

# Caplin Trader Client

## Customizing the Content

### 1.2

March 2008 ■ Confidential



# Contents

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Preface.....</b>                                     | <b>1</b>  |
| 1.1      | What this document contains.....                        | 1         |
| 1.2      | Who should read this document.....                      | 1         |
| 1.3      | Related documents.....                                  | 1         |
| 1.4      | Typographical conventions.....                          | 2         |
| 1.5      | Feedback.....   | 2         |
| 1.6      | Acknowledgments.....                                    | 2         |
| 1.7      | Technical assumptions and restrictions.....             | 3         |
| <b>2</b> | <b>About Caplin Trader Client.....</b>                  | <b>4</b>  |
| 2.1      | Overview of the User Interface.....                     | 5         |
| 2.2      | Layout Concepts and the Webcentric Framework.....       | 6         |
| <b>3</b> | <b>Preliminary Steps.....</b>                           | <b>7</b>  |
| <b>4</b> | <b>Adding Simple HTML Content.....</b>                  | <b>8</b>  |
| 4.1      | Create the HTML.....                                    | 8         |
| 4.2      | Modifying the Default Layout.....                       | 9         |
| <b>5</b> | <b>Adding a Website.....</b>                            | <b>10</b> |
| 5.1      | Modifying the Default Layout.....                       | 10        |
| <b>6</b> | <b>Adding a Fusion Chart.....</b>                       | <b>12</b> |
| 6.1      | Modifying the Default Layout.....                       | 13        |
| <b>7</b> | <b>Adding a Grid.....</b>                               | <b>15</b> |
| 7.1      | Inline Grid Definitions.....                            | 15        |
| 7.2      | Inheriting Grid Definitions.....                        | 19        |
| 7.3      | The Grid Definitions File.....                          | 21        |
|          | Naming the Grid Definitions File .....                  | 26        |
| 7.4      | Modifying the Default Layout.....                       | 27        |
| <b>8</b> | <b>Re-populating the User Preferences Database.....</b> | <b>28</b> |
| <b>9</b> | <b>Glossary of terms and acronyms.....</b>              | <b>29</b> |

# 1 Preface

## 1.1 What this document contains

This document describes how to add content to the layout components of the Caplin Trader Client Reference Implementation and then describes how to add custom components. The document applies to Caplin Trader Client, Version 1.2.

## 1.2 Who should read this document

This document is intended for System Administrators and Developers who need to configure the content and appearance of Caplin Trader Client. It assumes a basic working knowledge of XML (to modify the content) and JavaScript (to add new components or widgets).

## 1.3 Related documents

- ◆ **Installing Caplin Trader for Evaluation**

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

- ◆ **Caplin Trader Overview**

Describes how the Caplin Trader product provides real-time market data and trading capabilities.

- ◆ **Caplin Trader Client: Customizing the Appearance**

Describes how to configure the on-screen layout and 'look and feel' of the Caplin Trader Client.

- ◆ **Caplin Trader XML Configuration Reference**

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

- ◆ **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.

## 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 thoughts to [documentation@caplin.com](mailto:documentation@caplin.com).

## 1.6 Acknowledgments

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

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

*Adobe* and *Adobe Flash* are registered trademarks of Adobe Systems Incorporated in the United States and/or 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
- ◆ Microsoft Internet Explorer® version 6
- ◆ Microsoft Internet Explorer® version 7

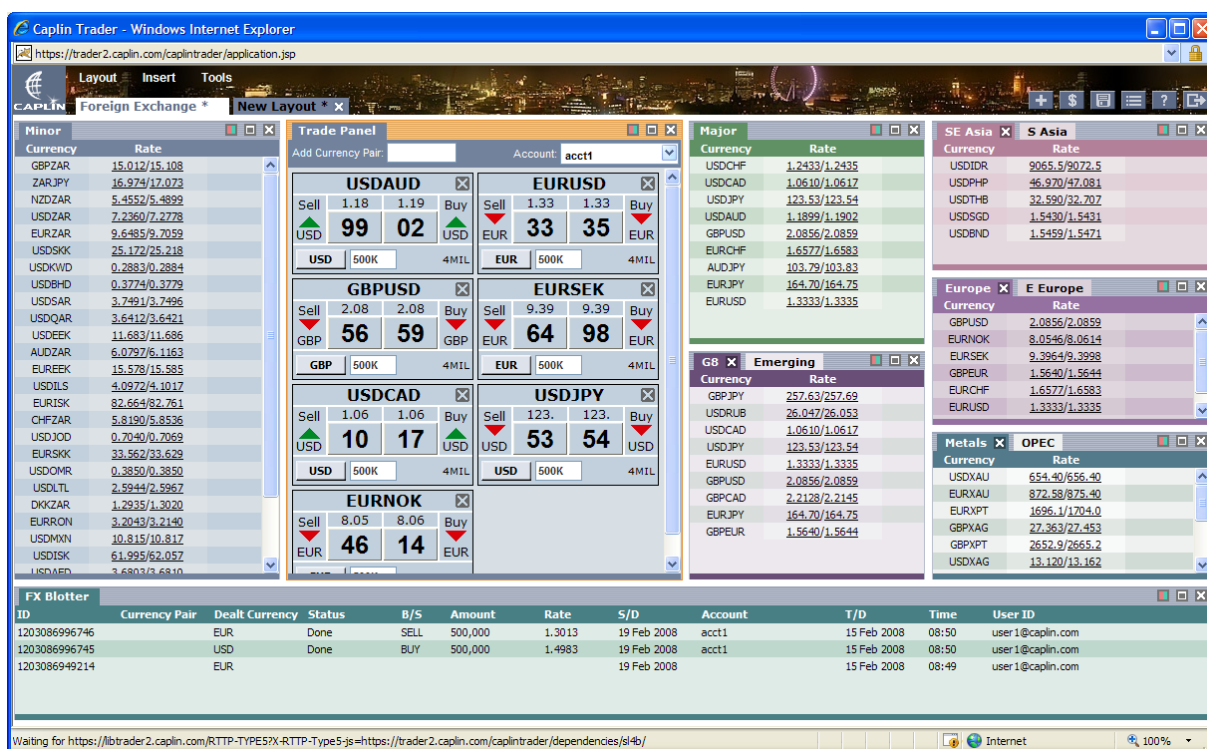
**Note:** The examples presented in this document relate to the Caplin Trader Client Reference Implementation that is supplied with the Caplin Trader Evaluation kit. This document assumes that you have installed the Caplin Trader Evaluation kit on a Linux server in accordance with the installation instructions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

## 2 About Caplin Trader Client

Caplin Trader Client is a Web browser based application for displaying real-time market data and placing trades on various financial instruments. Although Caplin Trader Client is deployed as a web page in a browser it uses an Ajax programming framework (called webcentric) that allows it to emulate many features of a desktop application including:

- ◆ Many windows on one screen, like a Multiple Document Interface (MDI).
- ◆ You can "drag and drop" windows to other parts of the application.
- ◆ You can resize, minimize and maximize windows.
- ◆ You can switch between layered windows via "tabs".

Caplin Trader Client is a major component of the Caplin Trader product. Further information about Caplin Trader can be found in the document **Caplin Trader Overview**.



### The Caplin Trader Client Reference Implementation layout and appearance

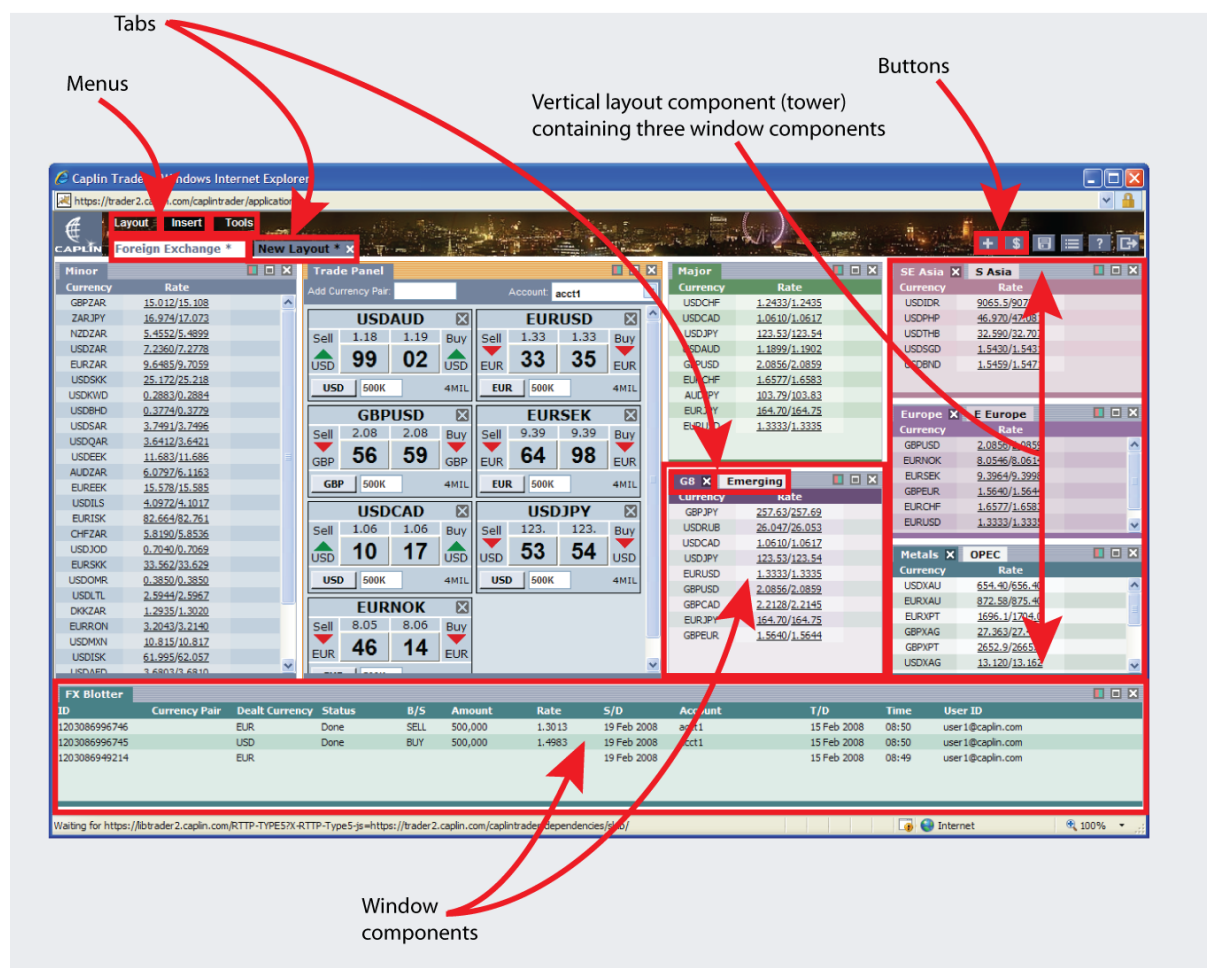
This document focuses on how to customize the content of Caplin Trader Client. You do this by editing a set of XML configuration files to customize the content that can be displayed when the application starts and by writing JavaScript to add custom components. The [Overview](#) introduces to you the main concepts behind the Caplin Trader Client interface.

## 2.1 Overview of the User Interface

The Caplin Trader Client interface is constructed from a number of tiled and nested components:

- ◆ Window components are rectangular areas that show various kinds of data in tabular and other formats.
- ◆ Layout components are used to arrange the layout of groups of windows.
- ◆ Tab, Menu, and Button components help the user interact with the windows.

These interface components are shown in the diagram below.



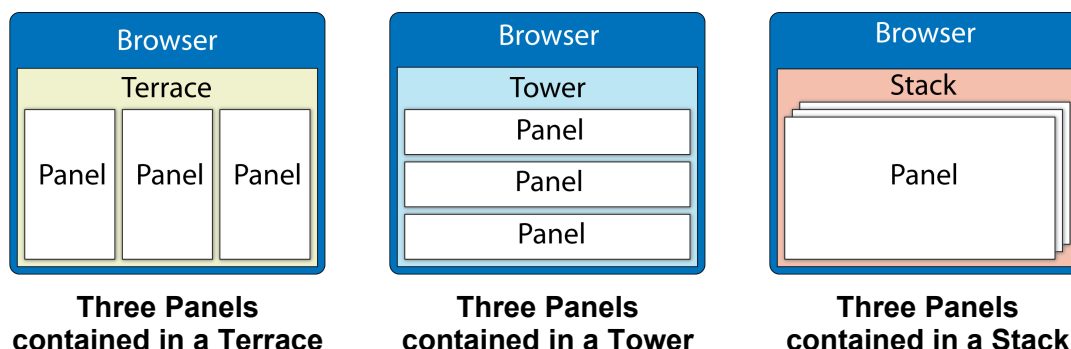
### Different types of interface component

The interface that is presented to a user when the Caplin Trader Client application starts can be customized to suit the needs of your business. This document will focus on how to customize the content of this interface.

## 2.2 Layout Concepts and the Webcentric Framework

Caplin Trader Client is given the look and feel of a windows style desktop application through a client-side portal framework called webcentric. Webcentric acts as a container for indicative market data and trade information, such as data grids, trade tickets, trade tiles, and trade blotters. The basic building blocks provided by webcentric are:

- ◆ The Panel – a rectangular area for displaying content and interactive components (widgets).
- ◆ The Terrace – used to group Panels horizontally.
- ◆ The Tower – used to group Panels vertically.
- ◆ The Stack – used to stack Panels on top of each other.



A Panel can display content in a layout when the content is defined using the `<Panel>` tag, the `<Frame>` tag, or the `<caplin:Panel>` tag.

- ◆ The `<Panel>` tag is used to add simple HTML content to a layout (see [Adding Simple HTML Content](#)<sup>[8]</sup>).
- ◆ The `<Frame>` tag is used to add a Website to a layout (see [Adding a Website](#)<sup>[10]</sup>).
- ◆ The `<caplin:Panel>` tag is used to add more complex components to a layout, such as a Fusion chart (see [Adding a Fusion Chart](#)<sup>[12]</sup>) or a grid (see [Adding a Grid](#)<sup>[15]</sup>).

You will find further information about webcentric, and the XML-based configuration that defines the layout of Caplin Trader Client, in the document **Caplin Trader: XML Configuration Reference**.

You will also find tutorials that describe how to customize the look and feel of Caplin Trader Client in the document **Caplin Trader Client: Customizing the Appearance**. The tutorials describe how to create new layouts and how to customize the appearance of existing layouts. It is recommended that you read that document in conjunction with this one, so that you become familiar with the layout terms and concepts that it introduces.



### 3 Preliminary Steps

This document discusses how to add new content to Caplin Trader Client. The examples in the document add content to the Caplin Trader Client Reference Implementation, allowing you to see how content can be added to a real application.

Since the directory where the Caplin Trader Client software is installed will not be the same for every installation, we suggest that you follow the procedure below to record this directory in an environment variable.

- Find the directory where the Caplin Trader software is installed.

For example:

```
/home/CaplinTrader
```

- Then find the directory where the Caplin Trader Client software is installed.

For example:

```
/home/CaplinTrader/apps/webapps/caplintrader/applications/CaplinTrader/
```

- Define the environment variable `CTC_INSTALL_DIR` for this directory.

For example:

```
export CTC_INSTALL_DIR=/home/CaplinTrader/apps/webapps/  
caplintrader/applications/CaplinTrader/
```

**Note:** In the rest of this document the directory where the Caplin Trader Client software is installed will be referred to using the environment variable `$CTC_INSTALL_DIR`.

Content will be added to the default "Foreign Exchange" layout that is displayed when Caplin Trader Client launches. To add content to this layout you will need to edit the file *Default\_FX\_Layout.xml* using a suitable editor. This file can be found at:

`$CTC_INSTALL_DIR/build/xml/layouts`

**Tip:** We recommend making a backup copy of this file before you make any changes to it, so that you can restore the original default layout whenever you wish.

If you change the default layout by amending *Default\_FX\_Layout.xml* then you must re-populate the user preferences database to make the changes available to your application (see [Re-populating the User Preferences Database](#) <sup>[28]</sup>).

Now create the directory `$CTC_INSTALL_DIR/source/mybank` (you can give this directory any name – this is just the name we chose for the examples in this document). This is where you will store some of the files you create when you add content to the Caplin Trader Client Reference Implementation. We will refer to this directory in the rest of this document and in the XML configuration code that we add later on.

When you follow the examples in this document please make sure that you have the Caplin Trader Client Reference Implementation open in a browser window.

## 4 Adding Simple HTML Content

You can use the Panel component to display HTML when the file containing the HTML resides on the same server as the Caplin Trader Client application. This is the simplest way to display HTML in a layout. The XML code that will add HTML content to a Panel is:

```
<Panel background="#ccf" src="source/mybank/html/panelcontent.html" caption="HTML Panel"
drop_target="SNAP_FRAMEITEM" colour="colour-12" decorators="basicDecorator" />
```

The `src` attribute of the `<Panel>` tag identifies the URL of the file that contains the HTML to be rendered. In this example the HTML is contained in the file `source/mybank/html/panelcontent.html`. The location of this file is relative to the directory of the running application, which for the Reference Implementation is `$CTC_INSTALL_DIR`.

The other attributes of the `<Panel>` tag shown above define the behaviour and appearance of the Panel when it is rendered.

**Tip:** You will find tutorials on modifying the layout and appearance of Caplin Trader Client in the document **Caplin Trader Client: Customizing the Appearance**.

### 4.1 Create the HTML

We will now add an HTML Panel to the default "Foreign Exchange" layout of the Reference Implementation. First create the directory that will contain the HTML source file; for this example the directory is `$CTC_INSTALL_DIR/source/mybank/html`.

Now open an HTML editor and enter the HTML that you want to be rendered in the Panel. An example is shown below:

```
<div><br />This text has been placed in a Panel. The source of the HTML is
a file that resides on the same server as the Caplin Trader Client application.</div>
```

Save this file as `$CTC_INSTALL_DIR/source/mybank/html/panelcontent.html`.

## 4.2 Modifying the Default Layout

We will now add the `<Panel>` tag to the default layout.

Open the file *Default\_FX\_Layout.xml* in a suitable editor. This file contains the XML configuration for the default "Foreign Exchange" layout and can be found at:

`$CTC_INSTALL_DIR/build/xml/layouts/`

We will place the new Panel at the left hand side of the layout. To do this, move to the start of the file, insert the highlighted code at the position shown below, and then save the file.

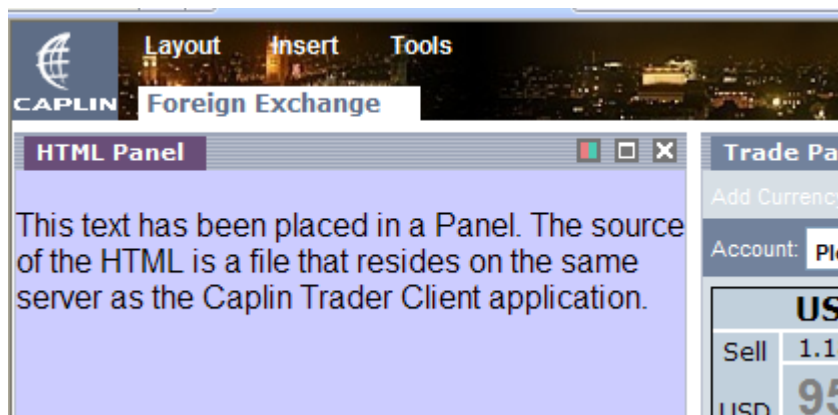
```
<Tower xmlns:caplin="www.caplin.com" splitters="true">
  <FrameItems>
    <Terrace splitters="true">
      <FrameItems>

<!-- Adding HTML to a Panel -->

<Panel background="#ccf" src="source/mybank/html/panelcontent.html" caption="HTML Panel"
  drop_target="SNAP_FRAMEITEM" colour="colour-12" decorators="basicDecorator" />
```

Since Caplin Trader Client does not read this XML file directly you must re-populate the user preferences database after you have made the changes; see [Re-populating the User Preferences Database](#) <sup>[28]</sup>.

When you have saved the XML file and re-populated the database, clear your browser cache and refresh Caplin Trader Client. The changes you made will now be available to the application.



**Simple HTML Content Added to a Panel**

## 5 Adding a Website

You can add a page from a website to your layout using the Frame component, which is implemented as an HTML inline frame. Scroll bars are added when the size of the rendered page exceeds the Frame size. No navigation bar is provided but you can navigate to other web pages if the displayed page contains a hyperlink.

The XML code that will add a website to a Frame is:

```
<Frame src="http://www.caplin.com" caption="Website in Frame"
  drop_target="SNAP_FRAMEITEM" colour="colour-12" decorators="basicDecorator" />
```

The `src` attribute of the `<Frame>` tag identifies the URL of the website, which in this case is <http://www.caplin.com>. The other attributes of the `<Frame>` tag shown above define the behaviour and appearance of the Frame when it is rendered.

**Tip:** You will find tutorials on modifying the layout and appearance of Caplin Trader Client in the document **Caplin Trader Client: Customizing the Appearance**.

### 5.1 Modifying the Default Layout

We will now add a website to a Frame in the default "Foreign Exchange" layout of the Reference Implementation.

Open the file *Default\_FX\_Layout.xml* in a suitable editor. This file contains the XML configuration for the default "Foreign Exchange" layout and can be found at:

*\$CTC\_INSTALL\_DIR/build/xml/layouts/*

We will place the new Frame at the left hand side of the layout. To do this, move to the start of the file, insert the highlighted code at the position shown below, and then save the file.

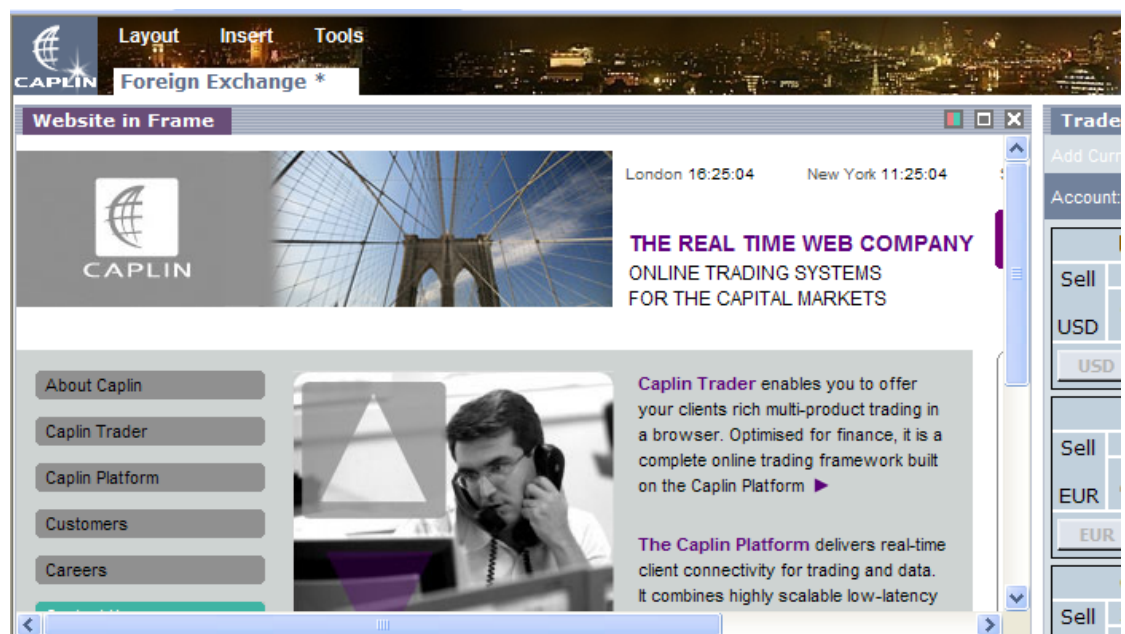
```
<Tower xmlns:caplin="www.caplin.com" splitters="true">
  <FrameItems>
    <Terrace splitters="true">
      <FrameItems>

<!-- Adding a Website to a Frame -->

      <Frame src="http://www.caplin.com" caption="Website in Frame"
        drop_target="SNAP_FRAMEITEM" colour="colour-12" decorators="basicDecorator" />
    
```

Since Caplin Trader Client does not read this XML file directly you must re-populate the user preferences database after you have made the changes; see [Re-populating the User Preferences Database](#)<sup>[28]</sup>.

When you have saved the XML file and re-populated the database, clear your browser cache and refresh Caplin Trader Client. The changes you made will now be available to the application.



Page from a Website Added to a Frame

## 6 Adding a Fusion Chart

A Fusion chart is an Adobe Flash® object that can be inserted in your Caplin Trader Client layout to render data driven charts. The Caplin Trader Client Reference Implementation can render four types of chart.

- ◆ Line – shows performance over a period of time.
- ◆ Column 3D – displays data using 3-dimensional vertical columns.
- ◆ Combination (Single Y) – superimposes different plots on the same y-axis.
- ◆ Combination (Dual Y) – superimposes different plots on two y-axis.

You insert a Fusion chart in a layout using the `<caplin:Panel>` tag in conjunction with the `<caplinx:fusion>` tag. The XML code to add a Line chart is:

```
<caplin:Panel caption="Treasury Price Trend"
  drop_target="SNAP_FRAMEITEM" colour="colour-12" decorators="basicDecorator">
  <state>
    <caplinx:fusion>
      <charttype>Line</charttype>
      <dataurl>source/xml/charts/30YTsy_Hist.xml</dataurl>
      <backgroundcolour>#fff</backgroundcolour>
    </caplinx:fusion>
  </state>
</caplin:Panel>
```

The container for a Fusion chart is a Panel, and the attributes of the `<caplin:Panel>` tag shown above define the behaviour and appearance of the Panel when it is rendered.

**Tip:** You will find tutorials on modifying the layout and appearance of Caplin Trader Client in the document **Caplin Trader Client: Customizing the Appearance**.

The XML between the `<state>` and `</state>` tags defines the content that will be placed in the container, and the `<caplinx:fusion>` tag identifies this content to be a Fusion chart component. The XML between the `<caplinx:fusion>` and `</caplinx:fusion>` tags identifies the chart that will be rendered.

The `<charttype>Line</charttype>` tag identifies the type of chart as a Line chart. The Fusion chart renderer `$CTC_INSTALL_DIR/source/charts/Line.swf` is used to render this type of chart and is supplied with the Reference Implementation. Files that render Fusion charts have the `.swf` file extension and are located in the `$CTC_INSTALL_DIR/source/charts` directory.

The `<dataurl>source/xml/charts/30YTsy_Hist.xml</dataurl>` tag identifies the XML file that contains the source data to be charted. Fusion chart renderers can only render data when it is supplied in a pre-defined XML format, and this file has the correct XML format for the Line type of Fusion chart.

**Tip:** You can find full details of the XML format that Fusion charts expect on the InfoSoft Global website at <http://www.fusioncharts.com>.

Finally the `<background>#fff</background>` tag identifies the background color of the charted data, which in this case is white (`#fff`).

If you want to display a Combination (Dual Y) chart rather than a Line chart then edit the XML to look like the code shown below.

```
<caplin:Panel caption="Treasury Yield Curve"
  drop_target="SNAP_FRAMEITEM" colour="colour-12" decorators="basicDecorator">
  <state>
    <caplinx:fusion>
      <charttype>MSCombiDY2D</charttype>
      <dataurl>source/xml/charts/yieldcurve.xml</dataurl>
      <backgroundcolour>#fff</backgroundcolour>
    </caplinx:fusion>
  </state>
</caplin:Panel>
```

In this case the container caption has been changed to `caption="Treasury Yield Curve"`, the Fusion chart renderer is now `MSCombiDY2D.swf`, and the XML data source is now `yieldcurve.xml`.

## 6.1 Modifying the Default Layout

We will now add a Combination (Dual Y) Fusion Chart to the default "Foreign Exchange" layout of the Reference Implementation.

Open the file `Default_FX_Layout.xml` in a suitable editor. This file contains the XML configuration for the default "Foreign Exchange" layout and can be found at:

`$CTC_INSTALL_DIR/build/xml/layouts/`

We will place the Fusion chart at the left hand side of the layout. To do this, move to the start of the file, insert the highlighted code at the position shown below, and then save the file.

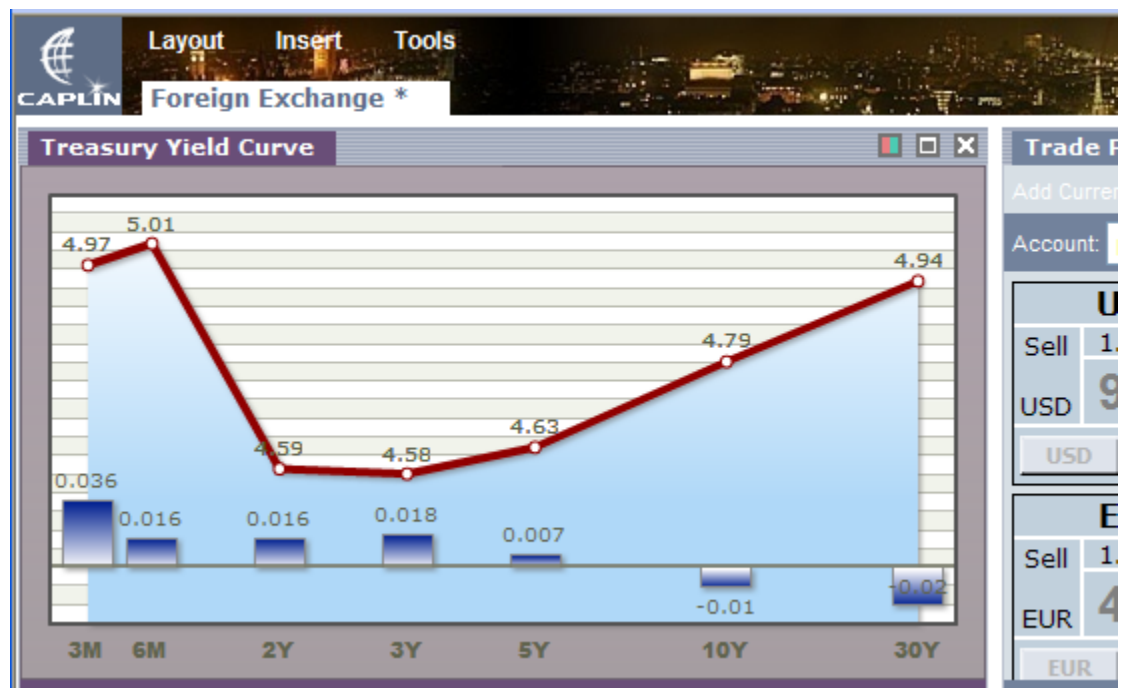
```
<Tower xmlns:caplin="www.caplin.com" splitters="true">
  <FrameItems>
    <Terrace splitters="true">
      <FrameItems>

<!-- Adding a Fusion Chart -->

<caplin:Panel caption="Treasury Yield Curve"
  drop_target="SNAP_FRAMEITEM" colour="colour-12" decorators="basicDecorator">
  <state>
    <caplinx:fusion>
      <charttype>MSCombiDY2D</charttype>
      <dataurl>source/xml/charts/yieldcurve.xml</dataurl>
      <backgroundcolour>#fff</backgroundcolour>
    </caplinx:fusion>
  </state>
</caplin:Panel>
```

Since Caplin Trader Client does not read this XML file directly you must re-populate the user preferences database after you have made the changes; see [Re-populating the User Preferences Database](#)<sup>[28]</sup>.

When you have saved the XML file and re-populated the database, clear your browser cache and refresh Caplin Trader Client. The changes you made will now be available to the application.



Combination (Dual Y) Fusion Chart added to a Layout



## 7 Adding a Grid

Caplin Trader Client can display data inside Panels in a grid format. Grids are an ideal mechanism for displaying large quantities of summary data in tabular format. In the example below a grid is being used to display real-time market data for major FX currency pairs.



| Currency | Rate          |
|----------|---------------|
| EURUSD   | 1.3339/1.3341 |
| USDJPY   | 123.60/123.61 |
| GBPUSD   | 2.0811/2.0814 |
| USDCHF   | 1.2434/1.2436 |
| AUDUSD   | 0.8381/0.8384 |
| USDCAD   | 1.0628/1.0637 |
| NZDUSD   | 0.7524/0.7527 |
| EURJPY   | 164.87/164.92 |
| EURGBP   | 0.6409/0.6411 |
| EURCHF   | 1.6586/1.6592 |

**Grid displaying FX  
Currency Pairs**

You insert a grid in the default layout using the `<caplin:Panel>` tag in conjunction with the `<grid>` tag. We look at two ways to define the grids you want to insert in [Inline Grid Definitions](#)<sup>[15]</sup> and [Inheriting Grid Definitions](#)<sup>[19]</sup>.

### 7.1 Inline Grid Definitions

The code below shows some XML code that will place a grid in the default layout by defining the grid inline. The grid displays Major FX instruments.

```

<caplin:Panel caption="Major"
  drop_target="SNAP_FRAMEITEM" colour="colour-2" decorators="basicDecorator">
  <state>
    <grid displayedColumns="description, rate">
      <gridRowModel>
        <rttpContainerGridDataProvider container="/CONTAINER/FX/Major" />
      </gridRowModel>
      <columnDefinitions>
        <column id="description"
          fields="InstrumentDescription"
          displayName="Currency" />
          width="70"/>
        <column id="rate"
          fields="Rate"
          displayName="Rate" />
          width="100"/>
        <column id="bestbid"
          fields="BestBid"
          displayName="Best Bid"
          width="100"/>
        <column id="bestask"
          fields="BestAsk"
          displayName="Best Ask"
          width="100"/>
      </columnDefinitions>
    </grid>
  </state>
</caplin:Panel>

```

The container for a grid is a Panel, and the attributes of the `<caplin:Panel>` tag shown above define the behaviour and appearance of the Panel when it is rendered.

**Tip:** You will find tutorials on modifying the layout and appearance of Caplin Trader Client in the document **Caplin Trader Client: Customizing the Appearance**.

The XML between the `<state>` and `</state>` tags defines the content that will be placed in the container, and the `<grid>` tag identifies this content to be a grid component. The JavaScript class that is responsible for creating grid components uses the XML code from this inline grid definition to configure the grid when it is created.

- ◆ The `<gridRowModel>` tag defines the row model that fills the grid; in this case the named RTTP container `<rttpContainerGridDataProvider>` obtains its data from `"/CONTAINER/FX/Major"`. This RTTP container supplies data for Major FX instruments.
- ◆ The `<columnDefinitions>` tag identifies the start of the column definitions and the `<column>` tag defines the individual grid columns. In this case there are four columns. The `id` attribute uniquely identifies each column, the `displayName` attribute defines the text that will appear in the column heading when the grid is rendered, the `fields` attribute identifies the data field that the column cell will display, and the `width` attribute defines the width of the column in pixels.
- ◆ The `displayedColumns` attribute of the `<grid>` tag identifies the columns that will be displayed when the grid is first rendered.

**Tip:** End users will be able to change the columns that are displayed in a grid in later releases of Caplin Trader Client.

You can insert the grid in the default layout by adding this inline grid definition to the file *\$CTC\_INSTALL\_DIR/build/xml/layouts/Default\_FX\_Layout.xml*. However grids that are defined in this file cannot be added to the Insert menu, and therefore end users will not be able to insert the grid in other layouts that they create.

## Adding a Grid to the Insert Menu

The XML code below defines the same grid but adds two attributes to the `<grid>` tag; the attributes `id` and `displayName`. If this inline grid definition is added to the file *\$CTC\_INSTALL\_DIR/conf/gridDefinitions.xml* (instead of *Default\_FX\_Layout.xml*) then the end user will be able to insert the grid from the Insert menu.

```
<caplin:Panel caption="Major"
  drop_target="SNAP_FRAMEITEM" colour="colour-2" decorators="basicDecorator">
  <state>
    <grid id="FX.Major" displayName="Major" displayedColumns="description, rate">
      <gridRowModel>
        <rttpContainerGridDataProvider container="/CONTAINER/FX/Major" />
      </gridRowModel>
      <columnDefinitions>
        <column id="description"
          fields="InstrumentDescription"
          displayName="Currency" />
        <column id="rate"
          fields="Rate"
          displayName="Rate" />
        <column id="bestbid"
          fields="BestBid"
          displayName="Best Bid"
          width="100"/>
        <column id="bestask"
          fields="BestAsk"
          displayName="Best Ask"
          width="100"/>
      </columnDefinitions>
    </grid>
  </state>
</caplin:Panel>
```

The `displayName` attribute defines the name of the grid that is added to the Insert menu. The `id` attribute uniquely identifies the grid and is used to refer to the grid when the grid is inserted in the default layout. Since the grid definition is now in a separate file, the XML code to insert this grid in the default layout is now as shown below.

```
<caplin:Panel caption="Major"
  drop_target="SNAP_FRAMEITEM" colour="colour-2" decorators="basicDecorator">
  <state>
    <grid baseGrid="FX.Major" />
  </state>
</caplin:Panel>
```

The `baseGrid` attribute of the `<grid>` identifies the grid that will be inserted, by referring to the unique `id` of the grid in the inline definition ("FX.Major" in this case). It is possible in the default layout to override the attributes of grids that are defined in *gridDefinitions.xml*, for example by re-defining column widths or headings using the

<columnDefinitions> and <column> tags.

The code below changes the heading for the column (id="description") from "Currency" to "Currency Pair" by overriding the displayName attribute.

```
<caplin:Panel caption="Major"
  drop_target="SNAP_FRAMEITEM" colour="colour-2" decorators="basicDecorator">
  <state>
    <grid baseGrid="FX.Major">
      <columnDefinitions>
        <column id="description">
          displayName="Currency Pair" />
        </columnDefinitions>
      </grid>
    </state>
  </caplin:Panel>
```

Grids have the ability to inherit characteristics from other grids and grid templates. In the two examples above, the grid in the default layout inherits the characteristics of the "FX.Major" grid by setting the baseGrid attribute of the <grid> tag to the unique id of the inline grid definition (baseGrid="FX.Major"). In [Inheriting Grid Definitions](#)<sup>19</sup>, we look at how inheritance can be used to minimize the XML code that is required to define grids.

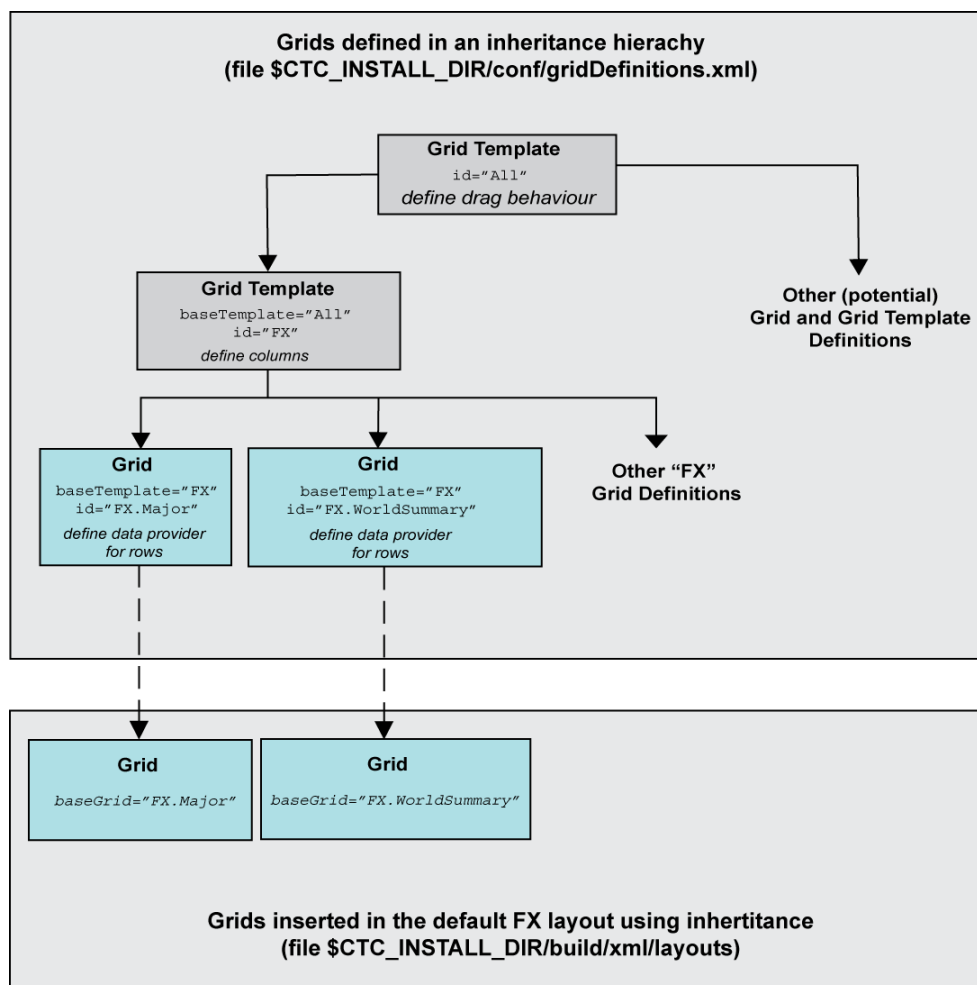
## 7.2 Inheriting Grid Definitions

The code below shows how two grids could be defined inline. The first grid will display Major FX instruments and the second World Cross Rates FX instruments.

```
<caplin:Panel caption="Major"
  drop_target="SNAP_FRAMEITEM" colour="colour-2" decorators="basicDecorator">
  <state>
    <grid id="FX.Major" displayedColumns="description, rate">
      <gridRowModel>
        <rttpContainerGridDataProvider container="/CONTAINER/FX/Major" />
      </gridRowModel>
      <columnDefinitions>
        <column id="description"
          fields="InstrumentDescription"
          displayName="Currency" />
        <column id="rate"
          fields="Rate"
          displayName="Rate" />
      </columnDefinitions>
    </grid>
  </state>
</caplin:Panel>

<caplin:Panel caption="World Cross Rates"
  drop_target="SNAP_FRAMEITEM" colour="colour-2" decorators="basicDecorator">
  <state>
    <grid id="FX.WorldSummary" displayedColumns="description, rate">
      <gridRowModel>
        <rttpContainerGridDataProvider container="/CONTAINER/FX/WorldSummary" />
      </gridRowModel>
      <columnDefinitions>
        <column id="description"
          fields="InstrumentDescription"
          displayName="Currency" />
        <column id="rate"
          fields="Rate"
          displayName="Rate" />
      </columnDefinitions>
    </grid>
  </state>
</caplin:Panel>
```

You will notice that some of the XML code is duplicated for each grid, as highlighted in the example above. This is because the grids share common characteristics; in this case they have identical column definitions. You can minimize this code duplication using inheritance, as illustrated in the diagram below.



## Grid Inheritance

The topmost grid template (`id="All"`) simply defines the drag behaviour of all grids that inherit from it. The `"FX"` grid template inherits this drag behaviour and adds column definitions. The `"FX.Major"` and `"FX.WorldSummary"` grids inherit from the `"FX"` template and define the specific data providers that will populate the rows of each grid. The JavaScript class that is responsible for creating grid components uses the XML code from this inherited grid definition to configure the grids when they are created.

The path to other potential grid and grid template definitions shows a possible inheritance path for defining other grids that inherit from the `"All"` grid template. This path is not discussed further in this document but illustrates one way to extend the inheritance hierarchy.

In the Reference Implementation of Caplin Trader Client all grids are defined in the file `$CTC_INSTALL_DIR/source/grid/gridDefinitions.xml`. If you want to add a new grid to the default layout using the `baseGrid` attribute then you must add to `gridDefinitions.xml` the XML code that defines the new grid. Grids defined in this file will also be available on the Insert menu for the end user to insert in a new layout. Grid templates define the common characteristics that other grids and templates can inherit, but they cannot be inserted in a layout or added to a menu.

**Tip:** Grids in the default layout can also inherit directly from grid templates (rather than grids) if the `baseTemplate` attribute of the `<grid>` tag is set to the `id` of the template that it will inherit from.

## 7.3 The Grid Definitions File

We will now look at some XML code from the file *gridDefinitions.xml*. Grids defined in this file use the inheritance hierarchy that is shown in the diagram of [Inheriting Grid Definitions](#)<sup>[19]</sup>.

Caplin Trader Client loads grid definitions from the file *\$CTC\_INSTALL\_DIR/conf/gridDefinitions.xml* each time the application opens in a browser window. This has the advantage that grids that have been modified or added by the system administrator will be available to an end user the next time the user logs in. This will be true even if the end user has modified their version of the grids, perhaps by changing column widths or even adding/removing columns the last time they logged in.

The code sample below is taken from *gridDefinitions.xml* and shows the XML code that defines the "FX.WorldSummary" and "FX.Major" grids. However, the inheritance principals described here can be applied to other grids that you add to this file.

**Tip:** The grid definitions for "FX.WorldSummary" and "FX.Major" have already been added to *gridDefinitions.xml*. You only need to modify this file if you want to add new grid definitions or modify existing grid definitions.

```

<?xml version="1.0"?>
<gridDefinitions xmlns="http://www.caplin.com/CaplinTrader/grid">

  <dataProviderMappings>
    <dataProviderMapping id="rttcpContainerGridDataProvider"
      className="caplin.grid.RtcpContainerGridDataProvider" />
  </dataProviderMappings>

  <decoratorMappings>
    <decoratorMapping id="dragDecorator"
      className="caplin.grid.decorator.DragDecorator" />
  </decoratorMappings>

  <templates>
    <gridTemplate id="All">
      <decorators>
        <dragDecorator />
      </decorators>
    </gridTemplate>

    <gridTemplate id="FX" baseTemplate="All" displayedColumns="description,rate">
      <columnDefinitions>
        <column id="description"
          fields="InstrumentDescription"
          displayName="Currency" width="70"/>

        <column id="rate"
          cellRenderer="caplin.dom.renderer.RateTextRenderer"
          fields="Rate"
          displayName="Rate"
          width="100"/>

        <column id="bestbid"
          cellRenderer="caplin.dom.renderer.TradableElementRenderer"
          fields="BestBid"
          displayName="Best Bid"
          width="100"/>

        <column id="bestask"
          cellRenderer="caplin.dom.renderer.TradableElementRenderer"
          fields="BestAsk"
          displayName="Best Ask"
          width="100"/>
      </columnDefinitions>
    </gridTemplate>
  </templates>

  <grids>
    <folders displayName="Foreign Exchange">
      <grid id="FX.Major" displayName="Major" baseTemplate="FX">
        <gridRowModel>
          <rttcpContainerGridDataProvider container="/CONTAINER/FX/Major" />
        </gridRowModel>
      </grid>

      <grid id="FX.WorldSummary" displayName="World Cross Rates" baseTemplate="FX">
        <columnDefinitions>
          <column id="description" headerRenderer="textfilter" />
        </columnDefinitions>
        <gridRowModel>
          <rttcpContainerGridDataProvider container="/CONTAINER/FX/WorldSummary" />
        </gridRowModel>
      </grid>
    </folders>

    <!-- Other grid definitions -->

  </grids>
</gridDefinitions>

```



## An explanation of the XML grid definition

Here is an explanation of the above XML grid definition and how this relates to what the end-user sees on the screen.

**Tip:** You will find a complete reference to the XML-based configuration that defines the layout and functionality of grids in the document **Caplin Trader Client: Grid Configuration XML Reference**.

- ◆ **<dataProviderMappings>** contains the definition of a single data provider that supplies data to a grid:

```
<dataProviderMappings>
  <dataProviderMapping id="rttContainerGridDataProvider"
    className="caplin.grid.RttContainerGridDataProvider" />
</dataProviderMappings>
```

The **<dataProviderMapping>** tag defines the JavaScript class that implements the data provider. It maps this class to an id (`rttContainerGridDataProvider`) that the rest of the XML configuration can use as a tag (`<rttContainerGridDataProvider>`) to refer to this particular data provider.

- ◆ **<decoratorMappings>** contains the definition of a grid decorator named `dragDecorator`:

```
<decoratorMappings>
  <decoratorMapping id="dragDecorator"
    className="caplin.grid.decorator.DragDecorator" />
</decoratorMappings>
```

Decorators define various aspects of what the grid looks like and how it behaves. In this particular case the drag decorator is specified. A drag decorator allows an instrument to be dragged out of a grid, so it can be dropped into another screen component, such as a trade panel.

The **<decoratorMapping>** tag defines the JavaScript class that implements the drag decorator. It maps this class to an id (`dragDecorator`) that the rest of the XML configuration can use as a tag (`<dragDecorator>`) to refer to this particular decorator.

- ◆ **<templates>** contains the definitions of two grid templates (`<gridTemplate>`):

```
<templates>
  <gridTemplate id="All">
    <decorators>
      <dragDecorator />
    </decorators>
  </gridTemplate>

  <gridTemplate id="FX" baseTemplate="All" displayedColumns="description,rate">
    ...
  </gridTemplate>
</templates>
```

Grid templates allow grids to be defined in an inheritance hierarchy, but unlike grids they cannot be added to the Insert menu. In this example there is a basic grid template named "ALL", which consists just of a drag decorator, so it allows instruments to be dragged out of it.

The grid template named "FX" inherits the characteristics of the "ALL" template (`baseTemplate="All"`), so it too can have instruments dragged out of it. The "FX" grid template additionally defines the columns of an FX grid:

```
<gridTemplate id="FX" baseTemplate="All" displayedColumns="description,rate">
  <columnDefinitions>
    <column id="description"
      fields="InstrumentDescription"
      displayName="Currency"
      width="70"/>

    <column id="rate"
      cellRenderer="caplin.dom.renderer.RateTextRenderer"
      fields="Rate"
      displayName="Rate"
      width="100"/>

    <column id="bestbid" .../>
    <column id="bestask" ... />
  </columnDefinitions>
</gridTemplate>
```

The `<column>` tags define four grid columns with the `id` identifier attributes `description`, `rate`, `bestbid`, and `bestask`.

Each `<column>` tag defines:

- The pixel width of the column (`width="70"` and `width="100"`).
- The data field or fields to be displayed in each cell of that column (`fields="InstrumentDescription"` and `fields="Rate"`).
- The text to be displayed in the heading of the column (`displayName="Currency"` and `displayName="Rate"`).
- An optional renderer that modifies the display format and behavior (if any) of the cells in the column (`cellRenderer="caplin.dom.renderer.RateTextRenderer"`).

The `displayedColumns` attribute of the template (`displayedColumns="description,rate"`) specifies that when an FX grid is first displayed, only the description and rate columns are to be shown on the screen.

- ◆ **<grids>** defines the actual grids that grids in a layout can inherit from:

```
<grids>
  <folders displayName="Foreign Exchange">
    <grid id="FX.Major" displayName="Major" baseTemplate="FX">
      <gridRowModel>
        <rttpContainerGridDataProvider container="/CONTAINER/FX/Major" />
      </gridRowModel>
    </grid>

    <grid id="FX.WorldSummary" displayName="World Cross Rates" baseTemplate="FX">
      <columnDefinitions>
        <column id="description" headerRenderer="textfilter" />
      </columnDefinitions>
      <gridRowModel>
        <rttpContainerGridDataProvider container="/CONTAINER/FX/WorldSummary" />
      </gridRowModel>
    </grid>
  </folders>

  <!-- Other grid definitions -->
</grids>
```

Each grid is defined in a **<grid>** tag. In this example the "FX.WorldSummary" and "FX.Major" grids inherit characteristics from the "FX" grid template, such that each grid displays description and rate information in columns defined by the "FX" grid template. Each grid also allows the end user to drag an instrument out of the grid; it inherits this behaviour from the top level "All" grid template.

The two grids use different row models, as defined by the **<gridRowModel>** tag.

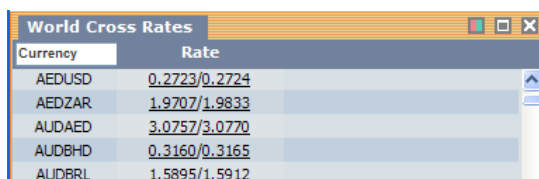
The row model defines the data provider that fills the grid; in this case a named RTTP container (**<rttpContainerGridDataProvider>**). Each grid obtains its data from different RTTP containers ("**/CONTAINER/FX/Major**" and "**/CONTAINER/FX/WorldSummary**") so the two grids display different sets of FX instruments.

The XML for the "FX.WorldSummary" grid also includes an additional column definition:

```
<columnDefinitions>
  <column id="description" headerRenderer="textfilter" />
</columnDefinitions>
```

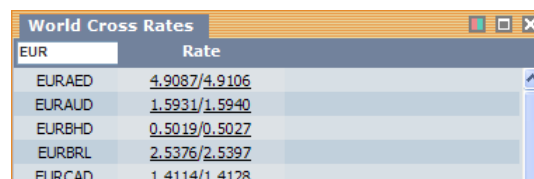
The "description" column has a header renderer (**headerRenderer="textfilter"**).

The "textfilter" renderer is supplied with Caplin Trader Client. It provides a text box into which the end user can enter text to filter the rows displayed in the grid. In this particular case the text filter allows the user to select a subset of the currency pairs. For example, entering "EUR" will cause all the currency pairs starting with EUR to be displayed (EURAUD, EURBHD, and so on).



| Currency | Rate          |
|----------|---------------|
| AEDUSD   | 0.2723/0.2724 |
| AEDZAR   | 1.9707/1.9833 |
| AUDAUD   | 3.0757/3.0770 |
| AUDBHD   | 0.3160/0.3165 |
| AUDBRL   | 1.5895/1.5912 |

**Grid with text filter entry box  
(before text is entered)**



| Currency | Rate          |
|----------|---------------|
| EURAUD   | 4.9087/4.9106 |
| EURAUD   | 1.5931/1.5940 |
| EURBHD   | 0.5019/0.5027 |
| EURBRL   | 2.5376/2.5397 |
| EURCAD   | 1.4114/1.4128 |

**Currency pairs filtered for text EUR**

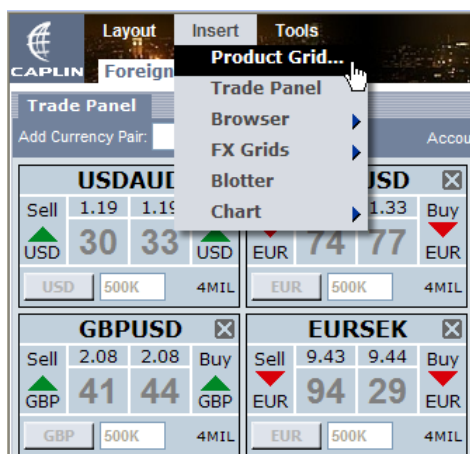
If the XML for the "FX.WorldSummary" grid did not specify a header renderer, then the default renderer is used; this just displays simple text in the header of every column.



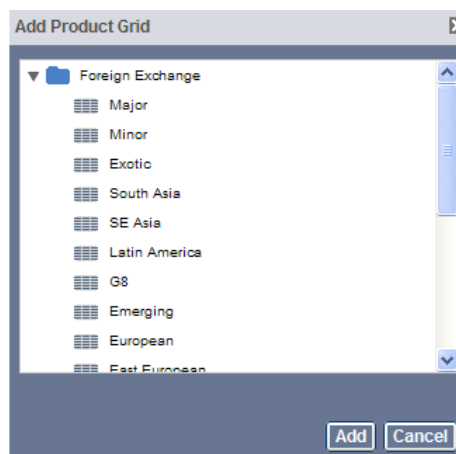
| Currency | Rate          |
|----------|---------------|
| AEDAUD   | 0.3247/0.3248 |
| AEDBHD   | 0.1012/0.1014 |
| AEDBRL   | 0.5181/0.5185 |
| AEDCAD   | 0.2896/0.2899 |

**Grid with default column headings**

The `<folders displayName="Foreign Exchange">` tag organizes the grids in a nested hierarchy. Grids added to the Insert menu will be organized using this hierarchy (like files in a directory) when the "Add Product Grid" dialog is displayed.



**Insert Menu**



**Grids organized in a hierarchy**

## Naming the Grid Definitions File

The URL of the file that contains the grid definitions is set in `$CTC_INSTALL_DIR/application.jsp` using the method `caplin.core.ApplicationProperties.setProperty()`. In the Reference Implementation of Caplin Trader

Client the URL is *conf/gridDefinitions.xml*. Grids defined in this file can be inserted in the default layout and will also be available on the Insert menu, allowing users to add them to new layouts.

## 7.4 Modifying the Default Layout

We will now add a grid that displays FX instruments from the RTTP container *"/CONTAINER/FX/WorldSummary"* to the default "Foreign Exchange" layout of the Reference Implementation.

Open the file *Default\_FX\_Layout.xml* in a suitable editor. This file contains the XML configuration for the default "Foreign Exchange" layout and can be found at:

*\$CTC\_INSTALL\_DIR/build/xml/layouts/*

We will place the new grid at the left hand side of the layout. To do this, move to the start of the file, insert the highlighted code at the position shown below, and then save the file.

```
<Tower xmlns:caplin="www.caplin.com" splitters="true">
  <FrameItems>
    <Terrace splitters="true">
      <FrameItems>

<!-- Adding a Grid -->

<caplin:Panel caption="World Cross Rates"
  drop_target="SNAP_FRAMEITEM" colour="colour-2" decorators="basicDecorator">
  <state>
    <grid baseGrid="FX.WorldSummary" />
  </state>
</caplin:Panel>
```

Since Caplin Trader Client does not read this XML file directly you must re-populate the user preferences database after you have made the changes; see [Re-populating the User Preferences Database](#) <sup>28</sup>.

When you have saved the XML file and re-populated the database, clear your browser cache and refresh Caplin Trader Client. The changes you made will now be available to the application.



| Currency | Rate          |
|----------|---------------|
| AEDAUD   | 0.3257/0.3258 |
| AEDBHD   | 0.1014/0.1016 |
| AEDBRL   | 0.5199/0.5203 |
| AEDCAD   | 0.2892/0.2894 |
| AEDCHF   | 0.3386/0.3387 |
| AEDCNY   | 2.0783/2.0789 |
| AEDDKK   | 1.5209/1.5215 |
| AEDDZD   | 17.772/17.927 |
| AEDEEK   | 3.1943/3.1957 |

**Grid added to a Layout**

## 8 Re-populating the User Preferences Database

Although Caplin Trader Client is configured in XML it does not read the XML files directly. If you make layout changes to an XML file, for example to *Default\_FX\_Layout.xml* or *application.xml*, then you must re-populate the user preferences database to make the changes available to your application.

- Open a terminal shell (if one is not already open).
- Type `cd $CT_INSTALL_DIR` (followed by <RETURN>) to change to the install directory.
- Type `java -jar kits/CaplinTrader/webcentric_database_populator.jar`  
`apps/webapps/caplintrader/applications/CaplinTrader/build/xml`  
(followed by <RETURN>).

The user preferences database will now be re-populated. Any changes you made to the XML will be available to Caplin Trader Client when you clear the browser cache and refresh the application.

## 9 Glossary of terms and acronyms

This section contains a glossary of terms and acronyms relating to the Caplin Trader Client product.

| Term   | Definition  |
|--|---|
| <b>Ajax</b>  | <u>A</u> syncronous <u>J</u> avaScript and <u>X</u> ML  |
| <b>API</b>   | <u>A</u> pplication <u>P</u> rogramming <u>I</u> nterface   |
| <b>blotter</b>                                       | A <b>Caplin Trader Client</b> component for displaying details of completed trades in tabular format.   |
| <b>Caplin Liberator</b>                              | A bidirectional streaming push server that delivers trade messages and market data to <b>Caplin Trader Client</b> over any network.   |
| <b>Caplin Trader</b>                                 | A framework for building multi-product financial trading portals.   |
| <b>Caplin Trader Client</b>                          | The client (browser) side components of <b>Caplin Trader</b> .  |
| <b>Caplin Trader Client Reference Implementation</b> | A reference implementation of a financial trading portal.   |
| <b>container (RTTP)</b>                              | See <b>RTTP container</b> .   |
| <b>container (webcentric)</b>                        | See <b>webcentric container</b> .   |
| <b>Default Layout</b>                                | The layout of Panels that is displayed by default (without intervention by the end user) when <b>Caplin Trader Client</b> opens in a web browser.   |
| <b>Flash</b>   | A set of multimedia technologies that add animation and interactivity to web pages.   |
| <b>Frame</b>   | A rectangular area in the <b>Caplin Trader</b> user interface in which external HTML content may be displayed.  |
| <b>Fusion Chart</b>                                  | A <b>Flash</b> object that can be inserted in a <b>Caplin Trader Client</b> layout to render data driven charts.  |
| <b>Grid</b>  | A <b>Caplin Trader Client</b> component for displaying large quantities of summary data in tabular format. The container for a <b>Grid</b> is a <b>Panel</b> .  |
| <b>Grid Template</b>                                 | Defines characteristics that <b>Grids</b> (and other Grid Templates) can inherit. Unlike a <b>Grid</b> , a Grid Template cannot be inserted in a layout or appear as a menu option.   |
| <b>Grid Definitions File</b>                         | A file that contains definitions for the grids that can be inserted in a layout. <b>Caplin Trader Client</b> loads <b>Grid</b> definitions from this file each time it opens in a web browser.  |
| <b>GUI</b>   | <u>G</u> raphical <u>U</u> ser <u>I</u> nterface  |
| <b>Grid Inheritance</b>                              | The ability of <b>Grids</b> and <b>Grid Templates</b> to inherit characteristics from other grids and grid templates. Inheritance can be used to minimize the amount of <b>XML</b> code that is required to define each <b>Grid</b> . |
| <b>JavaScript</b>                                    | A scripting language that is used for client-side web development. <b>Caplin Trader Client</b> is implemented in JavaScript.  |
| <b>Panel</b>   | A rectangular area in the <b>Caplin Trader Client</b> user interface for displaying content and interactive components ( <b>widgets</b> ).  |
| <b>Layout</b>  | An arrangement of <b>Panels</b> on the <b>Caplin Trader Client</b> page. More than one Layout can be inserted in a page, each displaying different information. Tabs allow the end user to switch between layouts.                    |

| Term                             | Definition   |
|----------------------------------|--|
| <b>RTTP</b>                      | <u>Real Time Text Protocol</u><br>Caplin's object-oriented, real-time, protocol for the distribution of financial data and trade messages over internet-protocol networks between client applications (such as <b>Caplin Trader Client</b> ) and <b>Caplin Liberator</b> .   |
| <b>RTTP Container</b>            | <b>RTTP container</b> objects allow clients, such as <b>Caplin Trader Client</b> , to subscribe to dynamically managed collections of objects through references. In <b>Caplin Trader Client</b> it is used to display the rows of <b>grids</b> and blotters.<br><br>In this document the term <b>RTTP container</b> refers to an <b>RTTP</b> container object (rather than a <b>webcentric container</b> ). |
| <b>Stack</b>                     | A <b>Caplin Trader</b> window component that acts as a container for other components and arranges them one layer on top of another. A number of Tabs are provided to allow the user to switch between the layers.   |
| <b>Tab</b>                       | A GUI element that allows the end user to switch between stacked components (such as when <b>Grids</b> are layered in a <b>Stack</b> ).  |
| <b>Terrace</b>                   | A <b>Caplin Trader</b> window component that acts as a container for other components and arranges them in a horizontal row.   |
| <b>Tower</b>                     | A <b>Caplin Trader</b> window component that acts as a container for other components and arranges them in a vertical column.  |
| <b>User Preferences Database</b> | A database that holds configuration details for each end user. When an end user saves changes to a <b>Layout</b> , the changes are saved to this database.   |
| <b>webcentric</b>                | A client-side portal framework that gives <b>Caplin Trader Client</b> the look and feel of a windows style desktop application.  |
| <b>webcentric container</b>      | In <b>webcentric</b> , a container is a <b>GUI</b> windows component. <b>Panels</b> , <b>Stacks</b> , <b>Terraces</b> and <b>Towers</b> are <b>webcentric containers</b> .<br><br>In the text of this document the term ' <b>container</b> ' refers to a webcentric container (rather than an <b>RTTP container</b> ).   |
| <b>widget</b>                    | An interactive component that can be inserted in a <b>Layout</b> .   |
| <b>XML</b>                       | <u>Extensible Markup Language</u> . <b>XML</b> files are used to configure <b>Caplin Trader Client</b> .   |





*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.*

## Contact Us

Caplin Systems Ltd.  
Triton Court  
14 Finsbury Square  
London EC2A 1BR  
UK

Telephone: +44 20 7826 9600  
Fax: +44 20 7826 9610

**[www.caplin.com](http://www.caplin.com)**