CAPLIN

Caplin Xaqua 1.0

Permissioning Overview And Concepts

August 2011

Contents

1 Preface

1.1 What this document contains

This document introduces permissioning concepts and terms, and shows the permissioning components of the Caplin Xaqua architecture.

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 Technical Managers, Enterprise Architects, System Architects, System Administrators, and Software Developers who want to integrate Caplin Xaqua with a Permissioning System.

1.3 Related documents

- **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.

- **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 Adapter**

  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 Trader: How To Add Permissioning At The Client**

  Describes how to add permissioning to Caplin Trader.

- **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.

- **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.

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
Glossary term	Items that appear in the "Glossary of terms and acronyms"
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.

Information about the applicability of a section is enclosed in a box like this.
For example: "This section only applies to version 1.3 of the product."

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.

Visit our feedback web page at https://support.caplin.com/documentfeedback/.

1.6 Acknowledgments

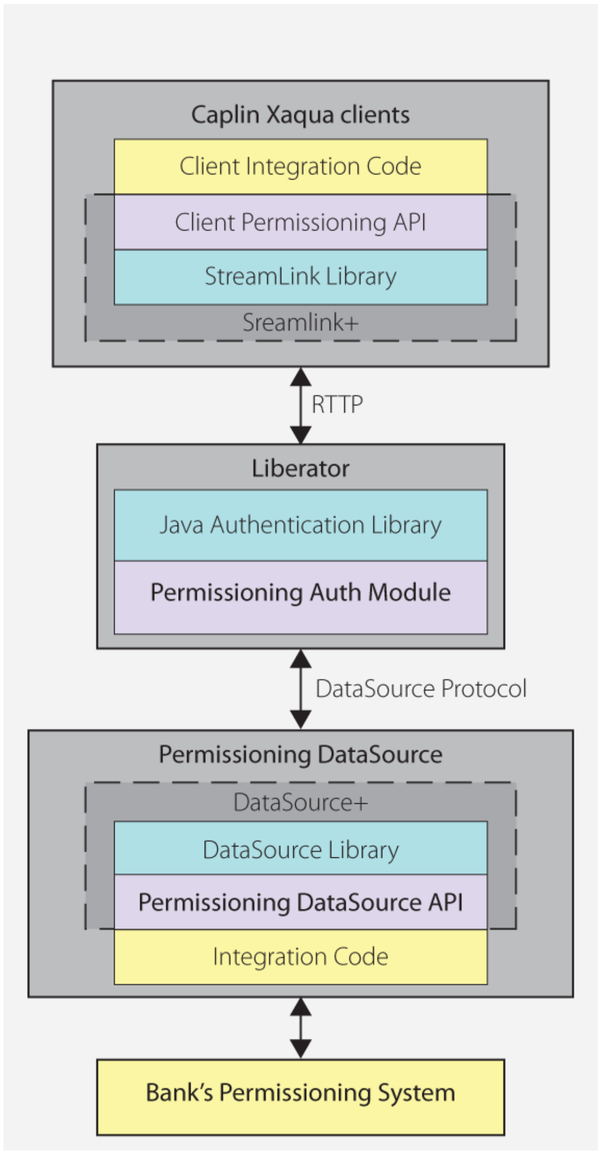
Adobe® Reader is a registered trademark of Adobe Systems Incorporated in the United States and/or other countries.

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

Java, JavaScript, and *JVM* are trademarks or registered trademarks of Oracle® Corporation in the U.S. and other countries.

2 Permissioning Components

Caplin Xaqua consists of a number of components that interact to provide end-users with a web-based financial trading application (see the document **Caplin Xaqua: Overview**). The components that are used to integrate a **Permissioning System** with Caplin Xaqua are shown in the diagram below.



Simplified Caplin Xaqua architecture showing only permissioning components

When a **Caplin Xaqua client** application such as **Caplin Trader** interacts with **Liberator**, for example when an end-user logs in or attempts to view or trade financial information, Liberator uses the services of a **Permissioning Auth Module** to decide if the interaction with Liberator is permitted or not. The Permissioning Auth Module enforces security constraints based on permissioning data provided to Liberator by the **Permissioning DataSource**.

Permissioning System

The Permissioning System is the source of the permissioning data that you want to integrate with Caplin Xaqua. Permissioning data identifies the end-users that are authorized to interact with Liberator, and the financial **products** that these end-users are permitted to view and trade.

Permissioning DataSource

The Permissioning DataSource is a **DataSource Adapter** that acts as the interface between Caplin Xaqua and your Permissioning System. Its purpose is to provide Liberator with the permissioning data that the Permissioning Auth Module will use to decide whether or not an interaction with Liberator is permitted.

To create a Permissioning DataSource, you write and compile a Java application that uses the Permissioning DataSource **API**. This simple API is built on top of the Caplin **DataSource** for Java API, allowing your application to send permissioning data to Liberator using the DataSource protocol, but without the need for your code to explicitly use the DataSource API.

Liberator can be configured to connect to one or more Permissioning DataSources.

Permissioning Auth Module

When a Permissioning DataSource is used to integrate Caplin Xaqua with a Permissioning System, the Liberator server must be configured to use the Permissioning Auth Module to make decisions about who is permitted to access the server and the type of access permitted. The Permissioning Auth Module is one of several authentication modules that are supplied with Caplin Xaqua, and uses the Java Authentication API to communicate with Liberator.

You will find further information about the authentication modules supplied with Caplin Xaqua in the **Caplin Liberator: Administration Guide**.

Permissioning at the Client

The components of the Graphical User Interface (GUI) at the client can be tailored to match the **permissions** of the **user** who is currently logged in. 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.

To add permissioning to a client application that is based on the Caplin Trader framework, you must modify the client application by adding JavaScript code that uses the Permissioning API of Caplin Trader. This simple API can retrieve permissioning data from Liberator and is built on top of the StreamLink for Browsers library. StreamLink for Browsers provides communication between Caplin Trader and the Liberator server using the RTTP protocol.

3 Permissioning Concepts

We will now look at some of the permissioning concepts that allow Liberator to enforce security constraints on the end-users of a Caplin Xaqua client application.

3.1 Products

A product is an entity on which end-users may be assigned permissions.

Note: In other Caplin Xaqua documentation a "product" is a term that refers only to a financial instrument. In permissioning documentation (including this document) a "product" is any entity on which an end-user may be assigned permissions (including financial instruments).

Examples of products are:

- "All FI Instruments"
- "FX Instrument GBPUSD"
- "LIBOR based swaps"
- "The Blotter"

You define products when you write the Permissioning DataSource application.

3.2 Actions

An **action** defines the interaction that a **user** can have with a product.

Examples of product actions are:

- "Viewing"
- "RFQ trading"
- "One-Click trading"

You define product actions when you write the Permissioning DataSource application.

3.3 Permissions

A **permission** determines whether an action on a product will be allowed or denied. When you define a permission you can also define a namespace. A namespace allows client applications to query related permissions, such as all permissions in the "tenor" namespace. If you do not define a namespace, then the permission will reside in the *default* namespace.

Examples of product permissions (each row in the table defines a permission):

Action	Product	Namespace	Authorization
Viewing	LIBOR based swaps		Allow
RFQ Trading	All FI Instruments		Allow

Action	Product	Namespace	Authorization
One-Click Trading	FX Instrument GBPUSD	Quick Trades	Deny
One month settlement	All Instruments	Tenor	Allow

Permissions are assigned to Users 8 and Groups 8 when you write the Permissioning DataSource application.

3.4 Users

A user represents an end-user of a Caplin Xaqua client application. A user can only log in to a Caplin Xaqua client application if permissioning data to authenticate the user has been supplied to Liberator. Users can also be assigned permissions for the products streamed by Liberator.

Users are defined when you write the Permissioning DataSource application. Data for constructing each user would typically be read from a configuration file or persistent Permissioning System. The **Demo Permissioning DataSource** that is supplied with the reference implementation of Caplin Trader reads this data from an XML file (see **Caplin Xaqua: How To Create A Permissioning DataSource**).

3.5 Groups

Users can be arranged in **groups**, where every member of a group has the same permissions. A user can be a member of more than one group, and groups can be members of other groups. This allows an inheritance tree of permissions to be created.

In the example below, user X is a member of two groups:

- Group "FI Traders". This group is permitted "RFQ trading" on "All FI Instruments"
- Group "FX Traders". This group is permitted "RFQ trading" on "All FX Instruments"

Since user X is a member of "FI Traders" and "FX Traders", they will be able to trade both FX and FI instruments.

You will find another example of an inheritance tree in Example Permissioning Hierarchy 15.

When permissions in an inheritance tree are in conflict, conventions are used to determine the permission that is assigned to the user. These conventions are described in Permissioning Hierarchy Conventions 12.

Groups are defined when you write the Permissioning DataSource application. Data for constructing each group would typically be read from a configuration file or persistent Permissioning System. The Demo Permissioning DataSource that is supplied with the reference implementation of Caplin Trader reads this data from an XML file (see **Caplin Xaqua: How To Create A Permissioning DataSource**).

3.6 Accounts

Accounts are a special type of grouping. Users inherit permissions from *every* group that they are a member of, but *only* inherit permissions from the particular account that they are using.

In the example below, user Z has two accounts:

- Account A. This account allows "Viewing and RFQ trading" on "All Instruments".

- Account B. This account allows "Viewing" on "All Instruments".

In this case user Z is *only* permitted RFQ trading when using Account A.

Accounts are defined when you write the Permissioning DataSource application.

3.7 Rules

In addition to defining permissions, you must also define permissioning **rules**. 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.

When a Caplin Xaqua client requests data from Liberator, or tries to publish data to Liberator, the Permissioning Auth Module inspects the content of the associated RTTP message and tries to match it with one or more of the defined rules. If a match is found then the rule will identify the permission that must be checked to determine whether this interaction with Liberator is to be allowed or denied.

If more than one rule matches an RTTP message, then every one of the identified permissions will be checked. If any one of these permissions denies access, then access to the product will be denied.

Rules *must* be defined for every interaction that involves a Caplin Xaqua client publishing data to Liberator, such as when a user attempts to trade a product. If a rule has not been defined for such an interaction, then the default permission is to *deny* the interaction.

Rules *cannot* be defined for interactions that involve a Caplin Xaqua client requesting data, such as when a user attempts to view a product. With this type of interaction a default rule is implemented, and the permission that has been defined for the "VIEW" action in the *default* namespace will be checked to see if the interaction is to be allowed or denied.

Simple Example of a Matching Rule

The following example shows a typical RTTP message that would be sent to Liberator when a user tries to spot trade a product.

RTTP Message

Message Subject	Message Fields			
	MsgType	*Trading-Type*	*Amount*	*Instrument*
/FT/TRADE	Execute	SPOT	1000000	/FX/GBPUSD

An example of a rule that would match this type of trade is:

- Check the user permission for the action "spot-trade" when the subject of the RTTP message to Liberator is "/FT/TRADE" and the "Trading-Type" is "SPOT". The product name to be checked is given by the "Instrument" field of the RTTP message.

Matching Rule

Subject Match	Field Match Criteria	Product Reference Field	Action	Namespace
/FT/TRADE	Trading-Type=SPOT	Instrument	spot-trade	

In this case the permission that has been defined for the action "spot-trade" in the *default* (undefined) namespace will be checked to see if the user is to be allowed or denied access to the product.

The permission must also match the product that is identified by the "Instrument" field of the RTTP message, which in this case is "/FX/GBPUSD".

An example of a permission that would apply to this type of trade is shown below.

Permission that would apply

Action	Product	Namespace	Authorization
spot-trade	/FX/GBP.*		Allow

The permission in the *default* (undefined) namespace is checked since a namespace was not defined by the matching rule.

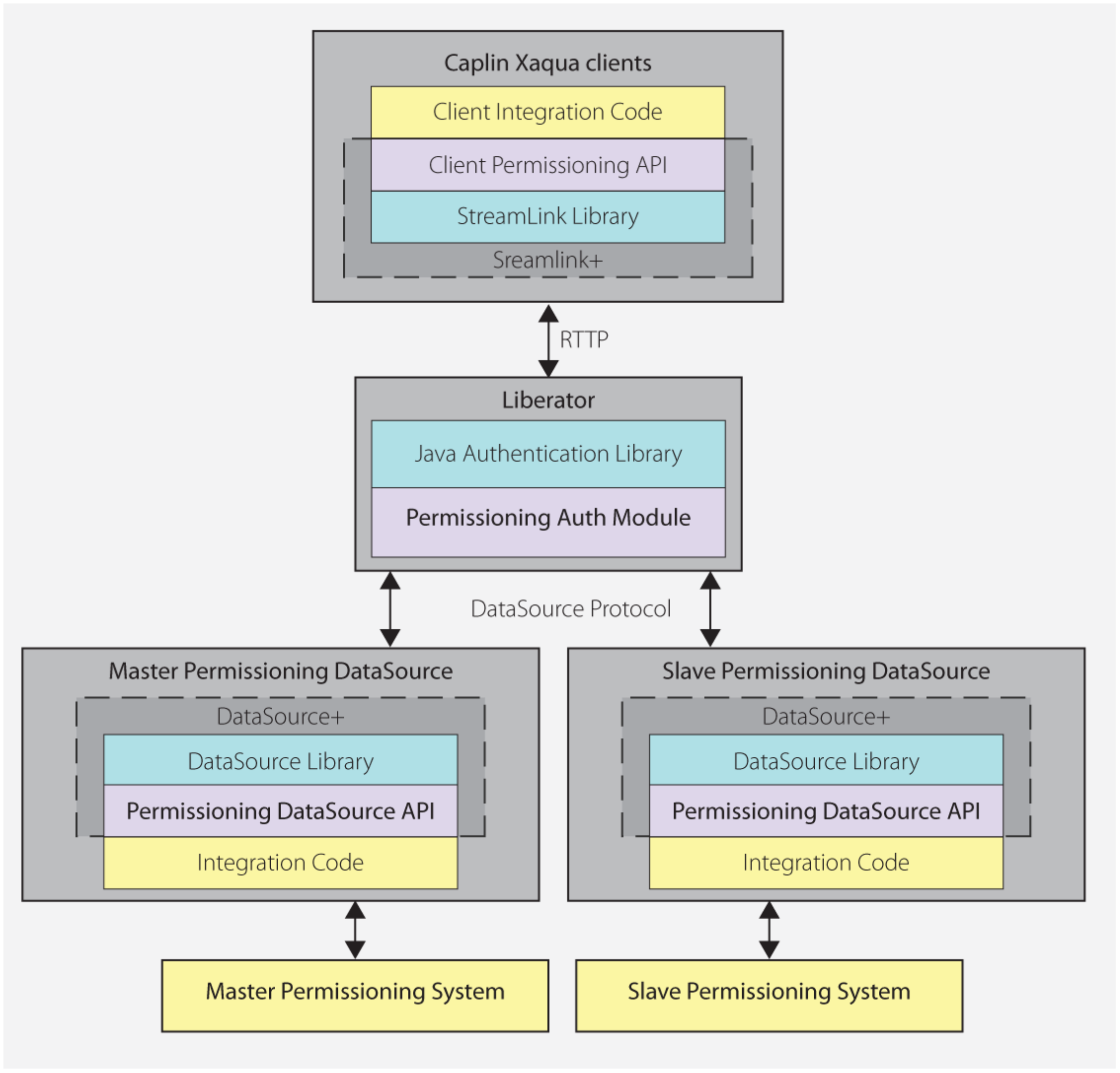
In this case "spot-trade" action on all "/FX/GBP" products will be allowed. The ".*" characters following "GBP" are metacharacters in a Java regular expression, such that "/FX/GBP.*" will match all GBP currency pairs.

Tip: You will find further information about Java regular expressions in the API documentation for the **java.util.regex** package, as supplied by Sun® Microsystems.

Rules are defined when you write the Permissioning DataSource application. Data for constructing each rule would typically be read from a configuration file or persistent Permissioning System. The Demo Permissioning DataSource that is supplied with the reference implementation of Caplin Trader reads this data from an XML file (see **Caplin Xaqua: How To Create A Permissioning DataSource**).

3.8 Roles

Roles determine whether a Permissioning DataSource has been designated as the **master** or **slave**.


Caplin Xaqua clients
Client Integration Code
Client Permissioning API
StreamLink Library
Sreamlink+
RTTP
Liberator
Java Authentication Library
Permissioning Auth Module
DataSource Protocol
Master Permissioning DataSource
DataSource+
DataSource Library
Permissioning DataSource API
Integration Code
Slave Permissioning DataSource
DataSource+
DataSource Library
Permissioning DataSource API
Integration Code
Master Permissioning System
Slave Permissioning System

Multiple Permissioning DataSource Adapters connected to Liberator (showing one master and one slave)

Liberator can be configured to accept permissioning data from multiple Permissioning DataSources. This allows permissions from different permissioning systems to be provided by different Permissioning DataSources. For example, one Permissioning DataSource could provide permissions for FX products and another permissions for FI products, with each Permissioning DataSource being administered by a different department.

When Permissioning data is sent to Liberator from more than one Permissioning DataSource, one Permissioning DataSource is designated the master and each of the other Permissioning DataSources are designated as slaves. A slave Permissioning DataSource can only send a limited a set of permissioning data to Liberator and there can only be one master.

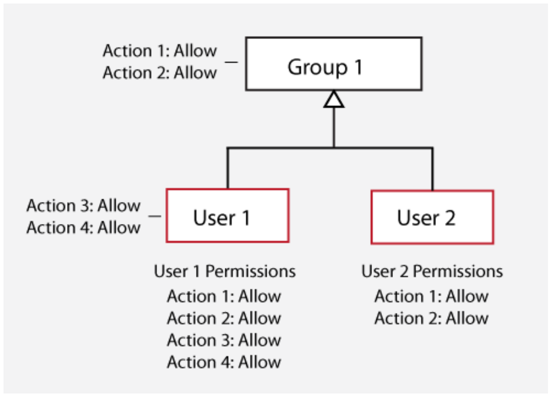
The document **Caplin Xaqua: How To Create A Permissioning DataSource Adapter** describes how to set the master and slave roles, and defines the limited set of permissioning data that a slave can send to Liberator.

3.9 Permissioning Hierarchy Conventions

We will now look at permissioning hierarchies and how permissioning conflicts are resolved.

Single Inheritance

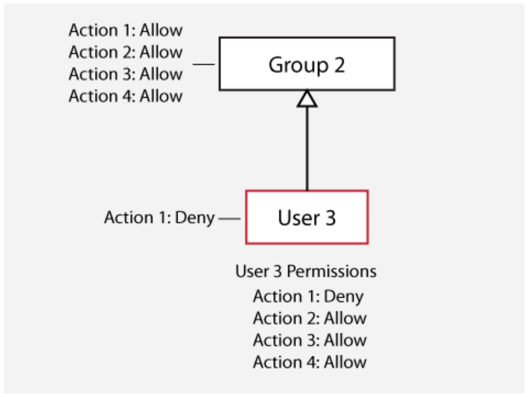
A user can be assigned their own permissions in addition to inheriting group permissions.



In this case, User 1 is assigned permissions and also inherits the permissions of Group 1, while User 2 is not assigned any permissions but inherits the permissions of Group 1.

Inheritance Masking

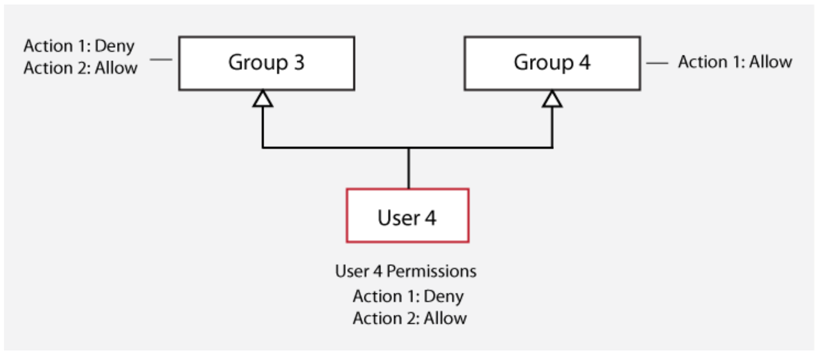
A permission assigned to a user or group masks permissions for the same action and product in parent groups.



In this case, the "Deny" permission assigned to User 3 for Action 1 masks the "Allow" permission for Action 1 in Group 2. The matching permission closest to the user in the permissioning hierarchy is always used to determine whether an action will be allowed or denied.

Multiple Inheritance Conflicts

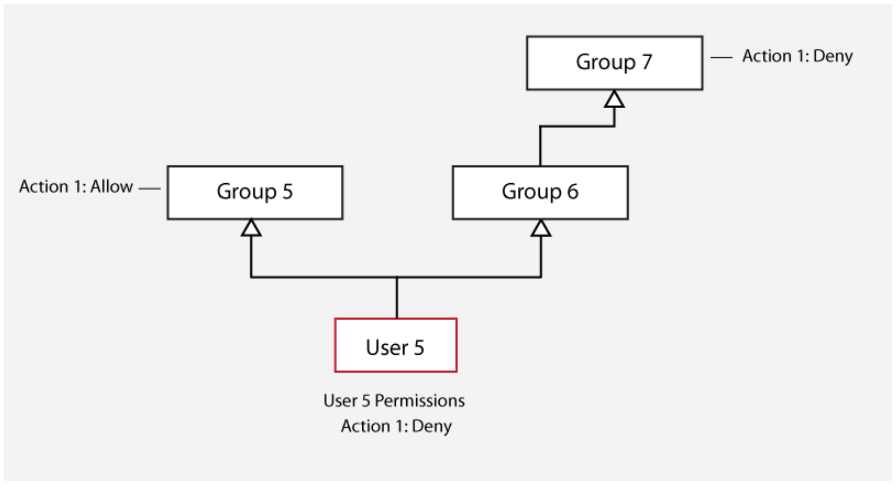
If a user is a member of more than one group and any of the inherited permissions deny an action, then this permission will override any inherited permissions that allow the action.



In this case, the "Deny" permission for Action 1 that User 4 inherits from Group 3 overrides the "Allow" permission for Action 1 that User 4 inherits from Group 4.

Multiple Inheritance Conflicts in a Complex Hierarchy

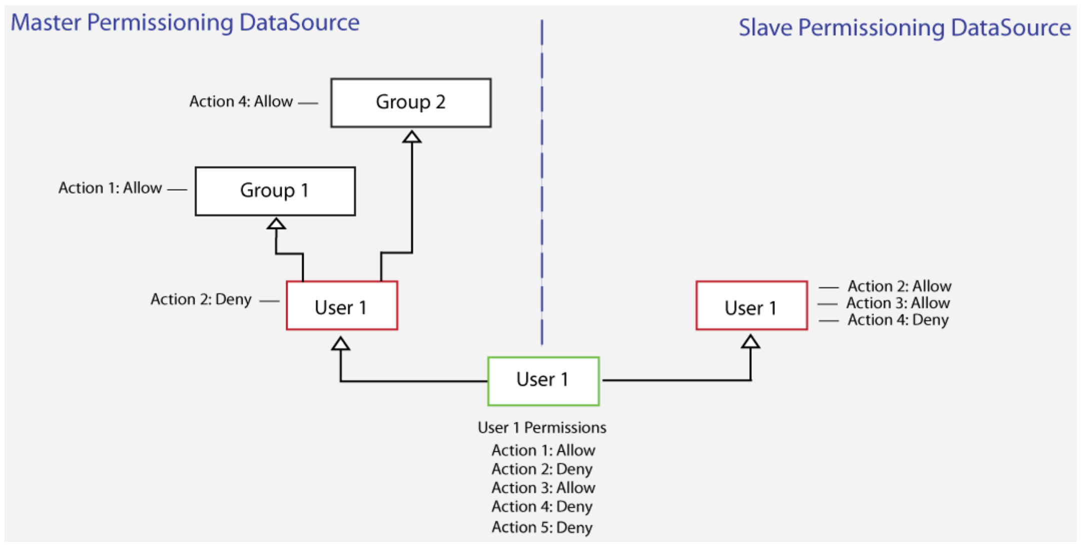
"Deny" permissions also override "Allow" permissions in more complex hierarchies.



In this case, the "Deny" permission that User 5 inherits from Group 7 overrides the "Allow" permission that User 5 inherits from Group 5. Group 5 was compared to Group 7 because Group 6 had no permission to compare.

Permissions from Multiple Permissioning DataSources

"Deny" permissions also override "Allow" permissions when permissions are set in the master and slave Permissioning DataSources. In addition, groups cannot exist in a slave Permissioning DataSource.



In this case:

- The "Deny" permission assigned to User 1 for Action 2 in the master overrides the "Allow" permission assigned to User 1 for Action 2 in the slave.
- The "Deny permission assigned to User 1 for Action 4 in the slave overrides the "Allow" permission for Action 4 that User 1 inherits from Group 2 in the master.
- Because permissions for Action 5 are not defined in the master or slave, the permission for Action 5 is "Deny".

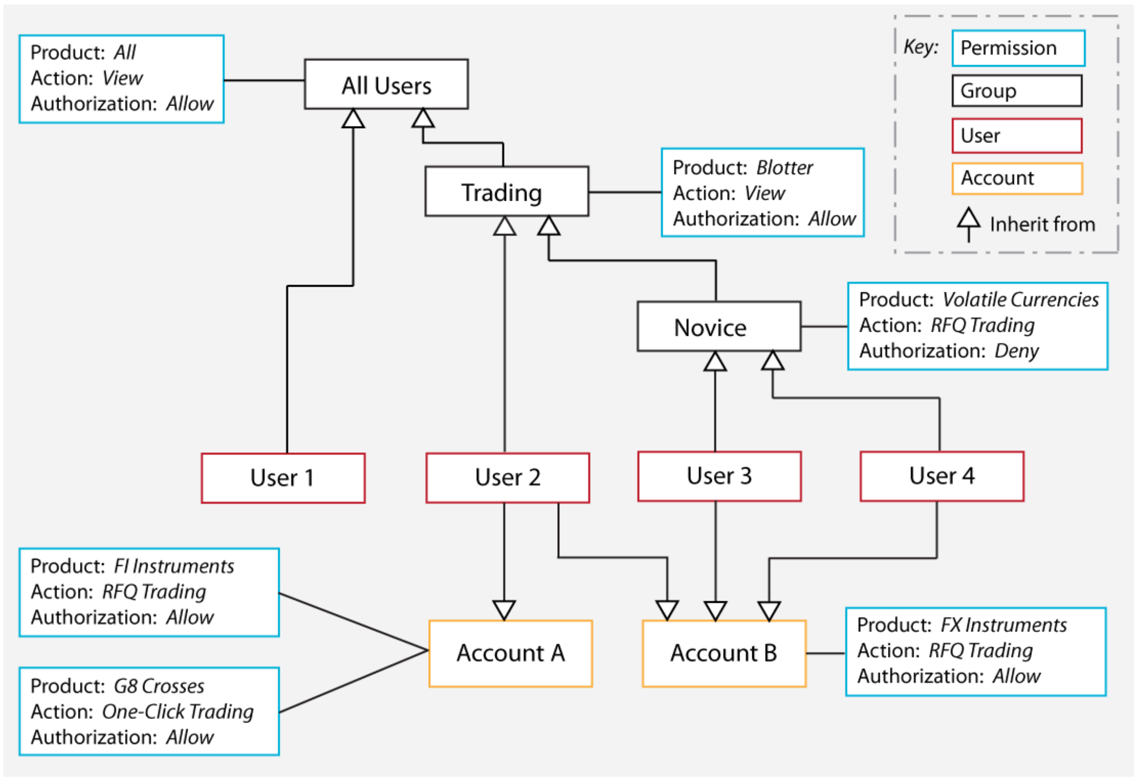
To summarize, when permissions for an action are sent to Liberator from master and slave Permissioning DataSources, permissioning conflicts are resolved as shown in the following table.

Master/Slave permissioning conflicts

Master Permission	Slave Permission	Combined Permission
"Allow"	"Allow"	"Allow"
"Allow"	Undefined	"Allow"
Undefined	"Allow"	"Allow"
"Allow"	"Deny"	"Deny"
"Deny"	"Allow"	"Deny"
Undefined	Undefined	"Deny"

3.10 Example Permissioning Hierarchy

The diagram below shows a permissioning hierarchy that uses the permissioning concepts introduced in this section. In this example, all permissions reside in the *default* namespace.



Example permissioning hierarchy

From this diagram you will see that users have been assigned permissions through inheritance.

User 1 is a view only user

- Allowed to view all products (inherited from "All Users")

User 2 is an experienced trader

- Allowed to view all products (inherited from "All Users")
- Allowed to view a blotter of previous trades (inherited from "Trading")
- Allowed to RFQ trade FI Instruments, but only when using Account A
- Allowed to One-Click trade G8 Crosses, but only when using Account A
- Allowed to RFQ trade FX Instruments, but only when using Account B

User 3 and User 4 are novice traders with the same product permissions

- Allowed to view all products (inherited from "All Users")
- Allowed to view a blotter of previous trades (inherited from "Trading")
- Allowed to RFQ trade FX Instruments *that are not volatile currencies* (inherited from "Novice"), but only when using Account B

4 Complex Permissioning Rules

Simple permissioning rules 9 were introduced when we discussed permissioning concepts 7. We will now look at how to construct more complex permissioning rules.

4.1 Matching the RTTP Message Subject

Java regular expressions and substitution tokens can be used when you define a rule or permission.

- Java regular expressions can be used in the Subject Match expression of a rule, and in the Product expression of a permission. For example "/F." matches "/FT" and "/FI", since the "." metacharacter matches any single character.

Tip: You will find further information about Java regular expressions in the API documentation for the **java.util.regex** package, as supplied by the Oracle® Corporation.

- Substitution tokens can also be used in Subject Match and Product expressions. The token %u matches the username of the logged in user, and the token %U matches the session name allocated by Liberator for a user session.

  For example, "/PRIVATE/%u/FX" matches "/PRIVATE/BOB/FX" if the user logged in as BOB, and "/PRIVATE/%U/FX" matches "/PRIVATE/BOB-0/FX" if the allocated session name is "BOB-0".

  A user can log in more than once, and is allocated a different session name each time they log in.

Tip: To place the literal string %u in a regular expression, escape the string with a "\" like this \%u.

Note: If substitution tokens are used in the definition of rules or permissions, corresponding object mappings must also be set up in the Liberator configuration (see **object-map** in the document **Caplin Liberator Administration Guide**).

Substitution tokens are used to prevent one user from accessing another user's data.

Using a substitution token in a rule

The following example shows an RTTP message that has a username (BOB) in the subject of the message.

RTTP Message

Message Subject	Message Fields			
	MsgType	*Side*	*Amount*	*Instrument*
/PRIVATE/BOB/FX/ONECLICK	Execute	Buy	500000	/FX/GBPUSD

An example of a rule that matches this request is:

Matching Rule

Subject Match	Field Match Criteria	Product Reference Field	Action	Namespace
/PRIVATE/%u/FX/ONECLICK		Instrument	ONE-CLICK	

Because the matching rule contains the %u substitution token, the Permissioning Auth Module compares the username of the logged in user with the username in the requested subject.

If the logged in user is "BOB", the rule is applied and the permission defined for "ONE-CLICK" action in the default namespace is checked to see if the request from "BOB" is to be allowed or denied.

If the logged in user is "JOHN" and not "BOB", the Permissioning Auth Module immediately denies the request.

Using a substitution token in a permission

The substitution tokens %u and %U can also be used when you define a permission.

When a user attempts to view a product a default rule is implemented, and the permission that has been defined for "VIEW" action in the *default* namespace is checked to see if the interaction is to be allowed or denied.

The following example shows a typical RTTP message for this type of request, in which the user attempts to view a product, and the username of the logged in user (BOB) is in the subject of the message.

RTTP Message

Message Subject
/PRIVATE/BOB/FX/USDGBP

Because the user is attempting to view a product, the default rule is implemented. An example of a permission that would apply to this type of request is shown below.

Permission that would apply

Action	Product	Namespace	Authorization
VIEW	/PRIVATE/%u/FX/USDGBP		Allow

In this example, the Permissioning Auth module replaces the %u token in Product with the username of the logged in user, and compares this modified Product with the requested subject.

If the logged in user is "BOB", the modified Product matches the requested subject and the request from "BOB" is allowed.

If the logged in user is "JOHN" and not "BOB", the modified Product does not match the requested subject and the request from "JOHN" is neither allowed nor denied by this permission.

4.2 Field Match Criteria

A rule can have zero or more Field Match Criteria that map RTTP message fields and values. All field mappings described by Field Match Criteria must be present in the RTTP message, otherwise the rule will not match the message. In the example below, the rule will only match an RTTP message if the message field "Trading-Type" has the value "SPOT", and the message field "SIDE" has the value "Buy".

Example Rule

Subject Match	Field Match Criteria	Product Reference Field	Action	Namespace
/FT/TRADE	Trading-Type=SPOT, SIDE=Buy	Instrument	spot trade	

If the rule does not define any Field Match Criteria, then the RTTP message fields will be ignored when the rule is being matched to the message.

You will notice from the above example that it is not necessary to include the "Instrument" value of Product Reference Field in the Field Match Criteria when you define the rule. Product matching is described in The Product Reference Field 18.

4.3 The Product Reference Field

The Product Reference Field of a rule identifies the field in the RTTP message that contains the name of the product. If the rule matches the message, the permission for the identified product is checked to see if the action is allowed or denied (see Permissions 7).

Matching All Products

The Product Reference Field can be given the special value "ALL_PRODUCTS" (equivalent to the Java regular expression ".*"), in which case all product permissions for the action are checked to see if the action is allowed or denied. We will now look at an example RTTP message and a matching rule that uses this special value.

RTTP Message

	Message Fields			
Message Subject	*MsgType*	*SIDE*	*Amount*	*Instrument*
/FX/ONECLICK	Execute	Buy	500000	/FX/USDGBP

An example of a rule that matches this type of trade is shown below.

Matching Rule

Subject Match	Field Match Criteria	Product Reference Field	Action	Namespace
/FX/ONECLICK		ALL_PRODUCTS	ONE-CLICK	

In this case all product permissions for "ONE-CLICK" action in the *default* namespace are checked to see if the user is to be allowed or denied access. If any of these permissions deny "ONE-CLICK" action, access to "/FX/EURGBP" is denied.

An example of a permission that applies to this type of trade is shown below.

Permission that would apply

Action	Product	Namespace	Authorization
ONE-CLICK	/FX/EURGBP		Allow

Although this permission is for "/FX/EURGBP", it also applies to "/FX/USDGBP", since the rule specifies that all product permissions for "ONE-CLICK" action in the *default* (undefined) namespace must be checked to see if the action is allowed or denied.

Using Java Regular Expressions

The Product Reference Field can be a Java regular expression, in which case every field of the RTTP message that matches the regular expression will be considered to contain the name of a product. Java regular expressions can be used to define rules for multi-leg trading, as shown in the following example.

RTTP Message

	Message Fields			
Message Subject	*L1_*	*L2_*		
/TRADE/FX	/FX/GBPUSD	/FX/USDJPY		

An example of a rule that would match this type of trade is shown below.

Matching Rule

Subject Match	Field Match Criteria	Product Reference Field	Action	Namespace
/TRADE/FX		L\d_	TRADE	TRADER

The "\d" in the regular expression of the Product Reference Field will match any digit. In this case the permission that has been defined for the "TRADE" action in the "TRADER" namespace will be checked to see if the user is to be allowed or denied access to the products identified by fields "L1_" and "L2_".

An example of the permissions that would apply to this type of multi-leg trade is shown below.

Permissions that would apply

Action	Product	Namespace	Authorization
TRADE	/FX/GBPUSD	TRADER	Allow
TRADE	/FX/USDJPY	TRADER	Allow

In this case "TRADE" action on the products "/FX/GBPUSD" and "/FX/USDJPY" will be allowed, since the rule states that the permission applies to each of these products. If the user did not have "Allow" permission for each of these products, then the multi-leg trade would be denied.

4.4 The Action Reference Field

We have already looked at how you can define the action for a rule in the Action field of the rule 9. Rules can also be defined that derive the action from a field in the RTTP message.

In the following example, we look at a rule that uses the Action Reference Field to identify the field of the RTTP message that defines the action.

RTTP Message

Message Subject	Message Fields		
	Tenor	*Trading-Type*	*Instrument*
/TRADE/FX	1Month	RFQ	/FX/GBPUSD

An example of a rule that would match this type of trade is shown below.

Matching Rule

Subject Match	Field Match Criteria	Product Reference Field	Action	Action Reference Field	Namespace
/.*		Instrument		Tenor	TenorPermissions

The rule matches since the subject of the RTTP message is "/TRADE/FX", which matches the regular expression "/.*" in the Subject Match field of the rule.

An example of the permission that would apply to this type of trade is shown below.

Permissions that would apply

Action	Product	Namespace	Authorization
1Month	.*	TenorPermissions	Allow

In this case "RFQ" trading with a tenor of one month will be allowed. The permission matches because:

- The Action Reference Field of the rule identifies "Tenor" as the RTTP field that defines the action.
- The "Tenor" field of the RTTP message has the value "1Month".
- The permission allows "1Month" action for all products in the "TenorPermissions" namespace.

5 Additional Permissioning Capabilities

We will now look at some additional permissioning capabilities.

5.1 Subject Mapping

When a user attempts to view data, a default rule is implemented (see Rules 9). This default rule states that the Permissioning Auth Module must check the user's permission in the *default* namespace for "VIEW" action on the product identified by the subject of the RTTP message. If the action is permitted, then Liberator will stream data to the user for the requested product.

Subject mapping allows the subject of an RTTP message to be modified by Liberator. Subject mapping is transparent to the user and could be used, for example, to provide preferential data to selected users.

The Default Subject Mapper

A default **subject mapper** is provided with the Permissioning software that allows you to add a subject mapping for a user. This mapping specifies a subject pattern and a subject suffix. If the user attempts to view data where the subject of the RTTP message matches the subject pattern, then the subject suffix will be appended to the subject of the RTTP message when Liberator requests the data from the pricing DataSource.

To view the data, the user must be permitted "VIEW" action in the *default* namespace for the product identified by the *modified* subject of the RTTP message.

When Liberator streams the data to the user, the data will be streamed using the *original* RTTP message subject, but for the product identified by the *modified* RTTP message subject.

Only one subject mapping can be added for each user by a Permissioning DataSource using the default subject mapper. The subject mapping is added by calling the `User.setSubjectMapping()` method of the **Permissioning DataSource API**.

Adding Multiple Subject Mappings for a User

The `RegexSuffixSubjectMapper` class of the Permissioning DataSource API allows multiple subject mappings to be added for each user. This class is set as the subject mapper for a user by calling the `User.setSubjectMapper()` method of the Permissioning DataSource API, and subject mappings are added by calling `User.addSubjectMapping()`.

Tip: For a complete description of the Permissioning DataSource API, please refer to the **Permissioning DataSource: API Reference** documentation.

Creating a Custom Subject Mapper

If you want to calculate a subject mapping based on customized mapping logic, then you must create a custom subject mapper. The custom subject mapper must implement the `SubjectMapper` interface of the Permissioning DataSource API and be deployed in the Permissioning Auth Module.

For more information on how to implement and deploy a custom subject mapper, see **Caplin Xaqua: How to Create a Permissioning DataSource Adapter**.

Simple Example of Subject Mapping using the Default Subject Mapper

The following example shows how subject mapping can be applied to a user when Liberator supports price tiering. In this example the subject pattern is the regular expression "/PRICES/FX/.*" and the subject suffix is "-tier2".

Applied subject mapping

Subject Pattern	Subject Suffix
/PRICES/FX/.*	-tier2

If the user attempts to view "/PRICES/FX/GBPUSD", Liberator will request the data from the DataSource provider for "/PRICES/FX/GBPUSD-tier2". When Liberator streams the data to the user, it will be streamed from "/PRICES/FX/GBPUSD-tier2" but using the original RTTP message subject "/PRICES/FX/GBPUSD".

An example of a permission that would apply to this type of data request is shown below.

Permission that would apply

Action	Message Subject	Namespace	Authorization
VIEW	/PRICES/FX/.*-tier2		Allow

In this case VIEW action on "/PRICES/FX/GBPUSD-tier2" will be allowed, since "/PRICES/FX/.*-tier2" matches all "-tier2" currency pairs in "/PRICES/FX" and the authorization is "Allow". Note that the permission matches the *modified* message subject, and not the *original* message subject.

Global context data

The **global context** is an object at the Permissioning Auth module. This object allows custom subject mappers to access data that is common to all subject mappers and users. In this way a custom subject mapper can map subjects using logic based on this common data, and not just on subject mappings defined for the user.

For example, if a custom subject mapper uses FX rates to map a subject, it is more efficient to add these rates to the global context than to the subject mappings of every user.

A default global context class is provided with the Permissioning software, but you can also create a custom global context class that provides specialized methods to subject mappers.

The document **Caplin Xaqua: How to Create a Permissioning DataSource Adapter** describes how to create a custom global context class. The document also describes how a Permissioning DataSource can add data to the global context, and how a custom subject mapper can access this data.

5.2 User Attributes

A user can be assigned any number of attributes in the form of name/value pairs. User attributes are not processed by the Permissioning Auth Module 6, and therefore do not affect permissioning directly, but can be accessed by the client application to provide miscellaneous contextual information about the currently logged in user.

A typical use would be to send information to a Caplin Xaqua client about the maximum tradable amount that users are permitted to trade.

Defining and Retrieving User Attributes

In the Permissioning DataSource Adapter, user attributes are defined in Java using the `com.caplin.permissioning.User.setAttribute()` method of the Permissioning DataSource API. In the following example, the maximum tradable amount for a user is set to 5 million USD.

```
/* create new user with login name trader1 */
User maxTradeUser = new User(trader1);

...

/* set the max tradable USD (millions) */
maxTradeUser.setAttribute(maxTradeUSD, 5);
```

Tip: The document **Caplin Xaqua: How To Create A Permissioning DataSource** describes how to create a custom Permissioning DataSource adapter.

In a client application that is based on Caplin Trader, user attributes are retrieved in JavaScript using the `caplin.security.permissioning.PermissionService.getUserAttribute()` method of the Caplin Trader Permissioning API. In the following example, the maximum tradable amount in USD is retrieved for the currently logged in user.

```
/* get the max tradable USD (millions) */
maxTradableUSD =
    caplin.security.permissioning.PermissionService.getUserAttribute(maxTradeUSD);
```

Tip: The document **Caplin Trader: How to Add Permissioning At The Client** describes how to modify the appearance and behaviour of Caplin Trader to match the permissions of the currently logged in user.

6 Further Reading

If you would like to know how to create a custom Permissioning DataSource adapter, or how to add Permissioning to Caplin Trader, then the following documents provide this information. You may also be interested in reading some of the other Related documents 1.

How To Create A Permissioning DataSource Adapter

A Permissioning DataSource adapter is required to integrate Caplin Xaqua with a Permissioning System. The document **Caplin Xaqua: How To Create A Permissioning DataSource Adapter** describes how to create a custom Permissioning DataSource adapter by writing an application that uses the Permissioning DataSource API. The document also discusses the Demo Permissioning DataSource that is provided with the reference implementation of Caplin Trader from release 1.2.8.

How To Add Permissioning At The Client

The appearance and behaviour of Caplin Trader can be tailored to match the permissions of the currently logged in user. You will find further information about how to do this in the document **Caplin Trader: How To Add Permissioning At The Client**.

7 Glossary of terms and acronyms

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

Term	Definition
Account	A user must trade using an account. An account can be assigned permissions that allow users of the account to **action** certain products.
Action	The interaction that a user can have with a **product**.
API	Application Programming Interface
Caplin Trader	A web application framework for constructing browser-based financial trading applications (**Caplin Trader applications**).
Caplin Trader application	A **Caplin Xaqua client** that has been built using **Caplin Trader**.
Caplin Xaqua	A framework for building single-dealer platforms that enables banks to deliver multi-product trading direct to client desktops. Caplin Xaqua can also be short for a **Caplin Xaqua system**.
Caplin Xaqua client	A client desktop or web application that interfaces with **Caplin Xaqua** to deliver multi-product trading to end-users.
Caplin Xaqua system	A single-dealer platform that is built using **Caplin Xaqua**.
DataSource	An **API** and underlying code library that allows **DataSource applications** to communicate with each other.
DataSource adapter	A **DataSource application** that acts as the interface between **Caplin Xaqua** and an external (non-Caplin) system, exchanging data and/or messages with that system.
DataSource application	A **Caplin Xaqua** application that uses the **DataSource API** and code library to communicate with other Caplin Xaqua applications.
Demo Permissioning DataSource	The Demo Permissioning DataSource is an example of a **Permissioning DataSource** that gets its permissioning data from an XML file.
Global context	An object at the **Permissioning Auth Module**. The global context allows custom **subject mappers** to access data that is common to all subject mappers and **users**.
Group	A logical grouping of zero or more **users** and other groups, such that each group can be assigned zero or more **permissions**.
Liberator	A real-time financial internet hub that delivers trade messages and market data to and from subscribers over any network that supports web traffic.
Master	When permissioning data is sent to Liberator from multiple **Permissioning DataSource** adapters, one of the Permissioning DataSource adapters is designated the master, and the others are designated as **slaves**.
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**.

Term	Definition
Permissioning DataSource	A **DataSource adapter** that acts as the interface between **Caplin Xaqua** and your **Permissioning System**.
Permissioning DataSource API	The **API** that a **Permissioning DataSource** uses to send permissioning data to **Liberator**.
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 Xaqua** and **Caplin Trader** documentation a "product" is a term that refers only to a financial instrument.
Role	Roles determine whether a **Permissioning DataSource** is designated as a **master** or **slave** Permissioning DataSource.
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
Slave	When permissioning data is sent to Liberator from multiple **Permissioning DataSource** adapters, one of the Permissioning DataSource adapters is designated the **master**, and the others are designated as slaves.
Subject mapper	A subject mapper is a Java class that resides at the **Permissioning Auth Module**. A subject mapper can modify the message that Liberator receives when an end-user attempts to view or trade a product, and can be used to provide preferential data to selected **users**.
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