

# Caplin Trader Client 1.3

---

## How To Create A Product Finder Composite Component

February 2009

# Contents

<b>1</b>	<b>Preface.....</b>	<b>1</b>
1.1	What this document contains.....	1
	Use Cases .....	1
	About Caplin document formats .....	2
1.2	Who should read this document.....	2
1.3	Related documents.....	2
1.4	Typographical conventions.....	4
1.5	Feedback.....	4
1.6	Acknowledgments.....	4
1.7	Technical assumptions and restrictions.....	5
<b>2</b>	<b>Introduction to the product finder.....</b>	<b>6</b>
<b>3</b>	<b>Creating a product finder.....</b>	<b>7</b>
<b>4</b>	<b>Defining the layout of the product finder.....</b>	<b>9</b>
<b>5</b>	<b>Customizing the grid component.....</b>	<b>13</b>
5.1	Grid template "FI.PRODUCT.SEARCH".....	13
5.2	Grid template "FI".....	14
5.3	Grid template "All".....	15
5.4	Further reading about Grids.....	16
<b>6</b>	<b>Customizing the tree component.....</b>	<b>17</b>
6.1	The tree view template.....	17
	An explanation of the tree view XML configuration .....	19
6.2	Populating the tree dynamically.....	21
6.3	Further reading about trees.....	23
<b>7</b>	<b>Customizing the quick search filter.....</b>	<b>24</b>
7.1	XHTML for the quick search filter.....	24
7.2	XHTML for the Clear button.....	26
7.3	Further reading about Simple Forms.....	26
<b>8</b>	<b>Inserting a menu item for the product finder.....</b>	<b>27</b>
8.1	XML for the product finder menu item.....	28
<b>9</b>	<b>The product finder controller.....</b>	<b>29</b>
9.1	Changing filter behavior by modifying the product finder controller.....	29

**10 Glossary of terms and acronyms..... 30**

**Index..... 31**

# 1 Preface

## 1.1 What this document contains

This document 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.

### Use Cases

This document covers the following use cases.

Use Case	Discussed in Section
How do I change the title of the product finder composite component?	<a href="#">Defining the layout of the product finder</a> <sup>9</sup>
How do I define the layout of my composite component?	<a href="#">Defining the layout of the product finder</a> <sup>9</sup>
How do I add display components to the composite component layout?	<a href="#">Defining the layout of the product finder</a> <sup>9</sup>
How do I configure the fields that the quick search filters on?	<a href="#">XHTML for the Quick Search filter</a> <sup>24</sup>
How do I set-up my own data in the tree display component?	<a href="#">Customizing the tree component</a> <sup>17</sup>
How do I render icons in the tree display component?	<a href="#">An explanation of the tree view XML configuration</a> <sup>19</sup>
How do I dynamically populate the tree display component?	<a href="#">Populating the tree dynamically</a> <sup>21</sup>
How do I apply column header filters to the grid display component?	<a href="#">Grid template "FI"</a> <sup>14</sup>
How do I change the filter behavior of the product finder controller?	<a href="#">Changing the filter behavior</a> <sup>29</sup>
How do I insert a menu item for the Composite Component?	<a href="#">Inserting a menu item for the product finder</a> <sup>27</sup>

## 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 want to construct a composite component that functions as a product finder.

## 1.3 Related documents

- ◆ **Caplin Trader Client: API Reference**  
The API reference documentation provided with Caplin Trader Client. The classes and interfaces of this API allow you to extend the capabilities of Caplin Trader Client.
- ◆ **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: Customizing the Content**  
Describes how to add a grid (and other content) to the default layout of Caplin Trader Client.
- ◆ **Caplin Trader Client: Grid Configuration XML Reference**  
Describes the XML-based configuration that defines the layout and functionality of the grids displayed in Caplin Trader Client.
- ◆ **Caplin Trader Client: Simple Form Component Configuration Reference**  
Describes the XML and XHTML based configuration that defines a simple form display component in Caplin Trader Client.

◆ **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.

◆ **Ext JS API reference documentation**

Describes the API of the "Ext JS" component framework that is used by Caplin Trader Client to render trees in a layout.

◆ **Installing Caplin Trader for Evaluation**

Describes how to install Caplin Trader on a Linux server for evaluation purposes.

## 1.4    Typographical conventions

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

Type	Uses
<b>aMethod</b>	Function or method name
<i>aParameter</i>	Parameter or variable name
<i>/AFolder/Afile.txt</i>	File names, folders and directories
<div>Some code;</div>	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
<b>XYZ Product Overview</b>	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](mailto:documentation@caplin.com).

## 1.6    Acknowledgments

*Firefox* is a registered trademark of the Mozilla Foundation.

*Java*, *JavaScript*, and *JVM* are trademarks of Sun Microsystems, Inc. in the U.S. or other countries.

*Windows* and *Internet Explorer* are registered trademarks of Microsoft Corporation in the United States and other countries.

## 1.7 Technical assumptions and restrictions

This document allows you to modify the default layout of the Caplin Trader Client Reference Implementation. To run this application you need a suitable Web browser. The supported Web browsers are:

- ◆ Mozilla Firefox® version 1.5
- ◆ Mozilla Firefox® version 2.0
- ◆ Mozilla Firefox® version 3.0
- ◆ Microsoft Internet Explorer® version 6
- ◆ Microsoft Internet Explorer® version 7

**Note:** The examples presented in this document relate to the Caplin Trader Client reference implementation that is supplied with the Caplin Trader evaluation kit. This document assumes that you have installed the Caplin Trader evaluation kit on a Linux server in accordance with the installation instructions in the document **Installing Caplin Trader for Evaluation**. The examples shown in this document apply to a Linux server installation

**Note:** The product finder discussed in this document requires a custom JavaScript class to be written (the product finder controller) to filter the instruments being displayed in a grid. An example of such a product finder controller is supplied with the reference implementation of Caplin Trader Client.

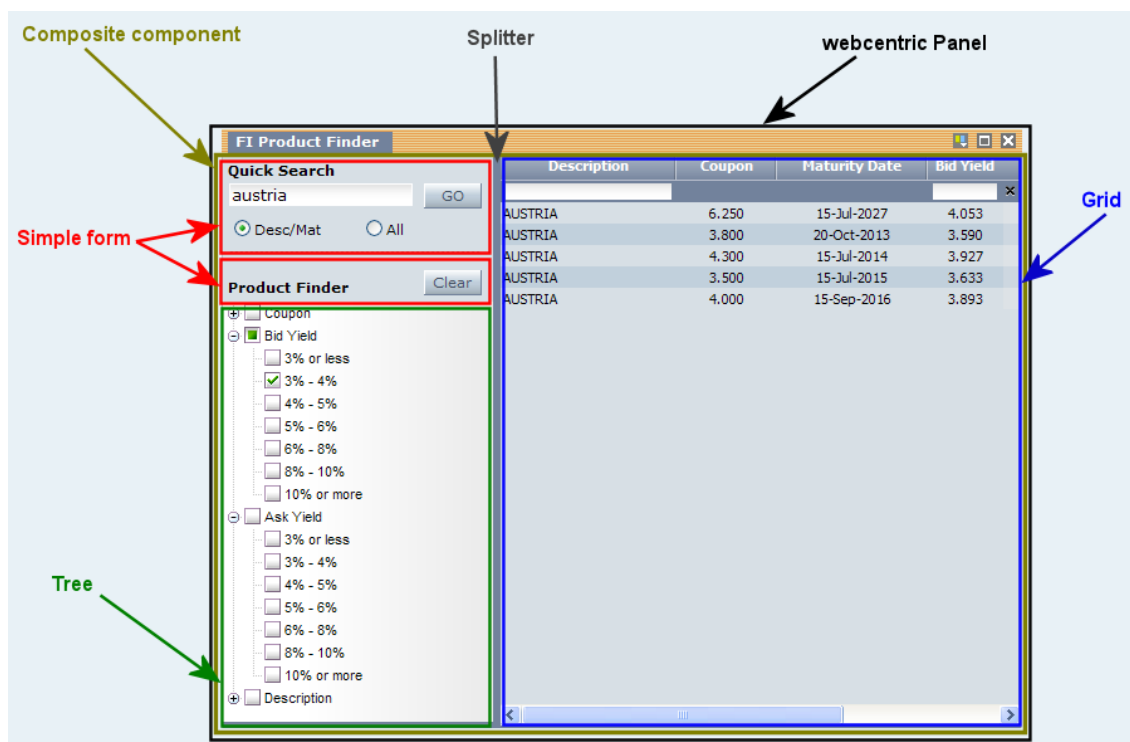


## 2 Introduction to the product finder

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 FI product finder composite component, identifying the display components and other elements that are used to construct it.



**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 Creating a product finder

In this document we look at how the FI product finder is constructed in the reference implementation of Caplin Trader Client. The steps to create a product finder are:

- Define the layout of the product finder
- Customize the grid component
- Customize the tree component
- Customize the quick search filter component
- Insert a menu item for the product finder
- Write the composite component controller

You will be able to use this information either to construct a new product finder (perhaps for FX instruments), or to modify the existing FI product finder.

**Note:** The code shown in this document is based on the FI product finder that is supplied with release 1.3.7 of the reference implementation of Caplin Trader Client. If your application is based on a later release of the reference implementation, then your code may be different from the code shown here.

Before we look at the product finder, we need to explain the Caplin Trader Client installation directory, the `caplinx` namespace, and the JavaScript API library requirements.

### The Caplin Trader Client installation directory

File references in this document are relative to the installation directory of the reference implementation of Caplin Trader Client. For example, if you installed Caplin Trader to the directory:

*/home/MyInstallation/*

then the installation directory of Caplin Trader Client is:

*/home/MyInstallation/apps/webapps/caplintrader/applications/CaplinTrader/*

In this document, the installation directory for Caplin Trader Client is abbreviated to:

*CaplinTrader/*

### The caplinx namespace

The reference implementation of Caplin Trader Client resides in the `caplinx` namespace. In the code samples that follow, replace `caplinx` with the namespace of your client application. For example, if the namespace of your client application is `mybank`, replace all occurrences of `caplinx` with `mybank` in any code that you write.

## **The JavaScript API Library**

From release 1.3.7, the Caplin Trader Client API library contains the JavaScript classes and interfaces that are required to implement a composite component. If your copy of the JavaScript library is earlier than this, then you must obtain and install a library upgrade before you modify your client application code. Please contact Caplin Support for further information about how to obtain and install this upgrade.

## 4 Defining the layout of the product finder

In the reference implementation of Caplin Trader Client, the layout of the product finder is defined in the file:

*CaplinTrader/source/xml/components/fi\_product\_search.xml*

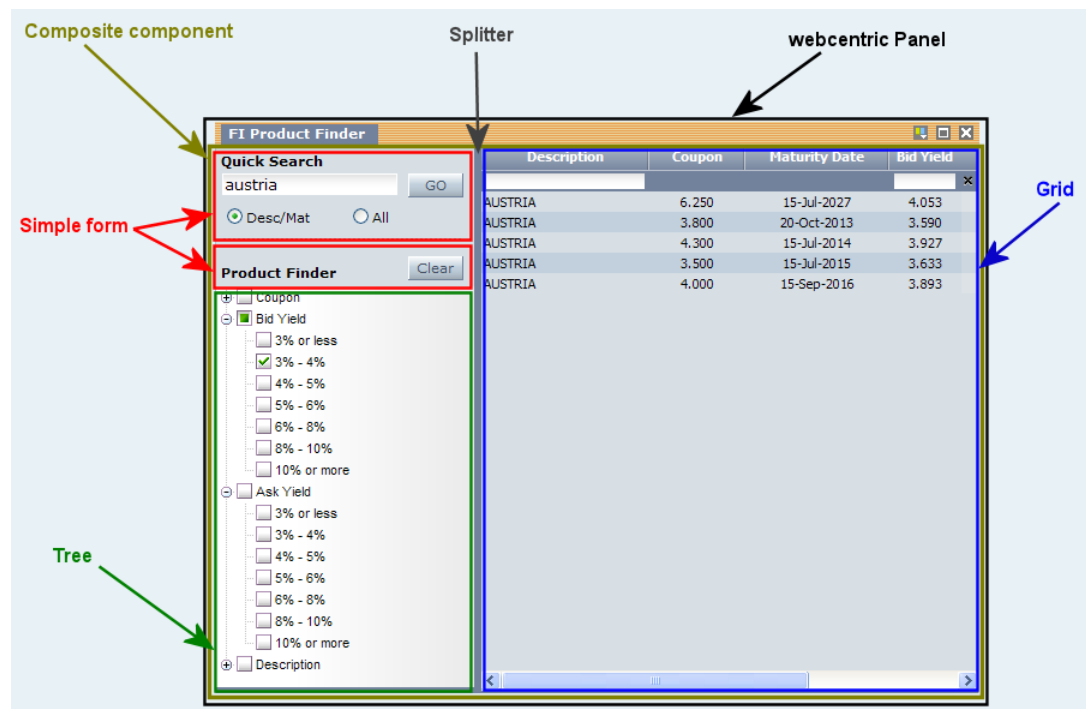
### XML that configures a composite display component as a product finder (*fi\_product\_search.xml*)

```
<caplin: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="/FT/TRADE/FX:TradeProtocol:ESP">
  <Decorators xref="Declarations/Decorators[@id='basicDecorator']"/>
  <state>
    <compositeComponent controller="caplinx.composite.FiProductFinderController">
      <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 baseGrid="FI.PRODUCT.SEARCH" />
        </component>
      </terrace>
    </compositeComponent>
  </state>
</caplin:Panel>
```

The XML configuration above defines 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.



Composite component configured as a product finder

## An explanation of the 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 XML configuration contains and how this relates to what the end-user sees on the screen.

- ◆ **<caplin:Panel>** The container for the composite component is a panel. The `caption="FI Product Finder"` attribute of the `<caplin:Panel>` tag defines the title of the product finder composite component. The other attributes of the `<caplin:Panel>` tag and `<Decorators>` tag define the behaviour and appearance of the panel when it is rendered.

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

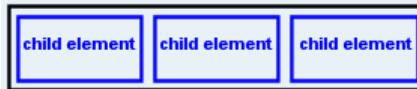
**Note:** There can only be one `<compositeComponent>` tag inside a `<caplin:Panel>` tag, which means that there can only be one composite component inside a panel.

- ◆ **<compositeComponent>** Contains a single **<terrace>** tag and starts the definition of the composite component.

```
<compositeComponent controller="caplinx.composite.FiProductFinderController">
  <terrace>
    ...
  </terrace>
</compositeComponent>
```

`controller="caplinx.composite.FiProductFinderController"`: Defines a custom 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.

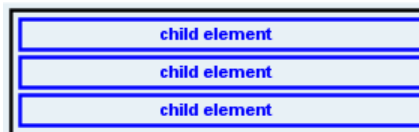
- ◆ **<terrace>** A layout element that arranges its children horizontally from left to right.



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

The **<terrace>** contains a **<tower>**, a **<splitter>**, and a **<component>**. Because the width of the terrace is not defined, its width is determined by the width of the **<tower>**, which in this case is 215 pixels (`<tower width="215px">`).

- ◆ **<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 baseGrid="FI.PRODUCT.SEARCH" />
</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 [Customizing the Quick Search filter](#) <sup>24</sup>.

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 its parent tower. The tree is constructed from a base template (baseTemplate="FiProductSearchTree").

See [Customizing the Tree component](#) <sup>17</sup>.

The component at the right hand side of the terrace (id="fiGrid") contains the product grid. The grid is constructed from a base grid (baseGrid="FI.PRODUCT.SEARCH").

See [Customizing the Grid component](#) <sup>13</sup>.

- ◆ **<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).

## 5 Customizing the grid component

The grid component of the product finder is defined by the `<grid>` tag in the XML layout definition of the product finder (see [Defining the layout of the product finder](#)<sup>[9]</sup>).

```
<component id="fiGrid">
  <grid baseGrid="FI.PRODUCT.SEARCH" />
</component>
```

The appearance and behavior of the grid is defined by grid templates in an inheritance hierarchy, where each template provides characteristics that can be inherited by other grids. The `baseGrid="FI.PRODUCT.SEARCH"` attribute of the `<grid>` tag identifies the first template in this inheritance hierarchy.

**Tip:** You will find further information about grid inheritance in the document **Caplin Trader Client: Customizing the Content**.

Grid templates are defined in the file *CaplinTrader/conf/gridDefinitions.xml*.

### 5.1 Grid template "FI.PRODUCT.SEARCH"

The grid template `id="FI.PRODUCT.SEARCH"` specifies the data provider that populates the grid. The XML for this template is defined in the file *CaplinTrader/conf/gridDefinitions.xml* and is reproduced below.

```
<gridTemplate id="FI.PRODUCT.SEARCH" displayName="COMBO_GRID" baseTemplate="FI">
  <gridRowModel>
    <rttpContainerGridDataProvider container="/CONTAINER/FI/ALL"/>
  </gridRowModel>
</gridTemplate>
```

◆ **<gridTemplate>**

The attribute `displayName="COMBO_GRID"` defines the display name of the grid. The display name is not used when the grid is part of the product finder composite component. Instead the `caption="FI Product Finder"` attribute of the `<caplin:Panel>` tag defines the display name of the whole composite component (see [Defining the layout of the product finder](#)<sup>[9]</sup>).

The attribute `baseTemplate="FI"` identifies the base template that this template inherits from.

◆ **<gridRowModel>**

Identifies the data provider (`<rttpContainerGridDataProvider>`) and container (`container="/CONTAINER/FI/ALL"`) that populates the grid.



## 5.2 Grid template "FI"

The grid template `id="FI"` defines the columns that the grid can display and adds grid and column decorators to the grid definition. The XML for this template is defined in the file

*CaplinTrader/conf/gridDefinitions.xml*

and is reproduced below.

```
<gridTemplate id="FI" baseTemplate="All" displayedColumns="description, cpn, mat,
    bidyield, price, askyield">
  <decorators>
    <dragDecorator ddGroup="fiInstruments" />
    <clearColumnFiltersDecorator/>
  </decorators>
  <columnDefinitions>
    <column id="description" fields="Description" headerRenderer="textFilter"
      displayName="Description" width="140" mandatory="true" />
    <column id="ccy" fields="Currency" displayName="Ccy" width="80"/>
    <column id="cpn" fields="CpnRate"
      cellRenderer="caplinx.renderer.YieldElementRenderer"
      displayName="Coupon" width="80" mandatory="true" />
    <column id="mat" fields="MaturityDate"
      cellRenderer="caplinx.renderer.DateElementRenderer"
      displayName="Maturity Date" width="120" mandatory="true" />
    <column id="price" cellRenderer="caplinx.renderer.BondSpreadElementRenderer"
      fields="BidPrice,AskPrice" displayName="Price" width="130"/>
    <column id="bidyield" headerRenderer="rangeFilter"
      cellRenderer="caplinx.renderer.YieldElementRenderer"
      fields="BidYield" displayName="Bid Yield" width="60"/>
    <column id="askyield" headerRenderer="rangeFilter"
      cellRenderer="caplinx.renderer.YieldElementRenderer"
      fields="AskYield" displayName="Ask Yield" width="60"/>
  </columnDefinitions>
</gridTemplate>
```

### ◆ <columnDefinitions>

Each <column> tag defines:

- The data field to be displayed in each cell of that column (`fields="..."`).
- The text to be displayed in the heading of the column (`displayName="..."`).
- The pixel width of the column (`width="..."`).
- An optional renderer that modifies the display format and behavior (if any) of the cells in the column (`cellRenderer="..."`).

The `headerRenderer` attribute of the "description", "bidyield", and "askyield" columns specifies that header renderers are displayed at the top of these columns. Header renderers provide a filter box into which the end user can enter data to filter the rows of the grid.

Two header renderers are defined: one that filters text (`headerRenderer="textFilter"`) and one that filters numeric data (`headerRenderer="rangeFilter"`).

The `mandatory` attribute of the "description", "cpn", and "mat" columns (`mandatory="true"`) specifies that the end user cannot remove these columns from a grid; the default is to allow end users to add and remove columns.

### ◆ <decorators>

The <dragDecorator> tag allows instruments to be dragged out of the grid and dropped into components that belong to the same drag-drop group (`ddGroup="fiInstruments"`).

The `<clearColumnFiltersDecorator/>` tag renders an icon inside the column filter box when rows are being filtered by data in the filter box. An end user can click this icon to clear the data from the filter box, which removes the applied filter.

Description	Coupon	Maturity Date	Bid Yield	Price	Ask Yield
10Y EURO	4.000	04-Jan-2018	3.898	101.02/101.05	3.895
15Y EURO	4.250	25-Oct-2023	4.354	98.96/99.00	4.350
20Y EURO	6.500	04-Jul-2027	3.801	126.99/127.15	3.785
2Y EURO	4.000	11-Dec-2009	3.873	101.27/101.29	3.871

#### ◆ `<gridTemplate>`

The `displayedColumns` attribute of the `<gridTemplate>` tag identifies the columns that will be shown when the grid is first displayed. The end user can show or hide the columns being displayed from a drop-down list at each column header. Mandatory columns cannot be hidden (see the `mandatory` attribute of the `<columnDefinitions>` tag above).

Description	Coupon	Maturity Date
AUSTRIA	3.500	15-5
AUSTRIA	3.500	15-5
AUSTRIA	3.500	15-5
AUSTRIA	3.500	15-5
AUSTRIA	3.500	15-5
AUSTRIA	3.500	15-5
AUSTRIA	3.500	15-5
AUSTRIA	3.500	15-5
AUSTRIA	3.500	15-5
AUSTRIA	3.500	15-5

The attribute `baseTemplate="All"` identifies the base template that this template inherits from.

## 5.3 Grid template "All"

The grid template `id="All"` adds decorators that define the basic appearance and behavior of the grid. The XML for this template is defined in the file `CaplinTrader/conf/gridDefinitions.xml` and reproduced below.

```
<gridTemplate id="All">
  <decorators>
    <columnMenuDecorator/>
    <loadingDataDecorator showInitialLoad="true"/>
    <noDataFoundDecorator message="Your search returned no results,
      please change your search criteria"/>
    <selectionDecorator/>
    <hoverDecorator/>
  </decorators>
</gridTemplate>
```

◆ **<decorators>**

The `<columnMenuDecorator>` tag adds a drop down menu to each column that allows columns to be inserted to the right hand side of the selected column, or existing columns to be removed.

The `<loadingDataDecorator>` tag causes a spinning "wait" icon to be displayed to the user while the grid waits for data to display. The `showInitialLoad` attribute specifies that the "wait" icon will be shown when the grid is initially rendered.

The `<noDataFoundDecorator>` tag displays the text defined by the `message` attribute when the grid contains no data.

The `<selectionDecorator>` tag allows rows to be selected and dragged out of the grid.

The `<hoverDecorator>` tag renders rows in a different style when the user hovers over the row. The styles are defined in the file *CaplinTrader/source/styles/hover-decorator.css*.

## 5.4 Further reading about Grids

The following documents provide further information about the grids used in Caplin Trader Client.

◆ **Caplin Trader Client: Customizing the Content**

Describes how to add a grid (and other content) to the default layout of Caplin Trader Client.

◆ **Caplin Trader Client: Grid Configuration XML Reference**

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

## 6 Customizing the tree component

The tree component of the product finder is defined by the `<tree>` tag in the XML layout definition of the product finder (see [Defining the layout of the product finder](#) <sup>9</sup>).

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

The `baseTemplate="FIProductSearchTree"` attribute of the `<tree>` tag identifies the tree view template that is used to construct the tree. Tree view templates are defined in the file *CaplinTrader/conf/treeViewDefinitions.xml*.

### 6.1 The tree view template

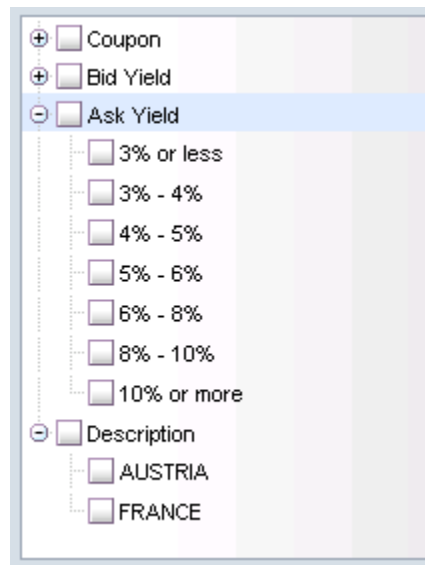
The tree view template `id="FIProductSearchTree"` specifies the layout of the tree component in the product finder. The XML for this template is defined in the file *CaplinTrader/conf/treeViewDefinitions.xml* and is reproduced below.

```

<treeViewTemplate id="FIProductSearchTree"
    toggleCheckOnDblClick="false"
    checked="false"
    nodeIcon="false"
    loadFolderContent="onInitialize"
    trackMouseOver="true"
    animate="false"
    enableDrag="false"
    rootVisible="false"
    ddGroup="fxInstruments"
    marginTop="0" marginRight="5" marginBottom="5" marginLeft="5"
    borderWidth="1" borderColor="#9CA4B7">
  <folder id="Coupon" value="" caption="Coupon">
    <item id="0-3" value="CpnRate=0-3" caption="3% or less"/>
    <item id="3-4" value="CpnRate=3-4" caption="3% - 4%"/>
    <item id="4-5" value="CpnRate=4-5" caption="4% - 5%"/>
    <item id="5-6" value="CpnRate=5-6" caption="5% - 6%"/>
    <item id="6plus" value="CpnRate=6+" caption="6% or more"/>
  </folder>
  <folder id="BidYield" value="" caption="Bid Yield">
    <item id="0-3" value="BidYield=0-3" caption="3% or less"/>
    <item id="3-4" value="BidYield=3-4" caption="3% - 4%"/>
    <item id="4-5" value="BidYield=4-5" caption="4% - 5%"/>
    <item id="5-6" value="BidYield=5-6" caption="5% - 6%"/>
    <item id="6-8" value="BidYield=6-8" caption="6% - 8%"/>
    <item id="8-10" value="BidYield=8-10" caption="8% - 10%"/>
    <item id="10plus" value="BidYield=10+" caption="10% or more"/>
  </folder>
  <folder id="AskYield" value="" caption="Ask Yield">
    <item id="0-3" value="AskYield=0-3" caption="3% or less"/>
    <item id="3-4" value="AskYield=3-4" caption="3% - 4%"/>
    <item id="4-5" value="AskYield=4-5" caption="4% - 5%"/>
    <item id="5-6" value="AskYield=5-6" caption="5% - 6%"/>
    <item id="6-8" value="AskYield=6-8" caption="6% - 8%"/>
    <item id="8-10" value="AskYield=8-10" caption="8% - 10%"/>
    <item id="10plus" value="AskYield=10+" caption="10% or more"/>
  </folder>
  <folder id="Description" value="" caption="Description">
    <item id="Description=AUSTRIA" value="Description=AUSTRIA" caption="AUSTRIA"/>
    <item id="Description=france" value="Description=france" caption="FRANCE"/>
  </folder>
</treeViewTemplate>

```

This configuration defines a tree that displays filter information in four folders with the captions "Coupon", "Bid Yield", "Ask Yield" and "Description".



**FI Tree displaying information in four folders**

## An explanation of the tree view XML configuration

Here is an explanation of what the XML configuration contains and how this relates to what the end-user sees on the screen:

◆ **<treeViewTemplate>** contains the template definition for the "FIProductSearchTree" tree view:

```
<treeViewTemplate id="FIProductSearchTree"
  toggleCheckOnDb1Click="false"
  checked="false"
  nodeIcon="false"
  loadFolderContent="onInitialize"
  trackMouseOver="true"
  animate="false"
  enableDrag="false"
  rootVisible="false"
  ddGroup="fxInstruments"
  marginTop="0" marginRight="5" marginBottom="5" marginLeft="5"
  borderWidth="1" borderColor="#9CA4B7">
  <folder id="Coupon" value="" caption="Coupon">
    ...
  </folder>
</treeViewTemplate>
```

The **<treeViewTemplate>** tag has a number of attributes.

**id="FIProductSearchTree"**: Identifies the template so the tree that is placed in the product finder can inherit from this template.

**toggleCheckOnDb1Click="false"**: Determines that a double click selects and expands the node if it is a parent node, or simply selects the node if it is a leaf node.

`checked="false"` Adds a check box to each node. Each check box will be unchecked when first displayed.

`nodeIcon="false"` Specifies that standard Ext JS node icons *will not* be added to each node. Set to `"true"` if you want icons.

`loadFolderContent="onInitialize"`: Specifies that folders in the tree will be populated when the tree is initially displayed, rather than when a folder is first expanded.

The remaining attributes are configuration options of the "Ext JS" `Ext.tree.TreePanel` class that affect the look and behaviour of the tree view component when it is rendered.

`trackMouseOver="true"`: Enable mouse over highlighting.

`animate="false"`: Disable animated expand/collapse.

`enableDrag="false"`: Disable drag and drop.

`rootVisible="false"`: Hide the root node of the tree.

`ddGroup="fxInstruments"`: The drag drop group the tree belongs to.

You can add other configuration options from the "Ext JS" `Ext.tree.TreePanel` class by adding name/value pairs for these attributes (see the description of the `<treeViewTemplate>` tag in the document **Caplin Trader Client:Tree View Configuration XML Reference**). For a complete description of the available "Ext JS" configuration options, please refer to the "Ext JS" API documentation.

◆ **<folder>** defines the folders that are displayed in the tree:

```
<folder id="Coupon" value="" caption="Coupon">
  ...
</folder>

<folder id="BidYield" value="" caption="Bid Yield">
  ...
</folder>

<folder id="AskYield" value="" caption="Ask Yield">
  ...
</folder>

<folder id="Description" value="" caption="Description">
  ...
</folder>
```

In this case four folders are defined, each with an `id`, `caption` and `value` attribute.

`id`: A unique identifier for the folder.

`caption`: The caption for the folder when the folder is displayed in the tree.

`value`: Folders in this tree do not have values associated with them.

◆ **<item>** each folder contains several items:

```
<folder id="AskYield" value="" caption="Ask Yield">
  <item id="0-3" value="AskYield=0-3" caption="3% or less"/>
  <item id="3-4" value="AskYield=3-4" caption="3% - 4%"/>
  <item id="4-5" value="AskYield=4-5" caption="4% - 5%"/>
  <item id="5-6" value="AskYield=5-6" caption="5% - 6%"/>
  <item id="6-8" value="AskYield=6-8" caption="6% - 8%"/>
  <item id="8-10" value="AskYield=8-10" caption="8% - 10%"/>
  <item id="10plus" value="AskYield=10+" caption="10% or more"/>
</folder>

<folder id="Description" value="" caption="Description">
  <item id="Description=AUSTRIA" value="Description=AUSTRIA" caption="AUSTRIA"/>
  <item id="Description=france" value="Description=france" caption="FRANCE"/>
</folder>
```

Each item is a leaf node of the tree and has an `id`, `caption` and `value` attribute.

`id`: A unique identifier for the item.

`caption`: The caption for the item when the item is displayed in the tree.

`value`: The value that is associated with the item. When the check box for the item is selected, the value is passed to the JavaScript class that controls the composite component. The controller filters the instruments displayed in the grid according to the items that are selected in the tree.

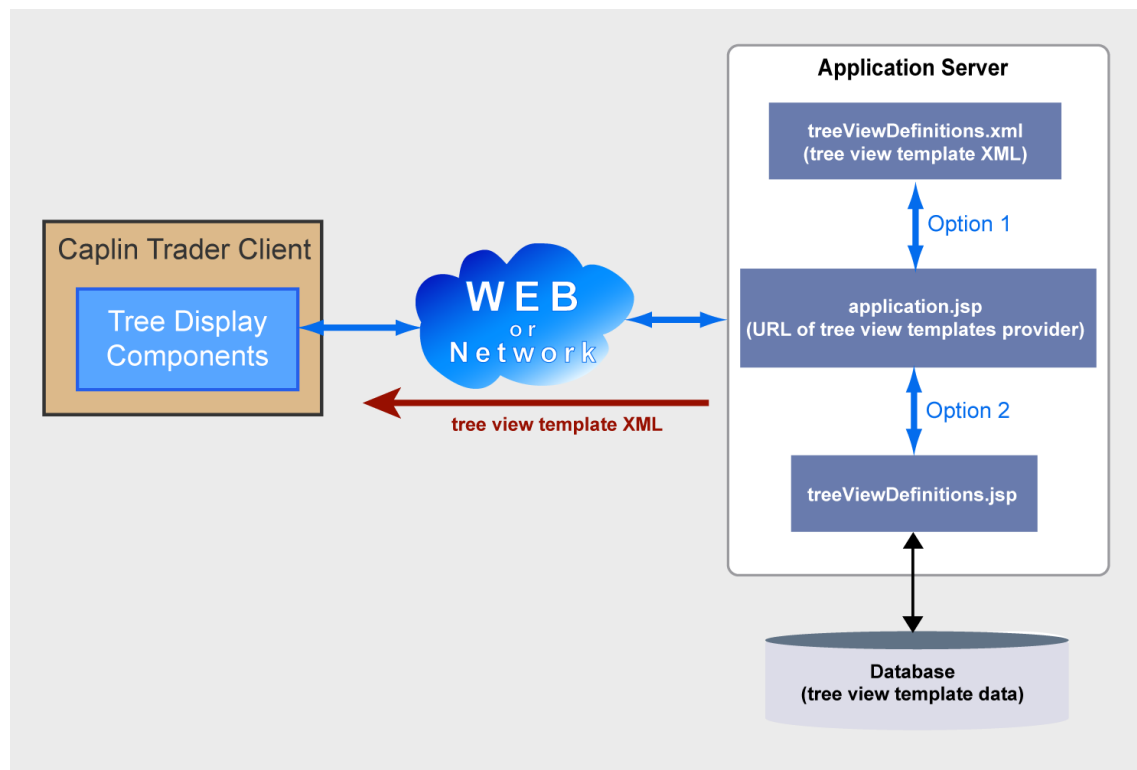
The JavaScript class that controls the composite component is defined by the `controller` attribute of the `<compositecompont>` tag in the layout definition for the product finder (see [Defining the layout of the product finder](#)<sup>[9]</sup>).

## 6.2 Populating the tree dynamically

Tree view templates are XML definitions of the trees that can be inserted in a Caplin Trader Client layout.

In the reference implementation of Caplin Trader Client, tree view templates are defined in the file *CaplinTrader/conf/treeViewDefinitions.xml*. If the tree view templates of your application are defined in another file, or if you want the XML to be provided to Caplin Trader client dynamically, then you must change the URL of the resource that provides this XML. The resource URL is defined in the file *CaplinTrader/application.jsp*.





### Options for serving tree view template XML to Caplin Trader Client

The picture above illustrates two options for providing tree view template definitions to Caplin Trader Client. Option 1 shows tree view templates being served from a static XML file, while option 2 shows how XML derived from tree view definitions in a database could be served dynamically from a custom JavaServer Page (JSP).

To serve the XML for tree view templates dynamically, three steps are necessary:

1. Create a database that contains the tree view definitions.
2. Create a JavaServer Page (*treeViewDefinitions.jsp*) that produces XML from the tree view definitions held in the database, and then serves the XML to Caplin Trader Client.
3. Modify the URL of the XML data provider in the file *CaplinTrader/application.jsp*.

The code extracts below are from *CaplinTrader/application.jsp*, and show the URL of the XML data provider for options 1 and 2 above. The URL is set by the second argument of the `caplin.core.ApplicationProperties.setProperty()` method.

#### Code extract for Option 1

```
...
<!-- ===== -->
<!-- Caplin Libraries -->
<!-- ===== -->
<script type="text/javascript" src="dependencies/core/core.js"></script>
<script type="text/javascript">

...

    caplin.core.ApplicationProperties.setProperty(
        "CAPLIN.COMPONENT.TREE.VIEW.DEFINITION.URL",
        "conf/treeViewDefinitions.xml");

    ...

</script>

...
```

#### Code extract for Option 2

```
...
<!-- ===== -->
<!-- Caplin Libraries -->
<!-- ===== -->
<script type="text/javascript" src="dependencies/core/core.js"></script>
<script type="text/javascript">

...

    caplin.core.ApplicationProperties.setProperty(
        "CAPLIN.COMPONENT.TREE.VIEW.DEFINITION.URL",
        "conf/treeViewDefinitions.jsp");

    ...

</script>

...
```

## Limitations

If the XML that defines tree view templates is changed when Caplin Trader Client is running, then new and modified tree definitions will not become available until the end user clears the browser cache and refreshes the browser.

## 6.3 Further reading about trees

The following documents provide further information about the tree views used in Caplin Trader Client.

- ◆ **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.

- ◆ **Ext JS API reference documentation**

Describes the API of the "Ext JS" component framework that is used by Caplin Trader Client to render trees in a layout.

## 7 Customizing the quick search filter

### The Quick Search filter component

The quick search filter of the product finder is a simple form component that is defined by a `<simpleForm>` tag in the XML layout definition of the product finder (see [Defining the layout of the product finder](#)).

```
<component id="fiQuickSearch" height="88px">  
  <simpleForm src="source/html-templates/quick-search.html" />  
</component>
```

The `src="source/html-templates/quick-search.html"` attribute of the `<simpleForm>` tag identifies the XHTML for the form.

### The Clear button for the product filter tree

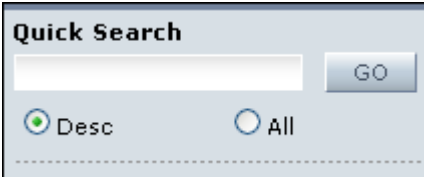
The Clear button that is situated above the product filter tree is also defined by a `<simpleForm>` tag in the XML layout definition of the product finder.

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

The `src="source/html-templates/product-finder-buttons.html"` attribute of the `<simpleForm>` tag identifies the XHTML for the form.

### 7.1 XHTML for the quick search filter

The XHTML form for the Quick Search filter provides a 'Quick Search' input box, a Go button, and two radio buttons.



The XHTML for this form is defined in the file *CaplinTrader/source/html-templates/quick-search.html* and is reproduced below.

```
<div id="FiQuickSearch">
  <div class="background">

    <div class="heading">Quick Search</div>
    <div id="quickSearchTextBoxHolder"><input type="text" name="QUICK_SEARCH_VALUE"
      class="quickSearchTextBox"/></div>
    <div id="submitButtonHolder"><input type="submit" id="SIMPLE_FORM_SUBMIT_ID"
      value="GO" class="submitButton"/></div>
    <div id="radioOneHolder" class="searchOption"><input type="radio"
      name="QUICK_SEARCH_FIELDS" value="Description" checked="true">Desc</input></div>
    <div id="radioTwoHolder" class="searchOption"><input type="radio"
      name="QUICK_SEARCH_FIELDS" value="ALL_FIELDS">All</input></div>

  </div>

  <div class="horizontalLine"/>
</div>
```

- ◆ **<div class="heading">**  
Provides the text for the Quick Search filter heading ("Quick Search").
- ◆ **<div id="quickSearchTextBoxHolder">**  
Provides a text input box. The end user types text into this box to filter the grid.
- ◆ **<div id="submitButtonHolder">**  
Provides a submit button labeled GO (value="GO"). The end user clicks this button to filter the grid according to the text that is typed into the input box (div id="quickSearchTextBoxHolder").
- ◆ **<div id="radioOneHolder">**  
Provides a radio button. The end user selects this button to restrict the filter to the 'Description' column of the grid (value="Description").
- ◆ **<div id="radioTwoHolder">**  
Provides a radio button. The end user selects this button to apply the filter to all columns of the grid (value="ALL\_FIELDS").
- ◆ **<div class="horizontalLine">**  
Provides a horizontal line below the two radio buttons.

## CSS Classes

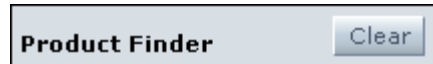
The classes that style the XHTML for the quick search filter form are defined in the file *CaplinTrader/source/styles/fi-product-finder.css*.

## Product Finder Controller

The JavaScript class that responds to events on the quick search filter form is defined by the controller attribute of the `<compositecompont>` tag in the layout definition for the product finder (see [Defining the layout of the product finder](#)<sup>[94]</sup>).

## 7.2 XHTML for the Clear button

An XHTML form provides the `Clear` button that is situated above the product finder tree.



The XHTML for this form is defined in the file *CaplinTrader/source/html-templates/product-finder-buttons.html* and is reproduced below.

```
<div id="FiButtons">
  <div class="background">
    <div class="quickButtons">Product Finder</div>
    <div id="submitButtonHolder"><input type="submit" name="clearButton"
      id="SIMPLE_FORM_SUBMIT_ID" value="Clear" class="submitButton"/></div>
  </div>
</div>
```

◆ **<div class="quickButtons">**

Provides the text to the left of the `Clear` button ("Product Finder").

◆ **<div id="submitButtonHolder">**

Provides a submit button labeled `Clear` (`value="Clear"`). The end user clicks this button to clear all checked items in the product finder tree, and to remove the grid filters that were applied by these checked items.

## CSS Classes

The classes that style the XHTML for the clear button form are defined in the file *CaplinTrader/source/styles/fi-product-finder.css*.

## Product Finder Controller

The JavaScript class that responds to events on the clear button form is defined by the `controller` attribute of the `<compositecompont>` tag in the layout definition for the product finder (see [Defining the layout of the product finder](#)<sup>[94]</sup>).

## 7.3 Further reading about Simple Forms

The following documents provide further information about the simple form display components used in Caplin Trader Client.

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

Describes the XML and XHTML based configuration that defines a simple form display component in Caplin Trader Client.

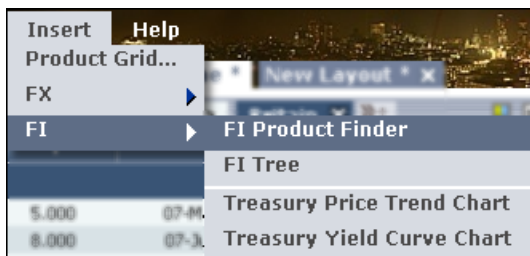
## 8 Inserting a menu item for the product finder

End users of Caplin Trader client can insert display components into a layout from the `Insert` menu.

The XML that describes the `Insert` menu is defined in the file `CaplinTrader/build/xml/application.xml`. The XML for the `FI` sub-menu is reproduced below.

```
<MenuItem label="FI" context="insert">
  <MenuItem menu_id="composite_component" label="FI Product Finder" context="insert">
    <addComponent>
      <component name="FI Product Finder"
        src="source/xml/components/fi_product_search.xml" />
    </addComponent>
  </MenuItem>
  <MenuItem menu_id="fi_tree" label="FI Tree" context="insert">
    <addComponent>
      <component name="FI Tree" src="source/xml/components/fi_tree.jsp" />
    </addComponent>
  </MenuItem>
  <MenuItem label="separator" context="insert" />
  <MenuItem label="Treasury Price Trend Chart" context="insert">
    <addComponent>
      <caplin:Fusion xmlns:caplin="http://www.caplin.com" caption="Treasury Price Trend"
        drop_target="SNAP_FRAMEITEM" colour="colour-1">
        <Decorators xref="Declarations/Decorators[@id='basicDecorator']" />
        <state><fusionChart><chartType>Line</chartType>
          <dataUrl>source/xml/charts/30YTsy_Hist.xml</dataUrl>
          <backgroundColor>#fff</backgroundColor></fusionChart>
        </state>
      </caplin:Fusion>
    </addComponent>
  </MenuItem>
  <MenuItem label="Treasury Yield Curve Chart" context="insert">
    <addComponent>
      <caplin:Fusion xmlns:caplin="http://www.caplin.com" caption="Treasury Yield Curve"
        drop_target="SNAP_FRAMEITEM" colour="colour-1">
        <Decorators xref="Declarations/Decorators[@id='basicDecorator']" />
        <state><fusionChart><chartType>MSCombiDY2D</chartType>
          <dataUrl>source/xml/charts/yieldcurve.xml</dataUrl>
          <backgroundColor>#fff</backgroundColor></fusionChart>
        </state>
      </caplin:Fusion>
    </addComponent>
  </MenuItem>
</MenuItem>
```

This menu configuration defines four display components that the end user can insert in a layout from the `Insert > FI` menu: 'FI Product Finder', 'FI Tree', 'Treasury Price Trend Chart', and 'Treasury Yield Curve Chart'.



## 8.1 XML for the product finder menu item

The XML that adds the 'FI Product Finder' option to the `Insert > FI` menu is reproduced below.

```
<MenuItem label="FI" context="insert">
  <MenuItem menu_id="composite_component" label="FI Product Finder" context="insert">
    <addComponent>
      <component name="FI Product Finder"
        src="source/xml/components/fi_product_search.xml" />
    </addComponent>
  </MenuItem>
  ...
</MenuItem>
```

### ◆ `<MenuItem menu_id="composite_component">`

Adds the Product Finder to the FI sub-menu.

`menu_id="composite_component"`: A unique identifier for the menu item.

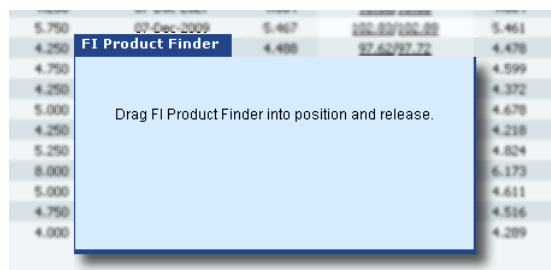
`label="FI Product Finder"`: The text that is added to the FI sub-menu.

`context="insert"`: Specifies that the component can be inserted in a layout.

### ◆ `<component>`

The display component that will be inserted in the layout when the menu item is selected.

`name="FI Product Finder"`: The title of the component *before* it is dropped into a panel. The `caption` attribute of the `<caplin: Panel>` tag defines the title of the component *after* it has been dropped into a panel (see [Defining the layout of the product finder](#))<sup>9</sup>.



`src="source/xml/components/fi_product_search.xml"`: The XML that defines the layout of the Product Finder (see [Defining the layout of the product finder](#))<sup>9</sup>.

## 9 The product finder controller

The example JavaScript class that controls the product finder is defined in the file *CaplinTrader/source/caplinx/composite/FiProductFinderController.js*.

The controller class registers itself as a listener on the product finder composite components (the tree, grid, and quick search simple forms), and filters the instruments that are displayed in the grid according to the filter criteria that the end user enters in the quick search box and product finder tree.

### 9.1 Changing filter behavior by modifying the product finder controller

In the reference implementation of Caplin Trader Client, an end user can filter the instruments in the product finder grid using any of the following filters.

- ◆ Filter boxes in the column headers of the grid.
- ◆ The quick search box at the top left hand side of the grid.
- ◆ The product finder tree that is situated below the quick search box.

Grid and Tree filters are applied additively. This means that if the end user filters the grid using column header filter boxes, and then filters the grid using the product finder tree, the two sets of filters will be applied to the grid.

You can change this filter behavior by modifying JavaScript code in the product finder controller. As an example, we will look at how to change the filter behavior so that:

1. When a node in the product finder tree is selected, column header filters are cleared *before* the tree filter is applied.
2. When a node in the product finder tree is selected, the quick search filter is cleared *before* the tree filter is applied.

To make these changes:

- Open the file *CaplinTrader/source/caplinx/composite/FiProductFinderController.js* and find the `onNodeSelected()` method.
- Add the lines of code that are highlighted below.

```
caplinx.composite.FiProductFinderController.prototype.onNodeSelected = function(oTreeNode)
{
    /* Add this code to clear filters when a tree node is selected */

    this._clearGridFilters();
    this.m_mActiveQuickSearchFiltersMap = {};

    this.m_mActiveFilterNodes[oTreeNode.getData().getNodeId()] = oTreeNode;
    var sNodeValue = oTreeNode.getData().getNodeValue();
    this._addTreeFilter(sNodeValue);
};
```

### Limitations

If the JavaScript code for the product finder is changed when Caplin Trader Client is running, then new and modified filter behaviors will not become available until the end user clears the browser cache and refreshes the browser.



## 10 Glossary of terms and acronyms

This section contains a glossary of terms, abbreviations, and acronyms used in this document.

Term	Definition
<b>API</b>	<u>A</u> pplication <u>P</u> rogramming <u>I</u> nterface
<b>Caplin Platform</b>	A suite of software products for on-line financial trading and Web delivery of real-time market data.
<b>Caplin Trader</b>	Caplin Trader is a complete platform and toolkit for building multi-product trading portals. It is built on the <b>Caplin Platform</b> .
<b>Caplin Trader Client</b>	Caplin Trader Client is a Web application written in Ajax that provides a rich trading workstation in a browser.
<b>Composite Component Controller</b>	A custom JavaScript class that controls the behavior of a <b>Composite Display Component</b> . An example JavaScript class that controls the behavior of the <b>Product Finder</b> is supplied with the reference implementation of Caplin Trader Client
<b>Composite Display Component</b>	A display component that is used to group together other display components.
<b>Grid Display Component</b>	A grid is a table of data where all of the rows of the table have a similar set of properties.
<b>Product Finder</b>	An implementation of a <b>Composite Display Component</b> that groups together display components that filter the instruments being displayed in a grid.
<b>Quick Search Filter</b>	An implementation of a <b>Simple Form</b> that is used to filter the instruments being displayed in a grid.
<b>Simple Form</b>	A display component that allows end-users of <b>Caplin Trader Client</b> to enter information into a displayed page.
<b>Tree Display Component</b>	A tree is a way of organizing items in a logical structure.
<b>User</b>	An end user of <b>Caplin Trader Client</b> .

# Index

## - A -

Abbreviations, definitions 30  
Acronyms, definitions 30  
adding JavaScript code 29  
API library requirements 7  
application.jsp 21  
application.xml 27  
attributes  
    baseGrid 13

## - B -

buttons  
    Clear 26  
    GO 24  
    radio 24  
    submit 24

## - C -

caplinx namespace 7, 29  
changing the URL of the tree XML definitions 21  
Clear button 24, 26  
column definitions 14  
components  
    grid 9  
    simple form 9  
    tree 9  
conventions 4  
creation process 7  
CSS classes 24, 26

## - D -

database 21  
document formats 2  
drag out of a grid 15  
dynamic tree population 21

## - E -

Ext JS configuration options 19

## - F -

FI sub-menu 27  
fi\_product\_search.xml 9  
filter behavior 29  
folder captions 17, 19

## - G -

Glossary 30  
GO button 24  
grid appearance and behavior 15  
grid template XLM 13  
gridDefinitions.xml 13, 14, 15

## - H -

hover over a row 15  
HTML tags  
    <div> 24, 26

## - I -

icons (tree) 19  
insert menu 27  
installation directory 7

## - J -

JavaScript API library 7  
JavaServer Page 21

## - L -

layout of the product finder 6, 9  
limitations 21, 29

## - N -

namespace 7

**- O -**

onNodeSelected method 29  
options for populating trees 21

**- P -**

product finder controller 29

**- Q -**

quick-search.html 24

**- R -**

radio buttons 24  
Readership 2  
related documents 2  
responding to events 24

**- S -**

simple form component 24  
spinning "wait" icon 15  
submit button 24

**- T -**

tags  
    <caplin:Panel> 9  
    <columnDefinitions> 14  
    <columnMenuDecorator> 15  
    <component> 9, 28  
    <compositeComponent> 9  
    <decorators> 14  
    <div> 24, 26  
    <folder> 19  
    <grid> 13  
    <gridRowModel> 13  
    <gridTemplate> 13, 14  
    <hoverDecorator> 15  
    <item> 19  
    <loadingDataDecorator> 15  
    <MenuItem> 28  
    <noDataFoundDecorator> 15

    <selectionDecorator> 15  
    <simpleForm> 24  
    <splitter> 9  
    <terrace> 9  
    <tower> 9  
    <tree> 17  
    <treeViewTemplate> 19  
Terms, glossary of 30  
tree configuration options 19  
tree icons 19  
tree view template 17  
treeGridComboController.js 29  
treeViewDefinitions.jsp 21  
treeViewDefinitions.xml 17, 21

**- U -**

use cases  
    How do I add display components to the composite component layout? 9  
    How do I change the filter behavior of the product finder controller? 29  
    How do I change the title of the product finder composite component? 9  
    How do I configure the fields that the quick search filters on? 24  
    How do I define the layout of my composite component? 9  
    How do I dynamically populate the tree display component? 21  
    How do I insert a menu item for the Composite Component? 27  
    How do I render icons in the tree display component? 19  
    How do I set-up my own data in the tree display component? 17

**- X -**

XML data provider 21

## 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](http://www.caplin.com)

The information contained in this publication is subject to UK, US and international copyright laws and treaties and all rights are reserved. No part of this publication may be reproduced or transmitted in any form or by any means without the written authorization of an Officer of Caplin Systems Limited.

Various Caplin technologies described in this document are the subject of patent applications. All trademarks, company names, logos and service marks/names ("Marks") displayed in this publication are the property of Caplin or other third parties and may be registered trademarks. You are not permitted to use any Mark without the prior written consent of Caplin or the owner of that Mark.

This publication is provided "as is" without warranty of any kind, either express or implied, including, but not limited to, warranties of merchantability, fitness for a particular purpose, or non-infringement.

This publication could include technical inaccuracies or typographical errors and is subject to change without notice. Changes are periodically added to the information herein; these changes will be incorporated in new editions of this publication.

Caplin Systems Limited may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time.

This publication may contain links to third-party web sites; Caplin Systems Limited is not responsible for the content of such sites.