AXCM Plugin Tutorial (Classic Schemes)

West Street Consulting

This tutorial is designed to give you a quick hands-on introduction to the AXCM plugin. It covers some of the basic functionality and should familiarize you with the core features that the software provides.

Before using this tutorial, note the following:

- This tutorial explores the "classic" scheme functionality that the software has supported since its inception. There is a different tutorial that explores the new XPath-based scheme functionality introduced with version 2.0. **IMPORTANT**: You should have an understanding of the distinction between the two, especially if you intend to play a leading role in the implementation of AXCM at your organization. See the *User Guide* for more information.

- The tutorial is designed for use with the sample files that install with AXCM. The instructions in this tutorial assume that the files have not been altered since the installation. However, you will be altering them during the course of the tutorial. If you or someone else will need the original files afterwards, consider making copies of the sample files and performing the tutorial on the copies. The files do not need to be in any particular location for this tutorial to work.

- The tutorial is designed to work with the sample schemes that installed with AXCM. These schemes are contained in the sample main settings file that is provided with AXCM. If your preferences are pointing to some main settings file that does not contain this set of sample schemes, the tutorial cannot be completed. In this case, please see the AXCM documentation or contact someone with AXCM experience for help. If you have just installed AXCM and have not altered your preferences, you should be fine.

- If you would like to experiment with tabular scheme data in "configuration files," all schemes that are used in this tutorial are also found in the sample configuration file that installs with AXCM: `AXCM_Sample_Config_File.fm`. You will need to interpolate the instructions as appropriate for configuration file usage. For more information on configuration files, see your *User Guide*.

- Some of the screen shots in this tutorial still display the previous name of the plugin, "ABCM." The screenshots should otherwise be accurate.

Part 1 – Setting up your "scratch" category

The main settings file that installs with AXCM includes a "Samples" category that contains scheme data for use with this tutorial. In the following steps, you will make a copy of this category which will be used for the remainder of the tutorial. By making a copy, you will be able to edit schemes without affecting the original scheme data that came with the plugin.

1. Open the classic scheme editor (**AXCM > Main Settings Configuration > "Classic" Schemes**).
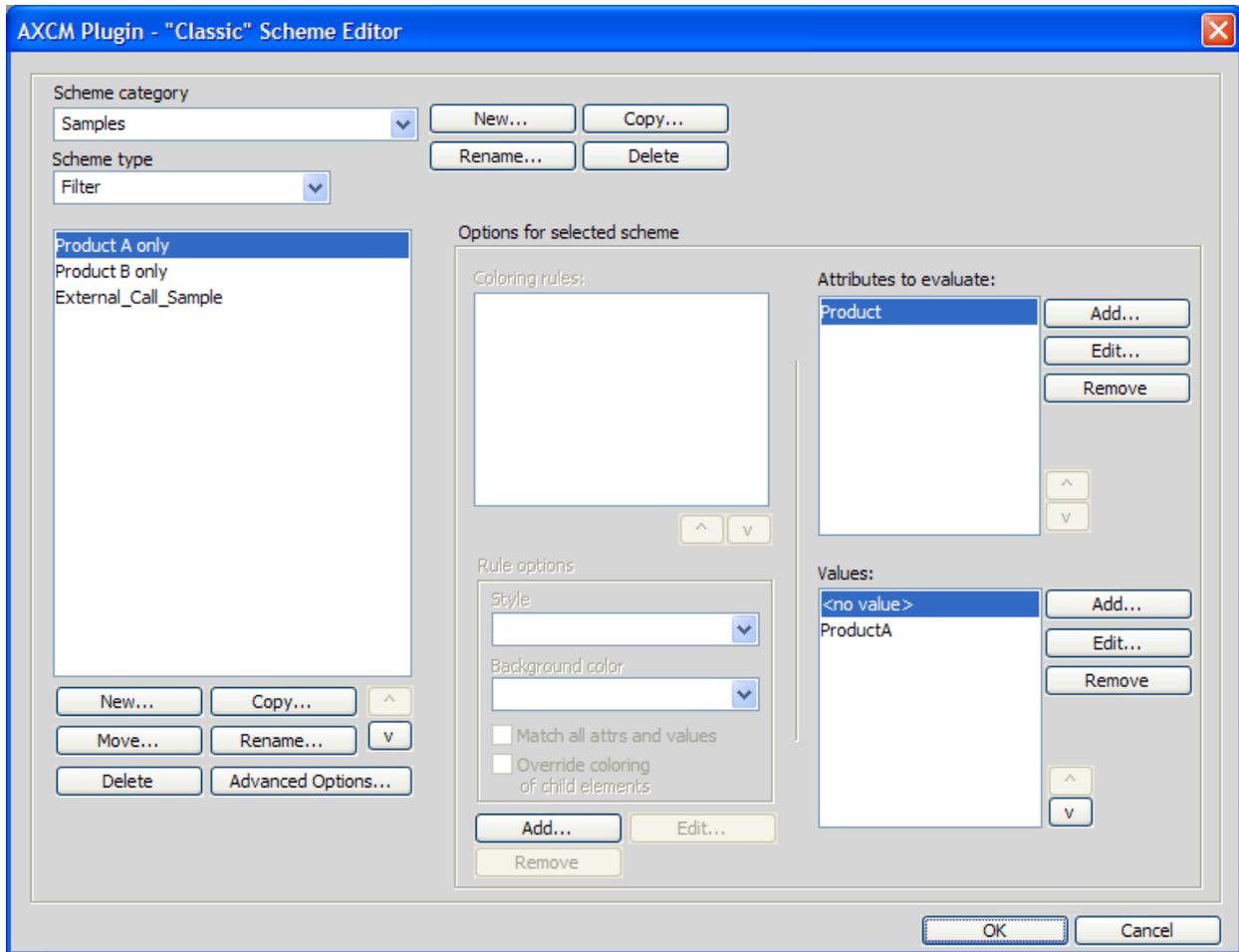
A majority of the work you do in this tutorial will focus on this dialog box. In the following steps, you will make one small change to prepare for the tutorial. Later, you will return to this dialog box to explore it in more detail.

2. Under **Scheme category**, select **Samples**.
3. To the right of the scheme category drop-down, click **Copy**.
4. In the new category prompt, type the name **Tutorial**, and click **OK**.
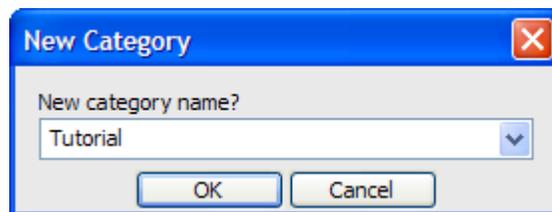
5. Note that the **Scheme category** list now contains **Samples** and **Tutorial**, which are currently exact duplicates.
6. Click **OK** to close the scheme editor.

Part 2 - Simple conditions – Coloring and filtering

Assume that you have a FrameMaker book that you use to generate a manual for both Product A and Product B. You want to use conditional text to tag information specific to one product or the other, and you've chosen to use

structured FrameMaker and AXCM for this. A conditional setup such as this is the most simple, using a single attribute and two separate values.

AXCM includes a sample file with this type of setup, using the `Product` attribute to denote the product condition with the values "ProductA" and "ProductB". Note that this is an example only, and that you may use any attributes and values you want to denote conditions.

1. Open the sample file `AXCM_Sample1_TwoSimpleConditions.fm` and close all other open files.
2. Open the Structure View window.
3. Note the use of the Product attribute to denote conditions, and how the text in the document is written to help you see where these conditions are.
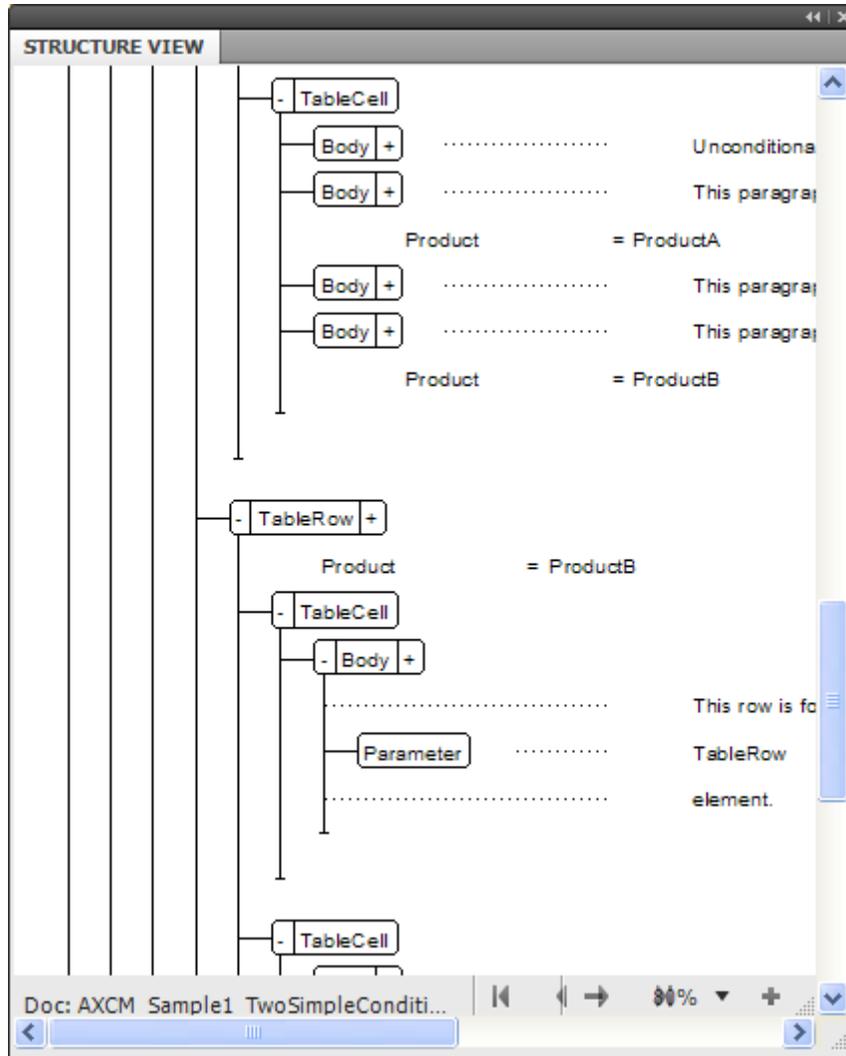
4. Select **AXCM > Coloring > Color Document**.
5. Select the **Tutorial** category, the **Product A vs. Product B** scheme, and click **OK**.
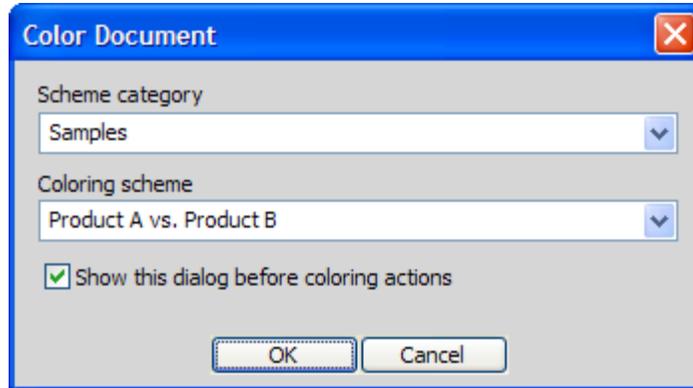
6. Note the coloring that was applied.
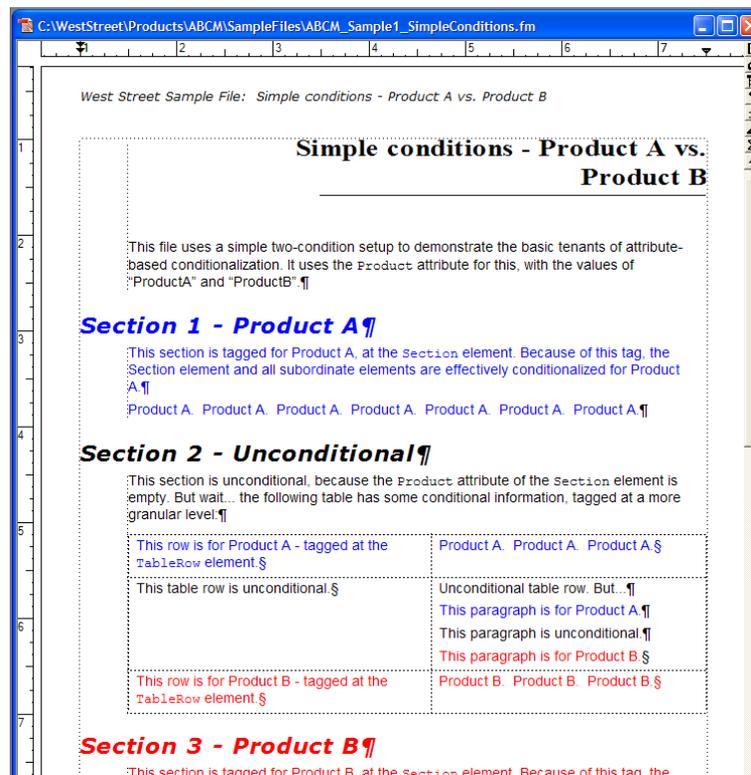
The coloring you see is defined in the scheme that you selected. The scheme setup will be examined in more detail later.

7. Select **AXCM > Filtering > Filter Document**.

8. Set up the Filter dialog as shown below, with the **Tutorial** category, the **Product A only** scheme, and the other settings.
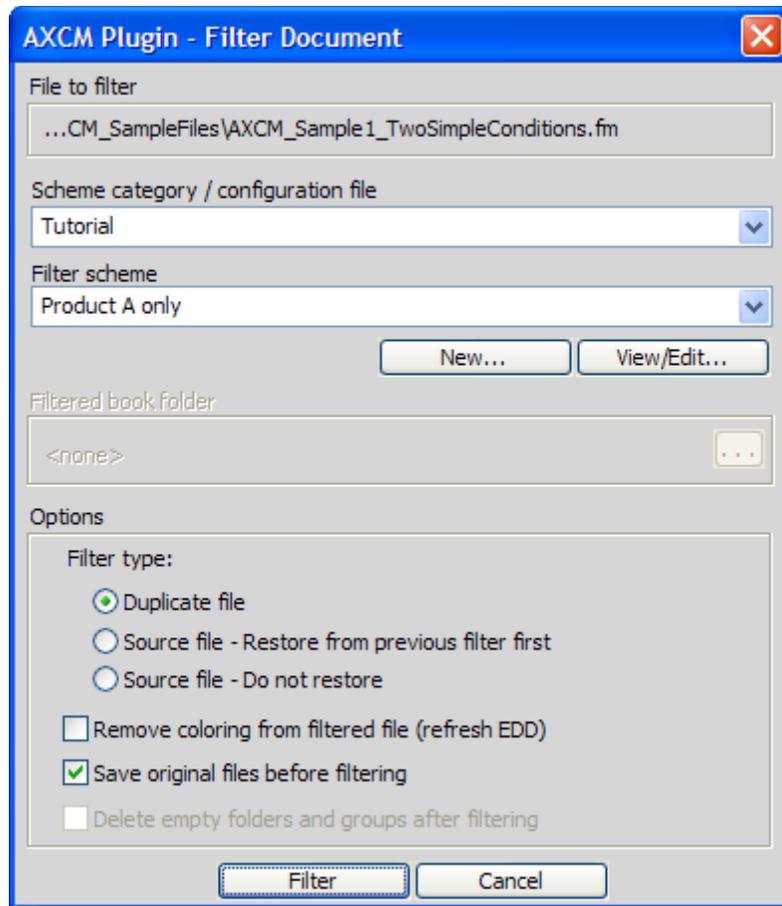
**9.** Click **Filter** and note the results.
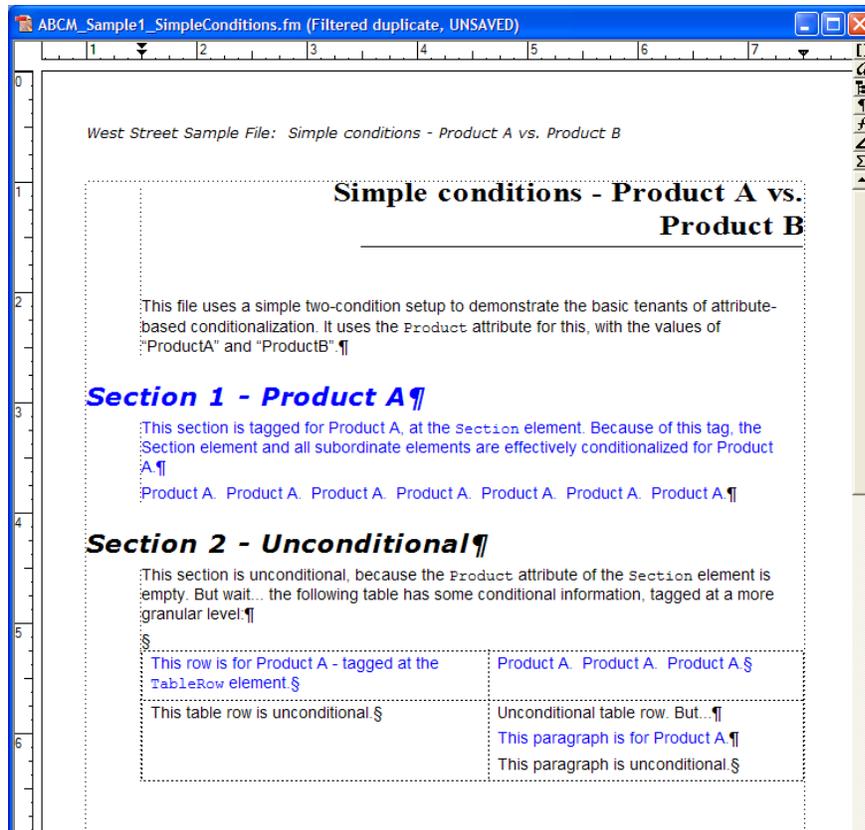
A duplicate file has been created with all the Product B (red text) information removed. The filter process is analogous to the Show/Hide process used for native conditional text, except that all the processing logic is programmed into the scheme. You do not need to think about individual conditions when running an AXCM filter… you need only choose a scheme and go.

10. Close the duplicate, filtered file without saving changes.

11. Run the filter again on the original document, using the **Product B only** scheme this time.

12. Note the results. This time, all Product A material has been filtered out.

West Street Sample File: Simple conditions - Product A vs. Product B

Simple conditions - Product A vs. Product B

This file uses a simple two-condition setup to demonstrate the basic tenants of attribute-based conditionalization. It uses the `Product` attribute for this, with the values of "ProductA" and "ProductB".¶

Section 2 - Unconditional¶

This section is unconditional, because the `Product` attribute of the `Section` element is empty. But wait... the following table has some conditional information, tagged at a more granular level:¶

This table row is unconditional.§	Unconditional table row. But...¶ This paragraph is unconditional.¶ This paragraph is for Product B.§
This row is for Product B - tagged at the `TableRow` element.§	Product B. Product B. Product B.§

Section 3 - Product B¶

This section is tagged for Product B, at the `Section` element. Because of this tag, the Section element and all subordinate elements are effectively conditionalized for Product B.¶

Product B. Product B. Product B. Product B. Product B. Product B. Product B.§

13. Close the filtered duplicate and the original file.

Part 3 – Introduction to scheme setup and editing

In this part, you will take a closer look at the schemes used in the previous procedure.

1. Open `AXCM_Sample2_ThreeSimpleConditions.fm`, and note that there is now a section for Product C in the document, with the `Product` attribute on the last `Section` element set to "ProductC".
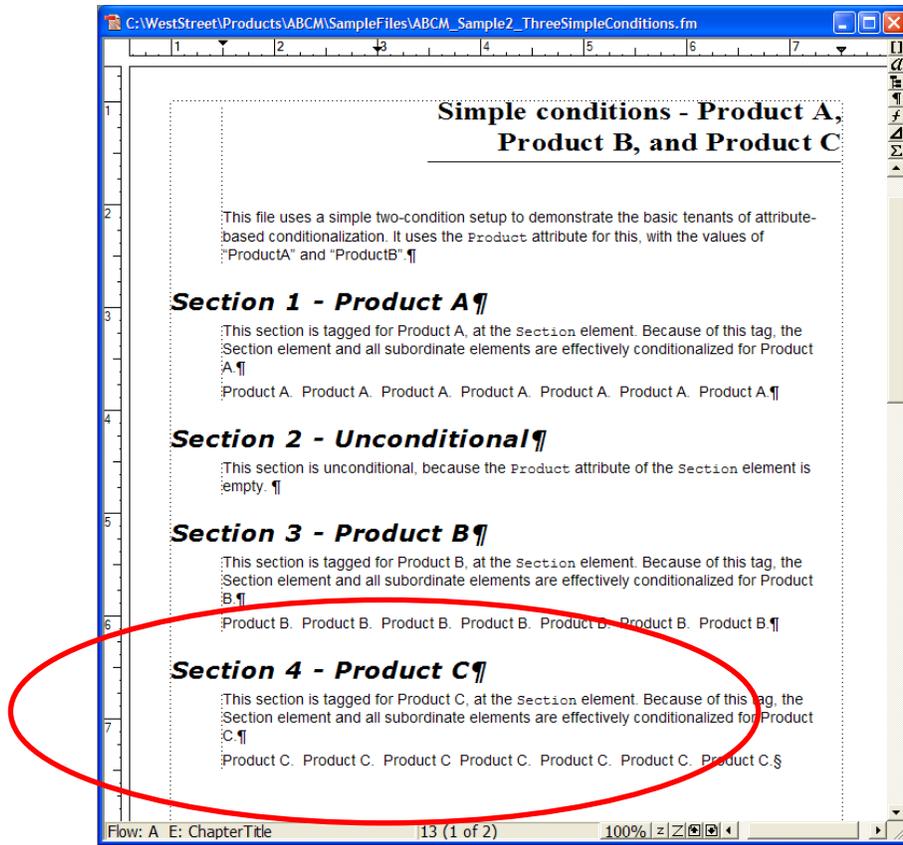
2. Select **AXCM > Main Settings Configuration > "Classic" Schemes** to open the classic scheme editor.
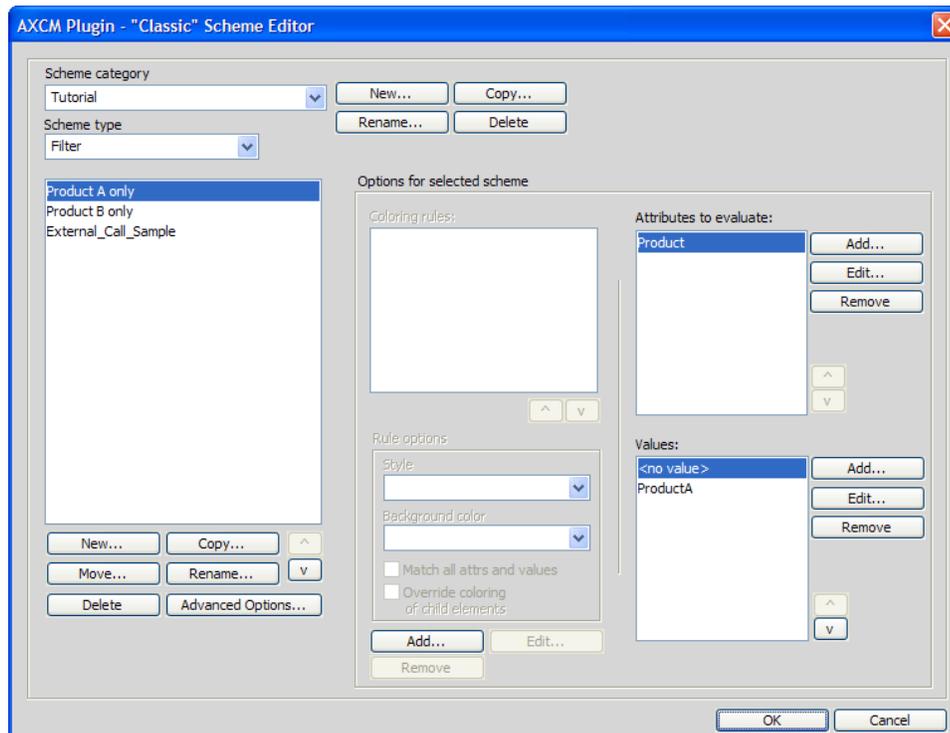
3. Under **Scheme category**, select **Tutorial**.
4. Under **Scheme type**, select **Filter**.
5. In the schemes list, select **Product A only**.
6. Take a close look at the right side of the dialog box, where the scheme parameters are specified. These parameters are nothing more than a set of attributes and corresponding values to evaluate during a filter action. Each element in the document will be evaluated according to the specified attribute(s) as applicable, and any that do not contain one or more of the specified values will be removed or hidden. In this case, any element that contains a Product attribute that is not empty or is not set to "ProductA" will be removed. Hence, the scheme is configured to produce a Product A-specific version of the document.
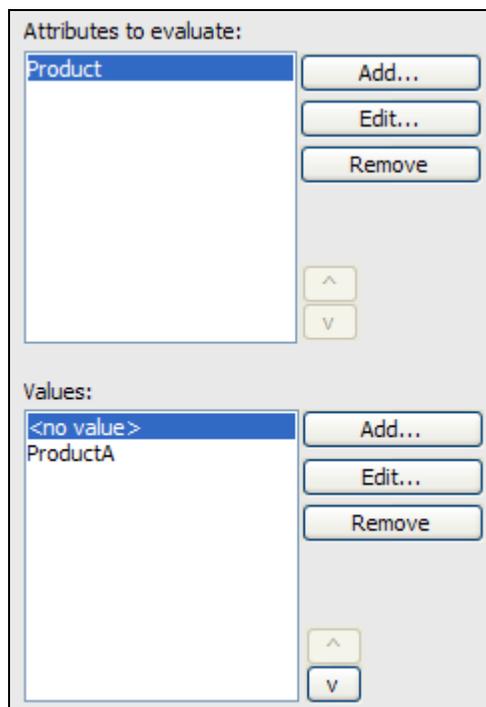
7. Under **Scheme type**, select **Coloring**.
8. Take a look at the **Product A vs. Product B** scheme. A coloring scheme is similar to a filter scheme, in that it is mostly just a set of attributes and values. The primary difference is that coloring schemes have coloring "rules," each of which is evaluated independently with its own set of attributes and values. In this scheme, there are two rules, one for Blue and Red. During a coloring action, AXCM evaluates each element based on these rules. If the attribute setup for the Blue rule matches, the respective element gets colored blue. If not, it checks the Red rule in a similar fashion. If all rules are exhausted with no match, the element gets no coloring at all. Note, however, that an element may become colored when an ancestor element matches a coloring rule, even if it doesn't match any rules itself.
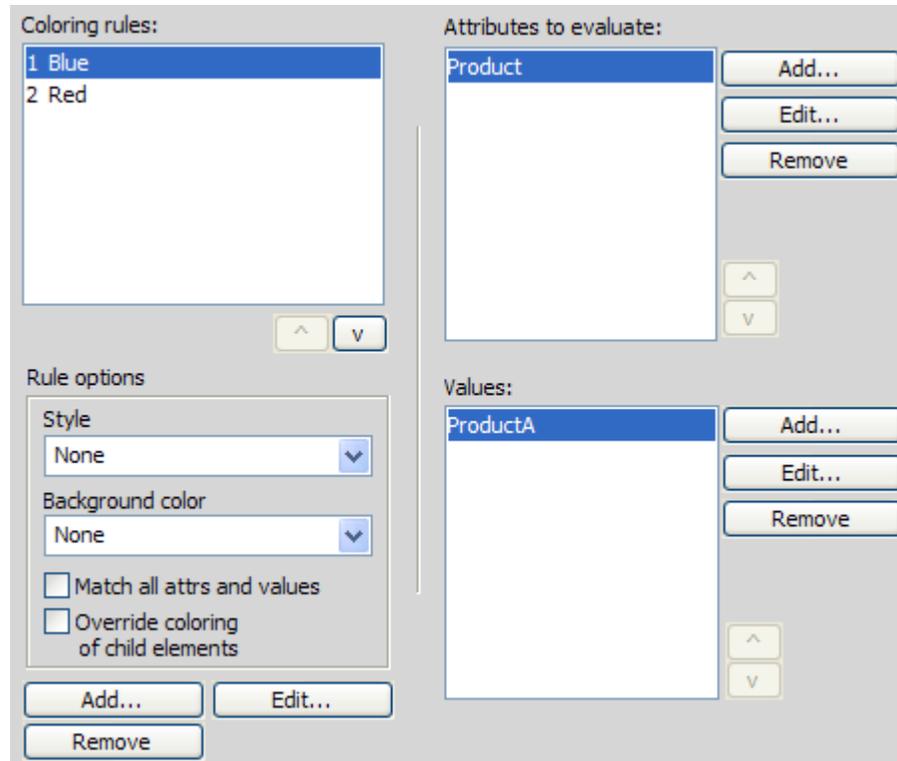
Now, you will alter the scheme setup to work with the sample document you just opened, `AXCM_Sample2_ThreeSimpleConditions.fm`. More specifically, you will change the schemes to accommodate the new condition.

9. With the **Product A vs. Product B** coloring scheme selected, click **Copy** under the scheme list.
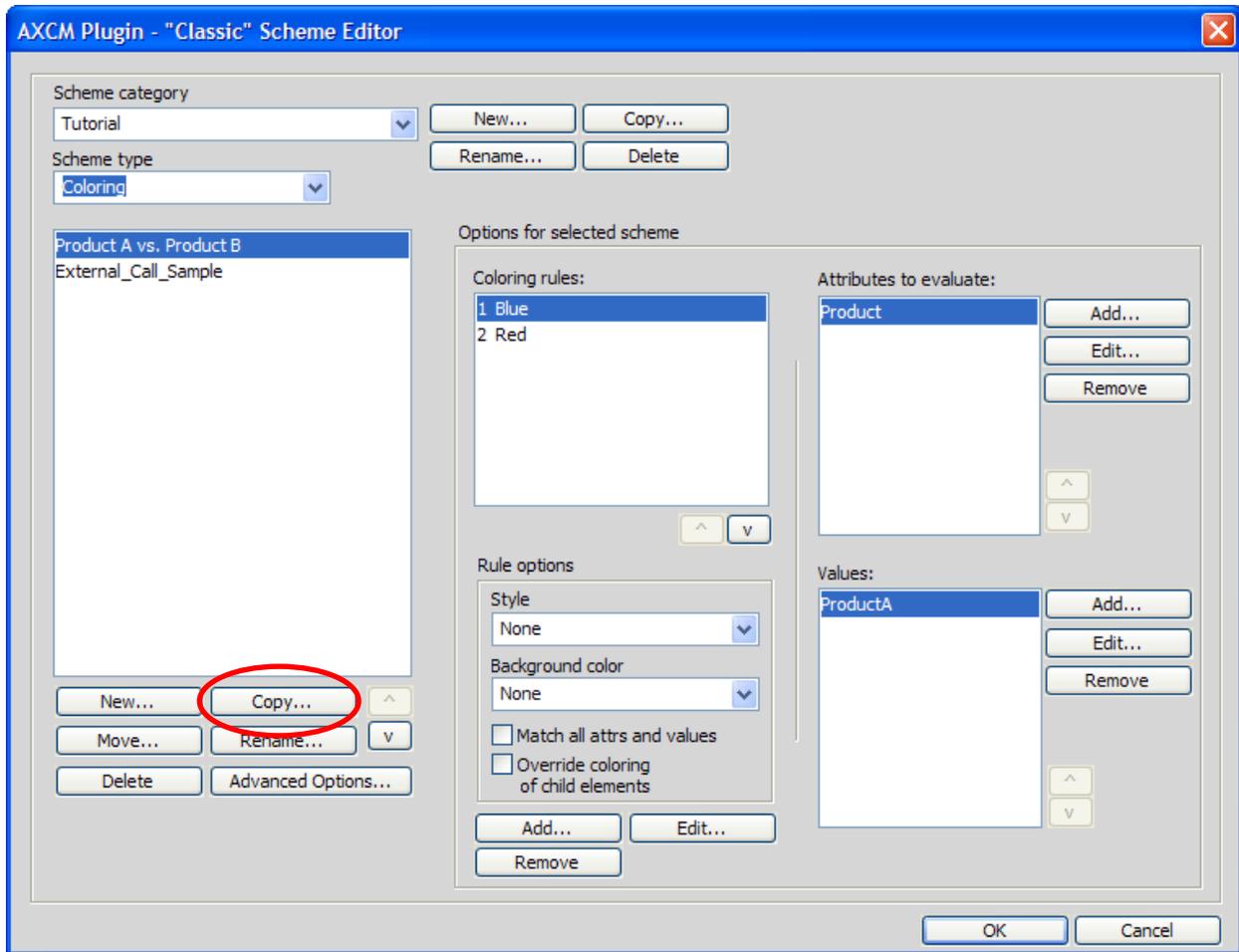
10. In the **New Scheme** dialog, enter a new scheme name such as **Products A, B, C**:
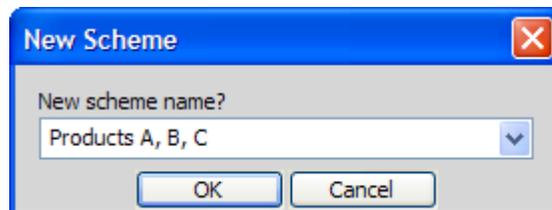

11. Click **OK**, and note that the new scheme appears in the list, and that it is an exact duplicate of the original scheme.

**12.** Under the list of rules, click **Add**.
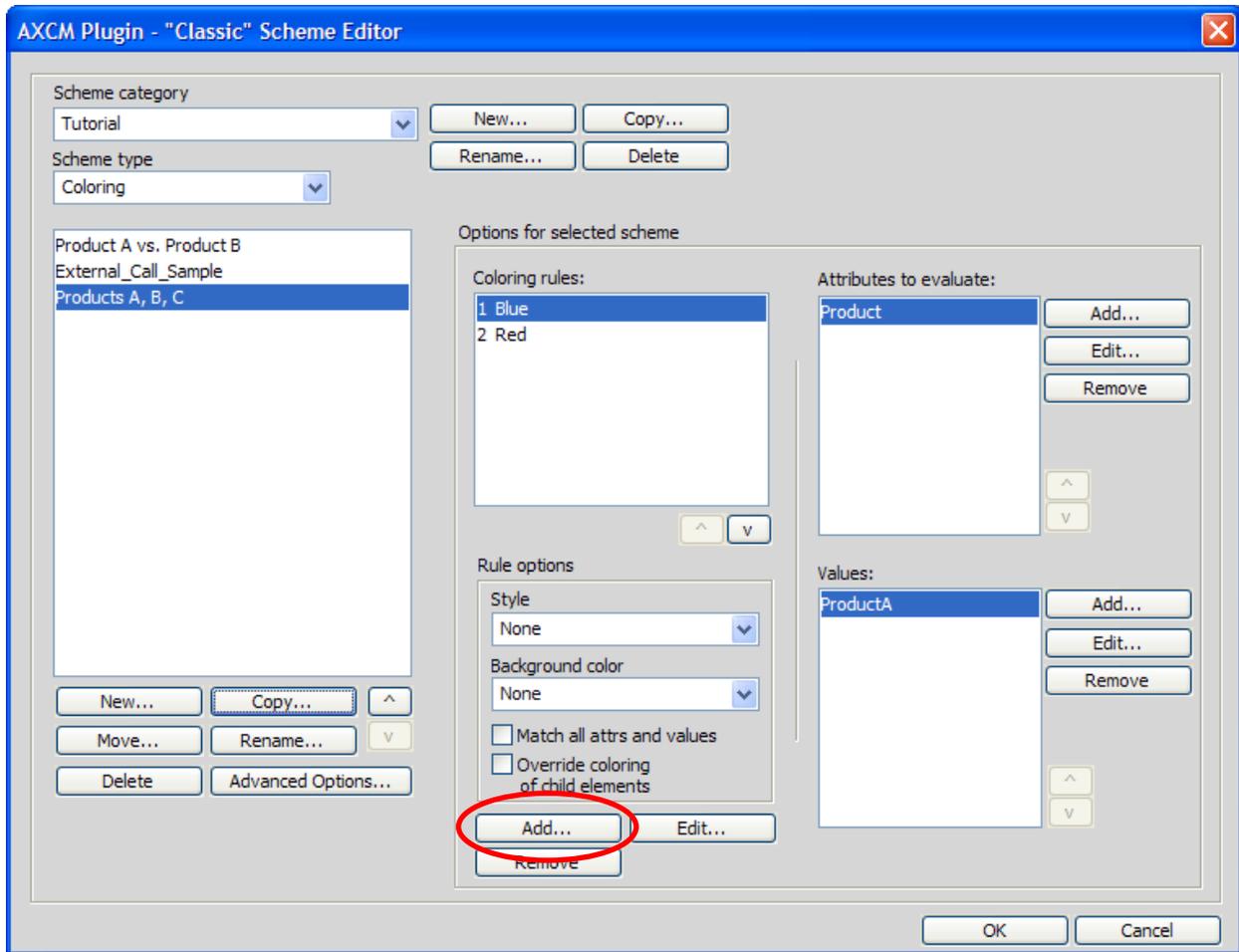
13. In the **New Rule** dialog, select a new color such as **Green** and click **OK**:
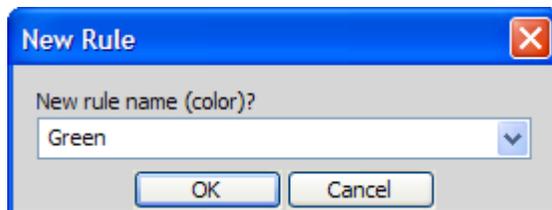
You will now set up the rule to evaluate elements for Product C and color them green.
**14.** To the right of the **Attributes** list, click **Add**.
15. In the **New Attribute** dialog, select or type **Product**, using that exact case.
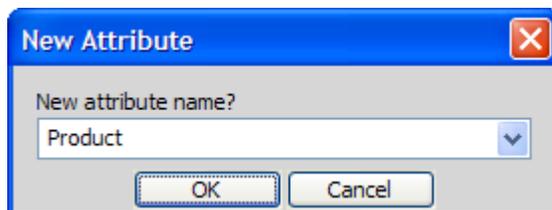
16. Click **OK**.

17. To the right of the **Values** list, click **Add**.

18. In the **New Value** dialog, select or type **ProductC**, using that exact case.
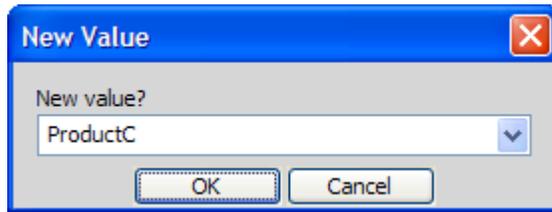
19. Click **OK**. The scheme editor should now appear as follows:
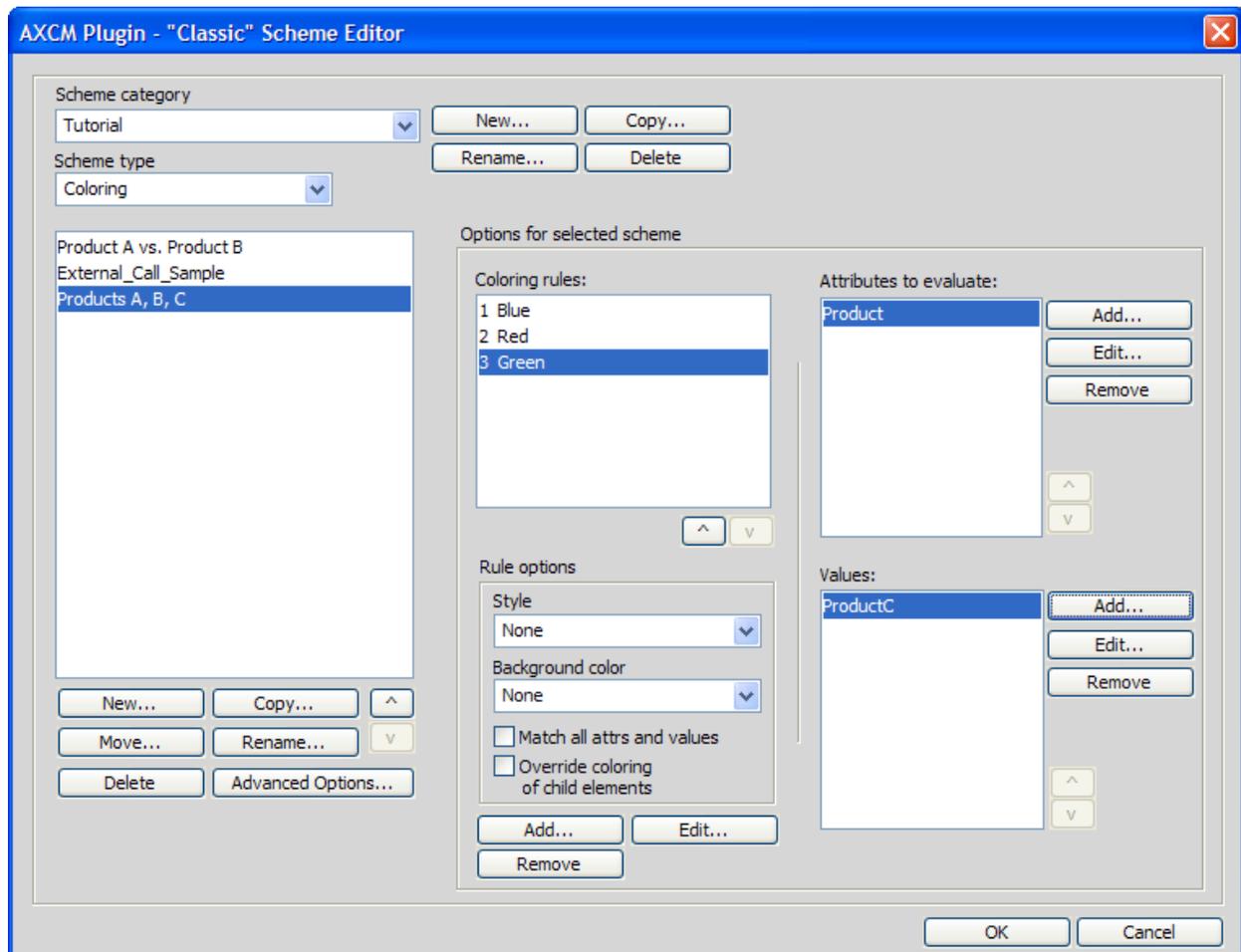
20. Click **OK** to close the scheme editor.

21. With the three-condition sample document active, select **AXCM > Coloring > Color Document**.

22. Select the **Tutorial** category, the new **Products A, B, C** scheme, and click **OK**. Note the coloring for the Product C section.
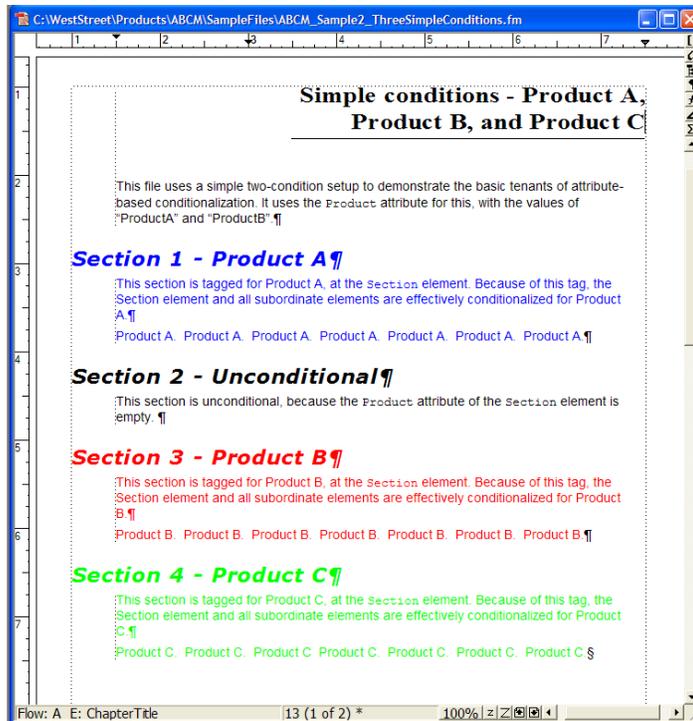
23. Select **AXCM > Filtering > Filter Document**, and filter the document with the **Product A** only scheme. Note that the Product B and Product C material is removed.
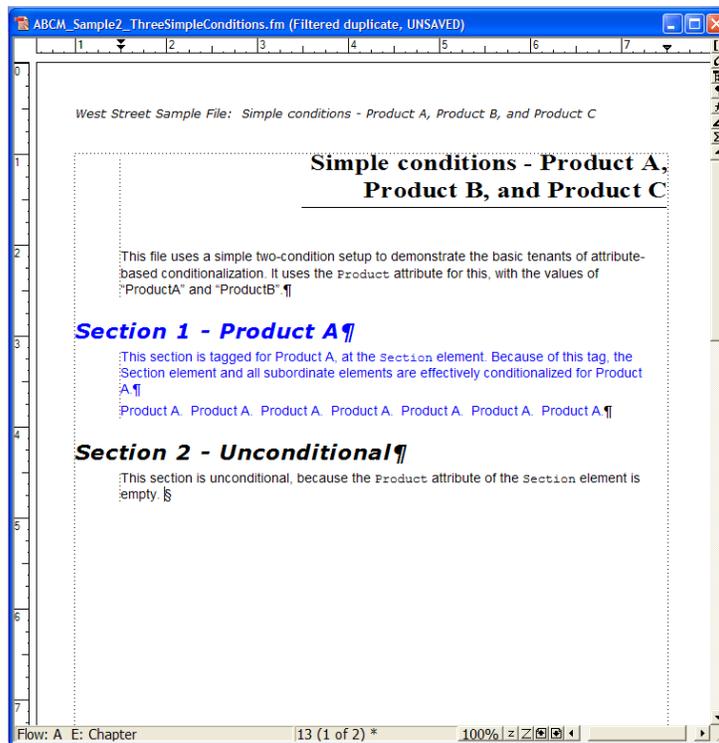
Also note that you did not have to change the filter scheme, even though a new condition was introduced to the document. This is a big advantage to AXCM filtering logic, versus native conditional text show/hide functionality. With a filter scheme, you are specifying the content you want to keep, and all other content is removed by

default. The addition of a new condition may not affect your filtering logic at all, because the desired conditions in the output are still the same.

24. Close the new, filtered duplicate.
25. Open the classic scheme editor again (**AXCM > Main Settings Configuration > "Classic" Schemes**)
26. Under **Scheme type**, select **Filter**.
27. Select the **Product A only** scheme.
28. Under the **Schemes** list, click **Copy** and give the new scheme the name **Products A and C**.
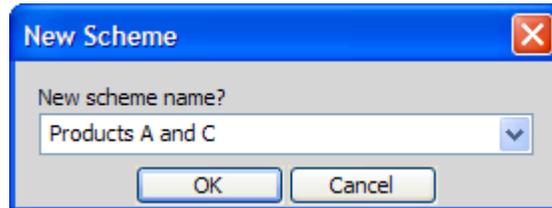
29. Click **OK** in the **New Scheme** dialog.
30. With the new scheme selected, click **Add** to the right of the **Values** list.
    You are now preparing to alter this scheme to produce a composite output for both Product A and Product C.
31. In the **New Value** dialog, select or type **ProductC**.
32. Click **OK** in the **New Value** dialog. The scheme editor should appear as follows:
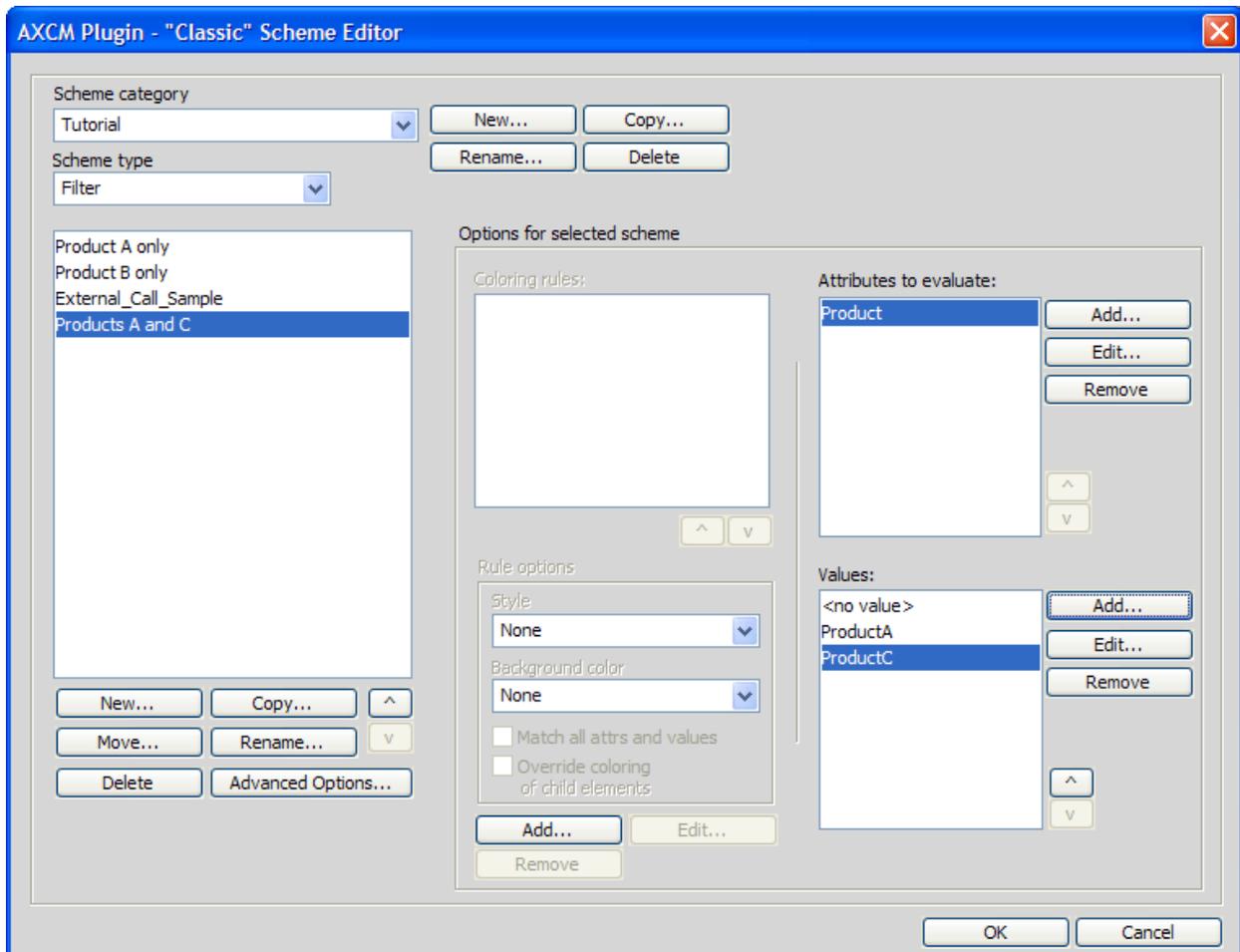
33. Click **OK** to close the scheme editor.

34. Filter the original document with the new scheme and note that the output contains Product A and Product C material.
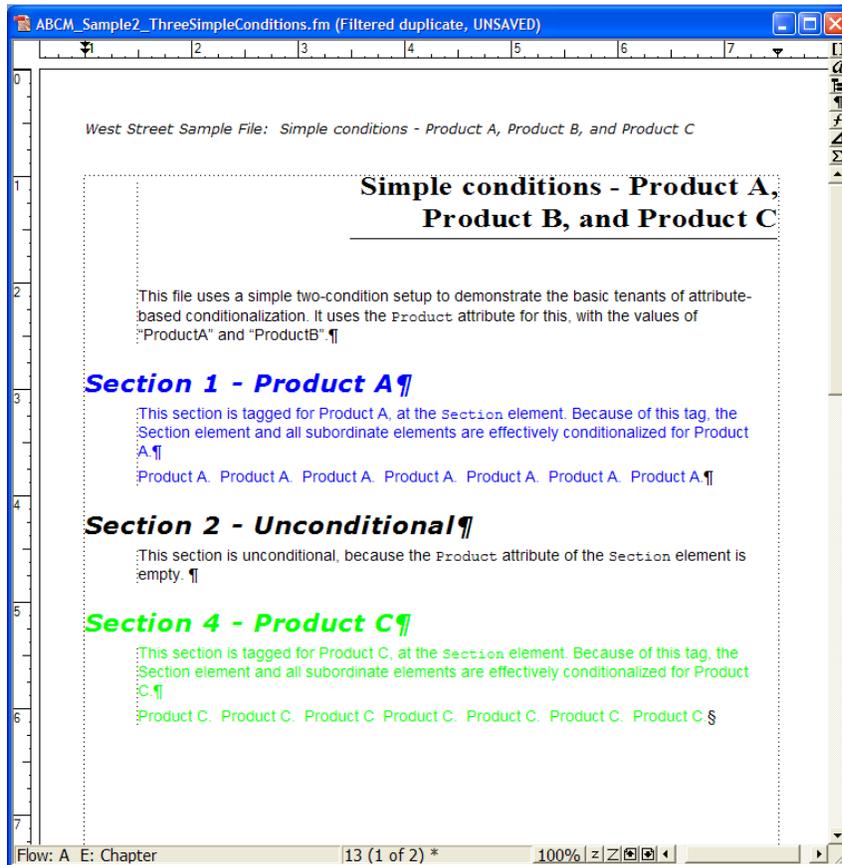
35. Close the filtered duplicate and the original document.

Part 4 – Simple overlapping conditions

In a structured document, an element can have multiple attributes, each of which may be configured to contain multiple values (as allowed by the EDD). When multiple attributes and/or values are used to denote conditions, this is equivalent to the concept of overlapping conditions. This situation, however, is much easier to handle when using structural attributes to denote conditions, rather than native conditional text tags.

1. Open `AXCM_Sample3_SimpleOverlappingConditions.fm`, and note that there is now a section for both Product A and Product B, with the `Product` attribute on the `Section` element set to "ProductA" and "ProductB".
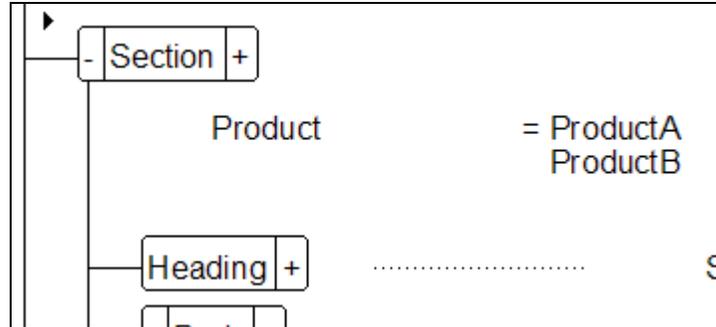
2. Filter the document (**AXCM > Filtering > Filter Document**) using the **Product A only** and the **Product B only** schemes, and note that in both cases the composite paragraph remains.
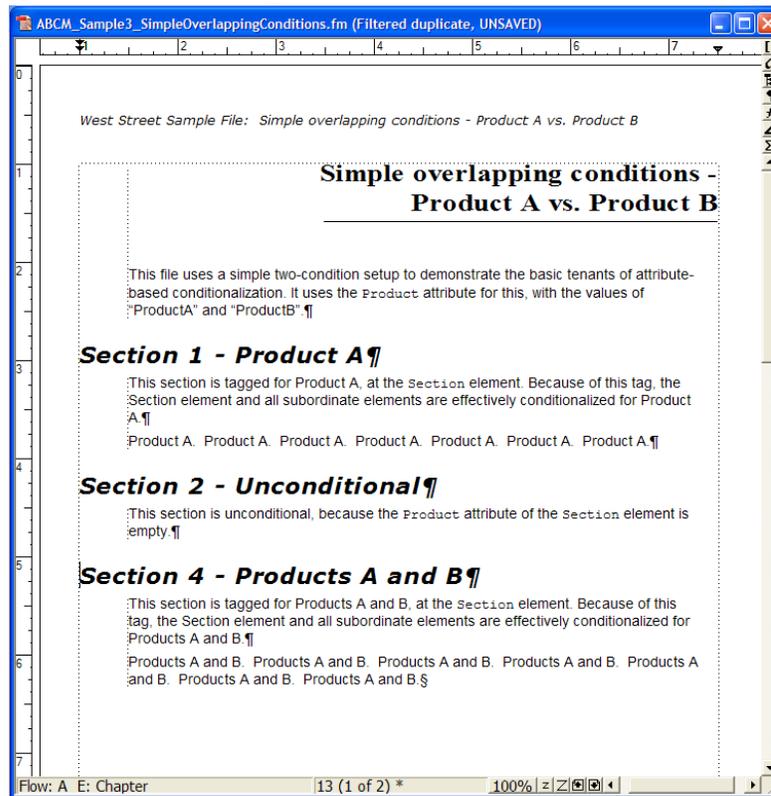
3. Close any filtered duplicates you created.
4. Color the document with the **Product A vs. Product B** scheme and note the results (**AXCM > Coloring > Color Document**).
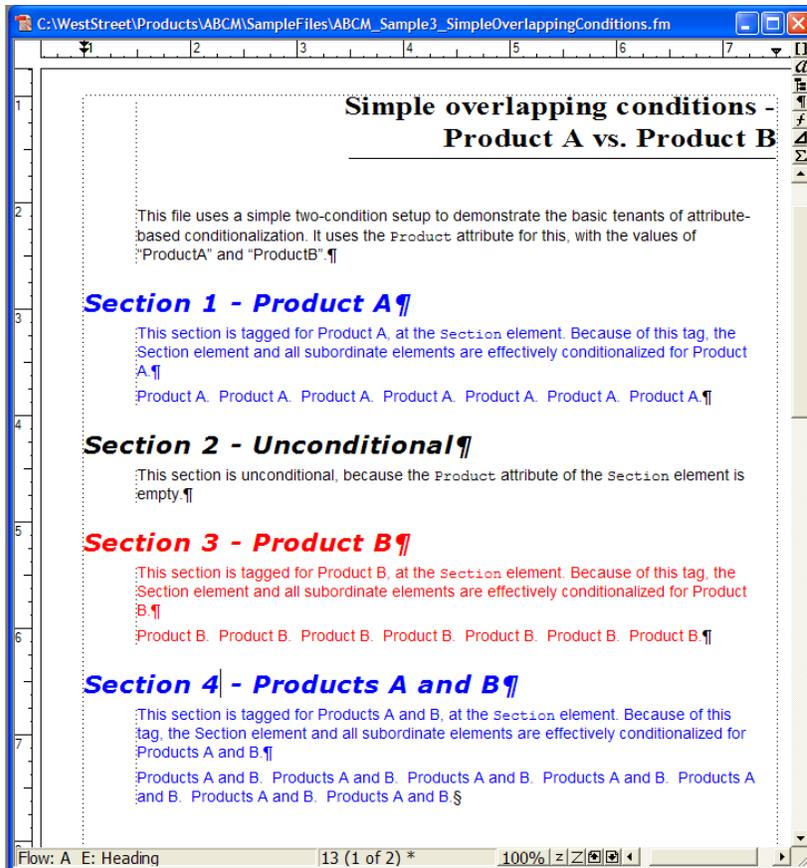
The composite section received blue, the color for Product A. This happened because the first rule in the scheme matched the composite section, because it found Product A there. Currently, there is no rule in the scheme designed specifically to match a composite Product A and B section, so AXCM simply used the first matching rule it found. In the following steps, you will change the scheme to include a rule for the composite material.

5. Open the classic scheme editor (**AXCM > Main Settings Configuration > "Classic" Schemes**).
6. Select the **Tutorial** category, the **Coloring** scheme type, and the **Product A vs. Product B** scheme.
7. Under the coloring rules, click **Add**.
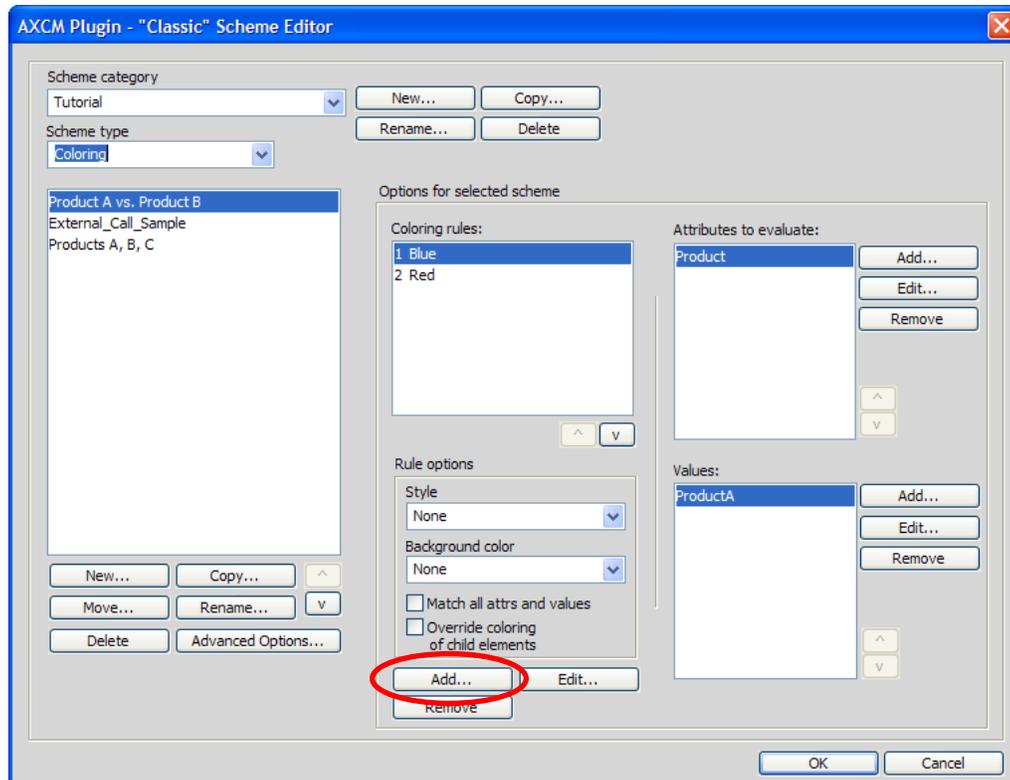
8. In the **New Rule** dialog, select a new color such as **Purple**.
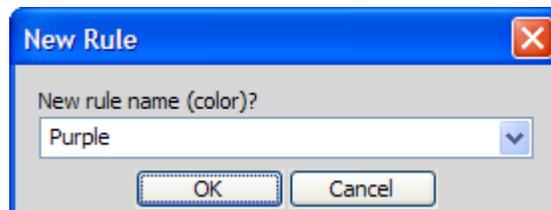
9. Use the "^" button under the rules to slide the new Purple rule to the top of the list.

    NOTE: This is a very important step, because rule order can have a significant impact on the behavior of a coloring scheme. For more explanation on this subject, see your *User Guide*.
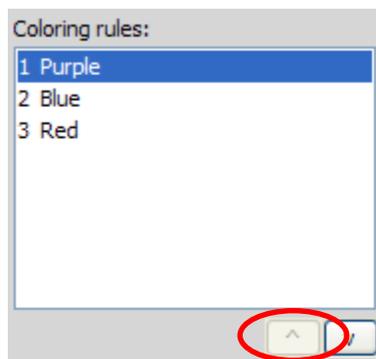
10. To the right, add **Product** to the list of attributes to evaluate.

11. For the **Product** attribute, add the values "**ProductA**" and "**ProductB**" below.

12. In the **Rule options**, check **Match all attrs and values**.
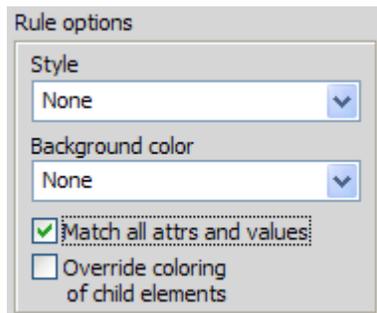
This option forces the rule to match all the attributes and values specified, rather than just one. That is, if an element has a Product attribute, it must contain both "ProductA" and "ProductB" to match this rule and get colored purple.

13. Ensure that the scheme editor appears as follows, and click **OK**.
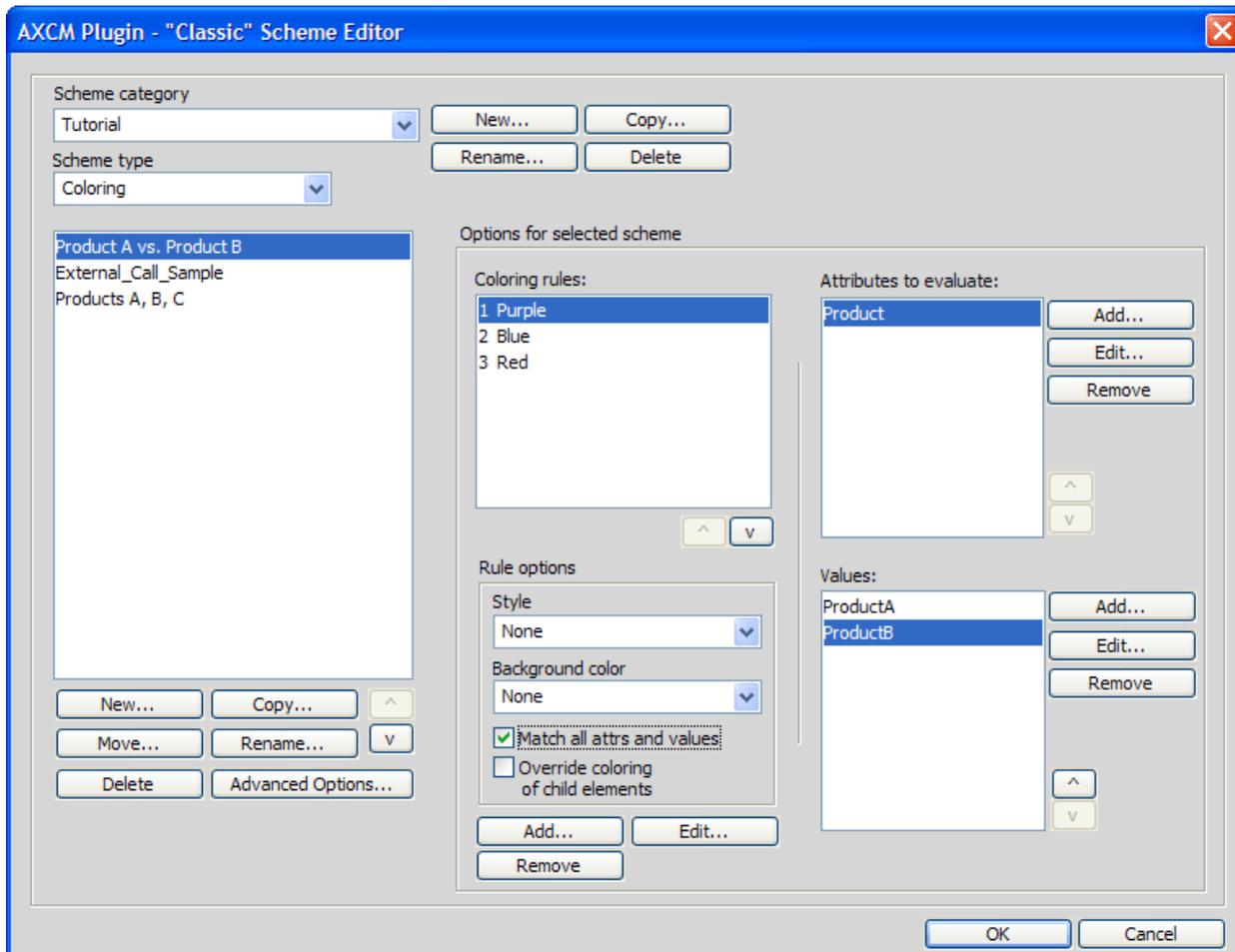
14. Color the document again (**AXCM > Coloring > Color Document**) with the **Product A vs. Product B** scheme and note the results.

The window title bar reads: C:\WestStreet\Products\ABCM\SampleFiles\ABCM_Sample3_SimpleOverlappingConditions.fm

Simple overlapping conditions -
Product A vs. Product B

This file uses a simple two-condition setup to demonstrate the basic tenants of attribute-based conditionalization. It uses the `Product` attribute for this, with the values of "ProductA" and "ProductB".¶

Section 1 - Product A¶

This section is tagged for Product A, at the `Section` element. Because of this tag, the Section element and all subordinate elements are effectively conditionalized for Product A.¶

Product A. Product A. Product A. Product A. Product A. Product A. Product A.¶

Section 2 - Unconditional¶

This section is unconditional, because the `Product` attribute of the `Section` element is empty.¶

Section 3 - Product B¶

This section is tagged for Product B, at the `Section` element. Because of this tag, the Section element and all subordinate elements are effectively conditionalized for Product B.¶

Product B. Product B. Product B. Product B. Product B. Product B. Product B.¶

Section 4 - Products A and B¶

This section is tagged for Products A and B, at the `Section` element. Because of this tag, the Section element and all subordinate elements are effectively conditionalized for Products A and B.¶

Products A and B. Products A and B. Products A and B. Products A and B. Products A and B. Products A and B. Products A and B.§

Flow: A   E: Heading   13 (1 of 2) *   100%

NOTE: All AXCM coloring with regards to overlapping conditions operates in this fashion. AXCM colors exactly what the respective scheme indicates, and nothing else. No more magenta text or automatically-generated composite colors, unless your scheme specifically says so!

15. Close the sample document.

Part 5 – Filtering by attribute name, rather than values

With classic schemes, AXCM includes a feature that allows you to filter out elements based on the mere presence of an attribute, regardless of its contents. This is particularly useful for a "blind" filtering out of elements that you know should not appear in your output. In this part, you will use the common scenario of authoring comments that should be filtered out of the output, and the particular contents of the element attributes are not important.

1. Open `AXCM_Sample4_AuthoringComments.fm`, and note the `Comment` elements. In particular, note that they each have an `Author` attribute, but set to different values.

NOTE: The Author attribute is unique to Comment elements in this document's EDD, so you can now configure a scheme to filter out Comment elements based on the presence of that attribute alone.

2. Select **AXCM > Main Settings Configuration > Attribute Library** to open the master attribute library.

The library is a scheme editing convenience only, providing data for some of the drop-down lists. It does not control the valid attributes and/or values you may use for conditions. This step is a brief diversion to introduce you to the library and how it works.

3. In the **Attribute Library**, select the **Product** attribute and note the list of valid values.
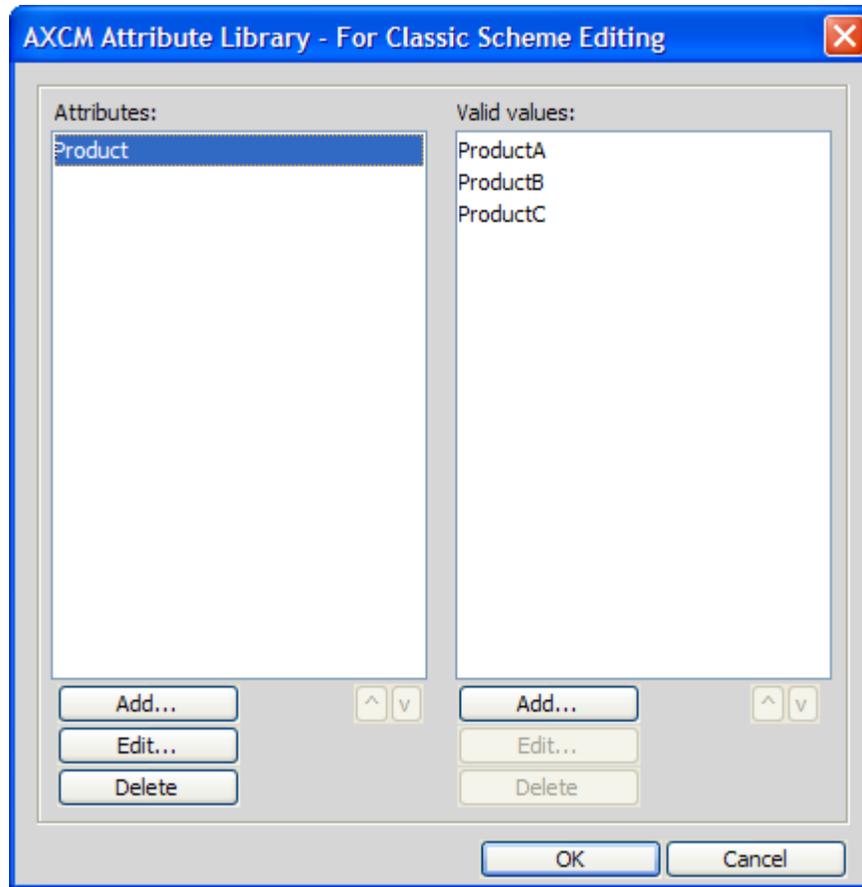
Earlier, when you added attributes and values to schemes, the drop-down lists were prepopulated with some values. This library is where those values came from. It is nothing more than a feature to help construct schemes more quickly and with more consistency. Note that it is applicable to classic scheme editing only, not XPath-based schemes.

4. Under the **Attributes** list, click **Add** and add the `Author` attribute.
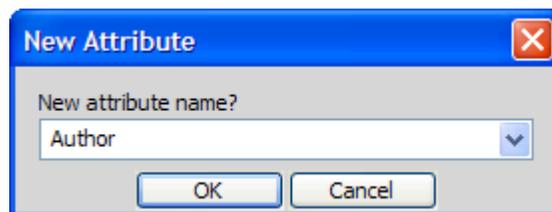
5. Click **OK** in the **Attribute Library** dialog.
6. Open the classic scheme editor (**AXCM > Main Settings Configuration > "Classic" Schemes**).
7. Select the **Tutorial** category, the **Filter** scheme type, and the **Product A only** scheme.
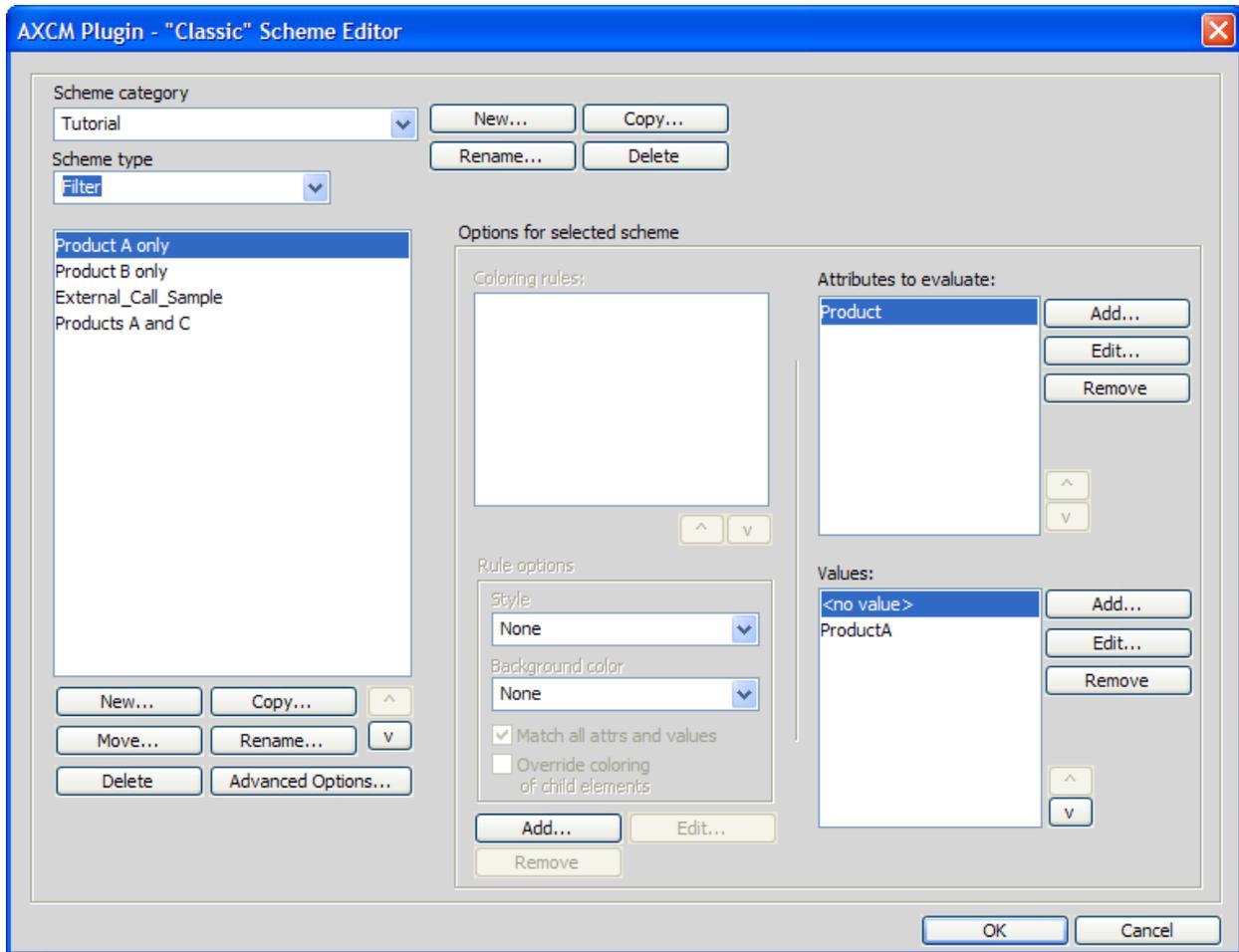
8. Under the **Attributes to evaluate** list, add the **Author** attribute, noting that it now appears in the drop-down list. Do not add any values.
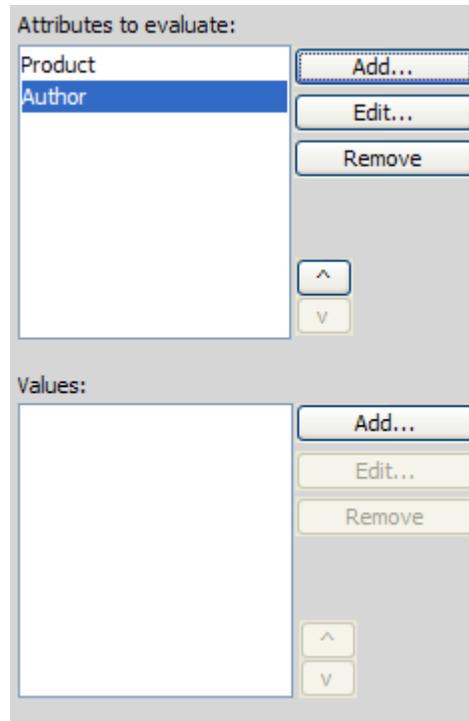
This setup indicates that any element with an `Author` attribute will be filtered out, regardless of the attribute's contents. In the sample document, this includes the `Comment` elements. Recall that for filter schemes, you are specifying the valid values for the content you want to keep. This setup literally says that there is no possible valid content for an `Author` attribute, so the mere presence of the attribute will trigger removal.

9. Click **OK** in the scheme editor
10. Filter the document with the **Product A only** scheme and note the results.

West Street Sample File: Multiple conditions and removing elements by attribute exister

Multiple conditions and removii elements by attribute existen

This file demonstrates the practice of element removal by looking for the mere exister of an attribute, in this case the Author attribute. It also shows how a scheme may conf more than one attribute for evaluation.¶

This file is intended for use with the AXCM tutorial.¶

Section 1 - Product A¶

This section is tagged for Product A, at the Section element. Because of this tag, th Section element and all subordinate elements are effectively conditionalized for Produ A.¶

Product A. Product A. Product A. Product A. Product A. Product A. Product A. ¶

Section 2 - Unconditional¶

This section is unconditional, because the Product attribute of the Section elemen empty.§

Flow: A E: Chapter |◄ ◄ [1] ► ►| 13 (1 of 2... [14] ►▣ —   100% ▼ ╋

Part 6 – Filtering a book

Book filtering is similar to document filtering, except that multiple files are processed together. One important feature of book filtering is the ability to conditionalize an entire chapter of a book and remove it during a filter process. This feature will be demonstrated in the following steps, used in conjunction with a duplicate file book filter.

1. Open `AXCM_SampleBook.book`, and all its chapter files.

2. Take a look at `AXCM_Sample5_ProductB_WholeDoc.fm`, and note how it is conditionalized at the highest-level element for Product B.
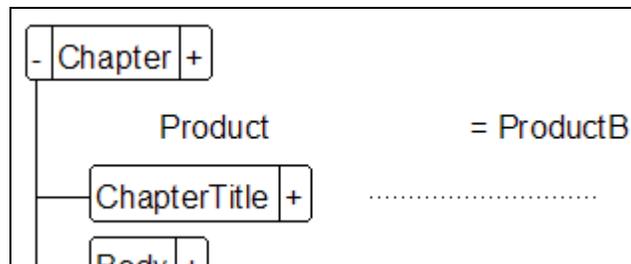
3. Bring the book window back to the front.
4. Select **AXCM > Coloring > Color Book**.
5. Color the book with the **Product A vs. Product B scheme**, under the **Tutorial** category, and note the results.
   All three files are colored, and the Product B file is colored entirely for the Product B color.
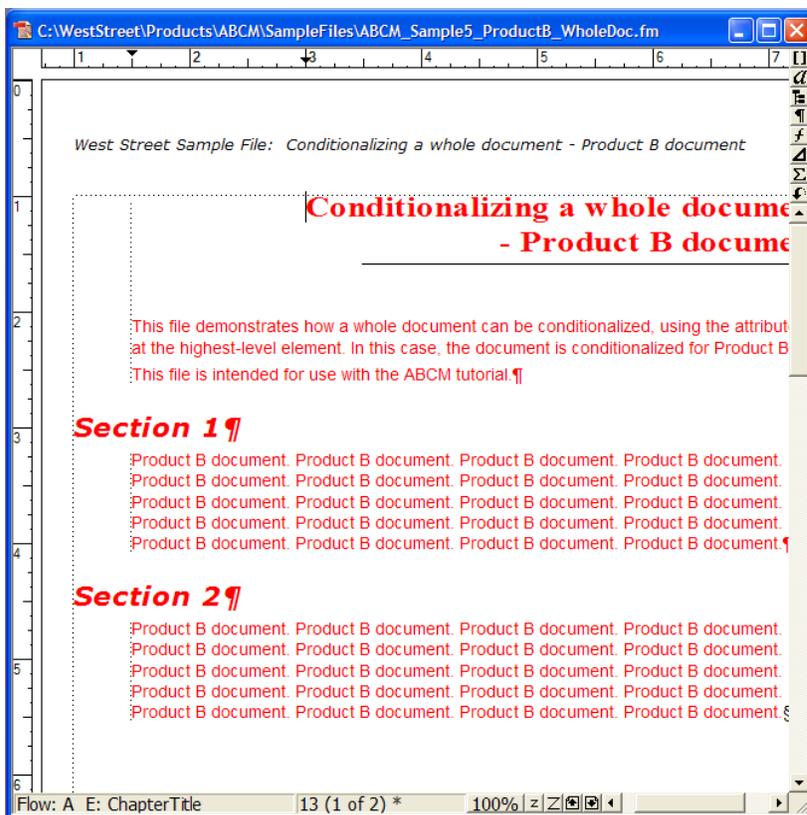
6. Using Windows Explorer or some other file management utility, create a subfolder called `Filtered` in your sample files area.
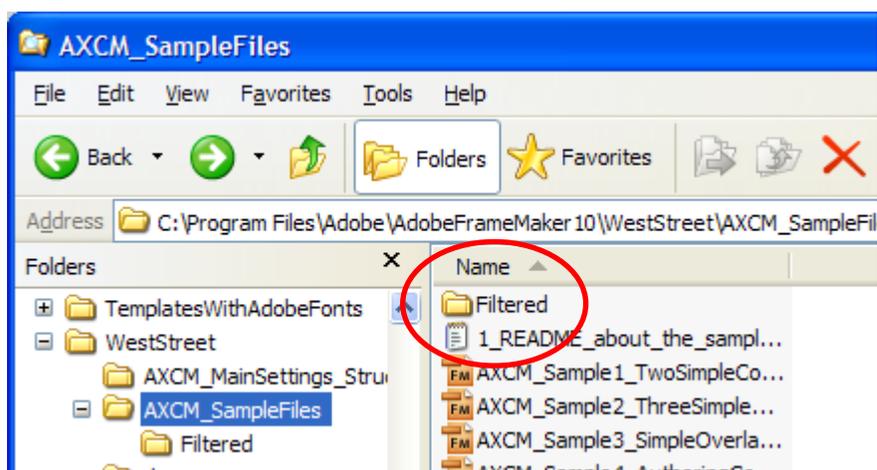

7. Back in FrameMaker, bring the book window to the front.
8. Select **AXCM > Filtering > Filter Book**.
9. Select the **Tutorial** category and the **Product A only** scheme.

10. Under **Filtered book folder**, click the "**. . .**" button and browse to the `Filtered` subfolder you just created.
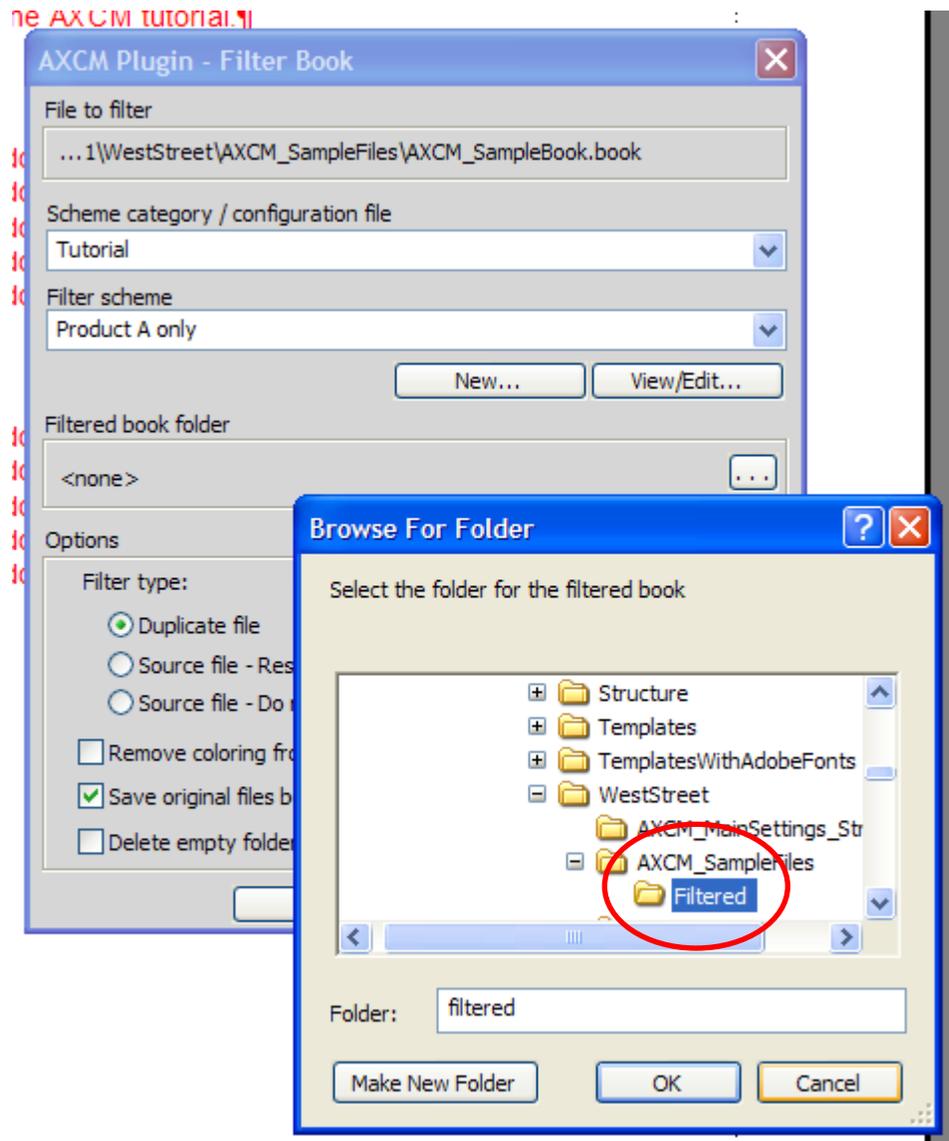
11. Ensure the **Filter** dialog now looks something like the following:

AXCM Plugin - Filter Book

File to filter

...1\WestStreet\AXCM_SampleFiles\AXCM_SampleBook.book

Scheme category / configuration file

Tutorial

Filter scheme

Product A only

New...   View/Edit...

Filtered book folder

...FrameMaker11\WestStreet\AXCM_SampleFiles\Filtered\

Options

Filter type:

⊙ Duplicate file
○ Source file - Restore from previous filter first
○ Source file - Do not restore

☐ Remove coloring from filtered file (refresh EDD)
☑ Save original files before filtering
☐ Delete empty folders and groups after filtering

Filter   Cancel

---

12. Click **Filter** and note the results. All files in the book are filtered, and the Product B chapter has been completely filtered out.

13. Close all the sample files.

Part 7 – Validation

This part will briefly introduce the attribute validation feature of AXCM. For more detailed information on how the validation schemes and rules work, see your *User Guide*.

1. Select **AXCM > Open Local Settings File**.

2. Ensure that the following settings are present. If not, edit them as shown, save and close the file, then select **AXCM > Read Local Settings From Settings File**.

```
VAL_RULE_1_ENABLED=TRUE

VAL_RULE_2_ENABLED=TRUE

VAL_RULE_3_ENABLED=FALSE

VAL_RULE_4_ENABLED=TRUE
```

3. Open the sample file AXCM_Sample6_ValidationErrorSample.fm.

4. Select **AXCM > Validation > Validate Document**.

5. Select the **Tutorial** category, the **Sample validation** scheme, and click **OK**.
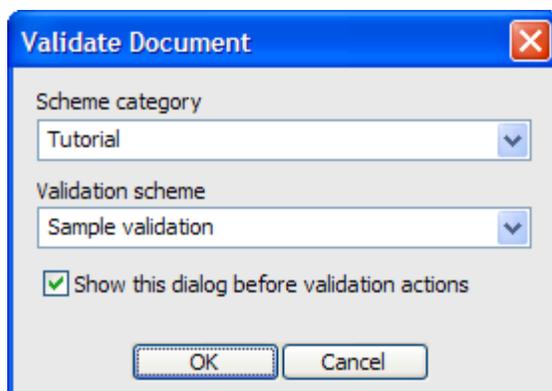
6. Note the results. As desired, examine the validation scheme setup and consult your *User Guide* to learn more about what happened.
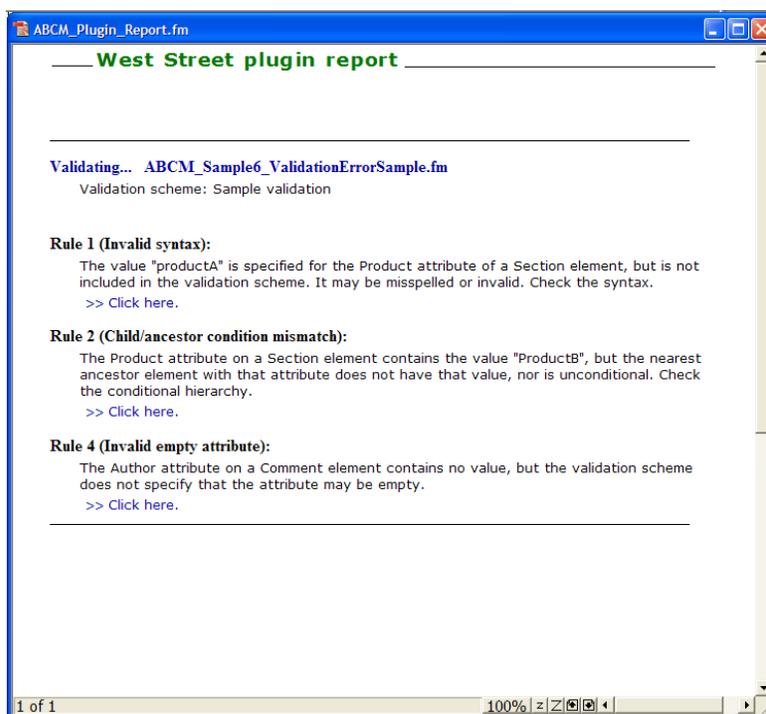
Part 8 – Finishing up and looking ahead

This tutorial introduced some of the fundamentals concerning AXCM. If you intend to incorporate the software into your workflow, the following are some things to keep in mind:

- In this tutorial, all conditionalization was done at the paragraph level or higher, for simplicity. That is, conditional attributes were set on paragraph-containing or higher-level attributes only. In practice, you can put conditional attributes on any element, including text spans, markers, table components, and variables, and the software will treat them the same way. AXCM cares about markup only… it doesn't care what the elements themselves contain.

- In this tutorial, all the schemes dealt with one or two attributes and values. Remember that schemes can address any number of attributes and values, as necessary to achieve the desired result.

- The software has many facets to the rule base used for filtering, coloring, and validation. This tutorial covered only a small subset of common features. If you plan to use AXCM for production work, consider giving the *User Guide* some time, such that you can discover ways to make the software work better for you.

Thank you for taking the time to check out AXCM. We hope that it can be of some use to you.