

Caplin Trader 1.3

How To Create A Permissioning DataSource Adapter

January 2009

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.....	3
1.5	Feedback.....	3
1.6	Acknowledgments.....	3
2	What is a Permissioning DataSource?.....	4
2.1	The Permissioning DataSource API.....	5
3	Creating a Permissioning DataSource Adapter.....	6
3.1	About Transactions.....	9
	API methods for starting a transaction	9
	When should an Image or Update transaction be used?	9
3.2	Updating Permissioning Data.....	10
	Creating Users	10
	Creating Groups	11
	Removing Users and Groups	11
	Setting a User's Password	12
	Changing a User's Permissions	12
	Removing a Permission from a User	13
	Changing a Group's Permissions	13
	Changing the Subject Mapping for a User	14
	Changing User Attributes	14
	Changing the Members of a Group	15
4	The Demo Permissioning DataSource.....	16
4.1	Starting and Stopping the Demo Permissioning DataSource.....	17
	Starting the Demo Permissioning DataSource	17
	Stopping the Demo Permissioning DataSource	17
4.2	Overview of the Demo Permissioning DataSource.....	18
5	The Demo Permissioning XML.....	19
5.1	Technical Assumptions and Restrictions.....	19
5.2	Ordering and Nesting of Tags.....	19

5.3	XML Reference Information.....	22
	<attributes>	22
	<fieldMatchCriteria>	22
	<group>	22
	<groupRef>	22
	<groups>	23
	<match>	23
	<members>	23
	<permission>	23
	<permissioning>	24
	<permissionSet>	24
	<productPermissionSet>	24
	<rule>	25
	<rules>	26
	<subjectMapping>	26
	<user>	26
	<userAttribute>	27
	<userRef>	27
	<users>	27
6	Further Reading.....	28
7	Glossary of terms and acronyms.....	29
	Index.....	30

1 Preface

1.1 What this document contains

This document describes how you can create a Permissioning DataSource adapter by writing an application that uses the Permissioning DataSource API. A Permissioning DataSource adapter is required to integrate Caplin Trader with a Permissioning System. The document also discusses the Demo Permissioning DataSource that is provided with the reference implementation of Caplin Trader from release 1.2.8.

Before reading this document, make sure you are familiar with the document **Caplin Trader: 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.

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 integrate Caplin Trader with a Permissioning System.

1.3 Related documents

- ◆ **Caplin Trader: Architecture**
Describes the architecture of Caplin Trader. It focuses on the use of the Caplin Platform in trading applications. It also identifies the areas in which the Platform can be integrated with your company's own and third-party systems.
- ◆ **Caplin Liberator: Administration Guide**
Describes how to install and configure Caplin Liberator and discusses the authentication modules that are provided with the server.

◆ **Caplin Trader: Permissioning Overview and Concepts**

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

◆ **Caplin Trader: Installing Permissioning Components**

Describes how to install the Permissioning Auth Module and Permissioning DataSource in an existing Caplin Trader 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 Trader Client: How To Add Permissioning At The Client**

Describes how to add Permissioning to Caplin Trader Client.

◆ **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 Trader.

◆ **Caplin Trader Client: API Reference**

The API reference documentation provided with Caplin Trader Client. 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 Client.

1.4 Typographical conventions

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

Type	Uses
aMethod	Function or method name
<i>aParameter</i>	Parameter or variable name
<i>/AFolder/Afile.txt</i>	File names, folders and directories
<div>Some code;</div>	Program output and code examples
The value=10 attribute is...	Code fragment in line with normal text
Some text in a dialog box	Dialog box output
Something typed in	User input – things you type at the computer keyboard
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

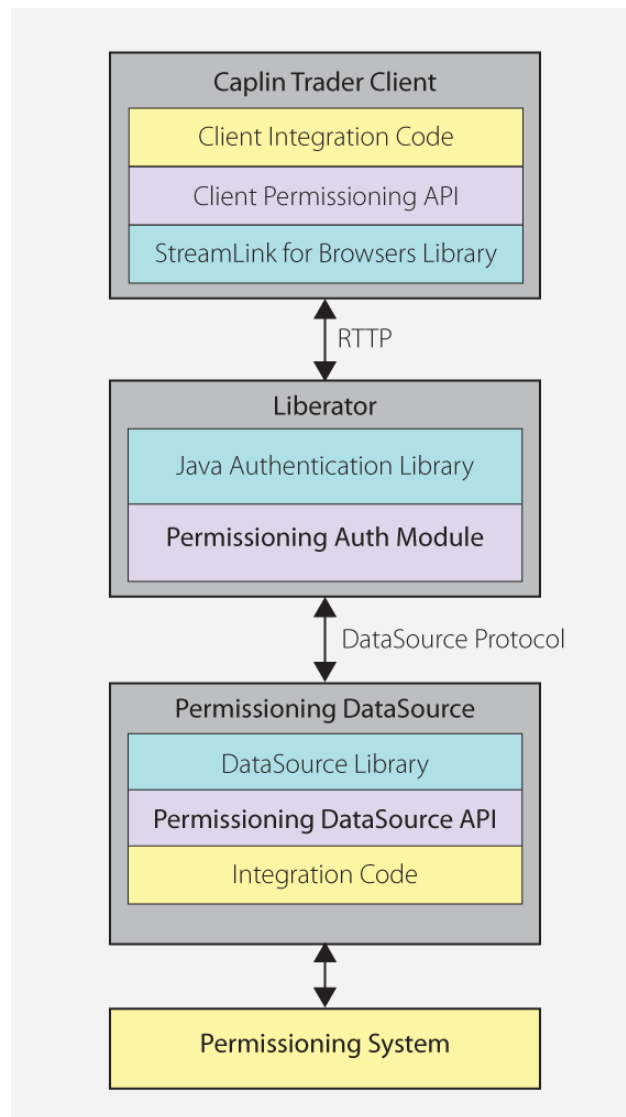
Firefox is a registered trademark of the Mozilla Foundation.

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

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

2 What is a Permissioning DataSource?

A Permissioning DataSource is a DataSource Adapter that acts as the interface between Caplin Trader 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.



**Simplified Caplin Trader architecture
showing only permissioning components**

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.

Tip: You will find further information about the permissioning components of the Caplin Trader architecture in the document **Caplin Trader: Permissioning Overview And Concepts**.

2.1 The Permissioning DataSource API

The Permissioning DataSource API is part of the Permissioning DataSource SDK (Software Development Kit) and allows you to write applications that can send permissioning data to Caplin Liberator. The SDK is delivered with Caplin Trader and contains the following components.

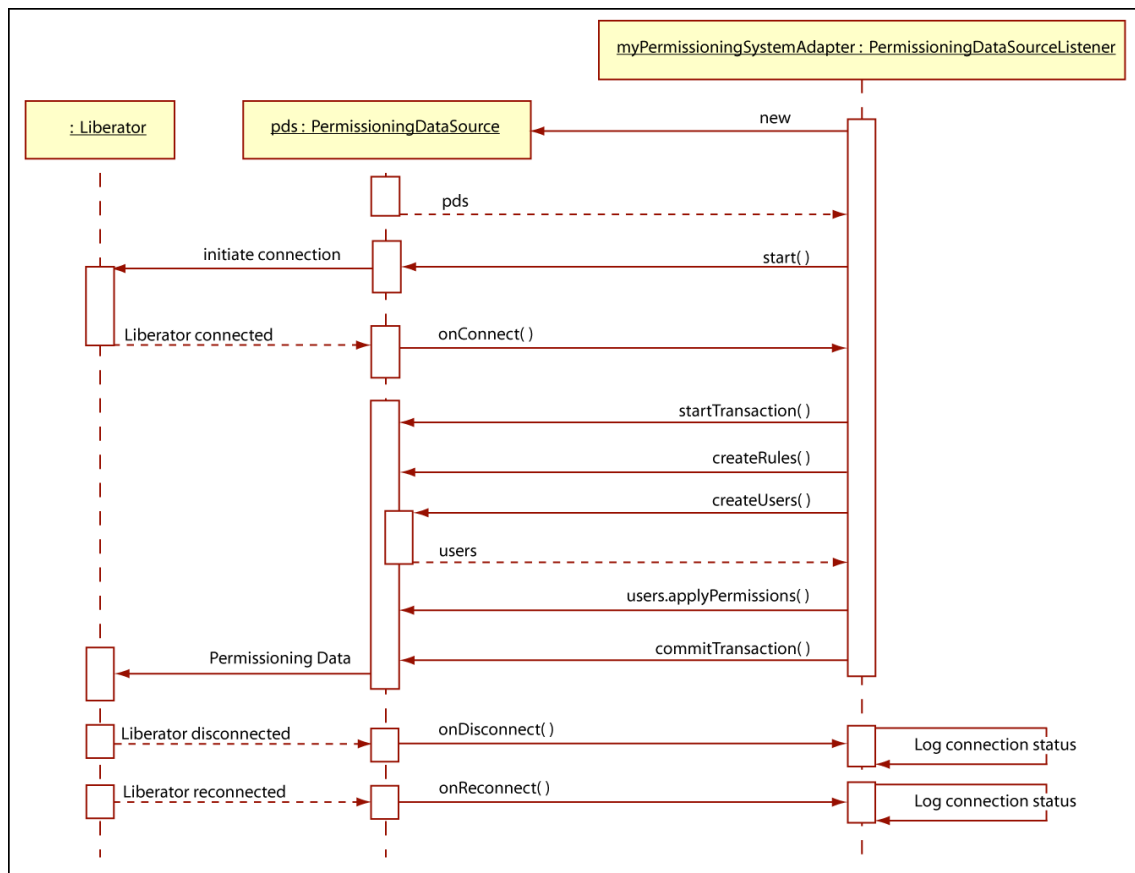
- ◆ The library of Java classes that provide the Permissioning DataSource API.
- ◆ **Permissioning DataSource: API Reference** that includes an overview, and package and class-level documentation.
- ◆ A [Demo Permissioning DataSource Adapter](#)¹⁶. This example application uses the Permissioning DataSource API to provide Liberator with permissioning data from an [XML file](#)¹⁹.

The Permissioning DataSource API is contained in a single package that provides the classes and interfaces you need to integrate Caplin Trader with a Permissioning System. The package also includes classes for assigning permissions to Users and Groups, classes for storing permissioning data, and classes for handling exceptions.

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

3 Creating a Permissioning DataSource Adapter

The Permissioning DataSource API provides the interface between the Caplin Platform and a Permissioning System. When you write an application that uses this API, your code must implement the `PermissioningDataSourceListener` interface and instantiate a `PermissioningDataSource`, as summarized in the following sequence diagram and in steps 1 to 3 below.



Typical sequence of events for sending permissioning data to Liberator

1. Implement the `PermissioningDataSourceListener` interface

This interface has three callback methods that your code must implement. The first of these callback methods, `onConnect()`, is triggered by the `PermissioningDataSource` when a new connection to Liberator is established.

Your implementation of `onConnect()` would typically respond to this trigger by starting a transaction, applying permissioning data to the `PermissioningDataSource`, and then committing the transaction. Data is applied to the `PermissioningDataSource` when you call one of the `createUser`, `createGroup` and `create-rule` methods as part of a transaction. Committing the transaction sends that data to Liberator.

The other two callback methods, `onDisconnect()` and `onReconnect()`, simply inform your application about the Liberator connection status. There is no need to send permissioning data or start a transaction, and your implementation would typically respond by logging the connection status.

2. Instantiate a PermissioningDataSource

The `PermissioningDataSource` has one constructor that expects three arguments in the following order:

- ◆ An instance of your `PermissioningDataSourceListener` implementation, as described in step 1 above.
- ◆ A DataSource XML configuration file (*conf/DataSource.xml*), in the form of an `InputStream`. This file configures the `PermissioningDataSource` as a DataSource adapter, and must contain network connection information for your particular network.
- ◆ A DataSource XML field mapping file (*conf/Fields.xml*), in the form of an `InputStream`. This file maps DataSource field names to field numbers, and must match the field name to number mappings that are used by Liberator.

[The Demo Permissioning DataSource](#)¹⁶ that is supplied with the SDK has an example DataSource XML configuration file and example DataSource XML field mapping file. You can either create your own version of these files or customize the supplied example files as required.

3. Start the PermissioningDataSource

You start a `PermissioningDataSource` when you call `PermissioningDataSource.start()`.

The following code sample is a trivial implementation of a `PermissioningDataSourceListener`, as summarized in steps 1 to 3 above.

```
// Step 1: Implement the PermissioningDataSourceListener interface
public class MyPermissioningSystemAdapter implements PermissioningDataSourceListener
{
    private PermissioningDataSource pds;

    public MyPermissioningSystemAdapter() throws IOException, SAXException
    {
        // Step 2: Instantiate a PermissioningDataSource,
        // passing this adapter as a listener
        pds = new PermissioningDataSource(this,
                                         <DataSource.Config.Stream>,
                                         <Fields.Config.Stream>);

        // Step 3: Start the PermissioningDataSource
        pds.start();
    }

    // Implement the onConnect() callback
    public void onConnect()
    {
        // start a PermissioningDataSource image transaction
        pds.startTransaction();

        // create some Rules
        pds.createActionRule( ... );
        pds.createActionRefRule( ... );

        // create some Users and configure them
        User user1 = pds.createUser( ... );
        user1.applyPermission( ... );
        user1.setSubjectMapping( ... );

        User user2 = pds.createUser( ... );
        user2.applyPermission( ... );

        // create some Groups and configure them
        Group group1 = pds.createGroup( ... );
        group1.applyPermission( ... );
        group1.addMember( user1 );

        Group group2 = pds.createGroup( ... );
        group2.applyPermission( ... );
        group2.addMember( user1 );
        group2.addMember( user2 );

        // send the permissioning data by committing the transaction
        pds.commitTransaction();
    }

    // Implement the onDisconnect() callback
    public void onDisconnect()
    {
        System.out.println("Disconnected from Liberator!")
    }

    // Implement the onReconnect() callback
    public void onReconnect()
    {
        System.out.println("Reconnected to Liberator!")
    }
}
```

In the code sample above:

- ◆ Text within <angled brackets> represents parameters that must be defined in your code.
- ◆ Text shown as (...) represents parameters that have been omitted for simplicity.

3.1 About Transactions

Transactions ensure that one or more operations on permissioning data are sent to Liberator as a single atomic unit. A typical sequence of events would be:

1. Start a transaction.
2. Apply permissioning data to the `PermissioningDataSource` (for example add and remove users, groups and permissions).
3. Commit the transaction.

Permissioning data is sent from the `PermissioningDataSource` to Liberator when the transaction is committed. The Permissioning Auth Module (which is embedded in Liberator) will not apply any permissioning data until all the data for a transaction is received.

API methods for starting a transaction

The Permissioning DataSource API provides two methods for starting a transaction.

◆ `startImageTransaction()`

Call this method when you want to apply a new set of permissioning data to Liberator. When you commit the transaction, all permissioning data in the `PermissioningDataSource` is sent to Liberator. Liberator replaces any permissioning data from previous transactions with this new permissioning data. Rules *must* be applied as part of an image transaction.

◆ `startUpdateTransaction()`

Call this method when you want to update permissioning data. When you commit the transaction, only changes to permissioning data are sent to Liberator. Liberator updates any permissioning data from previous transactions with this new permissioning data. Rules *cannot* be applied as part of an update transaction.

When should an Image or Update transaction be used?

The table below shows the type of transaction that is required (image or update) to send permissioning data to Liberator. The `startImageTransaction()` method starts an image transaction, and the `startUpdateTransaction()` method starts an update transaction (see [About Transactions](#)⁹).

Situation	Type of transaction required
When an <code>onConnect()</code> callback is received.	Start an image transaction. The permissioning data that you send will replace any existing permissioning data in Liberator.
When an <code>onReconnect()</code> callback is received.	This callback is for information only, you <i>do not</i> need to start a transaction or send permissioning data to Liberator.
When permissioning data in your Permissioning System changes (for example, when a new user is added to your Permissioning System).	Start an update transaction. The permissioning data that you send will modify the existing permissioning data in Liberator.
When you want to replace an existing set of permissioning data with a new set of permissioning data.	Start an image transaction. The permissioning data that you send will replace any existing permissioning data in Liberator.

Situation	Type of transaction required
When you want remove all permissioning data and eject all users currently logged in to Caplin Trader Client and/or Liberator.	Send an empty image transaction. This will clear all permissioning data from the <code>PermissioningDataSource</code> and from Liberator.

3.2 Updating Permissioning Data

The following examples show you how to update the permissioning data that has already been sent to Liberator. You update permissioning data as part of an update transaction.

Creating Users

This example creates a new user in the `PermissioningDataSource` (`pds`). When the transaction is committed, the data for this user is sent to Liberator.

```
pds.startUpdateTransaction();
pds.createUser("John.Smith", "johnsPassword");
pds.commitTransaction();
```

The `getUser()` method can later be used to get a reference to the user "John Smith" (see [Setting a User's Password](#)^[12]).

Applying Permissions

Permissions can either be applied as part of the same transaction in which the user is created, or in later transactions.

The following example creates a new user and then gives this user the permission to "SPOT-TRADE" all products in the "TradeType" namespace.

```
pds.startUpdateTransaction();
User newUser = pds.createUser("John.Smith", "johnsPassword");
Set products = new HashSet();
products.add("/.*");
newUser.applyPermission(products, "TradeType", "SPOT-TRADE", Authorization.ALLOW);
pds.commitTransaction();
```

We look at how to change the permissions of an existing user in [Changing a User's Permissions](#)^[12].

Creating Groups

The following example creates a new group, applies a permission to the group, and then adds an existing user to the group. When the transaction is committed, the data for this group is sent to Liberator.

```
pds.startUpdateTransaction();

// create a new Group
Group newGroup = pds.createGroup("RFQ-Traders");

// build up a product set
Set products = new HashSet();
products.add("/.*");

// apply the permission to the Group
newGroup.applyPermission(products, "TradeType", "RFQ", Authorization.ALLOW);

// retrieve an existing user from the permissioning datasource
User existingUser = pds.getUser("John.Smith");

//add the user as a member of the new Group
newGroup.addMember(existingUser);
pds.commitTransaction();
```

In the example above, `pds.getUser()` retrieves an existing user from the `PermissioningDataSource`. This user, who was created in an earlier transaction (see [Creating Users](#) ^[10], now inherits the permissions of the new group to "RFQ" trade all products in the "TradeType" namespace.

Removing Users and Groups

In this example we remove the user and group that we created in previous transactions (see [Creating Users](#) ^[10] and [Creating Groups](#) ^[11]).

```
pds.startUpdateTransaction();
Group group = pds.getGroup("RFQ-Traders");
pds.removeGroup(group);

User user = pds.getUser("John.Smith");
pds.removeUser(user);
pds.commitTransaction();
```

When you remove a group that has members, the members are not removed from the inheritance hierarchy but they no longer inherit permissions from the removed group or any of its parents.

When you remove a user, the user is automatically removed from all parent groups and will no longer be able to log in to Caplin Trader Client. If the removed user was already logged in to Caplin Trader Client, then they will be disconnected.

When you remove a user or group, references to the removed user or group object can no longer be used and should be de-referenced so that the object can be garbage collected. If you need to re-create a removed user or group, use `createUser()` or `createGroup()` inside a transaction to create a new object for that user or group.

Setting a User's Password

In this example we change a user's password.

```
pds.startUpdateTransaction();
User user = pds.getUser("John.Smith");

// set the new password
user.setPassword("new-password");
pds.commitTransaction();
```

If a user's password is changed when the user is logged in to Liberator, they will be disconnected immediately and will have to log back in using the new password.

Changing a User's Permissions

The `User.applyPermissions()` method can either be used to add a new permission to a user or to modify an existing permission.

In this example the permission to "OneClick" trade the "FX/GBPUSD" product in the "TradeType" namespace is added to the permissions already assigned to this user.

```
pds.startUpdateTransaction();

// acquire a reference to the User
User user = pds.getUser("John.Smith");

// build up the product set
Set products = new HashSet();
products.add("/FX/GBPUSD");

// apply the permission
user.applyPermission(products, "TradeType", "OneClick", Authorization.ALLOW);
pds.commitTransaction();
```

This permission would replace any other permission the user had for this product, action and namespace.

Removing a Permission from a User

In this example we remove the permission to "OneClick" trade the "FX/GBPUSD" product that we assigned in the previous transaction (see [Changing a User's Permissions](#) ¹²).

```
pds.startUpdateTransaction();
User user = pds.getUser("John.Smith");
Set products = new HashSet();
products.add("/FX/GBPUSD");

// remove the OneClick permission in the TradeType namespace for /FX/GBPUSD
user.removePermission(products, "TradeType", "OneClick");
pds.commitTransaction();
```

Attempting to remove a permission that has not been assigned has no effect.

Changing a Group's Permissions

The `Group.applyPermissions()` method can either be used to add a new permission to a group or to modify an existing permission.

In this example we remove the permission to trade all FX products, and add a permission to trade a small set of FX products.

```
pds.startUpdateTransaction();

// acquire a reference to the group
Group group = pds.getGroup("JuniorTraders");

// remove the promiscuous permission for all FX products
Set oldProducts = new HashSet();
oldProducts.add("/FX/.*");
group.removePermission(oldProducts, "TradeType", "OneClick");

// allow "OneClick" action on a small, explicit set of FX products
Set newProducts = new HashSet();
newProducts.add("/FX/GBPUSD");
newProducts.add("/FX/GBPAUD");
group.applyPermission(newProducts, "TradeType", "OneClick", Authorization.ALLOW);

pds.commitTransaction();
```

Attempting to remove a permission that has not been assigned has no effect.

Changing the Subject Mapping for a User

The `setSubjectMapping()` method can be used to add a new subject mapping or to change an existing subject mapping.

The following example shows a subject mapping being changed for one user, and a subject mapping being removed for another user.

```
pds.startUpdateTransaction();

// modify User with existing subject-mapping
User userWithChangedMapping = pds.getUser("John.Smith");
userWithChangedMapping.setSubjectMapping("/FX/.*", "-tier2");

// remove a User's subject-mapping
User userWithRemovedMapping = pds.getUser("Jane.Davis");
userWithRemovedMapping.removeSubjectMapping();
pds.commitTransaction();
```

Because a user can only have one subject mapping, the `removeSubjectMapping()` method does not require any parameters.

Attempting to remove a subject mapping that has not been assigned has no effect.

Changing User Attributes

A user can be assigned any number of attributes in the form of name/value pairs.

In this example we change the value of the "MaxTradeDollars" attribute to 3 million for an existing user.

```
pds.startUpdateTransaction();
User user = pds.getUser("John.Smith");

// modify an existing attribute (assumes MaxTradeDollars already set - not shown here)
user.setAttribute("MaxTradeDollars", "3000000");
pds.commitTransaction();
```

The next example shows how to remove the "MaxTradeDollars" attribute from the same user.

```
pds.startUpdateTransaction();
User user = pds.getUser("John.Smith");

// remove an attribute
user.removeAttribute("MaxTradeDollars");
pds.commitTransaction();
```

Attempting to remove an attribute that has not been assigned has no effect.

Changing the Members of a Group

The members of a group can be changed using the methods `Group.addMember()` and `Group.removeMember()`. Adding and removing group members affects every child that inherits from the group.

In this example we give an existing user a new parent and grandparent.

```
pds.startUpdateTransaction();
User user = pds.getUser("John.Smith");

// create the parent Group and add the User as a member
Group parent = pds.createGroup("Parent");
parent.addMember(user);

// create the grandparent group and add the earlier parent group as a member
Group grandparent = pds.createGroup("Grandparent");
grandparent.addMember(parent);

pds.commitTransaction();
```

The user will now inherit permissions (not shown in this example) from both the parent and the grandparent.

We now remove the parent group from the grandparent group.

```
pds.startUpdateTransaction();

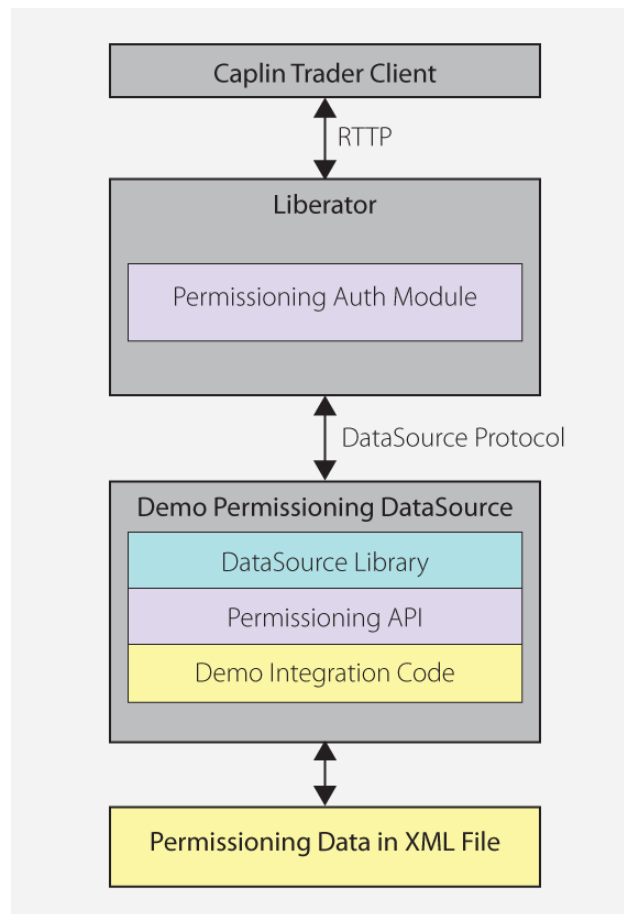
// acquire a reference to the two groups that are to be detached from each other
Group parent = pds.getGroup("Parent");
Group grandparent = pds.getGroup("Grandparent");

// sever the relationship
grandparent.removeMember(parent);
pds.commitTransaction();
```

The user continues to inherit permissions from the parent group but no longer inherits permissions from the grandparent group, because the grandparent is no longer an ancestor of this user.

4 The Demo Permissioning DataSource

The Demo Permissioning DataSource is an example of a Permissioning DataSource Adapter application that gets its permissioning data from an XML file. The application sends the permissioning data to Liberator when a connection to Liberator is established.



Demo Permissioning DataSource and XML File

From Caplin Trader release 1.2.8 onwards, the reference implementation of Caplin Trader Client is installed with a Permissioning Auth Module and Demo Permissioning DataSource example application. If you have an earlier release of Caplin Trader Client, then you must manually install and configure these components before you start using the Demo Permissioning DataSource (see **Caplin Trader: Installing Permissioning Components** for further information).

4.1 Starting and Stopping the Demo Permissioning DataSource

The Demo Permissioning DataSource is supplied with scripts that you can run to start and stop the example application.

Starting the Demo Permissioning DataSource

To start the Demo Permissioning DataSource, navigate to the *apps/caplin/PermissioningDataSource* directory and run the following command.

```
$ ./start.sh
```

This starts the application, passing the following files as arguments.

- ◆ *conf/Permissions.xml* (permissioning data in XML format)
- ◆ *conf/DataSource.xml* (DataSource configuration file)
- ◆ *conf/Fields.xml* (DataSource field mapping file)

When a connection to Liberator is established, the Demo Permissioning DataSource sends the permissioning data to Liberator.

Stopping the Demo Permissioning DataSource

To stop the Demo Permissioning DataSource, navigate to the *apps/caplin/PermissioningDataSource* directory and run the following command.

```
$ ./stop.sh
```

This stops the application and terminates the connection with Liberator.

4.2 Overview of the Demo Permissioning DataSource

The Demo Permissioning DataSource consists of one interface and two classes.

PermissionsLoader: This interface defines a service that will load permissioning data from a permissioning system.

XMLPermissionsLoader: This class implements the `PermissionsLoader` interface to load permissioning data into the `PermissioningDataSource` from the file *conf/Permissions.xml*.

DemoPermissioningDataSource: This class is initialized with an `XMLPermissionsLoader`. It creates a `PermissioningDataSource` to send the permissioning data to Liberator when a connection to Liberator is established. The class implements the `PermissioningDataSourceListener` interface of the Permissioning DataSource API. The principal methods of the class are summarized below.

- ◆ `main(String[] args)`

Creates the `DemoPermissioningDataSource` using the passed in arguments.

Starts the `DemoPermissioningDataSource`.

- ◆ `start()` Retrieves permissioning data from the permissioning system and initiates a connection to Liberator.

- ◆ `onConnect()` Called by the `PermissioningDataSource` when a Liberator connection is established.

- ◆ `terminate()` Shuts down the `DemoPermissioningDataSource`.

You will find fully commented source code for the Demo Permissioning DataSource in *apps/caplin/kits/permissioning-datasource-`<version>`/example-application* (where `<version>` = version number).

Tip: The `PermissioningDataSourceListener` interface and `PermissioningDataSource` class are described in the **Permissioning DataSource: API Reference**.

5 The Demo Permissioning XML

The Demo Permissioning DataSource gets its permissioning data from an XML file, and then sends that permissioning data to Liberator when a connection to Liberator is established. This part of the document describes the XML-based elements that define the structure and content of this permissioning data.

If you want to experiment with the demo by adding or modifying permissioning data for users, groups, or rules, then you must edit the file *apps/caplin/PermissioningDataSource/conf/Permissions.xml*.

5.1 Technical Assumptions and Restrictions

XML

The XML markup defined here conforms to XML version 1.0 and the XML schema version defined at <http://www.w3.org/2001/XMLSchema>.

5.2 Ordering and Nesting of Tags

Each top level tag is shown below, together with the child tags that it can contain.

Tip: Advanced users may wish to consult the Relax NG Schema (*Permissions.rnc*) for definitive information on the ordering and nesting of tags. This file is supplied with the Permissioning software.

For a description of each tag and its attributes, see the [XML Reference information](#) ²² section.

<permissioning>

This is the outermost tag.

```
<permissioning>
  <rules></rules> (zero or one)
  <users></users> (zero or one)
  <groups></groups> (zero or one)
</permissioning>
```

<rules>

```
<rules>
  <rule></rule> (one or more)
</rules>
```

<users>

```
<users>
  <user></user> (one or more)
</users>
```

<groups>

```
<groups>
  <group></group> (one or more)
</groups>
```

<rule>

```
<rule>
  <fieldMatchCriteria></fieldMatchCriteria> (zero or one)
</rule>
```

<user>

```
<user> (children in any order)
  <subjectMapping /> (zero or one)
  <attributes></attributes> (zero or one)
  <permissionSet></permissionSet> (zero or one)
</user>
```

<group>

```
<group>
  <permissionSet></permissionSet> (zero or one)
  <members></members> (zero or one)
</group>
```

<fieldMatchCriteria>

```
<fieldMatchCriteria>
  <match /> (one or more)
</fieldMatchCriteria>
```

<attributes>

```
<attributes>
  <userAttribute /> (one or more)
</attributes>
```

<permissionSet>

```
<permissionSet>
  <productPermissionSet></productPermissionSet> (one or more)
</permissionSet>
```

<members>

```
<members>
  <userRef /> (zero or more)
  <groupRef /> (zero or more)
</members>
```

<productPermissionSet>

```
<productPermissionSet>  
  <permission /> (one or more)  
</productPermissionSet>
```


5.3 XML Reference Information

The following sections describe the Permissioning XML tags. They are arranged in alphabetical order of tag name.

For each tag the attributes you can use within it are listed and described in a table. The "Req?" column indicates whether the attribute is always required ("Y") or is optional ("N"). If you do not supply an optional attribute within an instance of the tag then the runtime behavior will be according to the default value of the attribute.

<attributes>

<attributes>

A collection of one or more user attributes, with one attribute per child <userAttribute> tag.

Attributes: This tag has no attributes.

<fieldMatchCriteria>

<fieldMatchCriteria>

Contains a list of field match criteria. A rule can have zero or more field match criteria that map RTTP message fields and values. All defined field mappings must be present in the RTTP message, otherwise the rule will not match the message. Individual field mappings are defined using <match>.

Attributes: This tag has no attributes.

<group>

<group>

Defines a single permissioning group. A group can have zero or one <permissionSet> and zero or one <members>. Groups allow product permissions to be applied to the members of the group in an inheritance hierarchy. A user can be a member of more than one group, and groups can be members of other groups.

Attributes:

Name	Type	Default	Req?	Description
name	string	(none)	Y	The name of the group, which must be unique to each group. Other groups and users can become members of this group by referring to the group by this name.

<groupRef>

<groupRef>

Adds a group member to the group (see <group>). Groups can be members of more than one group, but cannot be members of their own or child groups.

Attributes:

Name	Type	Default	Req?	Description
nameRef	string	(none)	Y	The name of the group that you want to add. Only groups that have been defined using the name attribute of the <group> tag can be added to a group. Therefore nameRef must match the name attribute of a <group> tag.

<groups>

<groups>

Contains a list of one or more permissioning groups, with one group per child <group> tag.

Attributes: This tag has no attributes.

<match>

<match>

A child of <fieldMatchCriteria> that defines an individual field mapping for a key/value pair. The rule will only match the RTTP message if the field identified by criteria has the value identified by value.

Attributes:

Name	Type	Default	Req?	Description
criteria	string	(none)	Y	The field to match.
value	string	(none)	Y	The value to match.

<members>

<members>

Defines zero or more members of a group, where each member can be a user (<userRef>) or another group (<groupRef>).

Attributes: This tag has no attributes.

<permission>

<permission>

Defines a single permission. 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 that will restrict the scope of the permission. If you do not define a namespace, then the permission will reside in the default namespace.

Attributes:

Name	Type	Default	Req?	Description
action	string	(none)	Y	The action that the permission applies to. This value should match the action defined by a matching rule (see <rule>).
auth	string	(none)	Y	Whether the action will be allowed or denied. Permitted values are "ALLOW", "DENY", and "NO PERMISSION" (permission neither allowed nor denied).
namespace	string	(none)	N	The namespace in which the permission resides. This value should match the namespace for the action defined by a matching rule (see <rule>). If not defined, the permission will reside in the default namespace.

<permissioning>

<permissioning>

The outermost permissioning tag, with zero or one <rules>, zero or one <users>, and zero or one <groups>.

Attributes: This tag has no attributes.

<permissionSet>

<permissionSet>

Contains a list of one or more product permission sets, with one set per child <productPermissionSet> tag.

Attributes: This tag has no attributes.

<productPermissionSet>

<productPermissionSet>

Contains a list of one or more permissions for a set of products, with one permission per child <permission> tag.

Attributes:

Name	Type	Default	Req?	Description
productSet	string	(none)	Y	A comma delimited string. Each delimited section of the string must identify a single product (typically a product symbol such as "/FX/GBPUSD") or a regular expression that matches multiple products (such as "*USD").

<rule>

<rule>

Defines a single permissioning rule. Every rule must define either an action attribute or an actionRef attribute, but not both.

Attributes:

Name	Type	Default	Req?	Description
action	string	(none)	N	The user must have permission for this action if the rule matches the RTTP message. This attribute can be used to match an RTTP message to a single action, such as "Trade" or "SPOT". If the action attribute is used then the actionRef attribute must not be used, otherwise the XML will not be valid.
actionRef	string	(none)	N	The name of the field in the RTTP message that identifies the action. The user must have permission for this action if the rule matches the RTTP message. This attribute can be used to match the rule when the RTTP message could define one of several alternative actions. An example would be when the value of the TradeType field could be one of SPOT, FORWARD or SWAP. If the actionRef attribute is used then the action attribute must not be used, otherwise the XML will not be valid.
permissionNamespace	string	(none)	N	The namespace in which the user permission for the action must reside. If a namespace is not defined, then the user must have a permission for the action in the default namespace.
productRef	string	(none)	Y	The name of the field in the RTTP message that identifies the product that the user must have a permission to action. The reserved value ALL_PRODUCTS means that the rule will apply to any product.
ruleType	string	(none)	Y	This value must always be WRITE. WRITE rules apply when data is being contributed to Liberator, and READ rules when data is being requested from Liberator. At present a default READ rule is implemented by the Permissioning Auth Module when a user attempts to view data, but in future releases of Caplin Trader it may be possible to define READ rules in XML.
subjectNameMatch	string	(none)	Y	The subject of the RTTP message that will match this rule. The value can be a regular expression. For example "/F." would match "/FT" and "/FI", since the "." metacharacter will match any single character.

<rules>

<rules>

Contains a list of one or more permissioning rules, with one rule per child <rule> tag.

Attributes: This tag has no attributes.

<subjectMapping>

<subjectMapping>

Maps an RTTP message subject to a subject suffix. If the user attempts to VIEW data where the subject of the RTTP message matches subjectPattern, then subjectSuffix will be appended to the subject of the RTTP message before Liberator requests the data from a DataSource. Subject mappings can be used to get pricing data from different pricing tiers, depending on the user that requested the data.

Attributes:

Name	Type	Default	Req?	Description
subjectPattern	string	(none)	Y	A regular expression that will be compared with the subject of the RTTP message. If a match is found, then subjectSuffix will be appended to the subject of the RTTP message.
subjectSuffix	string	(none)	Y	The suffix that will be appended to the subject of the RTTP message.

<user>

<user>

Defines a single user and the user's name and password. A user can have zero or one <permissionSet>, which allows product permissions to be applied to the user; zero or one <subjectMapping>, which allows data to be requested from a pricing tier; and zero or one <attributes>, which map user attribute names to user attribute values.

Attributes:

Name	Type	Default	Req?	Description
name	string	(none)	Y	The user's login name.
password	string	(none)	Y	The user's login password. The reserved value "keymaster" indicates that the Caplin Keymaster single sign-on system will validate the user's password.

<userAttribute>

<userAttribute>

Defines a single user attribute. A user attribute maps an attribute name to an attribute value.

Attributes:

Name	Type	Default	Req?	Description
key	string	(none)	Y	The attribute name or key.
value	string	(none)	Y	The attribute value.

<userRef>

<userRef>

Adds a user member to the group (see <group>). Users can be members of more than one group.

Attributes:

Name	Type	Default	Req?	Description
nameRef	string	(none)	Y	The name of the user that you want to add. Only users that have been defined using the name attribute of the <user> tag can be added to a group. Therefore nameRef must match the name attribute of a <user> tag.


<users>

<users>

Contains a list of one or more users, with one user per child <user> tag. Users can have product permissions applied to them.

Attributes: This tag has no attributes.

6 Further Reading

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

An introduction to permissioning concepts and terms

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

Reference documentation for the Permissioning DataSource API

Reference material for this API can be found in the **Permissioning DataSource: API Reference**.

7 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	<u>Application Programming Interface</u>
Caplin Platform	A suite of software products for on-line financial trading and Web delivery of real-time market data.
Caplin Trader	Caplin Trader is a complete platform and toolkit for building multi-product trading portals. It is built on the Caplin Platform .
Caplin Trader Client	Caplin Trader Client is a Web application written in Ajax that provides a rich trading workstation in a browser.
DataSource	DataSources are software adapters within the Caplin Platform that connect the Platform 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 ^[16] is an example of a Permissioning DataSource application that gets its permissioning data from an XML file.
Group	A logical grouping of zero or more users and other groups, such that each group can be assigned zero or more permissions .
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 Trader .
Permissioning DataSource	A DataSource adapter that acts as the interface between Caplin Trader and your Permissioning System .
Permissioning System	The source of the permissioning data that you want to integrate with Caplin Trader .
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 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	<u>Software Development Kit</u>
User	An end user of Caplin Trader Client .

Index

- A -

Abbreviations, definitions 29
Acronyms, definitions 29
API
 Permissioning DataSource 5
Applying data to a PermissioningDataSource 9
Architecture 4

- C -

committing a transaction 6, 9
creating a Permissioning DataSource 6

- D -

DataSource Adapter 6
 demo permissioning 16
DataSource protocol 4
Demo Permissioning DataSource 16
 overview 18
 Permissioning XML 19
 scripts to start and stop 17
 starting 17
 stopping 17
Demo Permissioning XML 19
 ordering and nesting of tags 19
 reference information 22
 tags and attributes 22

- E -

example application 16
example transactions 10, 11, 12, 13, 14, 15

- G -

Glossary 29
Groups 11

- L -

live permissioning updates 9

- P -

Permissioning DataSource
 demo 16
Permissioning DataSource API
 using 5
permissions
 assigning 10, 11
 changing 12, 13
 changing group members 15
 password setting 12
 removing 13
 subject mapping 14
 User Attributes 14

- R -

Readership 1
real time updates 9

- S -

starting a transaction 6, 9
steps to create an application 6

- T -

tags and attributes 22
Terms, glossary of 29
transaction
 commit 6, 9
transactions 9

- U -

Users 10, 11

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.