CAPLIN

Caplin Trader Client 1.4

Composite Component Configuration XML Reference

July 2009

Contents

1 Preface

1.1 What this document contains

This reference document describes the XML-based configuration that defines the layout and functionality of the Composite Display Component in Caplin Trader Client.

The information in this document applies to Caplin Trader version 1.4.

About Caplin document formats

This document is supplied in three formats:

- Portable document format (*.PDF* file), which you can read on-line using a suitable PDF reader such as Adobe Reader®. This version of the document is formatted as a printable manual; you can print it from the PDF reader.
- Web pages (*.HTML* files), which you can read on-line using a web browser. To read the web version of the document navigate to the *HTMLDoc_m_n* folder and open the file *index.html*.
- Microsoft HTML Help (*.CHM* file), which is an HTML format contained in a single file.
  To read a *.CHM* file just open it – no web browser is needed.

For the best reading experience

On the machine where your browser or PDF reader runs, install the following Microsoft Windows® fonts: Arial, Courier New, Times New Roman, Tahoma. You must have a suitable Microsoft license to use these fonts.

Restrictions on viewing .CHM files

You can only read *.CHM* files from Microsoft Windows.

Microsoft Windows security restrictions may prevent you from viewing the content of *.CHM* files that are located on network drives. To fix this either copy the file to a local hard drive on your PC (for example the Desktop), or ask your System Administrator to grant access to the file across the network. For more information see the Microsoft knowledge base article at
http://support.microsoft.com/kb/896054/.

1.2 Who should read this document

This document is intended for System Administrators and Software Developers who need to compose display components in Caplin Trader Client.

1.3 Related documents

- **Caplin Trader Client: Tree View Configuration XML Reference**

  Describes the XML-based configuration that defines the layout and functionality of the Tree Views displayed in Caplin Trader Client.
- **Caplin Trader Client: Grid Configuration XML Reference**

  Describes the XML-based configuration that defines the layout and functionality of Grids displayed in Caplin Trader Client.

- **Caplin Trader Client: Simple Form Component Configuration Reference**

  Describes the XML-based configuration that defines the layout and functionality of the Simple Form Display Component in Caplin Trader Client.
- **Caplin Trader Client: How To Create A Product Finder Composite Component**

  Explains how to create a Product Finder using the Caplin Trader Client Composite Component. It refers to the Product Finder in the Caplin Trader Client reference implementation.
- **Caplin Trader Client: Customizing The Appearance**

  Describes how to configure the on-screen layout and 'look and feel' of Caplin Trader Client.
- **Caplin Trader Client: Layout Configuration XML Reference**

  Describes the XML-based configuration that defines the layout and appearance of Caplin Trader Client through webcentric.

1.4 Typographical conventions

The following typographical conventions are used to identify particular elements within the text.

Type	*Uses*
aMethod	Function or method name
aParameter	Parameter or variable name
/AFolder/Afile.txt	File names, folders and directories
`Some code;`	Program output and code examples
The `value=10` attribute is...	Code fragment in line with normal text
Some text in a dialog box	Dialog box output
`Something typed in`	User input – things you type at the computer keyboard
XYZ Product Overview	Document name
◆	Information bullet point
■	Action bullet point – an action you should perform

Note: Important Notes are enclosed within a box like this.
Please pay particular attention to these points to ensure proper configuration and operation of the solution.

Tip: Useful information is enclosed within a box like this.
Use these points to find out where to get more help on a topic.

1.5 Feedback

Customer feedback can only improve the quality of our product documentation, and we would welcome any comments, criticisms or suggestions you may have regarding this document.

Please email your feedback to documentation@caplin.com.

1.6 Acknowledgments

Adobe® Reader is a registered trademark of Adobe Systems Incorporated in the United States and/or other countries.

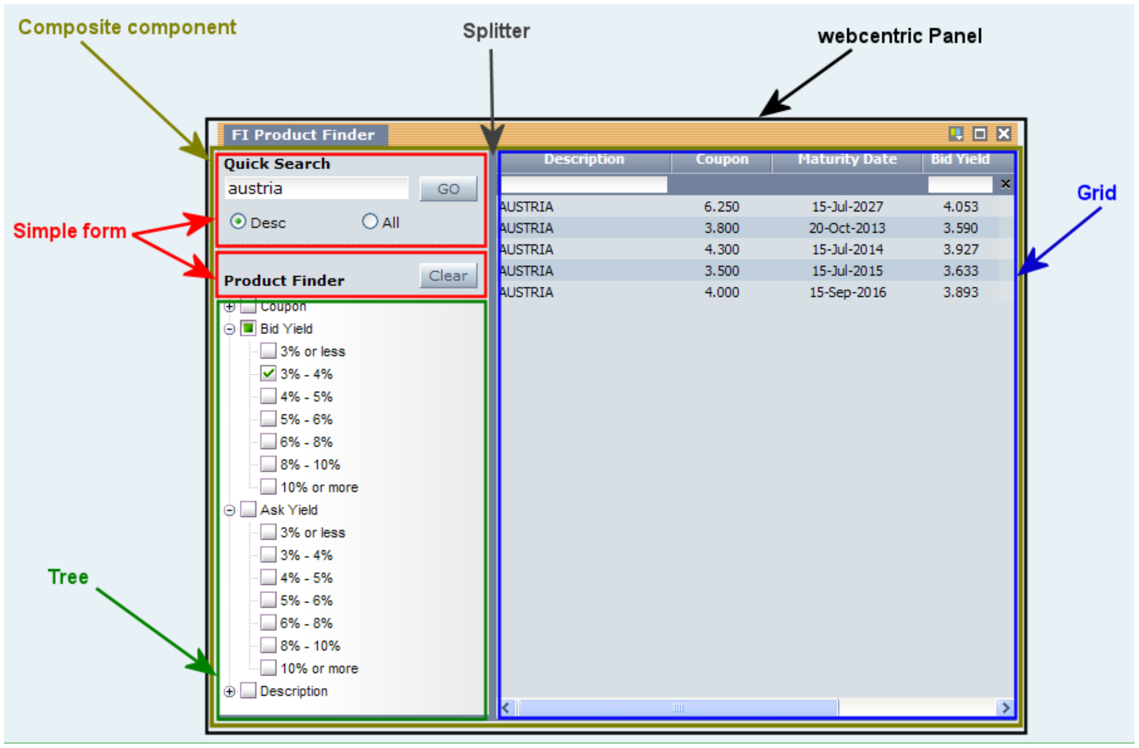
Windows is a registered trademark of Microsoft Corporation in the United States and other countries.

2 Introduction to the Composite Component

In Caplin Trader Client, a Composite Display Component is used to group together other display components, and to control and coordinate the behavior of these components in response to actions by the end user.

A typical example is the Product Finder, which allows an end user to filter the instruments that a grid displays by entering information into a 'quick search' text box and selecting options in a Product Finder tree.

The following picture shows the Product Finder Composite Component, identifying the display components and other elements that are used to construct it.



Description	Coupon	Maturity Date	Bid Yield
AUSTRIA	6.250	15-Jul-2027	4.053
AUSTRIA	3.800	20-Oct-2013	3.590
AUSTRIA	4.300	15-Jul-2014	3.927
AUSTRIA	3.500	15-Jul-2015	3.633
AUSTRIA	4.000	15-Sep-2016	3.893

Composite Component configured as a Product Finder

In the example above, if the end user types "Austria" (upper or lower case) in the 'Quick Search' box, then when Go is selected the grid only displays instruments with "AUSTRIA" in the 'Description' field. If the end user then selects a 'Bid Yield' of '3% - 4% in the 'Product Finder' tree, the grid only displays instruments with "AUSTRIA" in the 'Description' field and a 'Bid Yield' of 3% to 4%.

A custom "Product Search Controller" JavaScript class filters the instruments in the grid according to the content of the 'Quick Search' box and the selected 'Product Finder' nodes.

3 How the configuration works

The following sections explain how the various XML tags may be combined to define Composite Display Components for Caplin Trader Client.

3.1 An example configuration explained

Two example XML files describing a Product Finder Composite Display Component are shown below.

- *fi_product_search.xml*: Configures a Composite Display Component as a Product Finder for FI instruments.
- *CompositeComponentControllerMappings.xml*: Maps a set of Composite Display Component controllers to JavaScript classes; one of these controllers is used in the FI Product Finder.

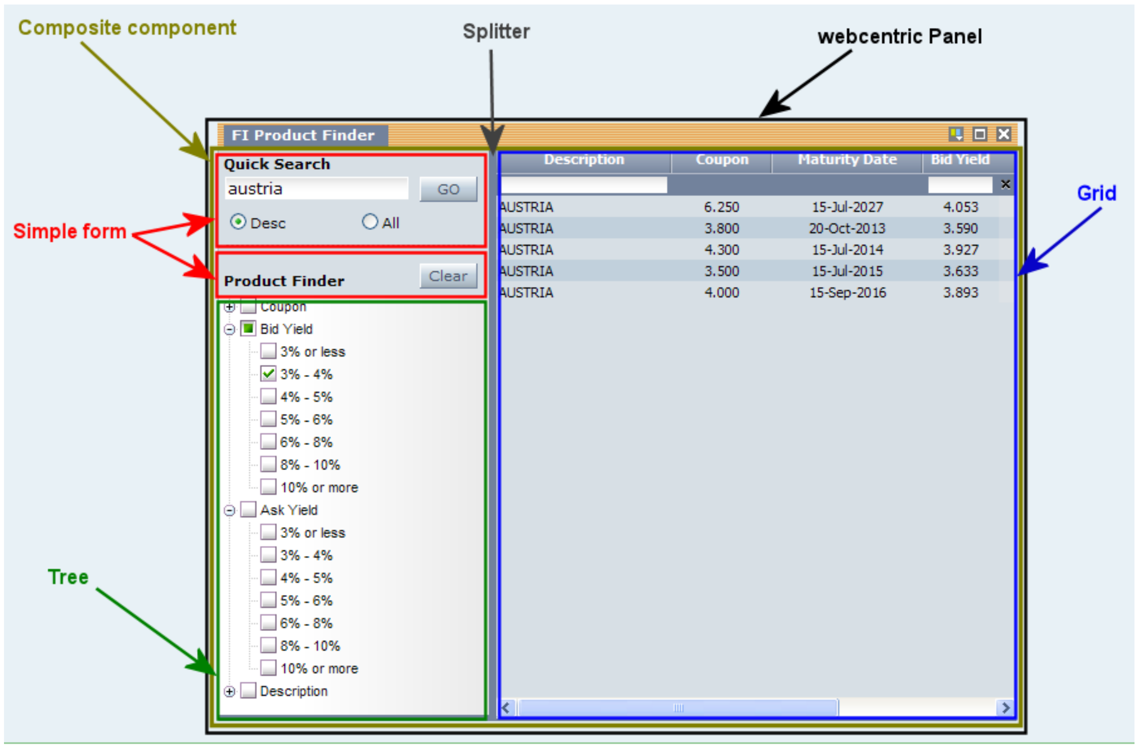
fi_product_search.xml: **XML that configures a Composite Display Component as a Product Finder**

```
<Panel xmlns:caplin="http://www.caplin.com" caption="FI Product Finder"
              drop_target="SNAP_FRAMEITEM" colour="colour-1" height="521" width="300"
              top="157" left="403" pkey="/CONTAINER/FI/ALL::VIEW">
   <Decorators xref="Declarations/Decorators[@id='basicDecorator']" />
   <state>
      <compositeComponent controller="FiProductFinder">
         <terrace>
            <tower width="215px">
               <component id="fiQuickSearch" height="88px">
                  <simpleForm src="source/html-templates/quick-search.html" />
               </component>

               <component id="productFinderButtons" height="30px">
                  <simpleForm src="source/html-templates/product-finder-buttons.html" />
               </component>

               <component id="fiTree">
                  <tree baseTemplate="FIProductSearchTree" />
               </component>
            </tower>

            <splitter />

            <component id="fiGrid">
               <grid baseTemplate="fiProductSearchGrid">
                  <gridRowModel>
                     <rttpContainerGridDataProvider container="/CONTAINER/FI/ALL" />
                  </gridRowModel>
               </grid>
            </component>
         </terrace>
      </compositeComponent>
   </state>
</Panel>
```

CompositeComponentControllerMappings.xml:
XML that maps Composite Display Component controllers to JavaScript classes

```
<controllerMappings>
   <controllerMapping id="FiProductFinder"
                      className="caplinx.composite.FiProductFinderController" />
   <controllerMapping id="TimeSeriesController"
                      className="caplinx.historicdata.TimeSeriesController" />
   <controllerMapping id="MarketOverviewController"
                      className="caplinx.historicdata.MarketOverviewController" />
   <controllerMapping id="BenchmarkController"
                      className="caplinx.historicdata.BenchmarkController" />
   <controllerMapping id="PortfolioManagerController"
                      className="caplinx.composite.PortfolioManagerController" />
</controllerMappings>
```

These configuration items define, between them, a Composite Component that consists of two simple forms (a 'Quick Search' box and a 'Product Finder' clear button), a tree, and a product grid.



Description	Coupon	Maturity Date	Bid Yield
AUSTRIA	6.250	15-Jul-2027	4.053
AUSTRIA	3.800	20-Oct-2013	3.590
AUSTRIA	4.300	15-Jul-2014	3.927
AUSTRIA	3.500	15-Jul-2015	3.633
AUSTRIA	4.000	15-Sep-2016	3.893

Composite Component configured as a Product Finder

An explanation of the example XML configuration

In the code samples that follow, text shown as (`...`) represents code fragments that have been omitted for simplicity. Here is an explanation of what the example XML configuration contains and how this relates to what the end-user sees on the screen.

In *fi_product_search.xml*:

- **`<Panel>`** The webcentric `<Panel>` tag is the container for the `<compositeComponent>`.
  The `caption="FI Product Finder"` attribute of the `<Panel>` tag defines the title of the Product Finder Composite Component. The other attributes of the `<Panel>` tag and `<Decorators>` tag define the behaviour and appearance of the panel when it is rendered.

```
<Panel ...>
  <Decorators .../>
  <state>
    <compositeComponent>
      ...
    </compositeComponent>
  </state>
</Panel>
```

The XML between the `<state>` and `</state>` tags defines the content that will be placed in the panel, and the `<compositeComponent>` tag identifies this content to be a Composite Component.

Note: There can only be one `<compositeComponent>` tag inside a `<Panel>` tag, which means that there can only be one Composite Component inside a panel.

For a definition of the `<Panel>` tag, see **Caplin Trader Client: Layout Configuration XML Reference**.

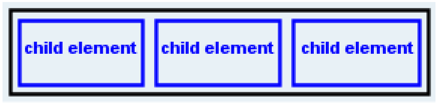
For an explanation of the webcentric Panel, see **Caplin Trader Client: Customizing The Appearance**.

- **`<compositeComponent>`** Contains a single `<terrace>` tag and starts the Composite Component definition.

```
<compositeComponent controller="FiProductFinder">
  <terrace>
    ...
  <terrace>
</compositeComponent>
```

`controller="FiProductFinder":`
This is the logical name (`id`) of a controller – the controller is a JavaScript class that controls the Composite Component. In this case the controller filters the instruments displayed in the grid according to the nodes that are selected in the 'Product Finder' tree, and the text that is typed into the 'Quick Search' box. The actual JavaScript class used is defined in the second XML file *CompositeComponentControllerMappings.xml*.

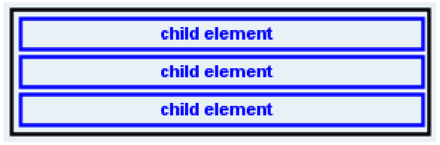
- **`<terrace>`** A layout element that arranges its children horizontally from left to right.



```
<terrace>
  <tower width="215px">
    ...
  </tower>
  <splitter />
  <component>
    ...
  </component>
</terrace>
```

The `<terrace>` contains a `<tower>`, a `<splitter>`, and a `<component>`. The `<terrace>` does not (and must not) have a width or height specified, because it is the outermost layout element of the Composite Component; its width and height are determined at run time.

- **`<tower>`** A layout element that arranges its children vertically from top to bottom.



```
<tower width="215px">
  <component id="fiQuickSearch" height="88px">
    ...
  </component>
  <component id="productFinderButtons" height="30px">
    <simpleForm src="source/html-templates/product-finder-buttons.html" />
  </component>
  <component id="fiTree">
    ...
  </component>
</tower>
```

This tower contains three components
(`id="fiQuickSearch"`, `id="productFinderButtons"`, and `id="fiTree"`).

- **`<component>`** A container for display components.

```
<tower width="215px">
  <component id="fiQuickSearch" height="88px">
    <simpleForm src="source/html-templates/quick-search.html" />
  </component>
  <component id="productFinderButtons" height="30px">
    <simpleForm src="source/html-templates/product-finder-buttons.html" />
  </component>
  <component id="fiTree">
    <tree baseTemplate="FiProductSearchTree" />
  </component>
</tower>

<splitter/>

<component id="fiGrid">
  <grid baseTemplate="fiProductSearchGrid">
  ...
  </grid>
</component>
```

The top component of the terrace (`id="fiQuickSearch"`) is 88 pixels high (`height="88px"`) and contains a simple HTML form (`<simpleForm>`) that provides the 'Quick Search' box and Go button. The attribute `src="source/html-templates/quick-search.html"` identifies the file that contains the XHTML for the form.
See **Caplin Trader Client: Simple Form Component Configuration Reference** for reference information on simple form configuration.

The middle component of the terrace (`id="productFinderButtons"`) is 30 pixels high (`height="30px"`) and contains another simple HTML form (`<simpleForm>`) that provides a `Clear` button. The attribute `src="source/html-templates/product-finder-buttons.html"` identifies the file that contains the XHTML for the form.

The bottom component of the terrace (`id="fiTree"`) contains the 'Product Finder' tree and takes up the remaining height of the parent tower. The tree is constructed from a base template (`<tree baseTemplate="FiProductSearchTree" />`).
See **Caplin Trader Client: Tree View Configuration XML Reference** for reference information on tree configuration.

The component at the right hand side of the terrace (`id="fiGrid"`) contains the product grid.
The grid is constructed from a grid template (`<grid baseTemplate="fiProductSearchGrid">`).
See **Caplin Trader Client: Grid Configuration XML Reference** for reference information on grid configuration.

- **`<splitter>`** Defines a bar that can be used to space display components apart. In this case it provides a gray vertical bar between the grid and the elements in the tower (two simple forms and a tree).

In ***CompositeComponentControllerMappings.xml***:

- **`<controllerMapping>`** maps the logical name of the Composite Component's controller (`id="FiProductFinder"`) to the JavaScript class that implements the controller (`className="caplinx.composite.FiProductFinderController"`).

```
<controllerMappings>
   <controllerMapping id="FiProductFinder"
                      className="caplinx.composite.FiProductFinderController" />
...
</controllerMappings>
```

The *CompositeComponentControllerMappings.xml* file contains the mappings for all the Composite Component controller classes used in Caplin Trader Client. This allows the class that implements a particular controller to be replaced by another one, without needing to modify the Composite Display Component configuration XML that refers to the controller.

Also see Composite Component Configuration Reference 11 and Controller Mappings Reference 18.

3.2 Technical assumptions and restrictions

XML

The XML markup defined in this document conforms to XML version 1.0 and the XML schema version defined at
http://www.w3.org/2001/XMLSchema.

4 Composite Component Configuration Reference

This is the reference information for the configuration XML that describes the Composite Display Component.

4.1 Ordering and nesting of tags

Each top level tag of the composite display configuration XML is shown below, together with the child tags that it can typically contain (the children are in no particular order).

Tip: Advanced users may wish to consult the Relax NG Schema (*compositeComponent.rnc*) for definitive information on the ordering and nesting of tags.

For a description of each tag and its attributes, see the Composite Display XML Reference 13 section.

<compositeComponent>

This is the outermost tag
```
<compositeComponent> (one only of the following)
   <tower></tower>
   <terrace></terrace>
   <component></component>
</compositeComponent>
```

<tower>

```
<tower> (at least one of the following)
   <terrace></terrace>  (zero or more)
   <tower></tower>  (zero or more)
   <component></component>  (zero or more)
   <splitter />  (zero or more)
</tower>
```

<terrace>

```
<terrace> (at least one of the following)
   <terrace></terrace>  (zero or more)
   <tower></tower>  (zero or more)
   <component></component>  (zero or more)
   <splitter />  (zero or more)
</terrace>
```

<component>

`<component>` (one only of the following)
   `<grid></grid> (see` **Caplin Trader Client: Grid Configuration XML Reference**`)`
   `<tree></tree> (see` **Caplin Trader Client: Tree View Configuration XML Reference**`)`
   `<simpleForm /> (see` **Caplin Trader Client: Simple Form Component Configuration Reference**`)`
`</component>`

<splitter>

`<splitter />` (no children)

4.2 Composite Display XML Reference

This section describes the XML tags that you can use to configure a Composite Display Component.

Note: The Composite Display Component configuration has XML tags called `<terrace>` and `<tower>`. The webcentric configuration defined in **Caplin Trader Client: Layout Configuration XML Reference** has similar tags called `<Terrace>` and `<Tower>`.
Although these two sets of tags have similar effects on the appearance of Caplin Trader Client, they are distinct and operate at different levels.
Always use the `<terrace>` and `<tower>` tags to configure a Composite Component; *never* use the `<Terrace>` or `<Tower>` tags.

Default attribute values

In the tables that follow, if an attribute is not required (Req? = 'N') and there is a default value specified, then not supplying the attribute is equivalent to setting the attribute to this default value. If an attribute is not required and the default is '(none)', then not supplying the attribute can result in one of two behaviors, depending on the particular attribute – either the behavior is as specified in the description column of the table, or there is no effect on the appearance or behavior of the component.

<component>

`<component>`

A component is a container for a registered display component, such as a <grid>, a <tree>, or a <simpleForm>, and must contain only one of these tags.

Attributes:

Name	Type	Default	Req?	Description
`height`	string	(none)	N	The height of the component as a positive integer, suffixed by 'px' (for pixels) or '%' (for percentage of the parent element's height). Omit this attribute if the parent is a <terrace>. If the parent is a <tower> and the component's 'height' attribute is omitted, the component's height is determined by the height of the tower and the height of the other elements inside the tower.
`id`	string	(none)	Y	An identifier for this component. The identifier must be unique across all Composite Components (<compositeComponent>).
`preventSerialization`	boolean	(none)	Y	When this attribute is set to "true", changes made to the Composite Component at run time are not saved when the component is serialized; when "false", any such changes are saved.

Name	Type	Default	Req?	Description
`width`	string	(none)	N	The width of the component as a positive integer, suffixed by 'px' (for pixels) or '%' (for percentage of the parent element's width). Omit this attribute if the parent is a <tower>. If the parent is a <terrace> and the component's 'width' attribute is omitted, the component's width is determined by the width of the terrace and the width of the other elements inside the terrace.

<compositeComponent>

`<compositeComponent>`

The outermost tag of the Composite Component definition XML. This tag must be placed inside a webcentric <Panel> tag, and must contain only one child tag. Valid children are the <tower>, <terrace>, and <component> tags.

Attributes:

Name	Type	Default	Req?	Description
`controller`	string	(none)	Y	CAPLIN TRADER 1.4.5 AND ABOVE: The logical name (id) of the Composite Component's controller. This name must match the id attribute of a <controllerMapping> tag in the Controller Mappings XML for the Composite Component (for example, in CompositeComponentControllerMappings. xml). The mapping XML associates the logical controller name with the name of a class that implements the controller. CAPLIN TRADER 1.4.4 AND BELOW: The fully qualified name of the JavaScript class that controls the Composite Component. This class must implement the "caplin.component.composite. CompositeComponentController" interface of the Caplin Trader Client API. For backwards compatibility, this usage is also supported in Caplin Trader 1.4.5 and above, but is DEPRECATED.

<splitter>

`<splitter>`

A splitter is a horizontal or vertical bar that can be inserted between components (<component>), terraces (<terrace>), and towers (<tower>) to space them apart. A splitter has no children.

Attributes:

Name	Type	Default	Req?	Description
`height`	string	'6px'	N	The height of the splitter as a positive integer, suffixed by 'px' (for pixels) or '%' (for percentage of the parent element's height). Omit this attribute if the splitter is inserted in a <terrace>. In a terrace, the height of the splitter is the height of the terrace.
`id`	string	(none)	N	An identifier for this splitter. The identifier must be unique within the scope of the Composite Component (<compositeComponent>).
`width`	string	'6px'	N	The width of the splitter as a positive integer, suffixed by 'px' (for pixels) or '%' (for percentage of the parent element's width). Omit this attribute if the splitter is inserted in a <tower>. In a tower, the width of the splitter is the width of the tower.

<terrace>

`<terrace>`

A terrace is used to arrange child components horizontally from left to right, and must contain at least one child. Valid children are the <terrace>, <tower>, <component>, and <splitter>. At least one of the children inside a terrace must have its width attribute omitted. You can set the width or height of a terrace, but not both.

Attributes:

Name	Type	Default	Req?	Description
`height`	string	(none)	N	The height of the terrace as a positive integer, suffixed by 'px' (for pixels) or '%' (for percentage of the parent element's height). Omit this attribute if the parent of the terrace is a <compositeComponent> or <terrace>. If the parent is a <tower> and the terrace's 'height' attribute is omitted, the terrace's height is determined by the height of the tower and the height of the other elements inside the tower.
`width`	string	(none)	N	The width of the terrace as a positive integer, suffixed by 'px' (for pixels) or '%' (for percentage of the parent element's width). Omit this attribute if the parent of the terrace is a <compositeComponent> or <tower>. If the parent is a <terrace> and the child terrace's 'width' attribute is omitted, the child terrace's width is determined by the width of the parent terrace and the width of the other elements inside the parent terrace.

<tower>

`<tower>`

A tower is used to arrange child components vertically on top of each other, and must contain at least one child. Valid children are the <tower>, <terrace>, <component>, and <splitter>. At least one of the children inside a tower must have its height attribute omitted. You can set the width or height of a tower, but not both.

Attributes:

Name	Type	Default	Req?	Description
`height`	string	(none)	N	The height of the tower as a positive integer, suffixed by 'px' (for pixels) or '%' (for percentage of the parent element's height). Omit this attribute if the parent of the tower is a <compositeComponent> or <terrace>. If the parent is a <tower> and the child tower's 'height' attribute is omitted, the child tower's height is determined by the height of the parent tower and the height of the other elements inside the parent tower.
`width`	string	(none)	N	The width of the tower as a positive integer, suffixed by 'px' (for pixels) or '%' (for percentage of the parent element's width). Omit this attribute if the parent of the tower is a <compositeComponent> or <tower>. If the parent is a <terrace> and the tower's 'width' attribute is omitted, the tower's width is determined by the width of the terrace and the width of the other elements inside the terrace.

5 Controller Mappings Reference

This is the reference information for the configuration XML that maps the identifier of a composite component controller to the class that implements the controller.

5.1 Ordering and nesting of tags

Each top level tag of the controller mappings configuration XML is shown below, together with the child tags that it can typically contain (the children are in no particular order).

Tip: Advanced users may wish to consult the Relax NG Schema (*compositeComponentControllerMappings.rnc*) for definitive information on the ordering and nesting of tags.

For a description of each tag and its attributes, see the Controller Mappings XML Reference 19 section.

<controllerMappings>

This is the outermost tag

```
<controllerMappings> (at least one of the following)
   <controllerMapping />
</controllerMappings>
```

<controllerMapping>

```
<controllerMapping /> (no children)
```

5.2 Controller Mappings XML Reference

This section describes the XML tags that map a reference to a composite component controller to the class that implements the controller.

Default attribute values

In the tables that follow, if an attribute is not required (Req? = 'N') and there is a default value specified, then not supplying the attribute is equivalent to setting the attribute to this default value. If an attribute is not required and the default is '(none)', then not supplying the attribute can result in one of two behaviors, depending on the particular attribute – either the behavior is as specified in the description column of the table, or there is no effect on the appearance or behavior of the component.

<controllerMapping>

`<controllerMapping>`

The <controllerMapping> tag maps the logical name (id) of a Composite Component's controller to the fully qualified name of a Javascript class that implements the controller.

Attributes:

Name	Type	Default	Req?	Description
`className`	string	(none)	Y	The fully qualified name of a JavaScript class that controls the Composite Component. This class must implement the "caplin.component.composite. CompositeComponentController" interface of the Caplin Trader Client API.
`id`	string	(none)	Y	The logical name (identifier) of the controller. This name is used in the XML that defines Composite Display Components.

<controllerMappings>

`<controllerMappings>`

The outermost tag of the Composite Component controller mappings XML. The <controllerMappings> tag simply acts as a container for <controllerMapping> tags.

Attributes: This tag has no attributes.

CAPLIN THE REAL TIME WEB COMPANY

Contact Us

Caplin Systems Ltd

Triton Court

14 Finsbury Square

London EC2A 1BR

Telephone: +44 20 7826 9600

Fax: +44 20 7826 9610

www.caplin.com