CAPLIN

Caplin Trader Client 1.4

Tile Configuration XML Reference

March 2009

CONFIDENTIAL

Contents

1 Preface 1

1.1 What this document contains 1
About Caplin document formats 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

2 Introduction to tile views 3

3 Getting started 5

3.1 Technical assumptions and restrictions 5
3.2 Using the XML configuration markup 5
Example tile view configuration 5
Ordering and nesting of tags 10

4 XML Reference information 12

4.1 <decoratorMapping> 12
4.2 <decoratorMappings> 12
4.3 <decorators> 12
4.4 <dropDecorator> 13
4.5 <header> 13
4.6 <headerMapping> 14
4.7 <headerMappings> 14
4.8 <templates> 14
4.9 <tile> 15
4.10 <tiles> 15
4.11 <tileView> 15
4.12 <tileViewDefinitions> 16
4.13 <tileViewTemplate> 16

5 Glossary of terms and acronyms 17

1 Preface

1.1 What this document contains

This reference document describes the XML-based configuration that defines the layout and functionality of tile views displayed in Caplin Trader Client.

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

About Caplin document formats

This document is supplied in three formats:

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

For the best reading experience

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

Restrictions on viewing .CHM files

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

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

1.2 Who should read this document

This document is intended for System Administrators and Software Developers who need to configure tile views for Caplin Trader Client.

1.3 Related documents

- **Caplin Trader Client: Customizing The Appearance**

  This document describes how to modify the layout of Caplin Trader Client Reference Implementation. It also explains how to change aspects of the look and feel of Caplin Trader Client.

1.4 Typographical conventions

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

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

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

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

1.5 Feedback

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

Please email your feedback to documentation@caplin.com.

1.6 Acknowledgments

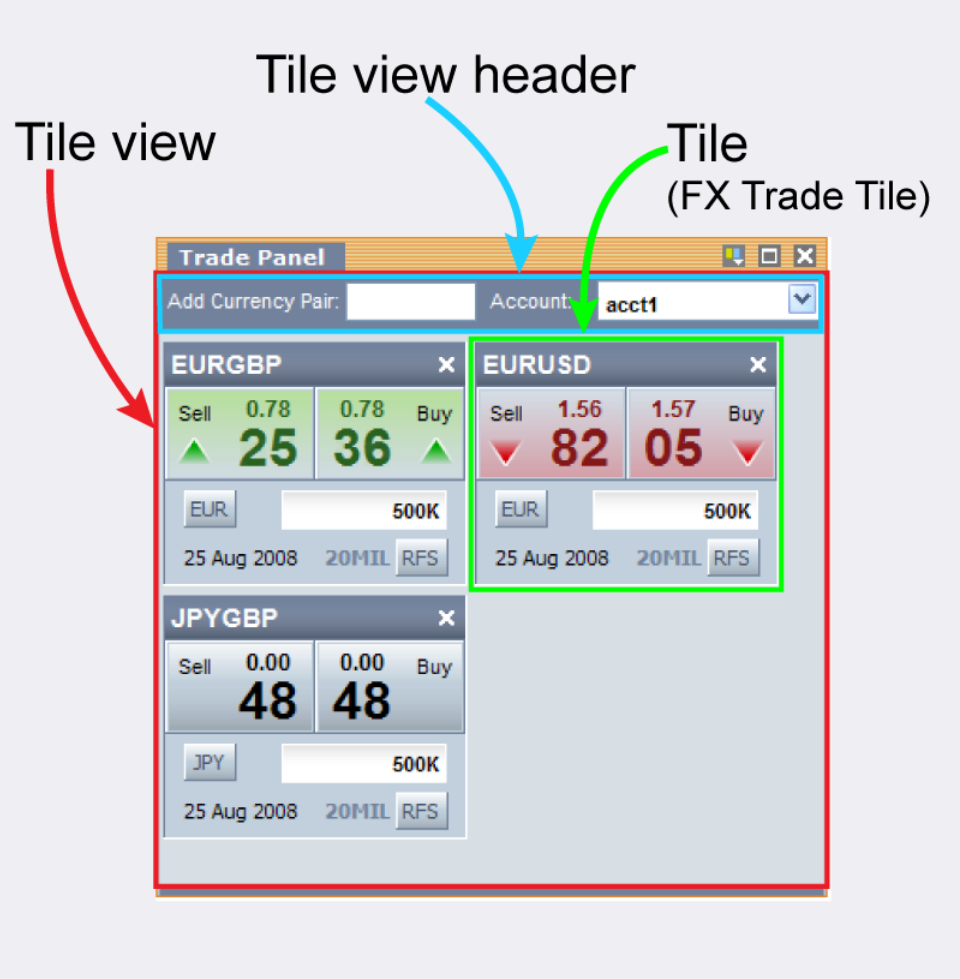
2 Introduction to tile views

Caplin Trader Client can display information about financial instruments in a tiled view, where there is a tile for each instrument. Typically end-users can perform actions on an instrument displayed in a tile, such as executing trades on that instrument.

For example, the following picture shows a Trade Panel containing three Trade Tiles, where each tile displays an FX currency pair whose buy and sell prices are updated in real time; users can trade a currency by clicking on the Buy or Sell button in the tile.

FX Trade Tiles displayed in Caplin Trader Client

You can configure the appearance and behavior of tiles using the XML configuration defined in this document. The XML uses some terminology related to how tiles are organized. This is shown in the following picture:



Tile organization and terminology

A **tile view** is a group of tiles displayed within a panel. The tiles in a tile view all have the same characteristics and behavior. For example, in the above picture the tile view contains three tiles, where each tile displays an FX currency pair and allows the user to trade the currency pair.

A tile view has a **tile view header** that allows information common to all the tiles in the view to be displayed. The tile view header can also have associated controls and behaviors. In the above example the tile view header has an Add Currency Pair box that allows users to dynamically insert new Trade Tiles in the panel, and a drop down list of accounts that determines which Trade Tiles are enabled for trading according to the account selected.

3 Getting started

The following sections explain how the various XML tags may be combined to define FX Trade Tiles for Caplin Trader Client.

3.1 Technical assumptions and restrictions

XML

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

3.2 Using the XML configuration markup

Example tile view configuration

The XML that defines the tile view illustrated in Introduction to tile views 3 is shown below.

The XML Reference information 12 section defines the XML tags and attributes you can use to define tile layouts in Caplin Trader Client pages. Also see the section Ordering and nesting of tags 10.

1) Definition of the tile view template for FX tiles:

```
<?xml version="1.0" encoding="UTF-8" ?>

<tileViewDefinitions xmlns="http://www.caplin.com/CaplinTrader/tile">

   <headerMappings>
      <headerMapping id="fxHeader"
        className="caplinx.trading.presentation.tile.FxTileViewHeader"/>
   </headerMappings>

   <decoratorMappings>
      <decoratorMapping id="dropDecorator"
       className="caplin.trading.presentation.tile.decorator.DropDecorator" />
   </decoratorMappings>

   <templates>
      <tileViewTemplate id="FX">
         <header refId="fxHeader" />
         <decorators>
            <dropDecorator" ddGroup="fxInstruments" />
         </decorators>
      </tileViewTemplate>
   </templates>

</tileViewDefinitions>
```

2) Defining a tile view within a layout:

```
<tileView baseTemplate="FX">
   <tiles>
      <tile instrument="/FX/EURGBP" amount="1000" dealtCurrency="EUR" />
      <tile instrument="/FX/EURUSD"  amount="1000" dealtCurrency="EUR" />
      <tile instrument="/FX/JPYGBP"  amount="1000000" dealtCurrency="JPY" />
   </tiles>
</tileView>
```

3) Defining a tile view invoked from a menu:

```
<tileView baseTemplate="FX" />
```

An explanation of the example XML configuration

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

The XML in **(1)** defines a tile view template. This template is then used in the XML **(2)** to define an actual tile view that is constructed when Caplin Trader Client starts up.

The tile view template

The tile view template definition XML **(1)** is structured as follows:

- **`<tileViewDefinitions>`** contains one or more template definitions and associated decorator and tile view header definitions:

```
<tileViewDefinitions xmlns="http://www.caplin.com/CaplinTrader/tile">

   <headerMappings>
   ...
   </headerMappings>

   <decoratorMappings>
   ...
   </decoratorMappings>

   <templates>
    ...
   </templates>

</tileViewDefinitions>
```

- **<headerMappings>** defines the tile view headers that can be used in tile views defined by the templates:

```
<headerMappings>
   <headerMapping id="fxHeader"
     className="caplinx.trading.presentation.tile.FxTileViewHeader"/>
</headerMappings>
```

In this case there is just one tile view header called `fxHeader`. The JavaScript class that implements this tile view header is **caplinx.trading.presentation.tile.FxTileViewHeader**.

- **<decoratorMappings>** defines the decorators that can be used in tile views defined by the templates:

```
<decoratorMappings>
  <decoratorMapping id="dropDecorator"
    className="caplin.trading.presentation.tile.decorator.DropDecorator" />
</decoratorMappings>
```

In this case there is just one decorator: a drop decorator called `dropDecorator`. The JavaScript class that implements this decorator is **caplin.trading.presentation.tile.decorator.DropDecorator**.

- **<templates>** defines all the tile view templates (`<tileViewTemplate>`) available within the configuration file:

```
<templates>
   <tileViewTemplate id="FX">
   ...
   </tileViewTemplate>
</templates>
```

In this case there is just one tile view template called `FX`.

- `<tileViewTemplate id="FX">` defines the tile view template for FX trade tiles

```
  <tileViewTemplate id="FX">
      <header refId="fxHeader" />
      <decorators>
         <dropDecorator" ddGroup="fxInstruments" />
      </decorators>
  </tileViewTemplate>
```

The template uses the tile view header called `fxHeader`, whose implementation is defined in `<headerMappings>`. The tiles that the view can contain are formatted by and have their behavior determined by the decorators listed inside the `<decorators>` tag. In this case there is just one decorator called `dropDecorator`. This decorator allows instruments to be dropped into any tile view that uses this tile view template, provided the instrument has been dragged from a display

component (such as a grid or tree view) whose configuration XML specifies the same `ddGroup` value in its drag decorator.

Note that tile view templates allow tile views to be defined in an inheritance hierarchy. So instead of defining the `dropDecorator` specifically in this FX tile view template, it could be defined in a general tile view template from which any other tile view template, including the FX template, can inherit:

```
<tileViewTemplate id="ALL">
   <decorators>
      <dropDecorator" ddGroup="fxInstruments" />
   </decorators>
</tileViewTemplate>

<tileViewTemplate id="FX" baseTemplate="ALL">
   <header refId="fxHeader" />
</tileViewTemplate>
```

Defining a tile view in a layout

- The tile view template is used in the XML **(2)** to define an actual tile view displayed in a layout within Caplin Trader Client. When the layout is constructed it contains the trade tiles specified by the tile view:

```
<tileView baseTemplate="FX">
   <tiles>
      <tile instrument="/FX/EURGBP" amount="1000" dealtCurrency="EUR" />
      <tile instrument="/FX/EURUSD"  amount="1000" dealtCurrency="EUR" />
      <tile instrument="/FX/JPYGBP"  amount="1000000" dealtCurrency="JPY" />
   </tiles>
</tileView>
```

The tile view is defined using a `<tileView>` tag and it inherits characteristics from the FX template: `baseTemplate="FX"`

The view contains three predefined tiles. Each individual tile that appears in the view as a display component is defined as a `<tile>` tag within the enclosing `<tiles>` tag.

The `instrument` attribute of `<tile>` defines the name of the financial instrument that the tile displays and which end users can trade via the tile. In this case these are FX instruments, defined by a symbol of the form `/FX/{currency-pair}`. The tile for each instrument is displayed with an initial amount and an initial setting for the dealt currency.

Defining a tile view invoked from a menu

- The XML **(3)** uses the tile view template to define a tile view that is constructed dynamically from a menu selection (for example when a user of the Caplin Trader Client Reference Implementation selects Insert --> FX --> Trade Panel):

```
<tileView baseTemplate="FX" />
```

Since the tile view does not contain any predefined tiles, it only needs to specify the required template and does not need a `<tiles>` tag.

Ordering and nesting of tags

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

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

For a description of each tag and its attributes, see the XML Reference information 12 section.

<tileViewDefinitions>

This is the outermost tag

```
<tileViewDefinitions>
   <headerMappings></headerMappings> (one or more)
   <decoratorMappings></decoratorMappings> (zero or one)
   <templates></templates> (zero or one)
<tileViewDefinitions>
```

<headerMappings>

```
<headerMappings>
   <headerMapping /> (one or more)
</headerMappings>
```

<decoratorMappings>

```
<decoratorMappings>
   <decoratorMapping /> (one or more)
</decoratorMappings>
```

<templates>

```
<templates>
   <tileViewTemplate></tileViewTemplate> (one or more)
</templates>
```

<tileViewTemplate>

```
<tileViewTemplate>
   <header /> (zero or one)
   <decorators></decorators> (zero or one)
   <tiles></tiles> (zero or one)
</tileViewTemplate>
```

<tileView>

```
<tileView>
   <header /> (zero or one>
   <decorators></decorators> (zero or one)
   <tiles></tiles> (zero or one)
</tileView>
```

<decorators>

```
<decorators>
   <dropDecorator /> (one or more)
</decorators>
```

<tiles>

```
<tiles>
   <tile /> (one or more)
</tiles>
```

<headerMapping>

`<headerMapping />` (no children)

<decoratorMapping>

`<decoratorMapping />` (no children)

<header>

`<header />` (no children)

<dropDecorator>

`<dropDecorator />` (no children)

<tile>

`<tile />` (no children)

4 XML Reference information

This is the reference information for the tile configuration XML.

4.1 <decoratorMapping>

`<decoratorMapping>`

Defines a decorator that can be used in a tile view or tile view template. It maps the concrete name of the tile view decorator class to a logical name. Only the logical names of tile view decorators are serialized, not the concrete class names. This allows the class implementing a tile view decorator to be swapped for a different one, without affecting the ability to restore previously saved tile views that use this decorator.

Attributes:

Name	Type	Default	Req?	Description
`className`	string	(none)	Y	The class that is constructed to implement the tile view decorator.
`id`	string	(none)	Y	The logical name of a tile view decorator; for example, id="dropDecorator". This name must correspond to the name of a tag that can appear inside the <decorators> tag. For example, <decorators> can contain the <dropDecorator> tag, so the <decoratorMapping> that defines the class implementing the drop decorator has an id called "decoratorMapping".

4.2 <decoratorMappings>

`<decoratorMappings>`

A list of <decoratorMapping> tags that define the tile view decorators available within the application. Tile view decorators allow the functionality and/or look and feel of the tile view to be augmented in some way. For example, a dropDecorator could be added to allow the user to drop instruments into the tile view.

Attributes: This tag has no attributes.

4.3 <decorators>

`<decorators>`

Specifies one or more decorators that are applied to a particular tile view (<tileView>) or tile view template (<tileViewTemplate>). The name of a decorator tag (such as <dropDecorator>) is the logical name of the decorator (such as "dropDecorator") as specified in the id attribute of a <decoratorMapping>.

Attributes: This tag has no attributes.

4.4 <dropDecorator>

`<dropDecorator>`

Specifies a decorator that is applied to a particular tile view (<tileView>) or tile view template <tileViewTemplate>. The drop decorator allows instruments to be dropped into tile views that have specified this decorator in their XML configuration. This decorator is supplied with Caplin Trader Client.

Attributes:

Name	Type	Default	Req?	Description
`ddGroup`	string	(none)	Y	The drop decorator group that defines the type of instrument that can be dragged and dropped into this tile view. Only instruments that that are dragged with a ddGroup that matches this attribute value can be dropped. For example, an instrument can only be dragged from a grid or a tree view and dropped into into the tile view if the dragDecorator in the XML defining the grid or tree view has a ddGroup attribute that matches this ddGroup.

4.5 <header>

`<header>`

Contains information about the type of header to be used in a tile view or tile view template. If a tile view does not inherit from a <tileViewTemplate> that contains a <header> then it must itself contain a <header>.

Attributes:

Name	Type	Default	Req?	Description
`account`	string	Defined by coding	N	The name of the account that is used by the tile view header when the tile view initially loads.
`className`	string	(none)	N	The class that is constructed to display the header for the physical tile view (<tileView>). This attribute must only be used if the refId attribute is omitted.
`refId`	string	(none)	N	The logical name for the tile view header. The refId should match the id of the corresponding header mapping (<headerMapping>). If this attribute is omitted, a className attribute must be defined.

4.6 <headerMapping>

`<headerMapping>`

Defines a tile view header that can be used in a tile view or tile view template. It maps the concrete name of the tile view header class to a logical name. Only the logical names of the tile view headers are serialized, not the concrete class names. This allows the class implementing a tile view header to be swapped for a different one, without affecting the ability to restore previously saved tile views using that particular header.

Attributes:

Name	Type	Default	Req?	Description
`className`	string	(none)	Y	The class that is constructed to display the header for the physical tile view (<tileView>).
`id`	string	(none)	Y	The logical name for the tile view header. This is the name that is referred to by the <header> tag embedded in <tileView> and <tileViewTemplate> tags.

4.7 <headerMappings>

`<headerMappings>`

A list of <headerMapping> tags that define the tile view headers available within the application.

Attributes: This tag has no attributes.

4.8 <templates>

`<templates>`

Contains a list of tile view templates (<tileViewTemplate>).

Attributes: This tag has no attributes.

4.9 <tile>

`<tile>`

Defines a tile that appears within a tile view (<tileView>) or tile view template (<tileViewTemplate>).

Attributes:

Name	Type	Default	Req?	Description
`account`	string	Defined by coding	N	The name of the account that is used by the tile when it initially loads.
`amount`	string	Defined by coding	N	The amount that is set on the tile when it initially loads.
`dealtCurrency`	string	Defined by coding	N	The dealt currency that is set on the tile when it initially loads.
`instrument`	string	(none)	Y	The name of the financial instrument that the tile displays and which end users can trade via the tile.

4.10 <tiles>

`<tiles>`

Defines one or more tiles that comprise a tile view (<tileView>) or tile view template (<tileViewTemplate>).

Attributes: This tag has no attributes.

4.11 <tileView>

`<tileView>`

A specific instance of a tile view that is displayed in a panel within Caplin Trader Client. The tile view can have a parent <tileViewTemplate> from which it inherits characteristics; it can then enhance its own configuration through defining additional and/or overriding characteristics via child tags. Alternatively the tile view can define all its characteristics itself. If the <tileView> does not inherit from a <tileViewtemplate> that contains a <header> then it must itself contain a <header>.

Attributes:

Name	Type	Default	Req?	Description
`baseTemplate`	string	(none)	N	The <tileViewTemplate>, if there is one, that this tile view inherits from.

4.12 <tileViewDefinitions>

`<tileViewDefinitions>`

The outermost tag of the trade tile definition XML.

Attributes: This tag has no attributes.

4.13 <tileViewTemplate>

`<tileViewTemplate>`

A tile view template is an abstract tile view definition. Multiple physical tile views (<tileView>) can be defined using the same template; the physical tile views inherit the characteristics of the template. This allows various aspects of the physical tile views, such as the headers (<header>) and decorators (<decorator>) used, to be reconfigured simply by modifying the template(s) associated with the tile view.

Attributes:

Name	Type	Default	Req?	Description
`baseTemplate`	string	(none)	N	The <tileViewTemplate>, if there is one, that this template inherits from.
`id`	string	(none)	Y	The identifier of this tile view template. The identifier must be unique across all tile views (<tileView>) and tile view Templates (<tileViewTemplate>).

5 Glossary of terms and acronyms

This section contains a glossary of terms and acronyms relating to the Caplin Trader tile configuration XML.

Term	Definition
Decorator	A basic display component that is embedded in larger display components such as Grids and Tiles and determines aspects of the look and behavior of the larger component. For example a dropDecorator, when applied to a tile, allows instruments to be dropped into the tile when dragged from other display components.
Display component	In the context of Caplin Trader a display component is a GUI component that can be embedded in a Caplin Trader Client page. The term also refers to the JavaScript code that generates the component and handles its user interaction. Caplin Trader Client has a number of pre-defined, customizable display components, such as Grids, **Tiles**, and the Blotter.
Header	See **tile view header**.
Tile	A **display component** that occupies a square or rectangular area on the page. The most typical implementation of a tile is the **Trade Tile**. See also Introduction to tile views 3.
Tile view	See Introduction to tile views 3.
Tile view header	See Introduction to tile views 3.
Trade Tile	A Caplin Trader **display component** that allows the user to trade on a product with a single mouse click. A trade tile is implemented in a **tile view**.

CAPLIN THE REAL TIME WEB COMPANY

Contact Us

Caplin Systems Ltd
Triton Court
14 Finsbury Square
London EC2A 1BR
Telephone: +44 20 7826 9600
Fax: +44 20 7826 9610
www.caplin.com