Customize code maps by editing the DGML files

11/04/2016 • 13 minutes to read • +4

In this article

To customize a code map, you can edit its Directed Graph Markup Language (.dgml) file. For example, you can edit elements to specify custom styles, assign properties and categories to code elements and links, or link documents or URLs to code elements or to links. For more information about DGML elements, see Directed Graph Markup Language (DGML) reference.

Edit the code map's .dgml file in a text or XML editor. If the map is part of your Visual Studio solution, select it in **Solution Explorer**, open the shortcut menu, and choose **Open With**, **XML (Text) Editor**.

Note

To create code maps, you must have Visual Studio Enterprise edition. When you edit a code map in Visual Studio, it cleans up any unused DGML elements and attributes by deleting them when you save the .dgml file. It also creates code elements automatically when you manually add new links. When you save the .dgml file, any attributes that you added to an element might rearrange themselves in alphabetical order.

Group code elements

You can add new groups or convert existing nodes into a group.

1. Open the .dgml file in a text or XML editor.

2. To convert a code element to a group, find the `<Node/>` element for that code element.

   - or -

   To add a new group, find the `<Nodes>` section. Add a new `<Node/>` element.

3. In the `<Node/>` element, add a `Group` attribute to specify whether the group appears expanded or collapsed. For example:

   XML Copy

```xml
<Nodes>
   <Node Id="MyFirstGroup" Group="Expanded" />
   <Node Id="MySecondGroup" Group="Collapsed" />
</Nodes>
```

4. In the `<Links>` section, make sure that a `<Link/>` element that has the following attributes exist for each relationship between a group code element and its child code elements:

   - A `Source` attribute that specifies the group code element
   - A `Target` attribute that specifies the child code element
   - A `Category` attribute that specifies a `Contains` relationship between the group code element and its child code element

     For example:

   XML Copy

```xml
<Links>
   <Link Category="Contains" Source="MyFirstNewGroup"
Target="FirstGroupChildOne" />
   <Link Category ="Contains" Source="MyFirstNewGroup"
Target="FirstGroupChildTwo" />
   <Link Category ="Contains" Source="MySecondNewGroup"
Target="SecondGroupChildOne" />
   <Link Category="Contains" Source="MySecondNewGroup"
```

```
Target="SecondGroupChildTwo" />
</Links>
```

For more information about the `Category` attribute, see Assign categories to code elements and links.

Change the style of the map

You can change the background color and border color of the map by editing the map's .dgml file. To change the style of code elements and links, see Change the style of code elements and links.

1. Open the .dgml file in a text or XML editor.

2. In the `<DirectedGraph>` element, add any of the following attributes to change its style:

   Background color

   XML Copy

   ```
   Background="ColorNameOrHexadecimalValue"
   ```

   Border color

   XML Copy

   ```
   Stroke="StrokeValue"
   ```

   For example:

   XML Copy

   ```
   <DirectedGraph Background="Green"
   xmlns="http://schemas.microsoft.com/vs/2009/dgml" >
      ...
      ...
   </DirectedGraph>
   ```

Change the style of code elements and links

You can apply custom styles to the following code elements:

- Single code elements and links
- Groups of code elements and links
- Groups of code elements and links based on certain conditions

Tip

If you have repeating styles across many code elements or links, you might consider applying a category to those code elements or links, and then applying a style to that category. For more information, see Assign Categories to Code elements and Links and Assign Properties to Code elements and Links.

To apply a custom style to a single code element

1. Open the .dgml file in a text or XML editor.

2. Find the code element's `<Node/>` element. Add any of these attributes to customize its style:

   Background color

   XML

   ```xml
   Background="ColorNameOrHexadecimalValue"
   ```

   Outline

   XML

   ```xml
   Stroke="ColorNameOrHexadecimalValue"
   ```

   Outline thickness

   XML

   ```xml
   StrokeThickness="StrokeValue"
   ```

   Text color

XML Copy

```
Foreground="ColorNameOrHexadecimalValue"
```

Icon

XML Copy

```
Icon="IconFilePathLocation"
```

Text size

XML Copy

```
FontSize="FontSizeValue"
```

Text type

XML Copy

```
FontFamily="FontFamilyName"
```

Text weight

XML Copy

```
FontWeight="FontWeightValue"
```

Text style

XML Copy

```
FontStyle="FontStyleName"
```

For example, you can specify `Italic` as the text style.

Texture

XML Copy

```
Style="Glass"
```

- or -

XML Copy

```xml
Style="Plain"
```

Shape

To replace the shape with an icon, set the `Shape` property to `None` and set the `Icon` property to the path with the icon file.

XML Copy

```xml
Shape="ShapeFilePathLocation"
```

For example:

XML Copy

```xml
<Nodes>
   <Node Id="MyNode" Background="#FF008000" Stroke="#FF000000"
   Foreground="#FFFFFFFF" Icon="...\Icons\Globe.png"/>
</Nodes>
```

To apply a custom style to a single link

1. Open the .dgml file in a text or XML editor.
2. Find the `<Link/>` element that contains both the names of the source code element and target code element.
3. In the `<Link/>` element, add any of the following attributes to customize its style:

   Outline and arrowhead color

   XML Copy

   ```xml
   Stroke="ColorNameOrHexadecimalValue"
   ```

   Outline thickness

   XML Copy

```
StrokeThickness="StrokeValue"
```

Outline style

XML Copy

```xml
StrokeDashArray="StrokeArrayValues"
```

For example:

XML Copy

```xml
<Links>
   <Link Source="MyFirstNode" Target="MySecondNode" Background="Green"
Stroke="#FF000000" StrokeDashArray="2,2"/>
</Links>
```

To apply custom styles to a group of code elements or links

1. Open the .dgml file in a text or XML editor.
2. If a `<Styles></Styles>` element does not exist, add one under the `<DirectedGraph></DirectedGraph>` element after the `<Links></Links>` element.
3. In the `<Styles></Styles>` element, under the `<Style/>` element and specify the following attributes:
   - `TargetType="Node | Link | Graph"`
   - `GroupLabel="` *NameInLegendBox* `"`
   - `ValueLabel="` *NameInStylePickerBox* `"`

   To apply a custom style to all target types, do not use a condition.

To apply a conditional style to groups of code elements or links

1. Open the .dgml file in a text or XML editor.
2. In the `<Style/>` element, add a `<Condition/>` element that contains an `Expression` attribute to specify an expression that returns a Boolean value.

For example:

XML Copy

```xml
<Condition Expression="MyCategory"/>
```

- or -

XML Copy

```xml
<Condition Expression="MyCategory > 100"/>
```

- or -

XML Copy

```xml
<Condition Expression="HasCategory('MyCategory')"/>
```

This expression uses the following Backus-Naur Form (BNF) syntax:

<Expression> ::= <BinaryExpression> | <UnaryExpression> | "("<Expression>")" | <MemberBindings> | <Literal> | <Number>

<BinaryExpression> ::= <Expression> <Operator> <Expression>

<UnaryExpression> ::= "!" <Expression> | "+" <Expression> | "-" <Expression>

<Operator> ::= "<" | "<=" | "=" | ">=" | ">" | "!=" | "or" | "and" | "+" | "*" | "/" | "-"

<MemberBindings> ::= <MemberBindings> | <MemberBinding> "." <MemberBinding>

<MemberBinding> ::= <MethodCall> | <PropertyGet>

<MethodCall> ::= <Identifier> "(" <MethodArgs> ")"

<PropertyGet> ::= Identifier

<MethodArgs> ::= <Expression> | <Expression> "," <MethodArgs> | <empty>

<Identifier> ::= [^.]*

<Literal> ::= single or double-quoted string literal

<Number> ::= string of digits with optional decimal point

You can specify multiple `<Condition/>` elements, which must all be true to apply the style.

3. On the next line after the `<Condition/>` element, add one or multiple `<Setter/>` elements to specify a `Property` attribute and a fixed `Value` attribute or a computed `Expression` attribute to apply to the map, code elements, or links that meet the condition.

   For example:

   XML Copy

   ```xml
   <Setter Property="BackGround" Value="Green"/>
   ```

   As a simple complete example, the following condition specifies that a code element appears green or red based on whether its `Passed` category is set to `True` or `False`:

XML Copy

```xml
<?xml version="1.0" encoding="utf-8"?>
<DirectedGraph xmlns="http://schemas.microsoft.com/vs/2009/dgml">
   <Nodes>
      <Node Id="MyFirstNode" Passed="True" />
      <Node Id="MySecondNode" Passed="False" />
   </Nodes>
   <Links>
   </Links>
   <Styles>
      <Style TargetType="Node" GroupLabel="Passed" ValueLabel="True">
         <Condition Expression="Passed='True'"/>
         <Setter Property="Background" Value="Green"/>
      </Style>
      <Style TargetType="Node" GroupLabel="Passed" ValueLabel="False">
         <Condition Expression="Passed='False'"/>
         <Setter Property="Background" Value="Red"/>
      </Style>
   </Styles>
</DirectedGraph>
```

The following table includes some example conditions that you can use:

Set the font size as a function of the number of lines of code, which also changes the size of the code element. This example uses a single conditional expression to set multiple properties, `FontSize` and `FontFamily`.

XML Copy

```xml
<?xml version="1.0" encoding="utf-8"?>
<DirectedGraph xmlns="http://schemas.microsoft.com/vs/2009/dgml">
<Nodes>
   <Node Id="Class1" LinesOfCode ="200" />
   <Node Id="Class2" LinesOfCode ="1000" />
   <Node Id="Class3" LinesOfCode ="20" />
</Nodes>
<Properties>
   <Property Id="LinesOfCode" Label="LinesOfCode" Description="LinesOfCode"
DataType="System.Int32" />
</Properties>
<Styles>
   <Style TargetType="Node" GroupLabel="LinesOfCode" ValueLabel="Function">
      <Condition Expression="LinesOfCode > 0" />
      <Setter Property="FontSize"
Expression="Math.Max(9,Math.Sqrt(LinesOfCode))" />
      <Setter Property="FontFamily" Value="Papyrus" />
   </Style>
</Styles>
</DirectedGraph>
```

Set the background color of a code element based on the `Coverage` property. The styles are evaluated in the order that they appear, similar to `if-else` statements.

In this example:

1. If `Coverage` is > 80, then set the `Background` property to green.

2. Else if `Coverage` is > 50, then set the `Background` property to a shade of orange based on the value of the `Coverage` property.

3. Else set the `Background` property to a shade of red based on the value of the `Coverage` property.

XML Copy

```xml
<?xml version="1.0" encoding="utf-8"?>
<DirectedGraph xmlns="http://schemas.microsoft.com/vs/2009/dgml">
<Nodes>
   <Node Id="Class1" Coverage="58" />
   <Node Id="Class2" Coverage="95" />
   <Node Id="Class3" Coverage="32" />
</Nodes>
```

```xml
<Properties>
   <Property Id="Coverage" Label="Coverage" Description="Code coverage as a
percentage of blocks" DataType="Double" />
</Properties>
<Styles>
   <Style TargetType="Node" GroupLabel="Coverage" ValueLabel="Good">
      <Condition Expression="Coverage > 80" />
      <Setter Property="Background" Value="Green" />
   </Style>
   <Style TargetType="Node" GroupLabel="Coverage" ValueLabel="OK">
      <Condition Expression="Coverage > 50" />
      <Setter Property="Background" Expression="Color.FromRgb(180 * Math.Max(1,
(80 - Coverage) / 30), 180, 0)" />
   </Style>
   <Style TargetType="Node" GroupLabel="Coverage" ValueLabel="Bad">
      <Setter Property="Background" Expression="Color.FromRgb(180, 180 *
Coverage / 50, 0)" />
   </Style>
</Styles>
</DirectedGraph>
```

Set the `Shape` property to `None` so that the icon replaces the shape. Use the `Icon` property to specify the location of the icon.

XML Copy

```xml
<DirectedGraph xmlns="http://schemas.microsoft.com/vs/2009/dgml">
<Nodes>
   <Node Id="Automation" Category="Test" Label="Automation" />
   <Node Id="C# Provider" Category="Provider" Label="C# Provider" />
</Nodes>
<Categories>
   <Category Id="Provider" Icon="...\Icons\Module.png" Shape="None" />
   <Category Id="Test" Icon="...\Icons\Page.png" Shape="None" />
</Categories>
<Properties>
   <Property Id="Icon" DataType="System.String" />
   <Property Id="Label" Label="Label" Description="Displayable label of an
Annotatable object" DataType="System.String" />
   <Property Id="Shape" DataType="System.String" />
</Properties>
<Styles>
   <Style TargetType="Node" GroupLabel="Group" ValueLabel="Has category">
      <Condition Expression="HasCategory('Group')" />
      <Setter Property="Background" Value="#80008080" />
   </Style>
   <Style TargetType="Node">
      <Setter Property="HorizontalAlignment" Value="Center" />
   </Style>
```

```
</Styles>
</DirectedGraph>
```

Assign properties to code elements and links

You can organize code elements and links by assigning properties to them. For example, you can select code elements that have specific properties so that you can group them, change their style, or hide them.

To assign a property to a code element

1. Open the .dgml file in a text or XML editor.

2. Find the `<Node/>` element for that code element. Specify the name of the property and its value. For example:

   XML Copy

   ```xml
   <Nodes>
      <Node Id="MyNode" MyPropertyName="PropertyValue" />
   </Nodes>
   ```

3. Add a `<Property/>` element to the `<Properties>` section to specify attributes such as its visible name and data type:

   XML Copy

   ```xml
   <Properties>
      <Property Id="MyPropertyName" Label="My Property"
   DataType="System.DataType"/>
   </Properties>
   ```

To assign a property to a link

1. Open the .dgml file in a text or XML editor.

2. Find the `<Link/>` element that contains both the names of the source code element and target code element.

3. In the `<Node/>` element, specify the name of the property and its value. For example:

XML Copy

```xml
<Links>
   <Link Source="MyFirstNode" Target="MySecondNode"
MyPropertyName="PropertyValue" />
</Links>
```

4. Add a `<Property/>` element to the `<Properties>` section to specify attributes such as its visible name and data type:

XML Copy

```xml
<Properties>
   <Property Id="MyPropertyName" Label="My Property Name"
DataType="System.DataType"/>
</Properties>
```

Assign categories to code elements and links

The following sections demonstrate how you can organize code elements by assigning categories to them, and how you can create hierarchical categories that help you organize code elements and add attributes to child categories by using inheritance.

To assign a category to a code element

- Open the .dgml file in a text or XML editor.
- Find the `<Node/>` element for the code element that you want.
- In the `<Node/>` element, add a `Category` attribute to specify the name of the category. For example:

XML Copy

```xml
<Nodes>
   <Node Id="MyNode" Category="MyCategory" />
</Nodes>
```

Add a `<Category/>` element to the `<Categories>` section so that you can use the `Label` attribute to specify the display text for that category:

XML Copy

```xml
<Categories>
   <Category Id="MyCategory" Label="My Category" />
</Categories>
```

To assign a category to a link

1. Open the .dgml file in a text or XML editor.

2. Find the `<Link/>` element that contains both the names of the source code element and target code element.

3. In the `<Link/>` element, add a `Category` attribute to specify the name of the category. For example:

   XML Copy

   ```xml
   <Links>
      <Link Source="MyFirstNode" Target="MySecondNode" Category="MyCategory"
   </Links>
   ```

4. Add a `<Category/>` element to the `<Categories>` section so that you can use the `Label` attribute to specify the display text for that category:

   XML Copy

   ```xml
   <Categories>
      <Category Id="MyCategory" Label="My Category" />
   </Categories>
   ```

To create hierarchical categories

1. Open the .dgml file in a text or XML editor.

2. Add a `<Category/>` element for the parent category, and then add the `BasedOn` attribute to the child category's `<Category/>` element.

   For example:

   XML Copy

```
<Nodes>
   <Node Id="MyFirstNode" Label="My First Node" Category= "MyCategory" />
   <Node Id="MySecondNode" Label="My Second Node" />
</Nodes>
<Links>
   <Link Source="MyFirstNode" Target="MySecondNode" />
</Links>
<Categories>
   <Category Id="MyCategory" Label="My Category"
BasedOn="MyParentCategory"/>
   <Category Id="MyParentCategory" Label="My Parent Category"
Background="Green"/>
</Categories>
```

In this example, the background of `MyFirstNode` is green because its `Category` attribute inherits the `Background` attribute of `MyParentCategory`.

Link documents or URLs to code elements and links

You can link documents or URLs to code elements or to links by editing the map's .dgml file and adding a `Reference` attribute to the `<Node/>` element for a code element or the `<Link/>` element for a link. You can then open and view that content from the code element or link. The `Reference` attribute specifies the path of that content. This can be a path relative to the location of the .dgml file or an absolute path.

Caution

If you use relative paths, and the .dgml file is moved to a different location, then those paths will no longer resolve. When you try to open and view the linked content, an error stating that the content cannot be viewed will occur.

For example, you might want to link the following code elements:

- To describe the changes to a class, you might link the URL of a work code element, document, or another .dgml file to the code element for a class.
- You might link a dependency diagram to a group code element that represents a layer in the software's logical architecture.

- To show more information about a component that exposes an interface, you might link a component diagram to the code element for that interface.
- Link a code element to a Team Foundation Server work item or bug, or some other information that is related to the code element.

To link a document or URL to a code element

1. Open the .dgml file in a text or XML editor.
2. Find the `<Node/>` element for the code element that you want.
3. Perform one of the tasks in the following table:

   A single code element

   - In the `<Node/>` or `<Link/>` element, add a `Reference` attribute to specify the location of the code element.

     Note

     You can have only one `Reference` attribute per element.

     For example:

XML Copy

```xml
<Nodes>
   <Node Id="MyNode" Reference="MyDocument.txt" />
</Nodes>
<Properties>
   <Property Id="Reference" Label="My Document" DataType="System.String"
IsReference="True" />
</Properties>
```

   Multiple code elements

   a. In the `<Node/>` or `<Link/>` element, add a new attribute to specify the location of each reference.

   b. In the `<Properties>` section:

      i. Add a `<Property/>` element for each new type of reference.

ii. Set the `Id` attribute to the name of the new reference attribute.

iii. Add the `IsReference` attribute and set it to `True` to make the reference appear on the code element's **Go To Reference** shortcut menu.

iv. Use the `Label` attribute to specify the display text on the code element's **Go To Reference** shortcut menu.

For example:

XML Copy

```xml
<Nodes>
   <Node Id="MyNode" SequenceDiagram="MySequenceDiagram.sequencediagram"
ActiveBugs="MyActiveBugs.wiq"/>
</Nodes>
<Properties>
   <Property Id="SequenceDiagram" Label="My Sequence Diagram"
DataType="System.String" IsReference="True" />
   <Property Id="ActiveBugs" Label="Active Bugs" DataType="System.String"
IsReference="True" />
</Properties>
```

On the map, the name of the code element appears underlined. When you open the shortcut menu for the code element or the link, you will see a **Go To Reference** shortcut menu that contains the linked code elements for you to choose.

4. Use the `ReferenceTemplate` attribute to specify a common string, such as a URL, that is used by multiple references instead of repeating that string in the reference.

   The `ReferenceTemplate` attribute specifies a placeholder for the value of the reference. In the following example, the `{0}` placeholder in the `ReferenceTemplate` attribute will be replaced by the values of the `MyFirstReference` and `MySecondReference` attributes in the `<Node/>` element to produce a full path:

XML Copy

```xml
<Nodes>
   <Node Id="MyNode" MyFirstReference="MyFirstDocument"
MySecondReference="MySecondDocument"/>
   <Node Id="MySecondNode" MyFirstReference="AnotherFirstDocument"
MySecondReference="AnotherSecondDocument"/>
</Nodes>
<Properties>
   <Property Id="MyFirstReference" Label="My First Document"
```

```
DataType="System.String" IsReference="True"
ReferenceTemplate="http://www.Fabrikam.com/FirstDocuments/{0}.asp"/>
   <Property Id="MySecondReference" Label="My Second Document"
DataType="System.String" IsReference="True" ReferenceTemplate="
http://www.Fabrikam.com/SecondDocuments/{0}.asp"/>
</Properties>
```

5. To view the referenced code element or code elements from the map, open the shortcut menu for the code element or the link. Choose **Go To Reference** and then the code element.

See also

- Map dependencies across your solutions
- Use code maps to debug your applications
- Find potential problems using code map analyzers
- Browse and rearrange code maps
- Directed Graph Markup Language (DGML) reference