CAPLIN

Caplin Trader 1.5

How To Add Permissioning At The Client

November 2009

Contents

1   Preface

1.1   What this document contains

This document describes how to add permissioning to Caplin Trader so that display components behave in accordance with the permissions of the logged in user.

Before reading this document, make sure you are familiar with the document **Caplin Xaqua: Permissioning Overview and Concepts**.

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 Architects and Software Developers who want to add or modify permissioning in Caplin Trader.

1.3   Related documents

◆   **Caplin Trader: Permissioning Configuration XML Reference**

Describes the XML tags and attributes that you can use to configure permissioning in Caplin Trader.

◆ **Caplin Trader: API Reference**

The API reference documentation provided with Caplin Trader. The classes and interfaces of the `caplin.security.permissioning` package allow you to write JavaScript classes that extend the live permissioning capabilities of Caplin Trader.

◆ **Caplin Xaqua: Permissioning Overview And Concepts**

Introduces permissioning concepts and terms, and shows the permissioning components of the Caplin Xaqua architecture.

◆ **Caplin Xaqua: Installing Permissioning Components**

Describes how to install the Permissioning Auth Module and Permissioning DataSource in an existing Caplin Xaqua installation. You only need to install these components if your installation of Caplin Trader is earlier than release 1.2.8, as later releases include these permissioning components.

◆ **Caplin Xaqua: How To Create A Permissioning DataSource**

Describes how to create a Permissioning DataSource adapter using the Permissioning DataSource API. A Permissioning DataSource adapter is required to integrate Caplin Xaqua with a Permissioning System. The document also discusses the Demo Permissioning DataSource provided with the reference implementation of Caplin Trader from release 1.2.8.

◆ **Caplin Xaqua: Overview**

Provides a business and technical overview of Caplin Xaqua and includes an explanation of its architecture.

◆ **Caplin Liberator: Administration Guide**

Describes how to install and configure Caplin Liberator and discusses the authentication modules that are provided with the server.

◆ **Permissioning DataSource: API Reference**

The API reference documentation provided with the Permissioning DataSource SDK (Software Development Kit). The classes and interfaces presented by this API allow you to write a Java application that will integrate a Permissioning System with Caplin Xaqua.

1.4 Typographical conventions

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

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

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

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

1.5 Feedback

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

Please email your feedback to documentation@caplin.com.

1.6 Acknowledgments

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

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

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

2   Permissioning and Caplin Trader

The display components of Caplin Trader can be tailored to match the permissions of the currently logged in user. An example would be to display information that the user is allowed to view, and to hide information that the user is not allowed to view. This information could be anything from pricing data for currency pairs, to menu items, data grids, trade tiles, and tenors. Another example would be to enable or disable the one-click trading button to reflect the permissions of the logged in user.

To add permissioning at the client, you must modify the client application by adding JavaScript code that uses the [Permissioning API](#) [10] for Caplin Trader. The Permissioning API is part of the Caplin Trader JavaScript API, and is supplied with Caplin Trader from release 1.2.8.

For a complete description of the Permissioning API, please refer to the **caplin.security.permissioning** package of the **Caplin Trader: API Reference** documentation.

Security

User authentication and permissioning is enforced by the Permissioning Auth Module at the Liberator server. It is not possible to modify these product permissions from Caplin Trader. The classes of the Permissioning API only allow you to query the permissions that have been assigned to the currently logged in user; they do not allow you to change these permissions.

Latency of permissioning queries

When a user logs in to Caplin Trader, the application automatically subscribes to the permissioning data that Liberator holds for that user. At the start of the subscription, Liberator sends Caplin Trader all the permissioning data for that user, followed by subscription updates whenever a product permission changes.

The classes of the Permissioning API provide an interface to this permissioning data, allowing you to tailor your copy of Caplin Trader to match the permissions of the currently logged in user.

Since Caplin Trader automatically subscribes to permissioning data, the latency of each subsequent permissioning query is reduced. This is because Caplin Trader holds a local copy of the latest product permissions, and does not need to contact Liberator each time your code initiates a permissioning query.

2.1   About Product Permissions

Product permissions are sent to Liberator by the Permissioning DataSource (see **Caplin Xaqua: Permissioning Overview And Concepts** for further information). A user can have any number of product permissions assigned to them, where each permission determines whether an action on a product will be allowed or denied.

Permissions for actions can reside in either the global (default) namespace or a specific namespace. Permissions are typically defined in a specific namespace to group related actions. An example would be to group tenor permissions by defining these permissions in a "tenor" namespace. Your client application could then query the tenor namespace to determine the tenors that are allowed and those that are denied.

The table below shows four such permissions, where each row of the table defines a single product permission. Note that the final column of the table (Significance) is not part of the permission, and is only included here to provide a description of the permission.

Namespace	Product	Action	Authorization	*Significance*
null	.*	VIEW	ALLOW	*Allow the user to view all products.*
null	/FX/.*	TRADE	ALLOW	*Allow the user to trade all FX products.*
tenor	/FX/.*	1_WEEK	ALLOW	*Allow the user 1-week tenor for FX products.*
tenor	/FX/.*	2_WEEK	DENY	*Deny the user 2-week tenor for FX products.*

You will see from the table above that JavaScript regular expressions have been used to define the product. In this way the same permission can be applied to a range of products. When a user attempts to view or trade a product, Liberator consults pre-defined permissioning rules that indicate the permissions that the user must have in order to complete the task.

The permissions in the first two rows of the table above allow the user to view all products, and to trade any FX product. In this case the namespace is null, which indicates that the permission resides in the global (default) namespace. This type of permission would typically be used at the client to enable or disable the display of products or product grids.

The permissions in the last two rows of the table are defined in the "tenor" namespace. These permissions allow the user to trade any FX product with a 1-week tenor, but do not allow the user to trade any FX product with a 2-week tenor. These permissions would typically be used at the client to:

   * Enable the 1-week tenor option.

   * Disable (or hide) the 2-week tenor option.

3 Subscribing to Multiple Permissioning DataSources

Caplin Trader can subscribe to permissioning data from more than one Permissioning DataSource (see **Roles** in the document **Caplin Xaqua: Permissioning Overview and Concepts** for further information).
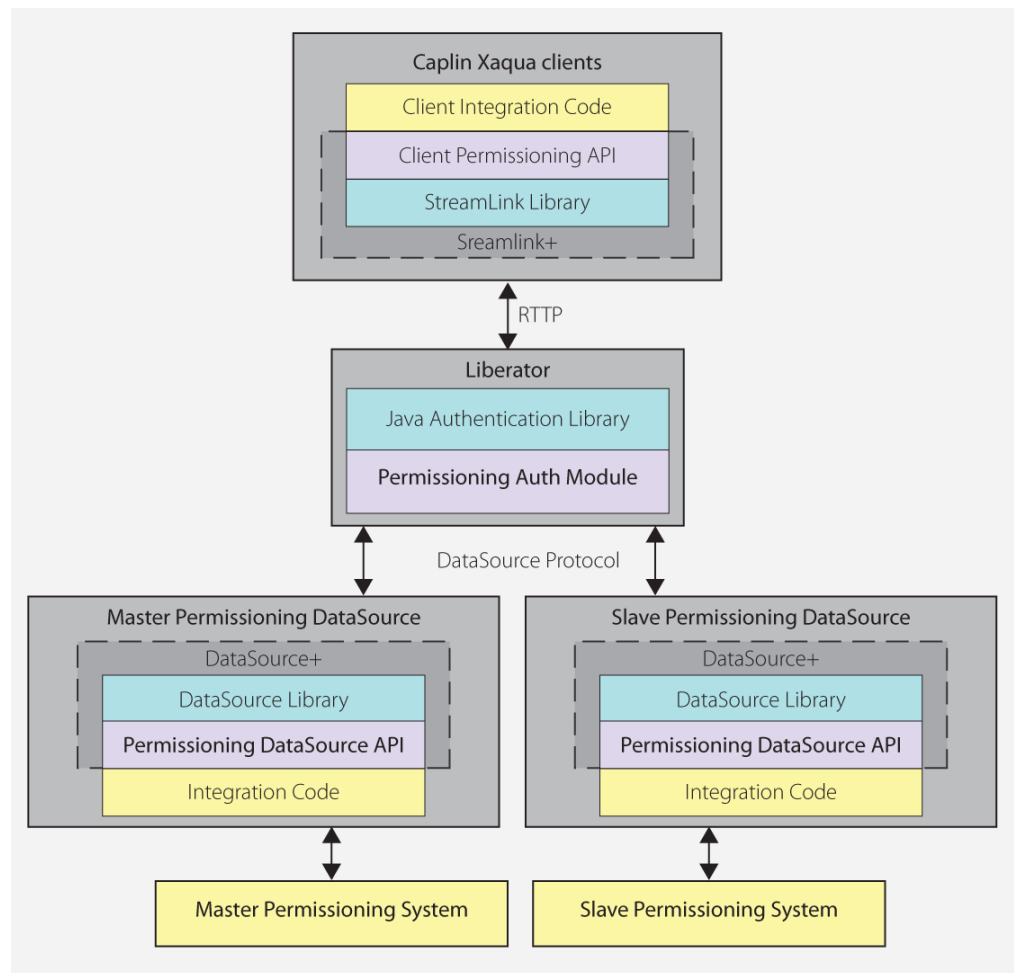
**Multiple Permissioning DataSource Adapters connected to Liberator
(showing one master and one slave)**

To subscribe to permissioning data from multiple Permissioning DataSources you must:

■ Create an XML configuration file that specifies the master and slave Permissioning DataSources that supply the required permissioning data.

■ Add a line to the file *CaplinTrader/application.jsp* that specifies the URL of the XML configuration file that you created (where *CaplinTrader* is the installation directory containing the Caplin Trader client application).

Creating the XML Configuration File

The code sample below is taken from an XML configuration file and specifies one master and two slave Permissioning DataSources.

```xml
<?xml version="1.0"?>
<permissionDatasourceDefinitions xmlns="http://schema.caplin.com/Security/permissions">
            <permissionsource name="MASTER" role="Master"/>
            <permissionsource name="SLAVE1" />
            <permissionsource name="SLAVE2" />
</permissionDatasourceDefinitions>
```

The name of the master must be "MASTER" and the role must be set to "Master". The name of a slave must match the name of the Permissioning DataSource that provides the permissioning data (the role defaults to "Slave" if not specified).

In this case Caplin Trader will subscribe to permissioning data from the following containers:

> `/PERMISSIONS/MASTER/CONTAINER/<username>`

> `/PERMISSIONS/SLAVE1/CONTAINER/<username>`

> `/PERMISSIONS/SLAVE2/CONTAINER/<username>`

where `<username>` is the logged in user.

The document **Caplin Trader: Permissioning Configuration XML Reference** describes the XML tags and attributes that you can use when you create a permissioning XML configuration file.

> **Tip:**   A permissioning container is mapped to a Permissioning DataSource in the Liberator configuration.
> See **Configuring Liberator to connect to multiple Permissioning DataSources** in the document **Caplin Xaqua: Installing Permissioning Components** for further information.

Legacy releases of Caplin Trader

If an XML configuration file is not created (or is invalid), then Caplin Trader will subscribe to permissioning data from the container:

> `/PERMISSIONS/CONTAINER/<username>`

This allows legacy client applications that do not subscribe to multiple Permissioning DataSources to continue working without modification if the Caplin Trader library has been upgraded, but the Permissioning DataSource library has not been upgraded.

See <u>Upgrading a Caplin Trader client application</u> [9] for further information.

> **Note:**   If the XML file is invalid, permissioning error messages will be logged to the console log. End users will be able to log in to Caplin Trader but will not be able to view or trade instruments.

Modifying application.jsp

In the code sample below, a line has been added to the file *application.jsp* that specifies the URL of the XML configuration file *conf/permissioningDataSourceDefinitions.xml*. The URL is set by the second argument of the `caplin.core.ApplicationProperties.setProperty()` method.

Code sample from application.jsp

```
...
<!-- ========================================================================= -->
 <!--                        Caplin Libraries                                  -->
 <!-- ========================================================================= -->
 <script type="text/javascript" src="dependencies/core/core.js"></script>
 <script type="text/javascript">
 ...
   caplin.core.ApplicationProperties.setProperty(
                           "CAPLIN.PERMISSION.CONFIG.URL",
                           "conf/permissionDatasourceDefinitions.xml");
   ...

 </script>
...
```

3.1 Upgrading a Caplin Trader client application

From release 1.5.0, the Caplin Trader library supports two versions of a Permissioning message protocol, each having a different (and mutually incompatible) message format.

Version 1 (the original protocol) has a message format that allows Caplin Trader to subscribe to only one Permissioning DataSource. Version 2 (a later protocol) has a message format that allows Caplin Trader to subscribe to both single (master) and multiple (master/slave) Permissioning DataSources.

If you are upgrading the Caplin Trader library and your Permissioning DataSource uses version 1 of the Permissioning message protocol, then you must ensure that your Caplin Trader client application continues to use version 1 of this protocol.

Caplin Trader will use version 1 of the protocol if you *do not* specify a valid permissioning XML configuration file in *application.jsp* (see Subscribing to Multiple Permissioning DataSources 6). This means that if the Permissioning DataSource only supports version 1 of the protocol, then you *do not* need to modify any code in either the client application or Permissioning DataSource when you upgrade the Caplin Trader library.

Caplin Trader will use version 2 of the protocol if you *do* specify a valid permissioning XML configuration file in *application.jsp*. You must create and specify a valid permissioning XML configuration file if the Permissioning DataSource uses version 2 of the Permissioning message protocol.

The following table shows the messaging protocols that are supported by each release of the Caplin Trader and Permissioning DataSource libraries.

Supported Permissioning message protocols:

Component	Release	Supported Permissioning Message Protocol	How to configure
Permissioning DataSource library (DataSource+)	4.5.3 and earlier	version 1 only	Not applicable
Permissioning DataSource library (DataSource+)	4.5.4 and 4.5.5	version 2 only	Not applicable
Permissioning DataSource library (DataSource+)	4.5.6 and later	versions 1 and 2	See the document **Caplin Xaqua: How To Create A Permissioning DataSource Adapter**.
Caplin Trader library (StreamLink+)	1.4.8 and earlier	version 1 only	Not applicable
Caplin Trader library (StreamLink+)	1.5.0 and later	version 1 and 2	Caplin Trader will use protocol version 1 if you *do not* specify a valid permissioning XML configuration file in *application.jsp*. Caplin Trader will use protocol version 2 if you *do* specify a valid permissioning XML configuration file in *application.jsp*.

4   Using the Permissioning API for Caplin Trader

The Permissioning API for Caplin Trader provides two classes.

`caplin.security.permissioning.PermissionServiceListener`: An interface. The classes of your GUI application must implement this interface if they want to be notified when user permissions change.

`caplin.security.permissioning.PermissionService`: A singleton class. The methods in this class return permissioning information about the currently logged in user. The methods fall into three categories.

◆ The first category (see Methods that return permissioning information when called ⌐11⌐) returns permissioning information when the method is called. This category can be used by the classes of your GUI application that respond to user interactions.

◆ The second category (see Methods that register listeners ⌐13⌐) allows you to register listeners that subscribe to current and future product permissions. This category can be used to register the classes of your GUI application that need to be notified when user permissions change. Classes that you register as listeners must implement the `PermissionServiceListener` interface.

◆ The third category (see User attributes ⌐14⌐) allows you to retrieve user attributes.

4.1   Methods that return permissioning information when called

Methods of the `PermissionService` class that return permissioning information when called are shown in the table below.

Query Type	Methods that can be used for this type of query	Return Value
Query an action on a single product in the given namespace.	`canUserPerformAction(product, namespace, action)`	*true* if action is allowed. *false* if action is denied.
Query an action on a single product in the *default* namespace.	`canUserPerformGlobalAction(product, action)`	*true* if action is allowed. *false* if action is denied.
Query an action on an array of products in the given namespace.	`getPermissionedProducts(products, namespace, action)`	A subset of products on which the action is *allowed*.
	`getUnpermissionedProducts(products, namespace, action)`	A subset of products on which the action is *denied*.
Query the permitted actions on a single product in the given namespace.	`getAllowPermissions(product, namespace)`	An array of *allowed* actions for this product.
	`getDenyPermissions(product, namespace)`	An array of *denied* actions for this product.
	`getAllPermissions(product, namespace)`	An array of *all* actions for this product (*allowed* and *denied*).

In the following example, we add code that responds to a user interacting with a GUI component. In particular, we add code to the part of the client application that responds to attempts by the user to open a trade ticket.

```
/* check that user is permitted to open this trade ticket */
if(caplin.security.permissioning.PermissionService.
                            canUserPerformGlobalAction("/FX/EURUSD", "TRADE"))
{
  // open the trade ticket
}
```

In above example, the trade ticket will only open if the user is permitted "TRADE" action for the product "/FX/EURUSD" in the "*default*" namespace.

Typical Opened Trade Ticket

4.2 Methods that register listeners

Methods of the `PermissionService` class that are used to register listeners are shown in the table below. The listeners that you register will typically be classes in your GUI application that need to receive notification of user permissions for a particular type of subscription. Listeners get notified of user permissions when the classes are first registered, and also when permissions change for the subscribed elements.

To receive permissioning notifications, the registered listener must implement one or more of the callback methods of the `PermissionServiceListener` interface, as indicated in the table.

Subscription Type	PermissionService method to register the subscription listener	Callback method the listener must implement	Notification
Subscribe to permissioning changes for an action on a single product in the given namespace.	```addPermission Listener(product, namespace, action, listener)```	```onSinglePermissionChanged(bIsAuthorized)```	`bIsAuthorized`: *true* if action is allowed. *false* if action is denied.
Subscribe to permissioning changes for an action on a single product in the *default* namespace.	```addGlobal PermissionListener(product, action, listener)```	```onSinglePermissionChanged(bIsAuthorized)```	`bIsAuthorized`: *true* if action is allowed. *false* if action is denied.
Subscribe to permissioning changes for an action on an array of products in the given namespace.	```addProduct PermissionsListener(products, namespace, action, isAuthorized listener)```	```onPermissionsChanged(permissions)```	`permissions`: An array (subset) of products for which the action is *allowed*, *denied*, or *allowed and denied*, depending on the value of `isAuthorized` (ALLOW, DENY, or ALL).
Subscribe to permissioning changes for permitted actions on a single product in the given namespace.	```addPermission SetListener(product, namespace, authtype, listener)```	```onPermissionsChanged(permissions)```	`permissions`: An array of *allowed*, *denied*, or *all* actions on the product, depending on the value of `authtype` (ALLOW, DENY, or ALL).

In the following example, we add code that responds to permissioning changes for an action on a single product. In particular, we add code to the part of the client application that is responsible for displaying a trade tile.

Unlike the previous example (see [Methods that return permissioning information when called]) 11 , where the trade ticket does not open until the user selects the currency pair that they want to trade, a trade tile is typically on display whether or not the end user uses it to trade. The tile must therefore always reflect the latest permissions that apply to it. It is for this reason that a listener must be registered to enable or disable the trade tile, rather than calling a method in response to a user action.

Register the listener

```
/* register a listener for the "TRADE" action * /
tile()
{
  addPermissionListener(product, namespace, "TRADE", this)
}
```

Implement the callback method

```
/* implement the callback method for this listener */
onSinglePermissionChanged(bIsAuthorized)
{
  if(bIsAuthorized)
  {
    /* enable the trade tile */
  }
  else
  {
    /* disable the trade tile */
  }
}
```

In above example, the trade tile is either enabled or disabled, depending on the permission that the user has for "TRADE" action on the product. If the action is allowed, then the trade tile is enabled. If the action is denied, then the trade tile is disabled.
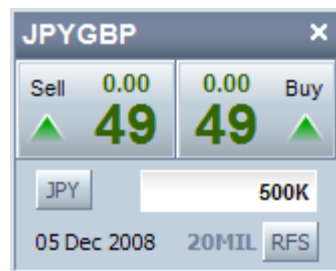
Enabled Trade Tile   **Disabled Trade Tile**

4.3   User attributes

User attributes are name/value pairs that are not processed by the Permissioning Auth Module and therefore do not affect permissioning directly. A typical use is to send information to Caplin Trader about the maximum tradable amount that a user is permitted to trade. Your application could then use this information to limit the tradeable amounts shown to the user.

You will find further information about user attributes in **Caplin Xaqua: Permissioning Overview and Concepts**, and in the **caplin.security.permissioning** package of the **Caplin Trader: API Reference**.

5 Further Reading

If you would like an introduction to permissioning concepts and terms, or to consult reference documentation for the Permissioning API, then the following documents provide this information. You may also be interested in reading some of the other Related documents [1].

An introduction to permissioning concepts and terms

The document **Caplin Xaqua: Permissioning Overview And Concepts** introduces permissioning concepts and terms, and shows the permissioning components of the Caplin Xaqua architecture.

Reference documentation for the Permissioning API

Reference documentation for the Permissioning API can be found in the `caplin.security.permissioning` package of the **Caplin Trader: API Reference**.

6 Glossary of terms and acronyms

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

Term	Definition
Action	The interaction that a user can have with a **product**.
API	Application Programming Interface
Caplin Trader	A **Caplin Xaqua client** application written in Ajax that provides a framework and comprehensive set of components for constructing browser-based trading applications.

Caplin Trader was formerly called "Caplin Trader Client". |
| **Caplin Xaqua** | A single-dealer platform that enables banks to deliver multi-product trading direct to client desktops.

Caplin Xaqua was formerly called "the Caplin Platform". |
Caplin Xaqua client	A client desktop application that interfaces with **Caplin Xaqua** to deliver multi-product trading to end users. The application can be implemented in any technology that is supported by Caplin Xaqua; for example Ajax, Microsoft .NET, Microsoft Silverlight™, Adobe Flex™, and Java™.
DataSource	DataSources are software adapters within **Caplin Xaqua** that connect Xaqua to external sources of real time data and external **Permissioning Systems**. In other Caplin documents DataSources are also called DataSource adapters.
Demo Permissioning DataSource	The Demo Permissioning DataSource is an example of a **Permissioning DataSource** application that gets its permissioning data from an XML file.
Liberator	Caplin Liberator is a bidirectional streaming push server designed to deliver market data and trade messages over any network that supports Web traffic.
Permission	Determines whether an **action** on a **product** will be allowed or denied.
Permissioning Auth Module	One of several authentication modules that are supplied with **Caplin Xaqua**.
Permissioning DataSource	A **DataSource** adapter that acts as the interface between **Caplin Xaqua** and your **Permissioning System**.
Permissioning System	The source of the permissioning data that you want to integrate with **Caplin Xaqua**.
Product	In permissioning documentation (including this document) a "product" is any entity on which a **User** may be assigned **permissions** (including financial instruments). In other **Caplin Trader** and **Caplin Xaqua** documentation, a "product" is a term that refers only to a financial instrument.
Rule	Rules link **permissions** to user interactions, and are used by **Liberator** to decide which of the many permissions that have been defined will apply when a **user** attempts to interact with a **product**.
SDK	Software Development Kit

Term	Definition
User	An end user of a **Caplin Xaqua client** application such as **Caplin Trader**.

Index

Single-dealer platforms for the capital markets

CAPLIN

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