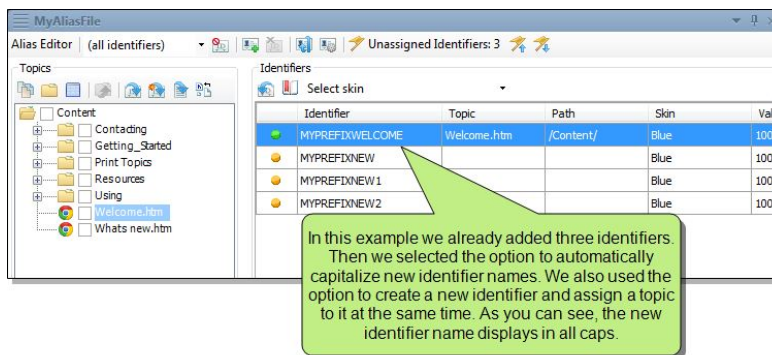
### INCLUDE TOPIC NAME IN IDENTIFIER NAME

You can specify that new identifiers should automatically include the assigned topic in the name.



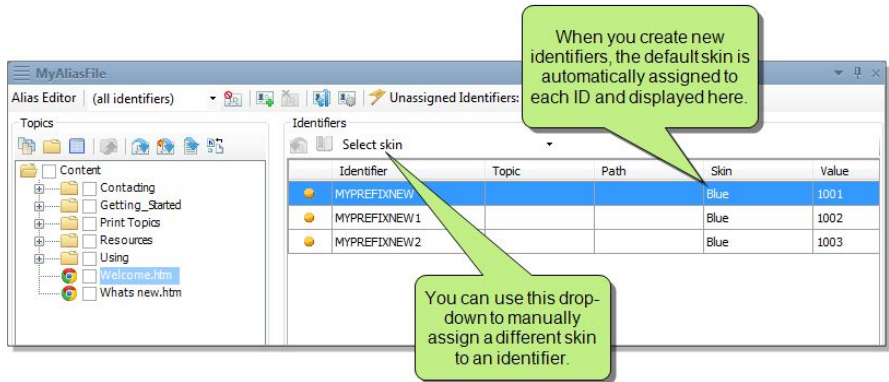
### CAPITALIZE IDENTIFIER NAMES

You can specify that the name that is automatically added for new identifiers should have all caps.



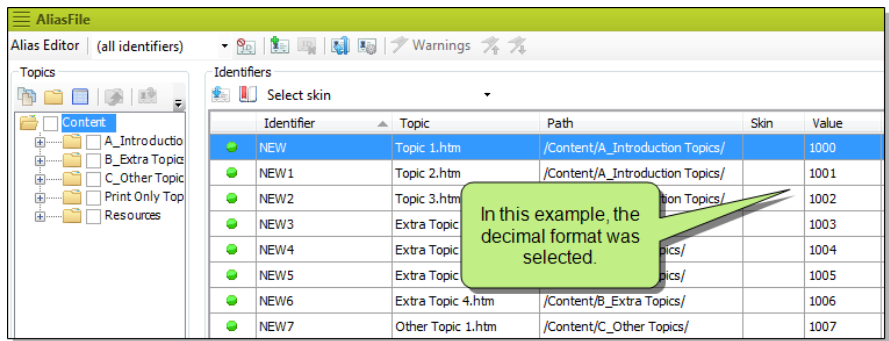
### SKINS

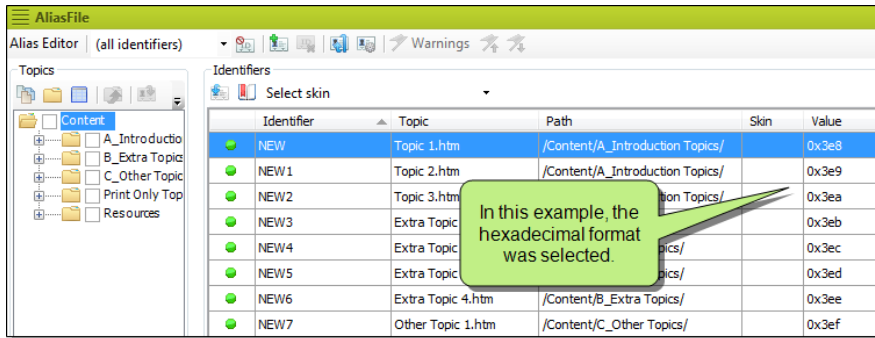
You can specify which skin should be assigned by default to new identifiers that are created. Of course, you can always manually select a different skin for any identifier, but when you first create a new identifier, it will initially be assigned to the default skin that you specified.






### VALUE FORMAT

You can specify whether the identifier values should use a decimal or hexadecimal format. Using hexadecimal values does not affect your CSH in a different way; it's simply another option in case your developers prefer that format.





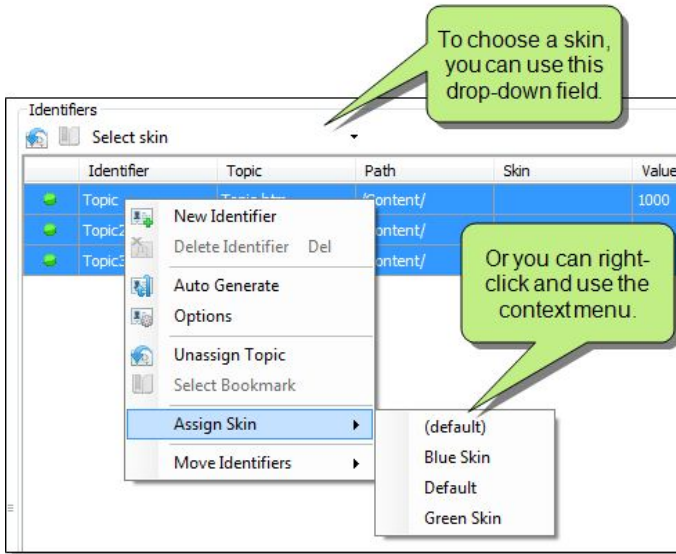
6. Click in the **Generate Identifiers for** field and select one of the following:
  - » **Unassigned topics only** Select this if you already have some identifiers in the header file that are assigned to topics. New identifiers will be created only for topics in your project that are not yet assigned to an identifier.
  - OR
  - » **All topics** Select this if you want new identifiers to be created for all topics in your project, whether some topics are already assigned to new topics or not.
7. If you selected the All topics option in the previous step, select one of the following in the **Existing Identifiers** field:
  - » **Keep** Select this if you want to keep the identifiers that you already have in the header file. As a result, you will have some topics that are assigned to multiple identifiers (i.e., the old identifier and the new one).
  - » **Delete** Select this if you want to delete the existing identifiers in the header file, creating new ones instead. This way, you will not have any topics that are assigned to multiple identifiers.
8. Click **Create** and in the next dialog click **OK**. In the Alias Editor, the header file is automatically selected in the local toolbar. Also, new rows are added in the Alias Editor with  (instead of ) next to them. The green icon indicates that the identifiers are assigned to topics. If you have previously set identifier options (see Step 2), the identifier includes at least some of the appropriate information already.
9. (Optional) If you want a certain identifier to point to a specific bookmark or header in the assigned topic, do the following:
  - a. Select the identifier row.
  - b. Click the  button located above the identifier list.
  - c. In the dialog that opens, select a bookmark or header.
  - d. Click **OK**.

10. (Optional) If you want to change an identifier name, click twice in the **Identifier** cell and type the name (e.g., Properties\_Dialog).



**Note:** Make sure you use underscores between words because spaces are not allowed.

11. (Optional) If you want to change the skin associated with one or more identifiers, do the following:
- Select the identifier row(s). You can hold the **SHIFT** key to select a range, or you can hold the **CTRL** key to select individual items.
  - Do one of the following:
    - » Click the arrow in the **Select skin** button Select skin ▾ (located above the identifier list) and choose a skin from the drop-down.
    - OR
    - » Right-click somewhere in the list. From the context menu select **Assign Skin** and then from the sub-menu choose the name of the skin.



12. (Optional) If you want to change the numerical value for an identifier, click twice in the **Value** cell and type the new value.



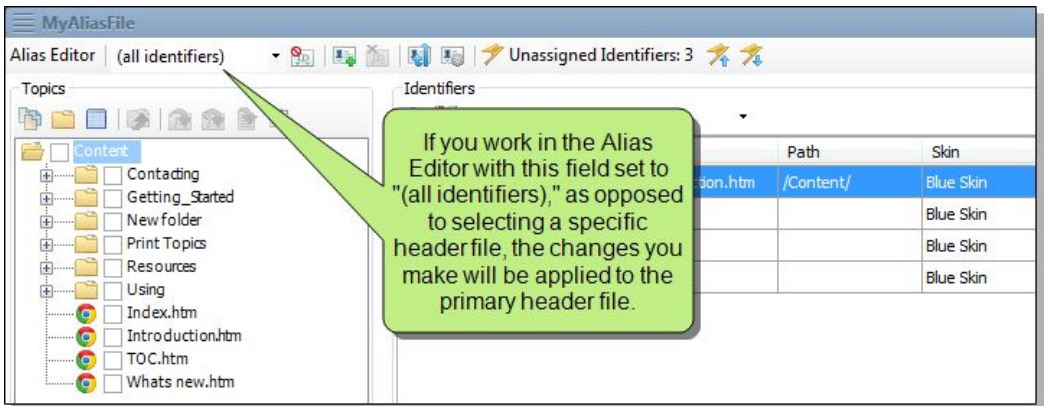
**Note:** You may need to use the horizontal scroll bar at the bottom of the editor to see this cell.

13. Click  to save your work.

#### HOW TO CREATE AND ASSIGN IDENTIFIERS AT THE SAME TIME

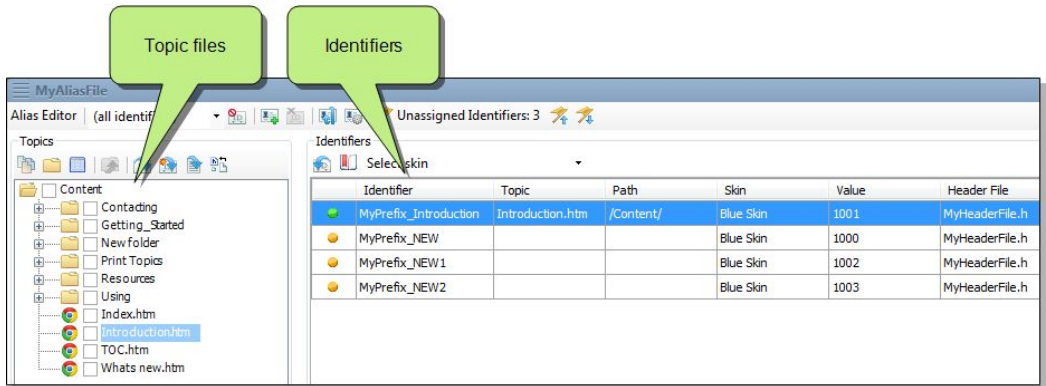
1. Open the alias file that you created.
2. (Optional) You can set options for new identifiers in advance. This will supply some of the information (e.g., starting value, prefix, include topic in ID name, assign skin) for you automatically as you create new identifiers. See "Setting Identifier Options" on page 23.
3. If you have more than one header file in your project, click the down arrow at the top-left side of the Alias Editor and select the header file for which you want to create identifiers. Otherwise, you can just leave (*all identifiers*) in the field.

What if you have multiple header files in your project and you do not select a header file, leaving (*all identifiers*) shown in the drop-down field? In that case, the changes you made in the Alias Editor are applied to the primary header file that you have selected when setting identifier options. See "Setting Identifier Options" on page 23.

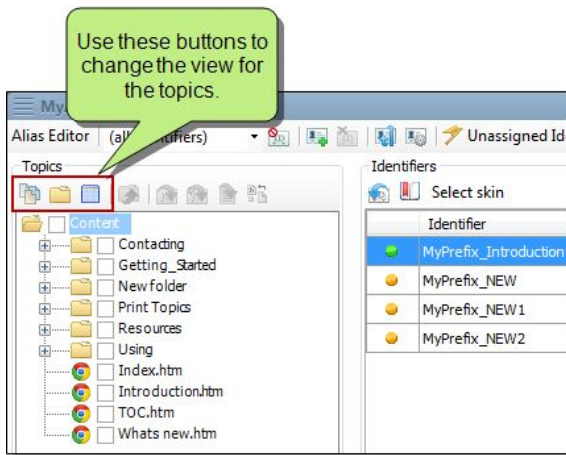










4. On the left side of the Alias Editor, select one or more topics that you want to assign to new identifiers.




You can click the show and hide buttons to help you better locate topics. If you use the options to split the view into two halves, you can select multiple topics on the right side by holding down the SHIFT or CTRL keys as you click.





	Shows all of the files in the project in a list in the area below. If you click the button again, it switches to a folder tree view. In the list, you can click the File, Type, or Path column headers to sort the list alphabetically by that column data.
	Shows or hides the folders that the files are stored in.
	Shows or hides the files. If you click this button when the Show Folders button  is selected, the area splits into two halves. The folder is shown on the left side, and the files and subfolders within it are shown on the right.
	If the Show Files button  is the only one selected, you can click this button to move up one folder level.

5. Do one of the following:


- >> Click the  button located above the file list.

OR

- >> Right-click a topic on the left side of the editor, and select **Assign Topic to New Identifier** from the context menu.

A new row is added with  (instead of ) next to it. The green icon indicates that the identifier is assigned to a topic. If you have previously set identifier options (see Step 2), the identifier includes at least some of the appropriate information already.

6. (Optional) If you want the identifier to point to a specific bookmark or header in the topic, do the following:

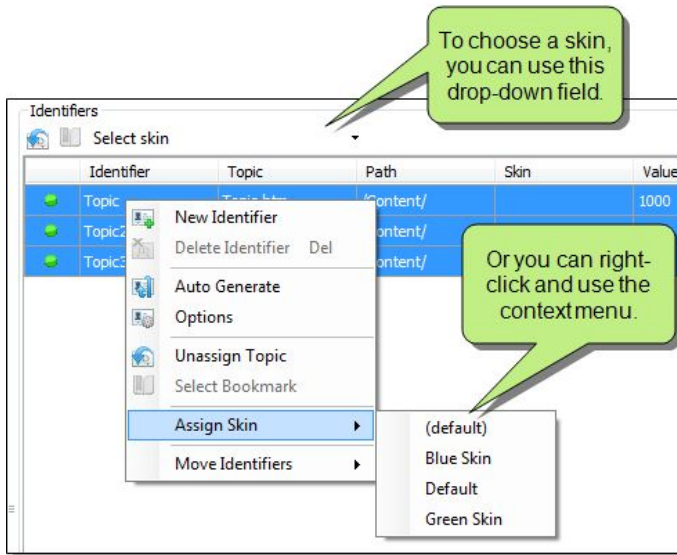
- a. Select the identifier row.
- b. Click the  button located above the identifier list.
- c. In the dialog that opens, select a bookmark or header.
- d. Click **OK**.

7. (Optional) If you want to change the identifier name, click twice in the **Identifier** cell and type the name (e.g., Properties\_Dialog).



**Note:** Make sure you use underscores between words because spaces are not allowed.


8. (Optional) If you want to change the skin associated with one or more identifiers, do the following:
- Select the identifier row(s). You can hold the **SHIFT** key to select a range, or you can hold the **CTRL** key to select individual items.
  - Do one of the following:
    - » Click the arrow in the **Select skin** button Select skin ▾ (located above the identifier list) and choose a skin from the drop-down.
    - OR
    - » Right-click somewhere in the list. From the context menu select **Assign Skin** and then from the sub-menu choose the name of the skin.



9. (Optional) If you want to change the numerical value for the identifier, click twice in the **Value** cell and type the new value. You can enter a decimal or hexadecimal value. Using hexadecimal values does not affect your CSH in a different way; it's simply another option in case your developers prefer that format.



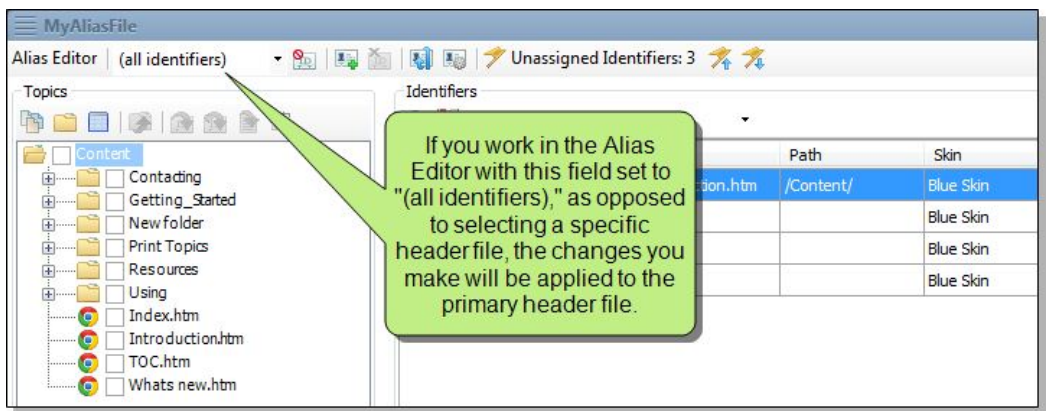
**Note:** You may need to use the horizontal scroll bar at the bottom of the editor to see this cell.


10. Repeat these steps for each identifier that you want to add. Each entry will represent a different dialog or window in the software application.
11. Click  to save your work.



### HOW TO CREATE IDENTIFIERS ONLY

1. Open the alias file that you created.
2. If you have more than one header file in your project, click the down arrow at the top-left side of the Alias Editor and select the header file for which you want to create identifiers. Otherwise, you can just leave (*all identifiers*) in the field.

What if you have multiple header files in your project and you do not select a header file, leaving (*all identifiers*) shown in the drop-down field? In that case, the changes you made in the Alias Editor are applied to the primary header file that you have selected when setting identifier options. See "Setting Identifier Options" on page 23.



3. Do one of the following:
  - » In the local toolbar click .
  - OR
  - » Right-click in the **Identifiers** side of the editor, and select **New Identifier** from the context menu.

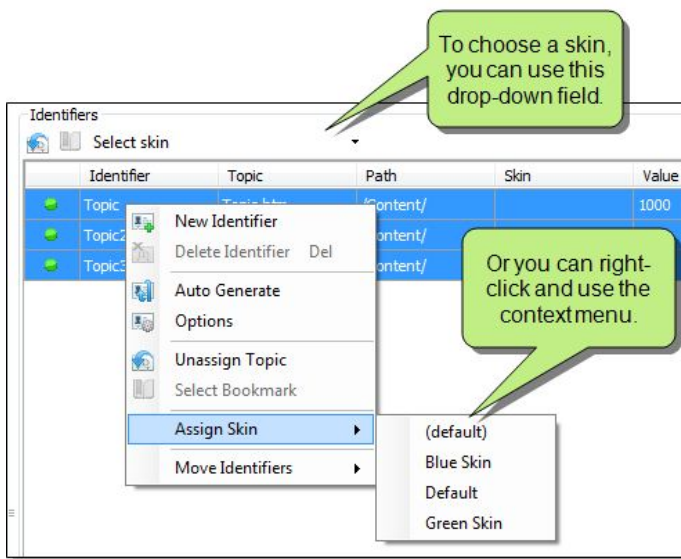
A new row is added with  (instead of ) next to it. The yellow icon indicates that the identifier is not yet assigned to a topic. If you have previously set identifier options (see Step 2), the identifier includes at least some of the appropriate information already.

4. (Optional) If you want to change the identifier name, click twice in the **Identifier** cell and type the name (e.g., Properties\_Dialog).



**Note:** Make sure you use underscores between words because spaces are not allowed.


5. (Optional) If you want to change the skin associated with one or more identifiers, do the following:
- Select the identifier row(s). You can hold the **SHIFT** key to select a range, or you can hold the **CTRL** key to select individual items.
  - Do one of the following:
    - » Click the arrow in the **Select skin** button Select skin ▾ (located above the identifier list) and choose a skin from the drop-down.
    - OR
    - » Right-click somewhere in the list. From the context menu select **Assign Skin** and then from the sub-menu choose the name of the skin.




6. (Optional) If you want to change the numerical value for the identifier, click twice in the **Value** cell and type the new value. You can enter a decimal or hexadecimal value. Using hexadecimal values does not affect your CSH in a different way; it's simply another option in case your developers prefer that format.

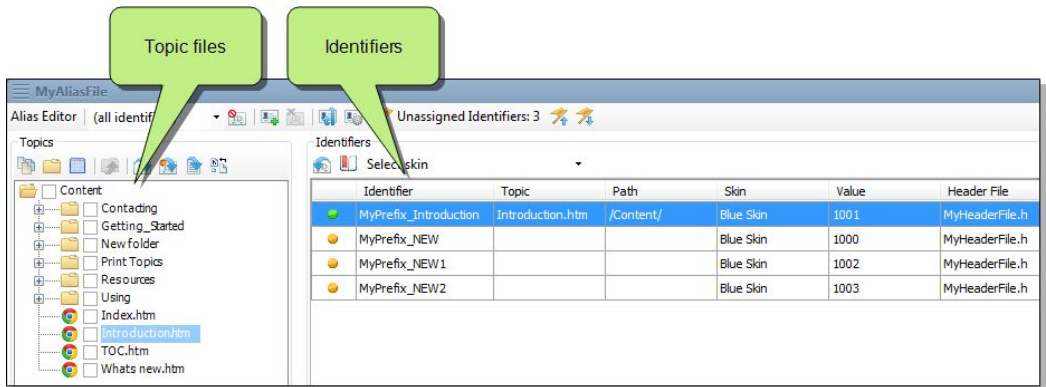


**Note:** You may need to use the horizontal scroll bar at the bottom of the editor to see this cell.

7. Repeat these steps for each identifier that you want to add (each one to represent a different dialog or window in the software application).
8. Click  to save your work.

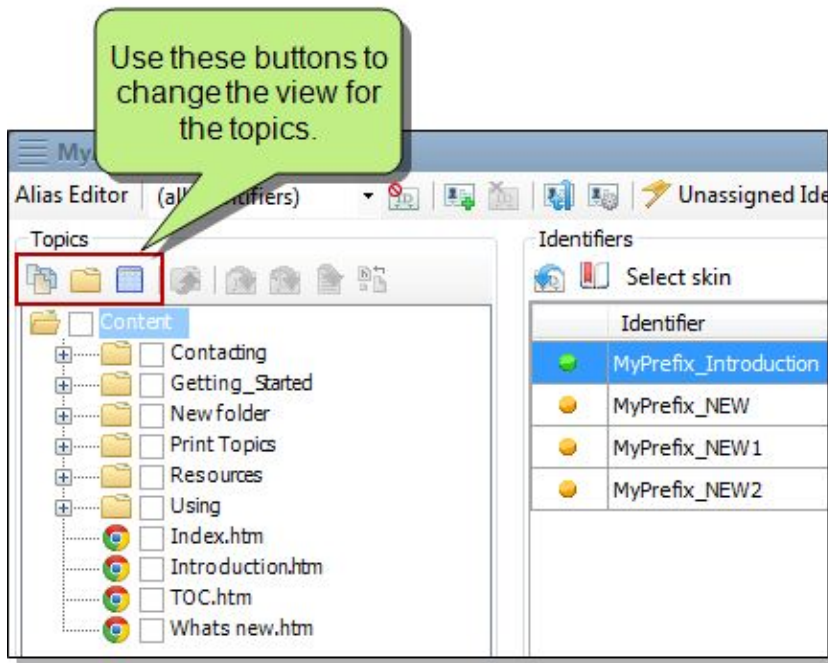
### HOW TO ASSIGN IDENTIFIERS TO TOPICS

1. Open the alias file that you created.
2. (Optional) If you want to see only the identifiers that are not yet assigned to topics, click  in the local toolbar of the Alias Editor. This hides identifiers that are already assigned to topics in the editor until you click the button again.
3. On the right side of the Alias Editor, select an identifier that you want to assign to a topic.
4. On the left side of the Alias Editor, find a topic that you want to assign to the identifier that you selected in the previous step.



You can click the show and hide buttons to help you better locate topics.






Shows all of the files in the project in a list in the area below. If you click the button again, it switches to a folder tree view. In the list, you can click the File, Type, or Path column headers to sort the list alphabetically by that column data.




Shows or hides the folders that the files are stored in.



Shows or hides the files. If you click this button when the Show Folders button  is selected, the area splits into two halves. The folder is shown on the left side, and the files and subfolders within it are shown on the right.




If the Show Files button  is the only one selected, you can click this button to move up one folder level.

5. Do one of the following:



>> Double-click the topic.

OR

>> Click the  button located above the file list.

OR

>> Right-click the topic and select **Assign Topic to Selected Identifier** from the context menu.

The topic name is added to the identifier row on the right side of the Alias Editor, and a green icon  (instead of ) is displayed next to it. The green icon indicates that the identifier is assigned to a topic.

6. (Optional) If you want the identifier to point to a specific bookmark or header in the topic, do the following:

a. Select the identifier row.

b. Click the  button located above the identifier list.








c. In the dialog that opens, select a bookmark or header.










d. Click **OK**.









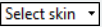

7. Click  to save your work.


**BUTTON AND CONTEXT MENU EXPLANATIONS**

Following are explanations of the various buttons and context menu options in the Alias Editor, some of which have been included in the steps above.

New	Description
	<p>Hide assigned identifiers</p> <p>This hides the identifiers that already have topics assigned to them. By doing this, you can concentrate solely on the identifiers that need topics assigned to them.</p>
	<p>Create new identifier</p> <div><b>Note:</b> You can also perform this task by right-clicking in the <b>Identifiers</b> side of the editor, and from the context menu selecting <b>New Identifier</b>.</div>
	<p>Delete selected identifier</p> <div><b>Note:</b> You can also perform this task by right-clicking a selection in the <b>Identifiers</b> side of the editor, and selecting <b>Delete Identifier</b> from the context menu.</div>
	<p>Generate identifiers</p> <p>This lets you automatically generate new identifiers for all of the topics in your project. This is a good feature to use if you are creating identifiers in a header file for the first time. When you click this button, the Generate Identifiers dialog opens. In this dialog you can create a new header file or select an existing one. You can also see the identifier options—such as starting value, prefix, and skin—that will be applied to all of the new identifiers that are created.</p> <div><b>Note:</b> You can also perform this task by right-clicking in the <b>Identifiers</b> side of the editor, and from the context menu selecting <b>Auto Generate</b>.</div>

New	Description
	<p>Change identifier options</p> <p>This lets you specify options in advance for how new identifiers are created. See "Setting Identifier Options" on page 23.</p> <div>  <b>Note:</b> You can also perform this task by right-clicking in the <b>Identifiers</b> side of the editor, and from the context menu selecting <b>Options</b>. </div>
 Unassigned Identifiers: 2	<p>Identifier warning</p> <p>This lets you see if there are any issues with identifiers in the editor, such as identifiers that are not yet assigned to topics.</p>
	<p>Previous warning</p> <p>This highlights the previous identifier in the list that has an issue.</p>
	<p>Next warning</p> <p>This highlights the next identifier in the list that has an issue.</p>
	<p>Assign topic to existing identifier</p> <div>  <b>Note:</b> You can also perform this task by right-clicking on the topic, and from the context menu selecting <b>Assign Topic to Selected Identifier</b>. </div>
	<p>Create new identifier and assign topic to it at the same time</p> <div>  <b>Note:</b> You can also perform this task by right-clicking on the topic, and from the context menu selecting <b>Assign Topic to New Identifier</b>. </div>

New	Description
	<p>Open selected topic</p> <p> <b>Note:</b> You can also perform this task by right-clicking on the topic, and from the context menu selecting <b>Open</b>.</p>
	<p>Locate topic in header file</p> <p>This displays all header files where the selected topic is assigned to an identifier.</p> <p> <b>Note:</b> You can also perform this task by right-clicking on the topic, and from the context menu selecting <b>Locate the Topic in Headers</b>.</p>
	<p>Unassign topic from identifier</p> <p> <b>Note:</b> You can also perform this task by right-clicking in the <b>Identifiers</b> side of the editor, and from the context menu selecting <b>Unassign Topic</b>.</p>
	<p>Select bookmark in selected topic</p> <p>First you can select an identifier row that has already been assigned to a topic and then click this new button. A dialog opens, letting you select a bookmark within that topic.</p> <p> <b>Note:</b> You can also perform this task by right-clicking in the <b>Identifiers</b> side of the editor, and from the context menu selecting <b>Select Bookmark</b>.</p>
	<p>Select skin for identifier</p> <p>You can click the down arrow in the <b>Select Skin</b> button and choose the appropriate skin for the identifier.</p>
	<p>Not assigned</p> <p>This icon displays next to any identifiers that are not yet assigned to a topic.</p>

New	Description
	Assigned This icon displays next to any identifiers that have already been assigned to a topic.




**Note:** You can move identifiers from one header file to another by using the right-click context menu. See "Moving Identifiers to Different Headers" on the following page.

## Moving Identifiers to Different Headers

In the Alias Editor, you can move identifiers to a different header file.

### *HOW TO MOVE IDENTIFIERS TO A DIFFERENT HEADER*

1. Open an alias file.
2. Select the identifiers that you want to move to another header file. You can hold the **SHIFT** key to select a range, or you can hold the **CTRL** key to select individual items.
3. Right-click and from the context menu select **Move Identifiers>[Name of Header File]**.
4. Click  to save your work.


## Importing Alias Files

Not only can you add new alias files to Flare, but you can also import existing alias files (FLALI files).

### HOW TO IMPORT AN ALIAS FILE

1. Do one of the following, depending on the part of the user interface you are using:
  - » **Ribbon** Select the **Project** ribbon. In the **Content** section select **New>Advanced>Alias File**.
  - » **Menu** Select **Project>Advanced>Add Alias File**.
  - » **Right-Click** In the Project Organizer, right-click on the **Advanced** folder and from the context menu select **Add Alias File**.

The Add File dialog opens.

2. In the **File Type** field at the top, make sure **Alias File** is selected.
3. Select **New from existing** and click .
4. Find and select the FLALI file that you want to import.
5. Click **Open**. The Source File field now contains the path to the file that you are importing. Also, the name of the file is displayed in the File Name field.
6. If you want to give the alias file a different name than that for the imported file, click in the **File name** field and replace the text.
7. Click **Add**. The alias file is added and opens in the Alias Editor.



## Opening Alias Files

When you want to work on an existing alias file, use the following steps to open it.


### *HOW TO OPEN AN ALIAS FILE*

1. Make sure the Project Organizer is open.
2. Double-click the **Advanced** folder. The alias file(s) in your project are displayed (next to any other files that you have created, such as browse sequences, search filter sets, or header files).
3. Double-click the alias file that you want to open. The Alias Editor opens to the right, with the alias file page shown.

## Associating an Alias File with a Target

If you have added more than one alias file for your project, you need to associate the appropriate alias file with the target that you plan to build. If you do not specify an alias file in a target, Flare uses the first alias file listed in the Project Organizer.

### *HOW TO ASSOCIATE AN ALIAS FILE WITH A TARGET*

1. Open the target.
2. In the Target Editor, click the **Advanced** tab.
3. Click the drop-down arrow in the **Alias File** field, and select the alias file that you want to associate with the target.
4. Click  to save your work.

# CHAPTER 4

## Additional Tasks for CSH

In addition to working with header and alias files, there are some additional tasks that you may perform when it comes to context-sensitive Help (CSH).

This chapter discusses the following:

Testing Context-Sensitive Help .....	60
Creating Topic Alias Markers in FrameMaker for CSH .....	63
CSH Calls for DotNet Help .....	65
CSH Calls for HTML Help .....	82
CSH Calls for HTML5 Output .....	85
CSH Calls for WebHelp .....	91
CSH Calls for WebHelp Mobile .....	98



## Testing Context-Sensitive Help

After you create context-sensitive Help (CSH), you should test it to verify that it works. You can do this a couple of different ways, and it is not a bad idea to do both.

First, you can test your CSH identifiers from inside your project. If you are testing DotNet Help, Microsoft HTML Help, WebHelp, or WebHelp Plus, you use the Context-Sensitive Help API Tester dialog. If you are testing HTML5 output, you use the CSH Test page. Second, you can test the CSH using a new build of the software application. The following steps show you the different ways to test CSH.

There are different ways to test CSH calls from inside a project. The method you use depends on the output type.


### HOW TO USE THE TEST CSH API CALL DIALOG—DOTNET HELP, HTML HELP, WEBHELP, AND WEBHELP PLUS

1. Do one of the following, depending on the part of the user interface you are using:
  - » **Ribbon** If you want to test the CSH for the primary target, select the **Tools** ribbon. In the **Context Sensitive Help** section, select **Test CSH API Calls**.
  - » **Menu** If you want to test the CSH for the primary target, select **Build>Test CSH API Calls - Primary**.
  - » **Right-Click** In the Project Organizer, right-click a target under the **Targets** folder, and select **Test CSH API Calls**.



**Note:** If the target's output is not up-to-date, click **Yes** when prompted to regenerate it.

The Context-Sensitive Help API Tester dialog opens.

2. (Optional) In the **Skin** drop-down list, select the desired skin for viewing the topic.
3. Next to each identifier, click . If the correct Help topic opens, the CSH link works.
4. When you have finished testing your topics, click **Close**.

### HOW TO USE THE CSH TEST PAGE—WEBHELP 2.0 (HTML5)

1. Do one of the following, depending on the part of the user interface you are using:
  - » **Ribbon** If you want to test the CSH for the primary target, select the **Tools** ribbon. In the **Context Sensitive Help** section, select **Test CSH API Calls**.
  - » **Menu** If you want to test the CSH for the primary target, select **Build>Test CSH API Calls - Primary**.
  - » **Right-Click** In the Project Organizer, right-click a target under the **Targets** folder, and select **Test CSH API Calls**.



**Note:** If the target's output is not up-to-date, click **Yes** when prompted to regenerate it.

The CSH Test page opens in a web browser. For Chrome, Firefox, and Internet Explorer users, the page opens in a new tab. For Safari users, the page opens in a new window.

2. Test the desired topic(s) using one of these options:
  - » **Enter a Topic ID and Value** In the **Topic ID** box, enter a topic value and select the topic identifier. You can find this information in the Value and Identifier columns of your alias file.
  - OR
  - » **Search for a Topic** Type a keyword in the **Search** box. Then under the **First Pick** label select **None**, **True**, or **False**.
3. (Optional) In the **Skin** drop-down list, select the desired skin for viewing the topic.
4. If you published your output to a web server and want to test a CSH call, type the web site path in the **Path to Help System** box. If you leave this box blank, Flare will test the CSH call to the Output folder.

### EXAMPLES

If you published your help system to the root directory on a web server, you might type:

```
http://help.example.com
```

OR

```
https://help.example.com
```

If you are publishing multiple outputs on the same web server, you might type one of the following (where [Flare Target Name] = the publishing destination folder of the target):

```
http://help.example.com/[Flare Target Name]
```

OR

```
https://help.example.com/[Flare Target Name]
```

5. Click **Show Help (Javascript)**. If the correct topic opens, the CSH link works.
6. When you have finished testing your topics, close your web browser.

#### *HOW TO TEST CONTEXT-SENSITIVE HELP USING A BUILD FROM THE APPLICATION*

1. After the developer finishes "hooking" the identifiers in the application, obtain a new build from the developer.
2. Launch the application build that you obtained from the developer.
3. Open a dialog or window that should be tied to a CSH topic.
4. Click the **Help** button in the dialog or window, or press **F1** on your keyboard. The topic associated with the dialog or window should open accordingly.
5. Test all of the CSH-related dialogs or windows in this same way.

## Creating Topic Alias Markers in FrameMaker for CSH

A topic alias marker is a special marker that you can insert into your FrameMaker source content wherever you want a new context-sensitive Help (CSH) identifier to be created in Flare (after you import the FrameMaker document). After the developer "hooks" the CSH identifier with the appropriate user interface element in the application, that imported FrameMaker content will display when a user opens the CSH in the application's user interface.

You can use these markers as an alternative to creating the identifiers in Flare. For more information see "Creating and Assigning Identifiers" on page 32 and "About Context-Sensitive Help" on page 5.

Following are steps for creating these markers in Adobe FrameMaker. For more information, please refer to the documentation provided with Adobe FrameMaker.

### *HOW TO CREATE A TOPIC ALIAS MARKER FOR CONTEXT-SENSITIVE HELP*

The following steps may vary in different versions of FrameMaker.

1. Open the FrameMaker document.
2. In FrameMaker, create the topic alias marker type that is required for this feature. This is a one-time step in the document. Afterwards, you can simply select the newly created marker type when applying it to content. To create the topic alias marker type, do the following:
  - a. In FrameMaker, select **Special>Marker**. The Marker dialog opens.
  - b. From the **Marker Type** field, click the down arrow and select **Edit**. The Edit Custom Marker Type dialog opens.
  - c. In the field, replace the existing text and type `TopicAlias`.
  - d. Click **Add**.
  - e. Click **Done**.

3. In the FrameMaker document, place your cursor at the location where you want to insert a topic alias marker.

Usually you will place the cursor at the beginning of a heading that you want to be the start of a new online topic in Flare.

4. If it is not already displayed, open the Marker dialog (**Special>Marker**).
5. If the topic alias marker type is not already displayed in the dialog, click the **Marker Type** down arrow and select **TopicAlias**.
6. In the **Marker Text** field, enter the text to be used as the topic ID.

You might simply type the same text that occurs in the paragraph heading. That way, it will be easily identified by both you and the developer. When typing the topic ID, make sure you use underscores between words, because spaces are not allowed (e.g., `Properties_Dialog`).

7. Click **New Marker**.

8. Close the Marker dialog.
9. Save the FrameMaker document.



**Note:** Flare supports FrameMaker 7.0 and newer versions.

After you import the FrameMaker document(s), you can open the alias or header file to see the new identifiers created from the topic alias markers.



## CSH Calls for DotNet Help

There are several ways that your Flare DotNet Help system can be "connected" to a software application.

### Available DotNet Help Features in Flare

Features that are available include the following.

- » **Detached or Integrated Help** The Help system can be connected so that the Help opens in a window separate from your software application (detached Help).

#### EXAMPLE

To see an example of detached Help in Flare's online Help, select **Tools>Manage Templates** and then press **F1**. The topic for that dialog opens in a separate window.

Alternatively, the Help can be connected so that the Help opens within the software application (integrated Help).

#### EXAMPLE

To see an example of integrated Help in Flare's online Help, select **Help>Contents**. The TOC for the Help opens within the application. If you click a topic page in the TOC, the topic opens in the application to the right.

- » **Same or Different Process** The Help system can be connected so that it uses the same process as the software application (embedded). If the Help uses the same process, the Help will close when the application is closed. The online Help in Flare is connected so that it uses the same process as Flare. Alternatively, the Help system can be connected so that it uses a different process than the software application. If the Help uses a different process, the Help can remain open even if the application is closed.

- » **Basic Help or Context-Sensitive Help (CSH)** The developer can connect the application to your basic DotNet Help output, rather than to a specific topic. For example, you might want a standalone version of the Help Viewer to open separately from the software application, displaying your startup topic and showing panes for the navigation elements (e.g., TOC, Index, Search, Browse Sequences) that you have created. Or maybe you want users to be able to open the different navigation elements within the application and access topics from each one.

#### EXAMPLE

To see an example of this in Flare's online Help, select **Help>Contents**. The TOC for the Help opens within the application. If you click a topic page in the TOC, the topic opens in the application to the right.

Alternatively, the developer can use CSH to connect the application to specific topics in the DotNet Help output (as long as you have created CSH in your Flare project and share the header file information with the developer).

#### EXAMPLE

To see an example of CSH in Flare's online Help, select **Tools>Manage Templates** and then press **F1**. The topic for that specific dialog opens.

- » **Dynamic Help** The developer can also incorporate your Help into the application using a unique feature called "Dynamic Help." This is a type of CSH where a Help window automatically displays topics from a Help system as an individual uses the application. It does not require any other action from the user (e.g., clicking a button or pressing a shortcut key). The Help system simply follows the actions of the user, automatically providing the appropriate Help content based on the area of the interface that is being clicked.

#### EXAMPLE

To see an example of Dynamic Help in Flare's online Help, select **Help>Dynamic Help**. Then start clicking anywhere in the Flare interface. As you do so, the Help topic changes accordingly.

- » **F1 Help** The developer can connect the application to the Help in such a way that the Help opens when a user presses the F1 key.

#### EXAMPLE

To see an example of this in Flare's online Help, simply click in any area of the interface or open any dialog. Then press **F1**. The appropriate Help topic for that area of the interface displays.

- » **Dialog's Help Button** The developer can connect the application to the Help in such a way that the Help opens when a user clicks the dialog's Help button (i.e., the "question mark" button).

#### EXAMPLE

To see an example of this in Flare's online Help, select **Tools>Manage Templates**. Then click the **?** button in the upper-right corner of the dialog.

## What You Need to Do

1. Work with your developer to determine which of the above features you want to use. You may want to use several of the features, as is the case with the Flare's online Help.
2. Create and build your DotNet Help project in Flare.
3. Provide the developer with the output files, as well as the CSH header file (if you plan to use CSH).
4. Provide the developer with the information in the following topics. The developer should begin with the "CSH Calls for DotNet Help—Developers" topic.
  - » "CSH Calls for DotNet Help—Developers" on the next page
  - » "HelpViewerClient API" on page 73
  - » "HelpViewerEmbeddedClient API" on page 75
  - » "IEmbeddedHelpSystem API" on page 79
  - » "ICSHIDProvider API" on page 77
  - » "Command Line" on page 71

## CSH Calls for DotNet Help—Developers

### Information for Developers

There are three ways to make DotNet Help calls from your Windows application. The most basic way is to use the command line functionality of the DotNet Help Viewer (see "Command Line" on page 71). The other two ways are to use the `HelpViewerClient` (HVC) (see "HelpViewerClient API" on page 73) and `HelpViewerEmbeddedClient` (HVEC) (see "HelpViewerEmbeddedClient API" on page 75) classes. Use the command line version only if your application doesn't have access to the latest .NET framework. Otherwise, you should use one of the other two methods.

#### Features of the DotNet Help CSH Engine

- » Open a topic of your choice in the MadCap Help Viewer.
- » Launch a Help system with the search results from a search string that you specify.
- » Embed Help topics or search results directly into your application.
- » Dynamic Help—A Help window automatically shows relevant information based on the user's focus in your application.

#### Key Differences between the HVC and HVEC Classes

- » HVC only supports basic, non-embedded CSH calls. It also does not provide Dynamic Help functionality.
- » HVEC contains built-in functionality for F1 help. F1 help is still possible with HVC, however the developer must manually handle the user pressing the F1 key.
- » HVC opens the Help Viewer in a separate process from your application, whereas HVEC opens the Help Viewer in the same process as your application. This means that with HVC, if your application closes, the Help window will remain open. On the other hand, with HVEC, if your application closes, the Help window will also close.

## Using the API

You can download the Help Viewer SDK at:

<http://www.madcapsoftware.com/downloads/redistributables.aspx>

You must add a reference to the appropriate DLL in your project to use the API.

In Visual Studio 2005, do the following.

1. In the Solution Explorer, right-click the project.
2. Click **Add Reference**.
3. Select the **Browse** tab.
4. Navigate to the appropriate .dll file and select it:
  - » Select **MadCap.HelpViewerClient.dll** to use the HelpViewerClient API.
  - OR
  - » Select the **MadCap.HelpViewerEmbeddedClient.dll** to use the HelpViewerEmbeddClient API.
5. Click **OK**.

## Sample Applications

There are two sample applications available in the SDK that demonstrate using the HelpViewerClient and HelpViewerEmbeddedClient classes.

## Distribution

Help Viewer must be installed on the user's machine that is using your application. The SDK contains a merge module that you can incorporate into an MSI-based installer to distribute the Help Viewer.

The following are instructions on how to include the merge module using Visual Studio 2005.

1. In the Solution Explorer, right-click the MSI installer project.
2. Click **Add**.
3. Click **Merge Module**.
4. Browse to the **InstallMadCapHelpViewer.msm** file and select it.
5. Click **Open**.

## Search Performance and Wildcards

When SQL Server Compact 3.5 SP1 is present, Help Viewer can use it to improve search performance and also allow wild-card searching. To take advantage of these capabilities, you can include SQL Server Compact 3.5 SP1 as part of your application's installation. For more information regarding SQL Server Compact 3.5 SP1 including redistribution rights, please visit:

<http://www.microsoft.com/Sqlserver/2008/en/us/compact.aspx>

# Command Line

## Information for Developers

Following is an explanation of how to use the command line options of the MadCap Help Viewer application. Use this method only if your application does not have access to the latest .NET framework. Otherwise, use either the HelpViewerClient API or HelpViewerEmbeddedClient API. See "HelpViewerClient API" on page 73 and "HelpViewerEmbeddedClient API" on page 75.

### Command Line Syntax

```
HelpViewer.exe -file <path to help system> [-cshid <identifier>] [-search <search_string>]
```

Parameter	Description
file	Path to the DotNet Help output file (.mchelp extension)
cshid	CSH ID of the desired topic to display. This can be either the identifier name or value. Alternatively, the ID may contain a topic path. When using a topic path, it must be relative to the Content folder of the Help system.
search	Search string that the Help system will automatically search for when it is opened

The following opens the specified Help system.

```
>HelpViewer.exe -file "C:\My Help System\Manual.mchelp"
```

The following opens the specified Help system and opens the topic associated with "MyID."

```
>HelpViewer.exe -file "C:\My Help System\Manual.mchelp" -cshid MyID
```

The following opens the specified Help system and opens the topic located at MyTopic.htm.

```
>HelpViewer.exe -file "C:\My Help System\Manual.mchelp" -cshid MyTopic.htm
```

The following opens the specified Help system and displays search results for "bikes and trikes."

```
>HelpViewer.exe -file "C:\My Help System\Manual.mchelp" -search "bikes and trikes"
```

The following opens the specified Help system, opens the topic associated with "MyID," and displays search results for "bikes and trikes."

```
>HelpViewer.exe -file "C:\My Help System\Manual.mchelp" -cshid MyID -search "bikes and trikes"
```



# HelpViewerClient API

Information for Developers

The `HelpViewerClient` API is used to make DotNet Help calls, including context-sensitive help (CSH) calls. `HelpViewerClient` only supports basic, non-embedded calls. It also does not provide Dynamic Help functionality. The `HelpViewerClient` API will open the Help Viewer in a separate process from your application.

If instead you require integrated Help, Dynamic Help, or opening the Help Viewer in the same process as your application, see "HelpViewerEmbeddedClient API" on page 75.

HOW TO USE THE HELP VIEWER CLIENT API

- 1. Create an instance of the class using the constructor.
- 2. Call **Load()** or **TryLoad()** to load the Help system of your choice.
- 3. Call **Search()** or **ShowTopic()**. The first time either of these two methods are called, a new Help Viewer window will be launched. Every call after that will open the desired topic or search results in the existing Help Viewer window.

Name	Description
<code>public void HelpViewerClient()</code>	Creates an instance of the <code>HelpViewerClient</code> .
<code>public void Load ( string helpsystem )</code>	Loads the Help system at the path specified in the <b>helpsystem</b> parameter. This method must be called once before calling <b>Search()</b> or <b>ShowTopic()</b> . If the Help system does not exist, an <b>ArgumentException</b> is thrown. See also <b>TryLoad()</b> .
<code>public void Search( string searchString )</code>	Performs a search for the string specified in the <b>searchString</b> parameter.
<code>public void Search( string searchString, string cshid )</code>	Performs a search for the string specified in the <b>searchString</b> parameter. The <b>cshid</b> parameter is used to override the default starting topic of the Help system. This can be either the identifier name or value. Alternatively, the ID may contain a topic path. When using a topic path, it must be relative to the Content folder of the Help system.

Name	Description
<code>public void ShowTopic ( string cshid )</code>	Opens the Help system with the topic specified in <b>cshid</b> . This can be either the identifier name or value. Alternatively, the ID may contain a topic path. When using a topic path, it must be relative to the Content folder of the Help system.
<code>public bool TryLoad( string helpsystem )</code>	See <b>Load()</b> . This method is equivalent to the Load() method. However, instead of throwing an exception, it returns true if the Help system was loaded successfully; it returns false otherwise.

## EXAMPLES

Refer to the sample application called "HelpViewerClientTester" for a fully functional application that uses the HelpViewerClient API. This sample application is available in the SDK, which you can download from:

<http://www.madcapsoftware.com/downloads/redistributables.aspx>

```
HelpViewerClient client = new HelpViewerClient();

if ( client.TryLoad( @"C:\My Project\Manual.mchelp" ) )
{
    client.ShowTopic( "MyID1" );
}
```

# HelpViewerEmbeddedClient API

Information for Developers

The `HelpViewerEmbeddedClient` API is used to make calls to the DotNet Help system, including context-sensitive help (CSH) calls. This API supports embedded CSH calls and can also provide Dynamic Help functionality. When you use this API, the MadCap Help Viewer opens in the same process as your application. If instead you require detached Help, opening the Help Viewer in a separate process from your application, see "HelpViewerClient API" on page 73.

HOW TO USE THE HELP VIEWER EMBEDDED CLIENT API

- 1. Create an instance of the class using the constructor.
- 2. Call **Load()** or **TryLoad()** to load the Help system of your choice.
- 3. Use the functionality of the Help System object. For more information see "IEmbeddedHelpSystem API" on page 79.

Name	Description
<code>public void HelpViewerEmbeddedClient()</code>	Creates an instance of the <code>HelpViewerEmbeddedClient</code> .
<code>public void HelpViewerEmbeddedClient( string title )</code>	Creates an instance of the <code>HelpViewerEmbeddedClient</code> . Sets the title of the Help system to the string specified in <b>title</b> .
<code>public void HelpViewerEmbeddedClient( string title, Icon icon )</code>	Creates an instance of the <code>HelpViewerEmbeddedClient</code> . Sets the title of the Help system to the string specified in <b>title</b> . Sets the icon of the help system to the icon specified in <b>icon</b> .
<code>public void Load( string helpsystem )</code>	Loads the Help system at the path specified in the <b>helpsystem</b> parameter. This method must be called once before calling <b>Search()</b> or <b>ShowTopic()</b> . If the Help system does not exist, an <b>ArgumentException</b> is thrown. See also <b>TryLoad()</b> .
<code>public bool TryLoad ( string helpsystem )</code>	See <b>Load()</b> . This method is equivalent to the <code>Load()</code> method. However, instead of throwing an exception, it returns true if the Help system was loaded successfully; it returns false otherwise.
<code>public IEmbeddedHelpSystem HelpSystem</code>	Gets an interface to the Help system. See "IEmbeddedHelpSystem API" on page 79 for more information.

Name	Description
<code>public virtual string Title</code>	Gets the title of the Help system.

## EXAMPLES

Refer to the sample application called "HelpViewerEmbeddedClientTester" for a fully functional application that uses the HelpViewerEmbeddedClient API. This sample application is available in the SDK, which you can download from:

<http://www.madcapsoftware.com/downloads/redistributables.aspx>

```
// Create a group box where the embedded help topic will be displayed

GroupBox mTopicPanel = new GroupBox();

mTopicPanel.Location = new System.Drawing.Point( 32, 204 );

mTopicPanel.Name = "mTopicPanel";

mTopicPanel.Text = "Help Topic";

// Display a topic embedded in the group box

HelpViewerEmbeddedClient client = new HelpViewerEmbeddedClient();

if ( client.TryLoad( @"C:\My Project\Manual.mchelp" ) )
{
    client.HelpSystem.DynamicHelpPanel = mTopicPanel;

    client.HelpSystem.LoadTopic( "MyID1" );
}
```

# ICSHIDProvider API

## Information for Developers

The ICSHIDProvider API is used to enable a System.Windows.Forms.Control to be a context-sensitive Help (CSH) ID provider. This is required to incorporate any of the following.

- » **Dynamic Help** This is a feature where a Help window automatically displays topics from a Help system that are relevant to the context as an individual uses an application. For example, if the user clicks on an area of your UI called "Project Organizer," the topic pertinent to the Project Organizer opens. You can see an example of Dynamic Help in Flare's online Help (select **Help>Dynamic Help**).
- » **F1 Help** This is a feature where the Help opens when a user presses the F1 key.
- » **Dialog's Help Button** This is a feature where the Help opens when a user clicks the dialog's Help button (i.e., the "question mark" button).

If your application uses any of these features, the DotNet CSH engine uses this interface to determine the desired topic to display. As an alternative to implementing this interface, you may also set the control's **Tag** property to the desired CSH ID string. In this case, you must prefix the string with "CSH:"

EXAMPLE

For example, if your CSH ID is "MyHelpTopic," you would set the Tag object to the string "CSH:MyHelpTopic"

Name	Description
<code>public string HelpSystemCSHID</code>	Gets the CSH ID of the topic that should be displayed in the Help system.

The following shows how to implement the ICSHIDProvider interface.

```
public class MyTextBox : TextBox, ICSHIDProvider
{
    private string mHelpSystemCSHID = "TestID1";

    public string HelpSystemCSHID
    {
        get { return mHelpSystemCSHID; } }

    ...
}
```

The following shows how to set the Tag object.

```
TextBox myTextBox = new TextBox();

myTextBox.Tag = "CSH:TestID1";
```

## IEmbeddedHelpSystem API

### Information for Developers

The IEmbeddedHelpSystem API is an interface that exposes functionality for context-sensitive Help (CSH), embedded Help, and Dynamic Help.

#### Detached CSH

To launch the Help system in a separate Help Viewer window, use the **ShowHelp()** method.

To make a CSH call and have the results displayed in a separate Help Viewer window, use the **ShowTopic()** method.

To display search results from a Help system in a separate Help Viewer window, use the **Search()** method.

These methods will launch an instance of the Help Viewer application. It will be launched in the same process as your application so when the user closes your application, the Help Viewer will also be closed.

#### Embedded CSH

To embed a topic from your Help system into your application, use the **LoadTopic()** method.

To embed search results from your Help system into your application, use the **EmbeddedSearch()** method.

You must set the **DynamicHelpPanel** property prior to calling **LoadTopic()**. Similarly, you must set the **SearchPanel** property prior to calling **EmbeddedSearch()**.

### EXAMPLES

Refer to the sample application called "HelpViewerEmbeddedClientTester." This sample application is available in the SDK, which you can download from <http://www.mad-capsoftware.com/downloads/utilities/redistributables.aspx>.

#### Dynamic Help

Dynamic Help is a feature where a Help window automatically displays topics from a Help system that are relevant to the context as an individual uses an application. For example, if the user clicks on an area of your UI called "Project Organizer," the topic pertinent to the Project Organizer opens. You can see an example of Dynamic Help in Flare's online Help (select **Help>Dynamic Help**).

To use this functionality, you must set the **EnableDynamicHelp** property to true. The CSH engine needs a way to determine the appropriate topic to display. Your application can supply this information in either of two ways.

- » One way is to implement the **ICSHIDProvider** interface. For more information see "ICSHIDProvider API" on page 77.
- » The other way is to set the control's **Tag** property to the desired CSH ID string. In this case, you must prefix the string with "CSH:"

For example, if your CSH ID is "MyHelpTopic," you would set the Tag object to the string "CSH:MyHelpTopic."

Name	Description
<code>public void EmbeddedSearch ( string searchString )</code>	Performs a search for the string specified in the <b>searchString</b> parameter. Displays the results embedded in the control that is set as the <b>SearchPanel</b> property.
<code>public void EnableHelpButton ( System.Windows.Forms.Form form )</code>	Enables the Help button in the form specified in <b>form</b> . See "ICSHIDProvider API" on page 77 if you are enabling this feature.
<code>public void LoadTopic ( string cshid )</code>	Loads the topic specified in <b>cshid</b> . This can be either the identifier name or value. Alternatively, the ID may contain a topic path. When using a topic path, it must be relative to the Content folder of the Help system. Displays the topic embedded in the control that is set as the <b>DynamicHelpPanel</b> property.
<code>public void Search ( string searchString )</code>	Performs a search for the string specified in the <b>searchString</b> parameter. Displays the results in a new Help Viewer window.
<code>public void Search ( string searchString, string cshid )</code>	Performs a search for the string specified in the <b>searchString</b> parameter. The <b>cshid</b> parameter is used to override the default starting topic of the Help system. This can be either the identifier name or value. Alternatively, the ID may contain a topic path. When using a topic path, it must be relative to the Content folder of the Help system. Displays the results in a new Help Viewer window.
<code>public void ShowHelp ()</code>	Opens the Help system in a new Help Viewer window.
<code>public void ShowTopic ( string cshid )</code>	Opens the Help system with the topic specified in <b>cshid</b> . This can be either the identifier name or value. Alternatively, the ID may contain a topic path. When using a topic path, it must be relative to the Content folder of the Help system.
<code>System.Windows.Forms.Control DynamicHelpPanel</code>	Gets or sets the control where the embedded Help topic will be loaded. This must be set before calling <b>LoadTopic()</b> .



Name	Description
<code>bool EnableDynamicHelp</code>	Gets or sets whether Dynamic Help is enabled for the form. See "ICSHIDProvider API" on page 77 if you are enabling this feature.
<code>bool EnableF1Help</code>	Gets or sets whether F1 Help is enabled for the form. See "ICSHIDProvider API" on page 77 if you are enabling this feature.
<code>System.Windows.Forms.Control SearchPanel</code>	Gets or sets the control where the embedded Help search results will be loaded. This must be set before calling <b>EmbeddedSearch()</b> .

## CSH Calls for HTML Help

You can work with your developer to map an HTML Help system to a software application. The developer can either connect the application directly to the HTML Help system or use context-sensitive Help (CSH) calls to map specific Help topics to the specific user interface elements:

- » **Basic Help** The developer can connect the application to your basic HTML Help output, rather than to a specific topic. The Help will open in the HTML Help viewer, displaying the startup topic that you designate and the navigation elements that you include.
- » **Context-sensitive Help (CSH)** The developer can use CSH to connect the application to specific topics in the HTML Help output (as long as you have created CSH in your Flare project and share the header file information with the developer).

### What You Need to Do

1. Work with your developer to determine how you want to connect the Help to the application (basic Help, CSH, or both).
2. Create and build your Microsoft HTML Help project in Flare.
3. Provide the developer with the output files, as well as the CSH header file (if you plan to use CSH).
4. Provide the developer with the information in the following topic: "CSH Calls for HTML Help—Developers" on the following page.

## CSH Calls for HTML Help—Developers

### Information for Developers

Use the following information to connect Microsoft HTML Help to an application:

- » Use the following syntax to call a topic using a map number:

```
HWND HtmlHelp(Window(), "c:\path\MyHelp.chm", HH_HELP_CONTEXT, Number);
```

- » Use the following syntax to call a topic using a file name:

```
HWND HtmlHelp (Window(), "c:\path\MyHelp.chm", HH_DISPLAY_TOPIC, "welcome.htm");
```

To hook context-sensitive Help (CSH) to an application, the code looks something like this:

```
HtmlHelp(hWnd, /*Window handle of program or dialog*/  
"CSHHelp.chm", /*Name of the CHM file*/  
HH_HELP_CONTEXT,  
dwMapNumber); /*Number from header file*/
```

Name	Description
<b>HtmlHelp (hWnd)</b>	This is the window handle of a program or dialog. A window handle helps identify a window so the HTML Help engine knows which application is performing the specific action.
<b>MyHelp.chm</b>	This is the name of a compiled HTML Help file (CHM) that includes the CSH. The actual name of the CHM file is determined by the Help author.
<b>HH_HELP_CONTEXT</b>	This is the command sent to the HTML Help engine for window-level Help.
<b>dwMapNumber)</b>	A map number from the header file.

## CSH Calls for HTML5 Output

You can work with your developer (or you can function as the developer yourself) to connect an HTML5 system to a software application or to open specific parts of your online documentation from a simple web link. The application or web link (s) can be connected to the basic Help output, to specific topics in the output (context-sensitive Help), or both.

- » **Basic Help** The developer can connect the application or web link to your basic HTML5 output, rather than to a specific topic. The Help will open in the browser window, displaying the startup topic that you designate and the navigation elements that you include.
- » **Context-sensitive Help (CSH)** The developer can use CSH to connect the application or web link(s) to specific topics in the HTML5 output (as long as you have created CSH in your Flare project and share the header file information with the developer).

### What You Need to Do

1. Work with your developer to determine how you want to connect the online output to the application or web link (s).
2. Create and build your HTML5 target in Flare.
3. Provide the developer with the output files, as well as the CSH header file.
4. Provide the developer with the information in the following topic: "CSH Calls for HTML5 Output—Developers" on the next page.

## CSH Calls for HTML5 Output—Developers

### Information for Developers

Use the following information if you are producing HTML5 and want to incorporate context-sensitive Help (CSH) into the software application.

There are two methods you can use.

- » **Method 1—JavaScript** Using this method requires calling a JavaScript function that Flare provides.
- » **Method 2—URL** Using this method, you can create a hyperlink to launch the Help system.

### Which Method is Best for You?

Each method has its unique benefits. Generally speaking, the JavaScript method lets you have more control, whereas the URL method is a bit more quick and simple.

One reason to choose the JavaScript method is to better control the window size and location. With the URL method, the browser window automatically starts to open at the same size and location as the previous time that browser window was opened. But if you have specified a different size and location for your output window, the window will visibly move and resize accordingly. The JavaScript method prevents this type of behavior by opening the window directly to the size and location you specified. You would set the window size and location in the skin. Then in the JavaScript call you would specify the appropriate skin.

Another benefit to using the JavaScript method is that it is required in order for the Browser Settings option to take effect. This option can be found on the Setup tab of the Skin Editor.

#### HOW TO USE JAVASCRIPT TO OPEN HTML5 OUTPUT

1. **Author** Add a header file to your project.
2. **Author** Add an alias file to your project.
3. **Author** In the Alias Editor, create and assign an identifier (ID) for the topic that you want to link to from the CSH call.

For example, if you have a topic called "Welcome.htm," you might create a new ID and also name it "Welcome." Then you might assign a value of 1000 to it. You would make sure that the topic Welcome.htm is assigned to the ID.

Finally, an optional step in the Alias Editor is to assign a skin to that ID. However, you can bypass this step and specify a skin later when you create the actual JavaScript, or you can choose not to select a skin at all.

4. **Author** Build your Help system using an HTML5 target and publish the output files to the final destination.

5. **Developer** Add a reference to the JavaScript file (which is created automatically when the author builds the output). This .js file should be named "csh.js." The reference to the JavaScript file should use the following format: `<script type="text/javascript" src="[path of file]/csh.js"></script>`.



**Note:** Make sure you use forward slashes (/) in the src path to the file, even if the file is referenced locally.

6. **Developer** Create a trigger and add the command to let users open the Help system. Here is a format that you can use to add a button.

```
<input type="button" value="Open Help System" onclick="MadCap.OpenHelp(ID, skin
name, search string, first pick search string value);" />
```

You can change the input type and the value if necessary. The most important parts that you will adjust are the elements within parentheses ID, skin name, search string, first pick search string value).

- » **ID** This is the CSH ID that the author created in Flare (see Step 3 above). This can be either the ID name or value. The topic and skin associated with the ID will be used. If no skin is associated with the ID in Flare, the skin name that you provide in this command will be used. Alternatively, the ID may contain a topic path. In this case, the specified topic will be loaded with the skin that is specified in this command. The topic path must be relative to the Content folder of the Flare project. You also have the option of entering "null" instead of an ID to use the Help system's default starting topic.
- » **Skin Name** This is the name of the skin to use when opening the Help system. If a skin has been assigned to the ID in Flare (see Step 3 above) *and* you enter a skin name in this command, the skin name in the command will take precedence. You also have the option of entering "null" instead of a skin name if you want to use the Help system's default skin or to use the skin that is associated with the CSH ID in Flare.
- » **Search String** This is an optional element that automatically performs a search for a specific string.
- » **First Pick Search String Value** This element can be used in conjunction with the search string. If you use the first pick option, you can include a true or false value. If the value is true, the first topic found with the specified search string will be opened automatically. If the value is false, the search results will simply be displayed; the first topic will not be opened automatically.

In the following example, the topic and skin associated with "Welcome" will be used. No search string information is included.

```
<input type="button" value="Open Help System" onclick="MadCap.OpenHelp('Welcome',
null, null, null);" />
```

In the following example, the topic associated with "Welcome" will be used. "BlueSkin" will override the skin associated with "Welcome." No search string information is included.

```
<input type="button" value="Open Help System" onclick="MadCap.OpenHelp('Welcome',  
'BlueSkin', null, null );" />
```

In the following example, the topic and skin associated with the ID value 1000 will be used. No search string information is included.

```
<input type="button" value="Open Help System" onclick="MadCap.OpenHelp(1000, null,  
null, null );" />
```

In the following example, the topic associated with the ID value 1000 will be used. "BlueSkin" will override the skin associated with ID value 1000. No search string information is included.

```
<input type="button" value="Open Help System" onclick="MadCap.OpenHelp(1000,  
'BlueSkin', null, null );" />
```

In the following example, "Company/Employees.htm" will be used with the default skin. No search string information is included.

```
<input type="button" value="Open Help System" onclick="MadCap.OpenHelp('Com-  
pany/Employees.htm', null, null, null );" />
```

In the following example, both the default topic and skin will be used. A search will automatically be performed for the words "quarterly report," but the first topic found in the search *will not* be opened automatically.

```
<input type="button" value="Open Help System" onclick="MadCap.OpenHelp(null, null,  
'quarterly report', false );" />
```



In the following example, the default topic will be used with "BlueSkin." A search will automatically be performed for the words "quarterly report," and the first topic found in the search *will* be opened automatically.

```
<input type="button" value="Open Help System" onclick="MadCap.OpenHelp(null,
'BlueSkin', 'quarterly report', true );" />
```

#### HOW TO USE A URL TO OPEN HTML5 OUTPUT

1. **Author** (Optional) Add a header file to your project.
2. **Author** (Optional) Add an alias file to your project.
3. **Author** (Optional) In the Alias Editor, create and assign IDs for the topics to which you want to provide links. If you do not want to create a header file, alias file, and IDs for topics, you can instead use the file names for the topics to which you want to link.

An optional step in the Alias Editor is to assign a skin to that ID. However, you can bypass this step and specify a skin later when you create the actual link, or you can choose not to select a skin at all.

4. **Author** Build your Help system using an HTML5 target and publish the output files to the final destination.
5. **Developer** Create a trigger and add a link to let users open a specific area of the Help system.

There is a certain amount of flexibility in terms of how you create the link and what you can include in it. Here is the basic structure of the link.

```
[main entry file].htm#cshid=[ID number, ID name, or topic path/name]
&searchQuery=[search string]&firstPick=true&skinName=[skin name]
```

After the hash tag (#), you can specify any combination of the parameters (cshid, searchQuery, firstPick, skinName), separated by ampersands (&). The order of the parameters does not matter.

- » **Main entry file** Provide the path to the main entry file for your output. The file name is determined by whatever you enter into the **Output File** field in the **General** tab of the Target Editor. If you do not provide a name in this field, the name "Default" will be used.
- » **cshid** This is the CSH ID that you created in Flare (see Step 3 above). This can be either the identifier name or value. The topic and skin that is associated with the ID will be used (unless you override it in this link by specifying a different skin name). Alternatively, you can enter the path and name of the specific topic to which you want to link. If you use this element, you do not need to create an ID as described above. The topic path must be relative to the Content folder of the Flare project.
- » **searchQuery** This is an optional element that automatically performs a search for a specific string.
- » **firstPick** This element can be used in conjunction with the search string. If you include the first pick option, the first topic found with the specified search string *will* be opened automatically. If you do not include this element, the search results will simply be displayed; the first topic *will not* be opened automatically.
- » **skinName** If you use a map ID in the link, the skin associated with that ID (if any) will be used to display the Help. However, you can specify a skin directly in the link, which will override the skin associated with that topic in the alias file.

## EXAMPLES

```
<a href="http://my.mycompany.com/Default.htm#cshid=1000&searchQuery=World Cup Standings&firstPick=true&skinName=Green">Click here to open</a>
```

```
<a href="http://my.mycompany.com/Default.htm#cshid=Soccer&searchQuery=World Cup Standings&firstPick=true&skinName=Green">Click here to open</a>
```

```
<a href="http://my.mycompany.com/Default.htm#cshid=Soccer.htm&searchQuery=World Cup Standings&firstPick=true&skinName=Green">Click here to open</a>
```

In these examples, the following were used.

- » Default.htm = main entry file name
- » 1000 = CSH ID value
- » Soccer = CSH ID name
- » Soccer.htm = topic in the project, at the root level of the Content Explorer
- » World Cup Standings = search term
- » Green = skin name

## CSH Calls for WebHelp

You can work with your developer to connect a WebHelp or WebHelp Plus system to a software application or to open specific parts of your online documentation from a simple web link. The application or web link(s) can be connected to the basic Help output, to specific topics in the output (context-sensitive Help), or both.

- » **Basic Help** The developer can connect the application or web link to your basic WebHelp or WebHelp Plus output, rather than to a specific topic. The Help will open in the browser window, displaying the startup topic that you designate and the navigation elements that you include.
- » **Context-sensitive Help (CSH)** The developer can use CSH to connect the application or web link(s) to specific topics in the WebHelp or WebHelp Plus output (as long as you have created CSH in your Flare project and share the header file information with the developer).

### What You Need to Do

1. Work with your developer to determine how you want to connect the online output to the application or web link(s).
2. Create and build your WebHelp or WebHelp Plus target in Flare.
3. Provide the developer with the output files, as well as the CSH header file.
4. Provide the developer with the information in the following topic: "CSH Calls for WebHelp and WebHelp Plus—Developers" on the next page .

## CSH Calls for WebHelp and WebHelp Plus—Developers

### Information for Developers

Use the following information if you are producing WebHelp or WebHelp Plus and want to incorporate context-sensitive Help (CSH) into the software application.

There are two methods you can use.

- » **Method 1—JavaScript** Using this method requires calling a JavaScript function that Flare provides.
- » **Method 2—URL** Using this method, you can create a hyperlink to launch the Help system.

### Which Method is Best for You?

Each method has its unique benefits. Generally speaking, the JavaScript method lets you have more control, whereas the URL method is a bit more quick and simple.

One reason to choose the JavaScript method is to better control the window size and location. With the URL method, the browser window automatically starts to open at the same size and location as the previous time that browser window was opened. But if you have specified a different size and location for your output window, the window will visibly move and resize accordingly. The JavaScript method prevents this type of behavior by opening the window directly to the size and location you specified. You would set the window size and location in the skin. Then in the JavaScript call you would specify the appropriate skin.

Another benefit to using the JavaScript method is that it is required in order for the Browser Settings option to take effect. This option can be found on the WebHelp Setup tab of the Skin Editor.

### HOW TO USE JAVASCRIPT TO OPEN WEBHELP OR WEBHELP PLUS

1. **Author** Add a header file to your project.
2. **Author** Add an alias file to your project.
3. **Author** In the Alias Editor, create and assign an identifier (ID) for the topic that you want to link to from the CSH call.

For example, if you have a topic called "Welcome.htm," you might create a new ID and also name it "Welcome." Then you might assign a value of 1000 to it. You would make sure that the topic Welcome.htm is assigned to the identifier.

Finally, an optional step in the Alias Editor is to assign a skin to that ID. However, you can bypass this step and specify a skin later when you create the actual JavaScript, or you can choose not to select a skin at all.

4. **Author** Build your Help system using a WebHelp or WebHelp Plus target and publish the output files to the final destination.
5. **Developer** Add a reference to the JavaScript file (which is created automatically when the author builds the output). This .js file is named after the WebHelp or WebHelp Plus output file and placed at the root level of the output folder. For example, if the output file is named "MyFirstHelp.htm," the JavaScript file is called "MyFirstHelp.js." The reference to the JavaScript file should use the following format: `<script type="text/javascript" src="path and name of file.js"></script>`.



**Note:** Make sure you use forward slashes (/) in the src path to the file, even if the file is referenced locally.

6. **Developer** Create a trigger and add the command to let users open the Help system. Here is a format that you can use to add a button.

```
<input type="button" value="Open Help System" onclick="FMCOpenHelp(ID, skin
name, search string, first pick search string value);" />
```

You can change the input type and the value if necessary. The most important parts that you will adjust are the elements within parentheses (ID, skin name, search string, first pick search string value).

- » **ID** This is the CSH ID that the author created in Flare (see Step 3 above). This can be either the identifier name or value. The topic and skin associated with the ID will be used. If no skin is associated with the ID in Flare, the skin name that you provide in this command will be used. Alternatively, the ID may contain a topic path. In this case, the specified topic will be loaded with the skin that is specified in this command. The topic path must be relative to the Content folder of the Flare project. You also have the option of entering "null" instead of an ID to use the Help system's default starting topic.
- » **Skin Name** This is the name of the skin to use when opening the Help system. If a skin has been assigned to the ID in Flare (see Step 3 above) *and* you enter a skin name in this command, the skin name in the command will take precedence. You also have the option of entering "null" instead of a skin name if you want to use the Help system's default skin or to use the skin that is associated with the CSH ID in Flare.
- » **Search String** This is an optional element that automatically performs a search for a specific string.
- » **First Pick Search String Value** This element can be used in conjunction with the search string. If you use the first pick option, you can include a true or false value. If the value is true, the first topic found with the specified search string will be opened automatically. If the value is false, the search results will simply be displayed; the first topic will not be opened automatically.

In the following example, the topic and skin associated with "Welcome" will be used. No search string information is included.

```
<input type="button" value="Open Help System" onclick="FMCOpenHelp('Welcome', null, null, null );" />
```

In the following example, the topic associated with "Welcome" will be used. "BlueSkin" will override the skin associated with "Welcome." No search string information is included.

```
<input type="button" value="Open Help System" onclick="FMCOpenHelp('Welcome', 'BlueSkin', null, null );" />
```

In the following example, the topic and skin associated with the ID value 1000 will be used. No search string information is included.

```
<input type="button" value="Open Help System" onclick="FMCOpenHelp(1000, null, null, null );" />
```

In the following example, the topic associated with the ID value 1000 will be used. "BlueSkin" will override the skin associated with ID value 1000. No search string information is included.

```
<input type="button" value="Open Help System" onclick="FMCOpenHelp(1000, 'BlueSkin', null, null );" />
```

In the following example, "Company/Employees.htm" will be used with the default skin. No search string information is included.

```
<input type="button" value="Open Help System" onclick="FMCOpenHelp('Company/Employees.htm', null, null, null );" />
```

In the following example, both the default topic and skin will be used. A search will automatically be performed for the words "quarterly report," but the first topic found in the search *will not* be opened automatically.

```
<input type="button" value="Open Help System" onclick="FMCOpenHelp(null, null, 'quarterly report', false );" />
```

In the following example, the default topic will be used with "BlueSkin." A search will automatically be performed for the words "quarterly report," and the first topic found in the search *will* be opened automatically.

```
<input type="button" value="Open Help System" onclick="FMCOpenHelp(null, 'BlueSkin', 'quarterly report', true );" />
```

#### HOW TO USE A URL TO OPEN WEBHELP OR WEBHELP PLUS

1. **Author** (Optional) Add a header file to your project.
2. **Author** (Optional) Add an alias file to your project.
3. **Author** (Optional) In the Alias Editor, create and assign IDs for the topics to which you want to provide links. If you do not want to create a header file, alias file, and IDs for topics, you can instead use the file names for the topics to which you want to link.

An optional step in the Alias Editor is to assign a skin to that ID. However, you can bypass this step and specify a skin later when you create the actual link, or you can choose not to select a skin at all.

4. **Author** Build your Help system using a WebHelp or WebHelp Plus target and publish the output files to the final destination.
5. **Developer** Create a trigger and add a link to let users open a specific area of the Help system.

There is a certain amount of flexibility in terms of how you create the link and what you can include in it. Here is the basic structure of the link.

```
[main entry file]_CSH.htm?[search string]|FirstPick#[ID or topic file name.htm]  
|[skin name]
```

- » **Main Entry file\_CSH.htm** This is the main entry file for your output. The file name is determined by whatever you enter into the **Output File** field in the **General** tab of the Target Editor. If you do not provide a name in this field, the name "Default" will be used. After the file name, it is important that you add an underscore followed by "CSH."



**Note:** You do not need to include the "\_CSH" portion in the file name of the topic in Flare. You only need to add "\_CSH" to the web page hyperlink connecting to that topic.

- » **Search String** This is an optional element that automatically performs a search for a specific string.
- » **First Pick** This element can be used in conjunction with the search string. If you include the first pick option, the first topic found with the specified search string will be opened automatically. If you do not include this element, the search results will simply be displayed; the first topic will not be opened automatically.
- » **ID** This is the CSH ID that you created in Flare (see Step 3 above). This can be either the ID name or value. The topic and skin that is associated with the ID will be used, unless you override it in this link by specifying a different skin name. If you do not want to use a map ID, you can use the topic file name (see below).
- » **Topic Name** This is the name of the specific topic to which you want to link. If you use this element, you do not need to create an ID as described above. The topic path must be relative to the Content folder of the Flare project. If you do not want to use the topic name, you can use an ID (see above).
- » **Skin Name** If you use an ID in the link, the skin associated with that ID (if any) will be used to display the Help. However, you can specify a skin directly in the link, which will override the skin associated with that topic in the alias file.

## EXAMPLES

```
<a href="http://my.mycompany.com/Default_CSH.htm#Soccer|Green">Click here to open</a>
```

```
<a href="http://my.mycompany.com/Default_CSH.htm#1000">Click here to open</a>
```

```
<a href="http://my.mycompany.com/Default_CSH.htm#Soccer.htm">Click here to open</a>
```

```
<a href="http://my.mycompany.com/Default_CSH.htm?World Cup Standings#1000">Click here to open</a>
```

```
<a href="http://my.mycompany.com/Default_CSH.htm?World Cup Standings|FirstPick#Soccer">Click here to open</a>
```

In these examples, the following were used.

- » Default.htm = main entry file name
- » 1000 = CSH ID value



- » Soccer = CSH ID name
- » Soccer.htm = topic in the project, at the root level of the Content Explorer
- » World Cup Standings = search term
- » Green = skin name

## CSH Calls for WebHelp Mobile

You can work with your developer (or you can function as the developer yourself) to open specific parts of your WebHelp Mobile documentation from web links. In order to accomplish this, you need to have created context-sensitive Help (CSH) in your Flare project and share the header file information with the developer.

### What You Need to Do

1. Work with your developer to determine how you want to connect the mobile output to the web link(s).
2. Create and build your WebHelp Mobile target in Flare.
3. Provide the developer with the output files, as well as the CSH header file.
4. Provide the developer with the information in the following topic: "CSH Calls for WebHelp Mobile—Developers" on the following page.

## CSH Calls for WebHelp Mobile—Developers

### Information for Developers

Use the following information if you are producing WebHelp Mobile and want to incorporate context-sensitive Help (CSH). This lets users open specific parts of the output from web links.

#### HOW TO CREATE CSH CALLS FOR WEBHELP MOBILE

1. **Author** (Optional) Add a header file to your project.
2. **Author** (Optional) Add an alias file to your project.
3. **Author** (Optional) In the Alias Editor, create and assign IDs for the topics that you want to provide links to. If you do not want to create a header file, alias file, and IDs for topics, you can instead use the file names for the topics to which you want to link.
4. **Author** Build your Help system using a WebHelp Mobile target and publish the output files to the final destination.
5. **Developer** Create a trigger and add a link to let users open a specific area of the Help system.

There is a certain amount of flexibility in terms of how you create the link and what you can include in it. Here is the basic structure of the link.

```
[main entry file]?cshid=[ID number, ID name, or topic path/name]&searchQuery=[search string]&firstPick=true
```

After the question mark, you can specify any combination of the parameters (cshid, searchQuery, firstPick), separated by ampersands (&). The order of the parameters does not matter.

- » **Main entry file** Provide the path to the main entry file for your output. The file name is determined by whatever you enter into the **Output File** field in the **General** tab of the Target Editor. If you do not provide a name in this field, the name "Default" will be used.
- » **cshid** This is the CSH ID that you created in Flare (see Step 3 above). This can be either the identifier name or value. Alternatively, you can enter the path and name of the specific topic to which you want to link. If you use this element, you do not need to create an ID as described above. The topic path must be relative to the Content folder of the Flare project.
- » **searchQuery** This is an optional element that automatically performs a search for a specific string.
- » **firstPick** This element can be used in conjunction with the search string. If you include the first pick option, the first topic found with the specified search string *will* be opened automatically. If you do not include this element, the search results will simply be displayed; the first topic *will not* be opened automatically.

## EXAMPLES

```
<a href="http://mycompany.com/Default.htm?cshid=1000&searchQuery=World Cup Standings&firstPick=true">Click here to open</a>
```

```
<a href="http://mycompany.com/Default.htm?cshid=Soccer&searchQuery=World Cup Standings&firstPick=true">Click here to open</a>
```

```
<a href="http://mycompany.com/Default.htm?cshid=Soccer.htm&searchQuery=World Cup Standings&firstPick=true">Click here to open</a>
```

In these examples, the following were used.

- » Default.htm = main entry file name
- » 1000 = CSH ID value
- » Soccer = CSH ID name
- » Soccer.htm = topic in the project, at the root level of the Content Explorer
- » World Cup Standings = search term

## PDF Guides

The following PDF guides are available for download from the online Help.

*Accessibility Guide*

*Analyzer Guide*

*Autonumbers Guide*

*Condition Tags Guide*

*Context-sensitive Help Guide*

*DotNet Help Guide*

*Eclipse Help Guide*

*Getting Started Guide*

*Global Project Linking Guide*

*HTML Help Guide*

*HTML5 Guide*

*Images Guide*

*Importing Guide*

*Index Guide*

*Key Features Guide*

*Language Support Guide*

*Movies Guide*

*Navigation Links Guide*

*Print-based Output Guide*

*Project Creation Guide*

*Pulse Guide*

*QR Codes Guide*

*Reports Guide*

*Reviews & Contributions Guide*

*Search Guide*

*SharePoint Guide*

*Shortcuts Guide*

*Skins Guide*



*Snippets Guide*

*Source Control Guide*

*Styles Guide*

*Tables Guide*

*Tables of Contents Guide*

*Targets Guide*

*Templates Guide*

*Topics Guide*

*Touring the Workspace Guide*

*Transition From FrameMaker Guide*

*Variables Guide*

*WebHelp Outputs Guide*

*What's New Guide*