

Caplin Trader Client 1.3

Tree View Configuration XML Reference

December 2008

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
2	Introduction to tree views.....	3
3	Getting started.....	6
3.1	Technical assumptions and restrictions.....	6
3.2	Using the XML configuration markup.....	6
	An example configuration file	6
	Ordering and nesting of tags	9
4	XML Reference information.....	11
4.1	<folder>.....	11
4.2	<item>.....	12
4.3	<templates>.....	12
4.4	<treeNodeMapping>.....	12
4.5	<treeNodeMappings>.....	13
4.6	<treeViewDefinitions>.....	13
4.7	<treeViewTemplate>.....	13
5	Glossary of terms and acronyms.....	15

1 Preface

1.1 What this document contains

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

The information in this document applies to Caplin Trader version 1.3.

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.

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 tree views in Caplin Trader Client.

1.3 Related documents

- ◆ **Ext JS API reference documentation**

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

1.4 Typographical conventions

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

Type	Uses
aMethod	Function or method name
<i>aParameter</i>	Parameter or variable name
/AFolder/Afile.txt	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
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

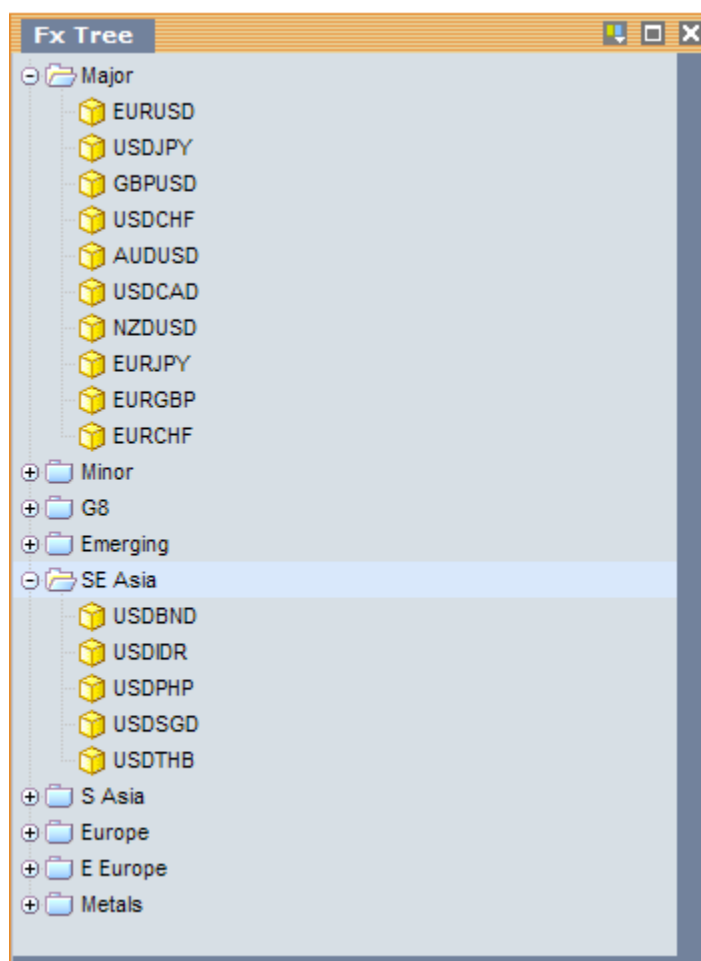
Customer feedback can only improve the quality of our product documentation, and we would welcome any comments, criticisms or suggestions you may have regarding this document.

Please email your feedback to documentation@caplin.com.

2 Introduction to tree views

Caplin Trader Client can arrange items in a tree structure that has an appearance similar to that found in a typical desktop file browser. A tree is a way of organizing items in a logical structure, and a tree view is how Caplin Trader Client displays that structure. In the reference implementation of Caplin Trader Client two trees have been pre-configured; one that displays FX instruments and another that displays FI instruments.

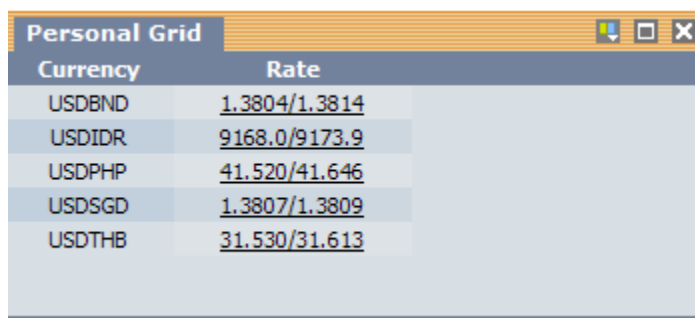
The example below shows a tree that has been configured to display FX instruments.



Tree view configured to display FX instruments

A tree view allows the end user to rapidly navigate the items in the tree by opening and closing folders. In the example above the folders **Major** and **SE Asia** have been opened, and you will see that each folder contains FX currency pairs arranged according to the title of the folder. The folders and the items inside folders can be copied to a Personal Grid or Trade Panel by dragging and dropping with the mouse.

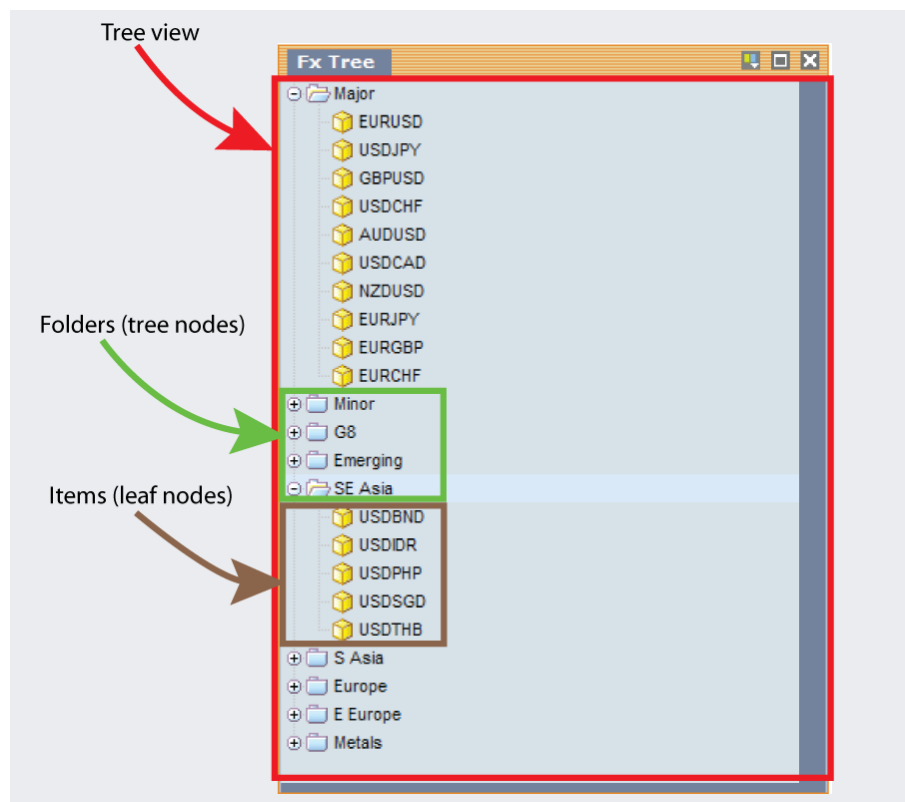
The example below shows the content of a Personal Grid after dragging the **SE Asia** folder from the tree view shown above. An end user would now be able to view current prices for these items and trade the items in the grid.



Currency	Rate
USDBND	<u>1.3804/1.3814</u>
USDIDR	<u>9168.0/9173.9</u>
USDPHP	<u>41.520/41.646</u>
USDSGD	<u>1.3807/1.3809</u>
USDTHB	<u>31.530/31.613</u>

Content of a Personal Grid after dragging the SE Asia folder from the tree view

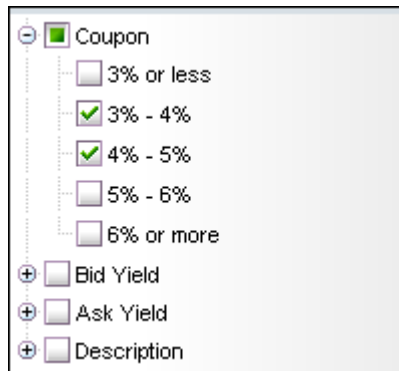
Trees are "Ext JS" components, but you can configure the appearance and behaviour of trees using the XML configuration defined in this document without writing any JavaScript code. The XML uses some terminology related to how trees are organized, as shown in the following diagram.



Tree organization and terminology

A **tree view** is a logical arrangement of items and folders. Items are the leaf nodes of the tree and do not have any children. Folders are the nodes of the tree and can have other folders or items as children. In the example shown above, each folder has FX currency pairs as children.

The nodes in a tree can be represented by standard "Ext JS" icons (as shown above), by selectable check boxes (as shown below), or by icons and check boxes.



**A check box at each node
instead of a node icon**

See also:

- [Getting started](#) ⁶
- [XML Reference information](#) ¹¹

3 Getting started

The following sections explain how the various XML tags may be combined to define tree views 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

An example configuration file

An example XML file describing a simple tree view configuration is shown below.

The XML Reference information section defines the XML tags and attributes you can use to define tree views in Caplin Trader Client pages. Also see the section [Ordering and nesting of tags](#).

XML for a simple tree view configuration

```
<treeViewDefinitions>
  <templates>
    <treeViewTemplate id="FXTree"
      animate="false"
      loadFolderContent="onInitialize"
      enableDrag="true"
      rootVisible="false"
      ddGroup="fxInstruments"
      nodeTextFields="InstrumentDescription">
      <folder id="HardCoded" value="HardCoded" caption="HardCoded">
        <item value="/FX/GBPUSD" caption="GBPUSD"/>
        <item value="/FX/EURGBP" caption="EURGBP"/>
        <item value="/FX/AUDCAD" caption="AUDCAD"/>
        <item value="/FX/JPYUSD" caption="JPYUSD"/>
        <item value="/FX/GBPCAD" caption="GBPCAD"/>
      </folder>
      <folder id="Major" caption="Major" container="true" value="/CONTAINER/FX/Major">
      </folder>
    </treeViewTemplate>
  </templates>
</treeViewDefinitions>
```

This configuration defines a tree that displays FX information in two folders, one with the caption "HardCoded" and the other with the caption "Major".



Simple FX tree view

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:

- ◆ **<treeViewDefinitions>** contains a single **<templates>** tag and starts the tree view definitions:

```
<treeViewDefinitions>
  <templates>
    ...
  </templates>
</treeViewDefinitions>
```

There can only be one **<templates>** tag inside a **<treeViewDefinitions>** tag.

- ◆ **<templates>** contains a tree view template (**<treeViewTemplate>**):

```
<templates>
  <treeViewTemplate id="FXTree"

  ...

</treeViewTemplate>
</templates>
```

In this case there is only one tree view template defined (`id="FXTree"`), but you can define multiple tree view templates inside a `<templates>` tag.

◆ `<treeViewTemplate>` contains the template definition for the "FX" tree view:

```
<treeViewTemplate id="FXTree"
    animate="false"
    loadFolderContent="onInitialize"
    enableDrag="true"
    rootVisible="false"
    ddGroup="fxInstruments"
    nodeTextFields="InstrumentDescription">
  <folder id="HardCoded" value="HardCoded" caption="HardCoded">
    ...
  </folder>
  <folder id="Major" caption="Major" container="true" value="/CONTAINER/FX/Major">
  </folder>
</treeViewTemplate>
```

The "FX" tree view has two folders, one with the caption "HardCoded" (`caption="HardCoded"`) and the other with the caption "Major" (`caption="Major"`). Captions are positioned on the right-hand-side of folders in the tree view.

The `<treeViewTemplate>` tag also has a number of attributes.

`id="FXTree"`: Identifies the template so that a tree that is placed in a layout can inherit from this template. Trees that can be placed in a layout are defined using the `<tree>` tag, which is not defined in this schema.

`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.

`nodeTextFields="InstrumentDescription"`: The name of the RTTP field that holds the caption for the items in the folder when the content of the folder is defined to be an RTTP container (see `<folder>` tag).

The remaining attributes (`animate`, `enableDrag`, `rootVisible`, and `ddGroup`) 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. 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 view:

```
<folder id="HardCoded" value="HardCoded" caption="HardCoded">
  <item value="/FX/GBPUSD" caption="GBPUSD" />
  <item value="/FX/EURGBP" caption="EURGBP" />
  <item value="/FX/AUDCAD" caption="AUDCAD" />
  <item value="/FX/JPYUSD" caption="JPYUSD" />
  <item value="/FX/GBPCAD" caption="GBPCAD" />
</folder>
<folder id="Major" caption="Major" container="true" value="/CONTAINER/FX/Major">
```

In this case two folders are defined, one that contains individually specified items (`id="HardCoded"`) and the other items from an RTTP container (`id="Major"`). Each item in the FX tree represents an FX currency pair.

When the `container` attribute of a folder is set to `true`, then the folder will be populated with items from the RTTP container identified by the `value` attribute of the folder. In this case the "Major" folder will be populated by the RTTP container `/CONTAINER/FX/Major`.

The "HardCoded" folder (`id="HardCoded"`) contains individual items that are defined by `<item>` tags rather than items from an RTTP container. Each `<item>` tag has two attributes:

`caption`: The caption for the item when the item is displayed in the tree or when the item is dragged out of the tree.

`value`: The value that will be passed to a component (such as a Personal Grid or Trade Panel) when the item is dropped into the component.

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 (*treeViewDefinitions.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 section.

<treeViewDefinitions>

This is the outermost tag

```
<treeViewDefinitions>
  <treeNodeMappings><treeNodeMappings/> (one only)
  <templates></templates> (one only)
</treeViewDefinitions>
```

<templates>

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

<treeViewTemplate>

```
<treeViewTemplate>
  <folder></folder> (zero or more)
  <item /> (zero or more)
</treeViewTemplate>
```

<folder>

```
<folder>
  <folder></folder> (zero or more)
  <item /> (zero or more)
</folder>
```

<treeNodeMappings>

```
<treeNodeMappings>  
  <treeNodeMapping /> (one or more)  
</treeNodeMappings>
```

4 XML Reference information

This is the reference information for the tree view configuration XML.

Default attribute values

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

4.1 <folder>

<folder>

Defines a node in the tree. A folder can either be populated from an RTTP container, from a custom JavaScript class (see <treeNodeMapping>), or with items (see <item>) and/or other folders.

Attributes:

Name	Type	Default	Req?	Description
caption	string	(none)	Y	The text that is placed next to the folder when the folder is rendered in a tree view.
id	string	(none)	Y	A unique identifier for this folder.
type	string	(none)	N	An optional attribute that identifies how the folder will be populated. If 'type' is the 'id' of a <treeNodeMapping>, then the folder is populated from a custom JavaScript class. If 'type' is set to "container", then the folder is populated by the RTTP container identified by the 'value' attribute of the folder. If the 'type' attribute is omitted, then the folder is populated by items (see <item>).
value	string	(none)	Y	If the folder 'type' is "container", then 'value' must identify the RTTP container that populates the folder. For example, setting 'value' to "/CONTAINER/FX/MAJORS" and 'type' to "container" populates the folder with the contents of the RTTP container "/CONTAINER/FX/MAJORS". If the folder 'type' is not "container", then 'value' can be set to any string of characters, including the empty string (value="").

4.2 <item>

<item>

Defines a leaf node of the tree. Items are contained inside folders and cannot have any children (see <folder>).

Attributes:

Name	Type	Default	Req?	Description
caption	string	(none)	Y	The text that is placed inside the item when the item is rendered in a tree view.
value	string	(none)	Y	A string that is passed to the drop decorator of a GUI component, such as Personal Grid or Trade Panel, when the item is dragged from the tree view and dropped on the component. If the item is an FX currency pair in Liberator, then the string must identify the subject of the currency pair, for example "/FX/GBPUSD".

4.3 <templates>

<templates>

Contains a list of tree view templates (see <treeViewTemplate>).

Attributes: This tag has no attributes.

4.4 <treeNodeMapping>

<treeNodeMapping>

Defines the JavaScript class that populates a node. This allows you to specify a JavaScript class that will populate a node with data from an external source, such as a web service or database.

Attributes:

Name	Type	Default	Req?	Description
className	string	(none)	Y	The fully qualified name of the JavaScript class that populates the node. This class must implement the "caplin.component.tree.TreeNode" interface of the Caplin Trader Client API.
id	string	(none)	Y	An identifier for this node. The identifier must be unique across all tree node mappings (<treeNodeMappings>). The node is placed in a tree using the 'type' attribute of the <folder> tag.

4.5 <treeNodeMappings>

<treeNodeMappings>

A list of one or more tree node mappings (see <treeNodeMapping>). A tree node mapping allows a folder to be populated from an external source, such as a web service or database.

Attributes: This tag has no attributes.

4.6 <treeViewDefinitions>

<treeViewDefinitions>

The outermost tag of the tree view definition XML.

Attributes: This tag has no attributes.

4.7 <treeViewTemplate>

<treeViewTemplate>

A tree view template is an abstract definition for a tree and can contain folders (see <folder>) and/or items (see <item>). Trees that are placed in a layout can inherit from tree view templates. Trees are defined using the <tree> tag, which is not defined in this schema.

Attributes:

Name	Type	Default	Req?	Description
anyAttribute		(none)	N	You can add attributes that are not part of this schema to the <treeViewTemplate> tag. These attributes are name/value configuration options for the "Ext.tree.TreePanel" class that affect the look and behavior of the tree view when it is rendered. Refer to the "Ext JS" API documentation for a list of valid configuration options.
baseTemplate	string	(none)	N	The identifier of an optional <treeViewTemplate> that this template inherits from.
checked	boolean	(none)	N	Adds a check box to each node in the tree. Valid values are "true" and "false". When set to "true", each check box is initially checked. When set "false", each check box is initially unchecked. If the 'type' attribute is omitted, then nodes are displayed without check boxes.
id	string	(none)	Y	An identifier for the tree view template. The identifier must be unique across all trees (<tree>) and tree view Templates (see <treeViewTemplate>). The <tree> tag is used to place a tree in a layout, but is not defined in this schema.

Name	Type	Default	Req?	Description
loadFolderContent	string	'onExpand'	N	An optional attribute that determines when the folders in a tree are populated. Valid values are "onInitialize" and "onExpand". When set to "onInitialize", each folder is populated when the tree is first displayed. With this option time and memory will be required to initially display the tree, but folders will expand rapidly. When set to "onExpand", a folder is not populated until the end user expands the folder. With this option the initial tree will display rapidly, but time and memory will be required to initially expand each folder.
nodeIcon	boolean	true	N	Determines whether the standard Ext JS node icon is added to each node in the tree. When set to "true" the icon is displayed, when set to "false" the icon is not displayed.
nodeTextFields	string	(none)	N	A comma-separated list of RTTP field names that contain captions. If a folder is populated from an RTTP container, then the captions will be concatenated to form a single caption that is displayed inside each item in the folder. For example, if nodeTextFields="InstrumentDescription, CpnRate", then the caption for each item will be an instrument description and coupon rate.
toggleCheckOnDbClick	boolean	true	N	An optional attribute that determines what happens when nodes are configured as checkboxes (see 'checked' attribute) and a node is double clicked. Valid values are "true" and "false". When set to "true", a double click toggles the checked state of the checkbox (for example from checked to unchecked). When set to "false", 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.

5 Glossary of terms and acronyms

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

Term	Definition
Ext JS	A JavaScript library that is used by Caplin Trader Client to render trees in a layout.
Folder	A folder is a node in a tree and can either be populated from an RTTP container or from individually specified items and folders.
Item	A leaf node of a tree . An example of a leaf node is an FX currency pair. Items are contained inside folders .
Leaf node	A leaf node is a node in a tree that cannot have any children. An example of a leaf node is an item .
Node	A node is an element in a tree and can contain other nodes and leaf nodes as children. An example of a node is a folder .
RTTP container	An RTTP container holds a set of references to other data items in Liberator. An example would be a set of references to information about financial instruments, such as FX currency pairs.
Tree	Information that is organized and displayed in a tree view . A tree can be inserted in a layout.
Tree view	A way of displaying an organized set of items in Caplin Trader Client. An example is a tree that displays FX currency pairs organized in folders .

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

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.