CAPLIN

Caplin Liberator 6.0

Administration Guide

September 2012

1 Preface

1.1 What this document contains

This document describes the Liberator and its place in the Caplin real-time data architecture. It includes instructions on how to install and configure the Liberator and details some simple user authorizing and object permissioning modules that are part of the installation.

It also includes a comprehensive listing of configuration options, debug messages, and example configuration files.

1.2 Who should read this document

This document is intended for people who need to install, configure and maintain the Liberator. Administrators are assumed to have a working knowledge of Linux procedures.

1.3 Related documents

- **Caplin Platform Overview**

  A technical overview of the Caplin Platform.

- **Caplin Platform: Guide to User Licensing**

  Describes the user-based licensing scheme used in the Caplin Platform.

- **Keymaster Administration Guide**

  Describes how to configure and operate Caplin's KeyMaster product to provide a secure and reliable user authentication service.

- **DataSource For C Configuration Syntax Reference**

  Describes the syntax of the configuration defined in “Appendix A: Configuration reference”.

- **Caplin DataSource for C API Documentation**

  The reference documentation for the C DataSource API.

- **Caplin Integration Suite for Java API Documentation**

  The reference documentation for the Caplin Trading Integration API for Java.

- **Caplin Trading: Integrating The Caplin Platform With A Trading System**

  Describes how a Caplin Trading Adapter allows you to integrate the Caplin Platform with your existing trading system. It explains trading concepts (such as Trade Models), and how to implement a Trading Adapter,

- **Liberator Authentication C API Reference**

  The API reference documentation for the C-based SDK that allows you to create your own Liberator authentication modules.

- **JavaAuth API Reference**

  The API reference documentation for the JavaAuth SDK that allows you to create your own Liberator authentication modules.

- **Server-side RTTP Logging**

  Describes the server-side logging of RTTP messages between the Liberator and a client communicating over RTTP.

1.4 Typographical conventions

This document uses the following typographical conventions to identify particular elements within the text.

Type	Use
Arial Bold	Function names and methods.
	Other sections and chapters within this document.
Arial Italic	Parameter names and other variables.
Times Italic	File names, folders and directories.
`Courier`	Program output and code examples.
❖	Information bullet point
■	Instruction

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

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (*http://www.openssl.org/*)

This product also includes cryptographic software written by Eric Young (*eay@cryptsoft.com*) and Tim Hudson (*tjh@cryptsoft.com*).

Enterprise Linux is a registered trademark of Red Hat, Inc. in the United States and other countries.

Oracle is a registered trademark of Oracle and/or its affiliates.

Java and *JMX* are trademarks of Oracle and/or its affiliates.

Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.

OS X is a trademark of Apple Inc., registered in the U.S. and other countries.

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

2 Overview

2.1 What is Caplin Liberator?

Caplin Liberator is a financial internet hub that delivers data and messages in real time to and from subscribers over any network. It provides a complete connectivity and subscription management system for streaming market data and trade messages over intranets, extranets and the Internet. It is a core component of the Caplin Platform (for more information about Liberator's role within the Caplin Platform, see the **Caplin Platform Overview** document).

Liberator runs on the Linux® operating system. Development versions are also available for the Microsoft Windows® and Mac OS X® Lion operating systems.

Note: *Caplin Systems does not provide support for production versions of Liberator running on Microsoft Windows or OS X.*

Liberator is capable of handling up to tens of thousands of concurrent users. It handles both HTTP and RTTP traffic (see "What is RTTP?" on page 69), and features a special high performance publishing engine capable of delivering hundreds of thousands of updates per second from a single server.

User permissioning and usage monitoring can be carried out in a variety of ways, for example using Caplin's XML Auth module, which enables programmers to use XML to create their own permissioning structures and control the entitlement of objects held on the Liberator - please refer to "Authentication and entitlement" on page 97 for further details.

The Liberator supports many RTTP data types, including text fields, fixed-format pages and page updates, logical records and news headlines.

Liberator exchanges data and messages with other DataSource applications, such as Caplin Transformer and Integration Adapters. In this document, such applications are called DataSource peers.

Caplin's Platform architecture

Figure 2-1 below shows a simplified implementation diagram and highlights the Liberator.

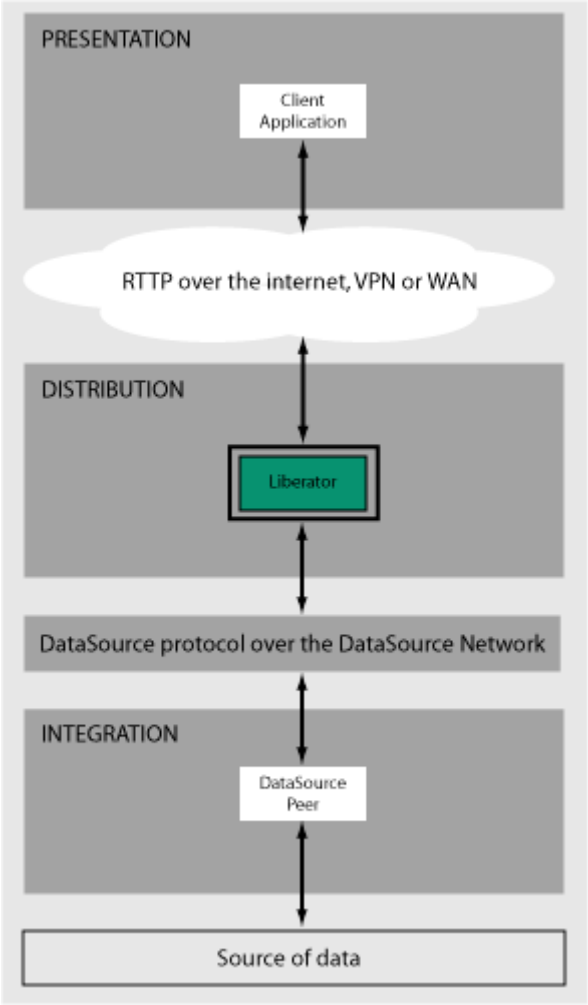


Figure 2-1: Liberator's place in Caplin's architectureInternal architecture

Figure 2-2 shows the components of an RTTP data source and Liberator and how they fit together. Contributing applications send market data to Liberator (see the section entitled “DataSource applications and peers” on page 14). Liberator then aggregates the data and publishes it over the Internet using RTTP, where it can be integrated into web pages or custom applications.

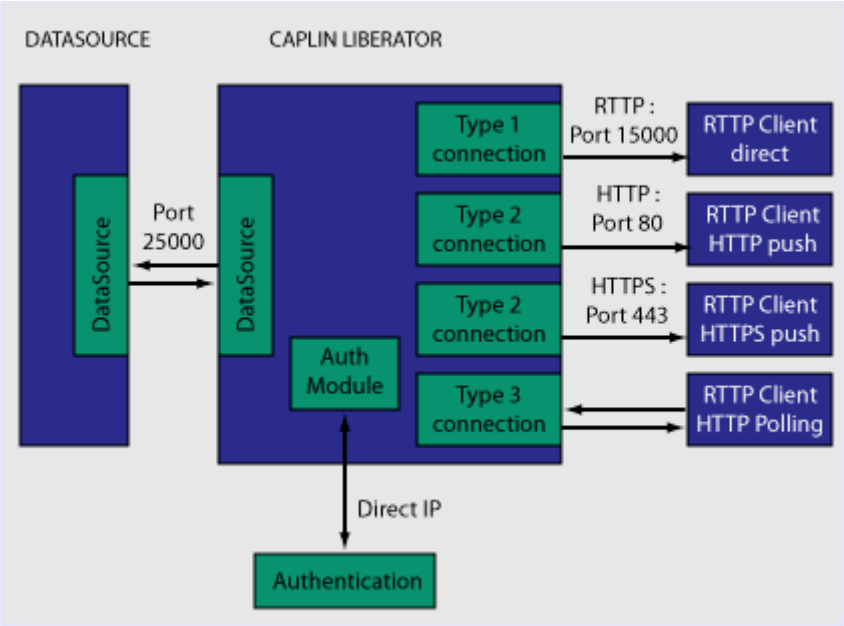


Figure 2-2: Liberator internal architecture

Figure 2-3 below shows a detailed illustration of Caplin's Platform architecture, including all the most common products, showing Liberator's position in the larger data distribution picture.

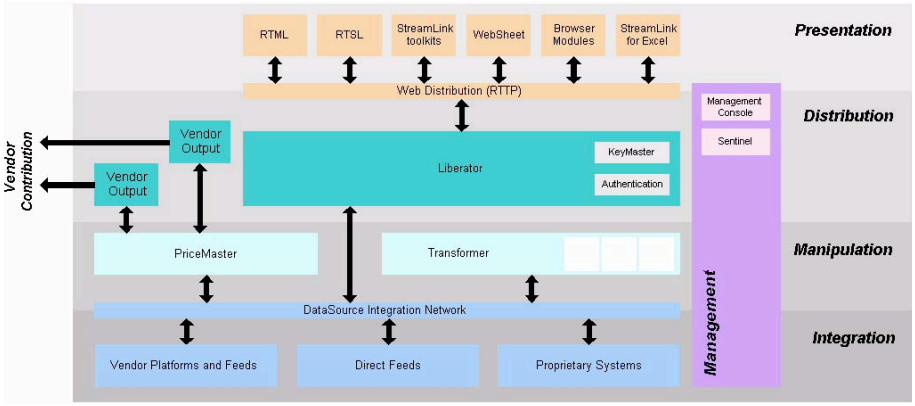


Figure 2-3: Caplin's architecture

2.2 What's new in Liberator version 6.0?

StreamLink JS

Liberator now ships with StreamLink JS, which is a version of StreamLink optimized for use with HTML5 browsers. The Liberator kit contains some StreamLink JS examples that use data from a demonstration Integration Adapter.

Improved validation of configuration

Liberator now fails to startup when a configuration error is identified, rather than continuing to run. This helps to ensure that your deployment. will run correctly.

Mac OS X supported for development

You can install and run Liberator, for development purposes only, on the Mac OS X Lion operating system.

Note: *Caplin Systems does not provide support for production versions of Liberator running on OS X.*

Improved CORS support

Liberator has improved support for Cross-origin resource sharing (CORS). This allows simpler deployment strategies and improves web browser security.

2.3 Architectural examples

For examples of how Liberator is used within the Caplin Platform, see the **Caplin Platform Overview** document.

2.4 Functions and features of the Liberator

Operational features

Publishing real-time data

Liberator reliably publishes data over the internet in real-time with very low latency. It is capable of publishing to tens of thousands of simultaneous users, employing a protocol called RTTP which tunnels through firewalls and proxy servers without their needing to be modified. It is also possible for users to contribute data back through Liberator to the source of the data.

- For details about how RTTP works see “About the data” on page 69.

Clustering

Each Liberator can be configured as a node within a cluster, in order to share information about the number of licenses, users logged on, and information about data and subscriptions.

- For details on how to configure the Liberator see “Running multiple Liberators from the same install location” on page 32.

Global caching for clusters

When Liberator is configured as a cluster it can be set up to share information about users subscriptions. This allows each Liberator to request the object itself or know the best DataSource peer to request it from.

- For details on how to configure global caching, see “Running multiple Liberators from the same install location” on page 32.

Multi-threading

Liberator is a multi-threaded application. It uses one thread per DataSource connection, and a configurable number of threads on the client session side.

- For details on how to configure optimal threading, see “Improving performance using threads” on page 151.

Per-object configurable throttling

This allows directories (or specific objects) to be given throttle times. This is configured through the add-object interface.

- For details on how to configure object throttling, see “Using throttling” on page 92.

Reconnecting clients

Update messages are stored in an output queue which can be resent when reconnecting to a client. This reduces the possibility that the server will not have any messages that the client missed while disconnected. The size of the output queue is configurable.

For details on how to configure output queues, see “Configuring buffering” on page 95.

Permissioning and security features

Authentication and permissioning modules

Liberator supports a modular system for handling authentication and authorization. Each "Auth" module allows users to be authenticated, objects to have permissions loaded, a user's read and write permissions for an object to be checked and object name mappings to be performed.

- For details on how to use Auth modules, see “Authentication and entitlement” on page 97.

Permissions objects

DataSource peers can determine user access to objects on the Liberator by sending the Liberator permissions objects. A custom Liberator auth module is responsible for interpreting updates to permissions objects and modifying access permissions accordingly.

Client applications can also subscribe to particular permissions objects. When the Liberator’s custom auth module receives updates to a permission object, it will pass the updates on, through the standard update mechanism, to any clients that have subscribed to the object. A client application can use the updated permission information to modify the way the application behaves.

- For details see “External authorization using permissions objects” on page 103.

Content-based permissioning

Content-based permissioning works by allowing users to see an object only if the object contains a certain value in one of its fields. Your Liberator installation includes an Auth Module which uses XML to handle content permissioning information, or you can create your own modules using the associated development kit.

- For details on how to permission objects based on content, see the companion documents **XML Auth Module User Guide** or **Liberator Auth Module Developer's Guide**.

HTTP authentication using Auth Modules

You can configure different HTTP directories with different realms for different users, and perform user authentication to allow access to the directory with an Auth Module.

For details on how to authenticate HTTP directories with an Auth Module, see “add-authdir” on page 176.

KeyMaster Integration

Liberator provides functionality to allow Auth Modules to use a KeyMaster generated encrypted signature as a password. This allows a client application to generate a signature based using a private key, which will then be verified by the Liberator to authenticate the user.

For details on KeyMaster, see the **KeyMaster Administration Guide**.

HTTP headers for authentication

A Liberator Auth Module has access to both the Cookie and the Authorization HTTP headers. This allows the Auth Module to work alongside or on top of an existing authentication system which utilises cookies or HTTP Authentication.

- For details on how to access HTTP headers from within a C Auth Module, see the **Liberator Authentication C API Reference** (which shows some example code in *demoauth.c*).

Note: *HTTP headers cannot be accessed from the JavaAuth API.*

The Liberator web site

The Liberator installation includes a local website, which serve as an introduction to the Liberator and enable you to:

- ❖ monitor the usage of the Liberator, including the number of client sessions connected, and information about its DataSource peers
- ❖ view or download documentation for the Liberator and associated components
- ❖ view demonstration applications written using the StreamLink JS API

- ■ To open the Liberator web pages, point your browser at http://<hostname>:<port number> where <hostname> is the host name or IP address of the machine you have installed the Liberator on. For example *http://liberator:8080*.

For more information on the contents of the Liberator web site, see “Liberator status web page” on page 135 and “Liberator demonstrations” on page 161

Restricting data

Liberator enables you to offer users a restricted RTTP service in the following forms.

Delayed data

Delayed delivery of data can be configured for any non-active DataSource peer feeding the Liberator.

For more information regarding delaying data please refer to the supporting documentation about DataSource.

Throttled data

Liberator can throttle different objects at different levels. Users can be mapped to these throttled objects seamlessly.

The timing of throttling in this way is per object, so if a user is looking at more than one object they will not all update at the same time. This can have a big impact on the loading of the Liberator, as it will be sending fewer updates at any given moment.

For more details on throttling objects, see “Using throttling” on page 92.

2.5 Liberator's data sources

DataSource applications and peers

Liberator is capable of exchanging data and messages with any application that uses the DataSource protocol, a protocol that enables most Caplin and RTTP-related products to communicate with each other. Such an application is called a DataSource application, and it is written using a DataSource API. Liberator is itself a DataSource application, because it too uses the DataSource protocol.

In this document, DataSource applications that are connected to a Liberator are called DataSource peers. Liberator's DataSource peers include Caplin Transformer and Integration Adapters.

You can use the DataSource SDK to write your own DataSource application (for example, an Integration Adapter) to connect to other DataSource applications. A DataSource-enabled application can contribute any logical record, page update or other type of data by using the DataSource API.

The DataSource SDKs available include:

- Caplin Integration Suite for Java, which includes the Java DataSource API
- DataSource SDK for C (Linux, Windows)

The development kits include comprehensive sets of sample applications and demonstration files to illustrate the use of all aspects of DataSource functionality.

DataSource features

Support for SSL data sources

The Liberator is capable of communicating with its data sources over SSL, providing an encrypted channel over which the data sources can publish their data. For information on how to configure DataSource connections to use SSL, see "Making SSL connections with DataSource peers" on page 123 and the ***datasrc-sslport*** configuration option in "Connecting to DataSource peers" on page 107.

Active data sources

An active data source is one that will keep track of which objects have been requested and send updates for those objects only. This improves performance by reducing network bandwidth requirements.

For details on how to use active data sources, see "Data services" on page 113.

Data services

This allows you to define which data source or set of data sources an active request for an object will be sent to. Regular expressions are used to match object names which are then sent to the relevant data source.

Each mapping can have many data sources defined; such a group of data sources is regarded as one data source. For each request directed to the group, the data source selected to service the request is the one with the smallest number of existing subscriptions – this achieves load balancing.

- For details on how to implement active data services, see page 113.

Auto Replay

The Liberator's Auto Replay capability means that previously-sent data can be reprocessed by stepping through its log files and replaying the data on startup.

Auto Replay is useful following a period when Liberator was down, as replaying data can return you to the state immediately before the Liberator shutdown. Auto Replay is not necessary if you are using active sources, as the data will simply be requested again.

- For details on how to implement auto replay, see “Replaying data from peers into Liberator” on page 120.

Message queues

If a data source loses its connection to the Liberator, messages will be queued until the connection can be reestablished. The queue is flushed when a reconnection is successful. The length of the queue is configurable on a per-peer basis.

- For details on how to configure message queues, see “datasrc-rerequest-timeout” on page 211.

UDP command interface

The Liberator now includes a UDP command interface that enables you to send UDP messages from a utility to Liberator in order to reset peer connections after failover, and change the verbosity of log messages.

- For details of the UDP command interface, see “UDP commands” on page 144.

- For details on how to reset connections with UDP commands, see “Reconnecting peers using the UDP interface” on page 112.

- For details on how to adjust logging levels with UDP commands, see “Debugging” on page 146.

Data features

Object sub-type mappings

Wildcard mappings can be configured to determine the sub-type of an object.

- For details on configuring object type mappings, see “add-type-mapping” on page 197.

Configurable startup objects

Any number of objects can be configured to be created when Liberator starts. This configuration includes name, type, flags and source.

- For details on how to configure startup objects, see add-object in “Configuring objects” on page 84, and “add-object” on page 191.

Purging of objects

Liberator can be configured to delete data held in cache at any time of day, on a per-object basis.

- For details on how to configure Liberator purging, see “add-object” on page 191. The purging of news headlines is set using the “news-purge-days” on page 241 and “news-purge-time” on page 241.

News headlines and stories

Liberator can handle news objects, including news headlines and associated stories. Liberator offers complex filtering of headlines based on either headline text or codes.

- For details on how to configure Liberator to handle news, see “Handling requests for news headlines” on page 91. For a description of RTTP news objects, see “News headline and news story” on page 71.

Type 2 and 3 record data

Liberator holds Type 2 (Level 2 quote data) and Type 3 (historic updates) for specially configured fields (see “About RTTP fields” on page 73). Data sources can control this store of data by sending flags to clear the cache or to filter out some entries based on the value of a particular field.

- For details on how Liberator handles these data types, see “Identifying the fields clients can request” on page 88.

Record filtering

Liberator can accept requests for record objects with a user-defined filter—only updates matching the expression given by the user will be sent to that user. These expressions are based around the field values in the update and can contain most standard logical and relational operators (NOT, OR, AND, equals, greater than, etc). For example, a user might specify that they only want to receive updates when the Volume field is greater than a certain amount.

- Record filtering is implemented by client applications. Please refer to relevant documentation for details.

Object name mapping

Liberator allows the configuration of object name mappings. Object mapping changes the internal name of an object when a user requests it. These mappings are global, but you can insert the username as part of the map. The user will be unaware of the new name which is only known to Liberator and its DataSource peers.

This functionality was previously only available within Auth Modules, but has been extended so that Liberator can perform the mappings where necessary.

- For details on how to configure object name mappings, see “Configuring objects” on page 84.

News headlines

As well as serving up cached headlines previously broadcast to Liberator, Liberator can actively collect historic news using a suitably-configured DataSource peer. This enables clients to request news from a certain date without being limited by Liberator's cache size.

DataSource documentation

For detailed information about implementing custom Integration Adapters and other DataSource applications, please refer to the relevant DataSource documentation:

- **Caplin Integration Suite for Java API Documentation**
- **Caplin DataSource for C API Documentation**
- To implement a Trading Adapter, see the document **Caplin Trading: Integrating The Caplin Platform With A Trading System**.
- For an example implementation of a Permissioning Adapter, see the Caplin Integration Suite.

3 Quick start

To quickly install Liberator for evaluation and development purposes, follow the instructions in the *README.txt* file supplied with the Liberator installation kit.

4 Installing and running Liberator

The following sections explain how to install and run Caplin Liberator in production environments.

4.1 Installing Liberator

Introduction

The method of installing Liberator described here allows for easy upgrades by separating the standard kit from your custom configuration.

Note: *If you are using the Caplin Deployment Framework, you can you can install Liberator by deplying it to the Framework, and you do not need to follow the instructions given here.*

*For more information about the Caplin Deployment Framework, see the documents **Caplin Platform:Deployment Framework Overview**, and **Caplin Platform: How To Use The Deployment Framework**. The latter document explains in detail how to install the Deployment Framework and use it to deploy core Caplin Platform components, such as Liberator.*

Conventions and Assumptions

In the following procedure, symbolic links are used to point to physical files. This allows multiple configurations of the software to be used at the same time and also means that changing from one version to another is fast and simple. The usage of the link command is:

```
ln -s TargetFileName AliasFileName
```

or

```
ln -s TargetDirectoryName AliasDirectoryName
```

The `-s` option makes the link a symbolic one rather than a hard one. The `TargetFileName` is the name of the file that is to be linked to and the `AliasFileName` is the name by which the file should be known.

This guide assumes that an appropriate license has been requested from Caplin to allow usage of multiple Liberators (if required) or to allow the Liberator to run for more than the default 30 minutes. To request a license, contact Caplin Support (support@caplin.com).

Note: *For more information about licensing and how Liberator manages licenses, see the document **Caplin Platform: Guide to User Licensing**.*

It will also be assumed that the Liberator will be installed to */apps/caplin*. In the rest of this document, this path will be referred to as *$INSTALL_DIR*. All paths given below are relative to */apps/caplin*. For example, *kits* actually refers to */apps/caplin/kits*.

Step-by-Step Standard Install

1. Create the following directories:

kits/liberator	This will contain the Liberator kit
liberator1	This will contain the symbolic links to the kit

   To create the directories inside *$INSTALL_DIR*, enter the following from inside that directory:

```
$ mkdir –p kits/liberator
$ mkdir –p liberator1
```

   Note: *The* `-p` *option creates parent directories if necessary.*

2. Copy the liberator kit into the *kits/liberator* directory. The example below copies the kit from */tmp* into the *kits/liberator* directory:

```
$ cp /tmp/Liberator-6.0.0-1-i686-pc-linux-gnu.tar.gz  kits/liberator
```

   Note: *Your kit will probably have a slightly different name.*

3. The Liberator kit is a compressed tar file that will need to be uncompressed and untarred. From the *kits/liberator* directory type:

   Linux

```
$ tar xzf Liberator-6.0.0-1-i686-pc-linux-gnu.tar.gz
```

4. Now, whilst still within the *kits/liberator* directory, create the symbolic link which will make upgrading to new versions easier. Enter the following to create the link:

```
$ ln -s Liberator-6.0.0-1 latest
```

Browsing to *latest*, should reveal the following directory structure:

Folder	Description
bin	Contains binary programs for the Liberator.
doc	Contains Liberator documents and example programs.
etc	Contains start-up scripts and configuration files for the Liberator.
htdocs	Contains the Liberator webpages.
include	Contains auth module development files and contains header files used to create custom auth modules.
lib	Contains auth libraries, modules and third party libraries.
users	Contains Liberator user statistics.
var	This directory will contain all Liberator log files.

5. Now set up an instance of Liberator by moving to the */liberator1* directory and creating symbolic links as shown below:

```
$ ln –s ../kits/liberator/latest/bin bin
$ ln –s ../kits/liberator/latest/doc doc
$ ln –s ../kits/liberator/latest/htdocs htdocs
$ ln –s ../kits/liberator/latest/lib lib
$ ln –s ../kits/liberator/latest/include include
$ cp –r ../kits/liberator/latest/etc etc
$ mkdir users
$ mkdir var
```

Note: *The etc directory does not have a symbolic link as it contains the configuration files that should not be overwritten by an upgrade.*

Note: *The normal directory for containing the Liberator's start-up scripts and configuration files is $INSTALL_DIR/etc; this is where the example configuration files in the install kit are located. However when the Liberator starts up it will first look for configuration files in its root directory $INSTALL_DIR. If it finds a required configuration file in $INSTALL_DIR*

(for example the main configuration file rttpd.conf), it will use that in preference to a file of the same name in the etc directory.

It is recommended that you do not put configuration files in $INSTALL_DIR; always keep them in the etc directory. In particular. avoid keeping old or back up copies of configuration files in $INSTALL_DIR (unless you rename them), since Liberator would then cease to respond to changes in the "live" versions of these files located in the etc directory.

Upgrading Liberator

Periodically new versions of Liberator are released. The release can be due to feature enhancements or bug fixes. Using the setup detailed above greatly simplifies the upgrade process. Repeat steps 2 to 4 above with the new Liberator kit to complete the upgrade. Once this has been completed, all instances of the Liberator will be upgraded to the new version.

4.2 Starting Liberator

Introduction

With the instance set up, configure the Liberator and the demonstration Integration Adapter that ships with the Liberator to confirm that the Liberator is working and can successfully connect to a data source.

Liberator receives data from Integration Adapters. In this installation we are going to connect to an example Integration Adapter called *demosrc* to prove the system is operating correctly. It is usually a good idea to perform this step in any install, but you could connect to a real source if you have one available.

This demo source can later be deleted and is not necessary for running the Liberator.

Step-By-Step Start-up

1. Liberator is started with the start-up script *rttpd* which can be found in the *etc* directory.

2. The Liberator has a number of ports that need to be defined, as shown in the following table::

Port configuration item	Setting when Liberator is installed	Use
http-port	`8080`	This is the port that listens for HTTP requests. The Liberator has an inbuilt webserver which hosts pages to check status info etc. See “Enabling clients to connect using RTTP (over HTTP)” on page 76.
udp-port	None	This is the port that listens for UDP messages. UDP messages can be used to issue commands to the Liberator while it is running. It is not defined when Liberator is installed, which means that, by default, Liberator does not listen for UDP commands. See “UDP commands” on page 144.
direct-port	`15000`	This is the port that listens for direct (type1) RTTP connections. See “Enabling clients to connect using RTTP (direct connection)” on page 82.
datasrc-port	`25000`	This is the port that listens for connections from DataSource peers. See “Connecting to DataSource peers” on page 107.
https-port	`8443`	This is the port that listens for HTTPS connections. This port is only used if HTTPS is enabled in the Liberator configuration. See “Enabling clients to connect using HTTPS” on page 77.

In a production environment ***http-port*** and ***https-port*** would be set to `80` and `443` respectively. This is to allow this traffic to pass through an unmodified firewall and be accessible to external users. All the port-related configuration items can be modified and set according to your requirements. An example configuration might be the following:

http-port	`80`
udp-port	`12000`
direct-port	`15000`
datasrc-port	`25015`
https-port	`443`

3. Now configure the Demo data source configuration file (*demosrc.conf*) to connect to the Liberator. To do this we must first edit the configuration in the Liberator configuration file (*rttpd.conf*) as shown below:

```
datasrc-port           25015
add-peer
      remote-id        1
      remote-name      demosrc
      label            demosrc
end-peer
```

Now edit the *demosrc.conf* as shown in the example below:

```
datasrc-id       1
add-peer
      port       25015
      ...
end-peer
```

The datasrc-port (25015) set in *rttpd.conf* must be the same as port specified in *demosrc.conf*. Also note that the remote-id (1) specified in the add-peer section for the demo data source in *rttpd.conf* must be the same as the datasrc-id specified in *demosrc.conf*.

4. To run the Liberator for longer than the default 30 minutes install the new license you received from Caplin Support. Perform the following steps to install it:

    1. Ensure the Liberator is not running. (Please refer to step 8 below for how to stop the Liberator).

    2. Copy the new license to the *etc* directory.

    3. Rename the license file to *license-rttpd.conf*.

    4. Empty the contents of the *users* directory.

Note: *For more information about licensing and how Liberator manages licenses, see the document* ***Caplin Platform: Guide to User Licensing****.*

5. This is the basic configuration required to install the Liberator. Start the Liberator and demo data source by entering the following from inside the *etc* directory:

```
$ ./rttpd start
$ ./demosrc start
```

    The order in which the Liberator and data source is started is not important, but it is good practice to start the Liberator first.

    Automatic restart

    Liberator can be configured to attempt to restart after an unexpected shutdown.

    Edit the file etc/rttpd and change the value of LIBERATOR_START from start-noloop to start-loop.

6. To ensure the Liberator has started and connected to the demo data source open up Internet Explorer and browse to the status page:

   http://hostname:8080 8080 is the http-port specified in the Liberator configuration file.

   If the Liberator home page appears then the Liberator has started correctly. To check the demo data source has connected click ‘Status’. You will be prompted for the following credentials:

   ```
   Username:       admin

   Password:       admin
   ```

   This username and password is configurable in the Liberator configuration file.

   The following two pictures show the relevant parts of an example status page:

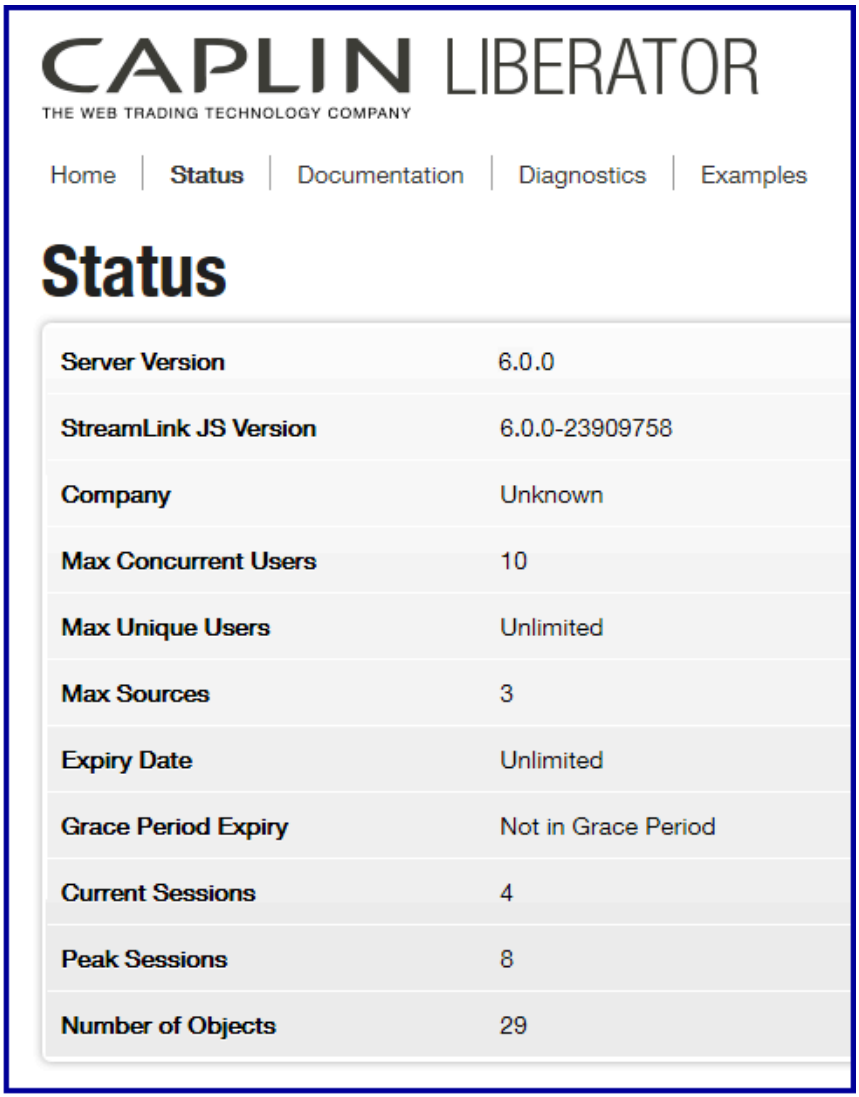


Figure 4-1: Example Liberator status page (top part) after initial step-by-step set up

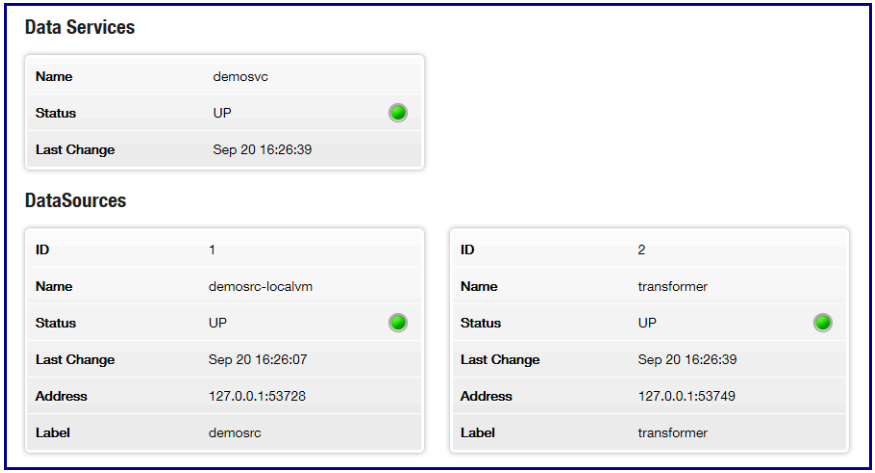


Figure 4-2: Example Liberator status page (middle part) after initial step-by-step set up

The important part to note is the DataSources section, which indicates whether demosrc has status UP or DOWN. If the status is DOWN, the usual cause is that incorrect ports are specified in *demosrc.conf*.

Note: *For an explanation of the other items displayed on the status page, see "Liberator status web page" on page 135.*

7. The Object Browser Tool, which ships with the Liberator, can be used to request some data from the demo Integration Adapter. Browse to Examples -> Object Browsing Tool and request `/EXAMPLES/PRICING/TYPE1`.

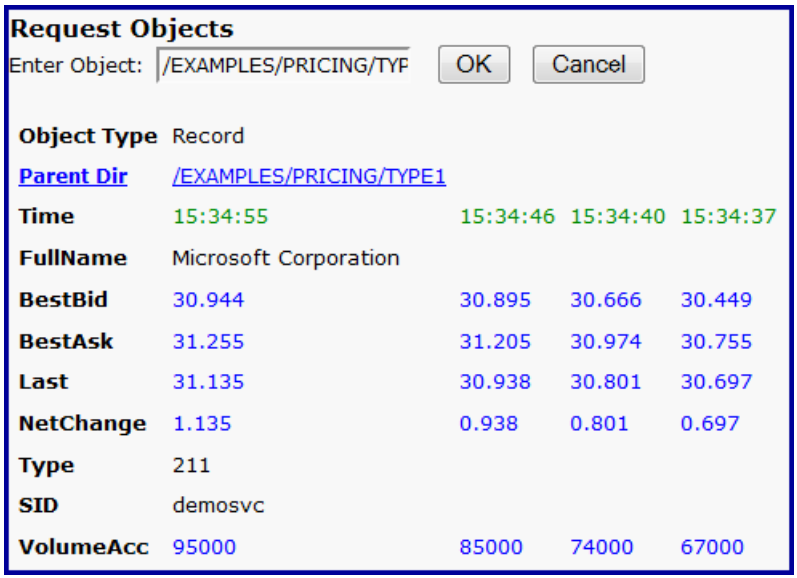


Figure 4-3: Object Browser Tool from the Examples page

You can also use the Liberator Explorer tool to browse objects; see “Liberator Explorer” on page 141.

A full list of available symbols is available within the demo data source configuration file (*demosrc.conf*).

8. To stop the Liberator and demo data source enter the following commands from inside the *etc* directory:

```
$ ./demosrc stop
$ ./rttpd stop
```

4.3 About your Liberator license

Liberator license details are contained in a system file called *licence-rttpd.conf*. This file can be found in the etc directory.

The file can contain several licenses which simplifies deployment by enabling one file to control several installations.

Note: *The default license causes Liberator to shut down after 30 minutes of operation. Contact Caplin for a full license to replace this.*

If you need to change, upgrade or renew your license agreement, a new version of this file will be made available on the Caplin Client Portal web site (https://support.caplin.com) for you to download and install.

For more information about licensing and how Liberator manages licenses, see the document **Caplin Platform: Guide to User Licensing**.

4.4 Full secure set up on Linux

If you want to use port 80, 443 or any other restricted port, or if your license contains a MAC address entry, you will need to enter the following to unpack the Liberator kit as a normal user, then make the binaries start as root but run as an unprivileged user (cap-run).

Note: *Locations may vary.*

This is the preferred method for a production install. It means that only the log files are writable by the user which the process runs as, providing additional security.

1. Login as root.

2. Create two users.

```
# /usr/sbin/useradd -d /opt/caplin caplin
# /usr/sbin/useradd -d /opt/caplin -s /usr/bin/false cap-run

# mkdir -p /opt/caplin
# chown caplin /opt/caplin

# passwd caplin [enter a password twice as prompted]
```

3. Login as caplin.

4. Unpack Liberator

```
$ cd /opt/caplin
$ zcat /tmp/Liberator-6.0.0-1.tar.Z | tar xf -
$ ln -s Liberator-6.0.0-1 Liberator
```

5. Login as root.

6. Configure runtime user.

```
# cd /opt/caplin/Liberator
# chown cap-run var users
# vi etc/rttpd.conf [Uncomment the line runtime-user cap-run, save
and exit]
```

7. Configure ports and set any other required parameters.

8. Log in as root.

9. Start Liberator as root.

```
$ etc/rttpd start
```

Liberator will start as root to allow it to open restricted ports, it will then change to run as 'cap-run' which only has access to write to the *var* and *users* directories. This provides a secure sandbox for the application to run in.

4.5 Running multiple Liberators from the same install location

It is possible to run multiple instances of the Liberator from the same installation. You will need a license which allows this, and the Liberators will have to use different interfaces or ports in their configuration. This can be done using the standard Liberator startup script:

1. Create two links to the startup script *etc/rttpd*, eg

```
$ ln -s rttpd rttpd-one
$ ln -s rttpd rttpd-two
```

2. Create separate configuration files, eg

```
$ cp rttpd.conf rttpd-one.conf
$ cp rttpd.conf rttpd-two.conf
```

3. Make relevant changes to the configuration files. The things that will or may need changing are either ports or interfaces for the following:

```
http
direct
datasrc
udp
cluster
```

4. Using the new startup scripts the revelant config files will be used, eg

```
$ ./etc/rttpd-one start
$ ./etc/rttpd-two start
```

Using this method the same binary (*bin/rttpd*) will be used, but you must use the convention of using the binary name as a prefix in the startup script links and the configuration files. It would be possible to use a name other than *rttpd*, but it would require you to rename or make links to the binary as well. In each case the startup script and the binary can be links to the original file, but the config files would need to be copies to allow changes to be made. The following table shows examples as an aid to understanding the startup script.

Startup Script	Binary it will start	Config file it will read
etc/rttpd	*bin/rttpd*	*etc/rttpd.conf*
etc/rttpd-one	*bin/rttpd*	*etc/rttpd-one.conf*
etc/lib1	*bin/lib1*	*etc/lib1.conf*
etc/lib-one	*bin/lib*	*etc/lib-one.conf*

4.6 Clustering and intelligent source routing

A cluster enables a group of Liberators to act as one, in order to monitor license use and numbers of users logged on. Clients can also contribute data to a cluster, for example when using the chat facility. You can configure the cluster to use a global cache, which means on failover each clustered Liberator can provide data from the same cache without having to rerequest it from the data source.

- Use the following parameters in the configuration file *rttpd.conf* to enable the clustering of multiple Liberators.

cluster-index	The index number of this cluster node. This states which of the add-cluster-node sections this node represents.
cluster-cache-request-objects	When this option is set to true all Liberators in a cluster will request an object when one of the Liberators requests it.
cluster-cache-source-routing	When this option is set to true the Liberators will share information about which Integration Adapter the object was requested from.
add-cluster-node	Identifies all the Liberators in the cluster.

- Make sure each Liberator identifies every node in the cluster, including itself. This list of nodes must be in the same order in each Liberator's configuration file.

- Make sure each Liberator in the cluster has a different cluster-index in its configuration file. Index numbers must start at 0 corresponding to the order of the ‘add-cluster-node’ entries.

Intelligent Source Routing

Enabling cluster cache source routing allows the other Liberators in the cluster to request the object from the same Integration Adapter. This can have two advantages, firstly it minimises the load on the Integration Adapters as they are not all serving up the same objects. and secondly it minimises the bandwidth used on the Integration Adapter to Liberator network as each update is only sent by a single Integration Adapter. This can be a significant advantage if the Integration Adapters are connected to the Liberators over a WAN.

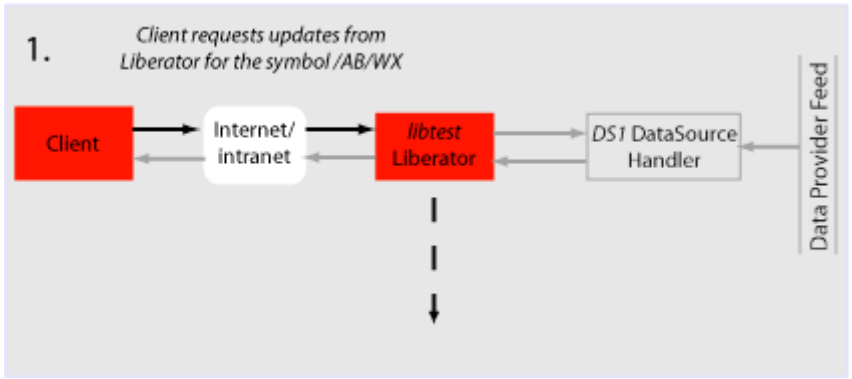
For cluster cache source routing to work, all Liberators in the cluster must be configured in a compatible way. The source routing works based on the labels given to DataSource peers (see “add-peer” on page 211 and “add-data-service” on page 224), so each Liberator must use the same labels for the relevant Integration Adapters.

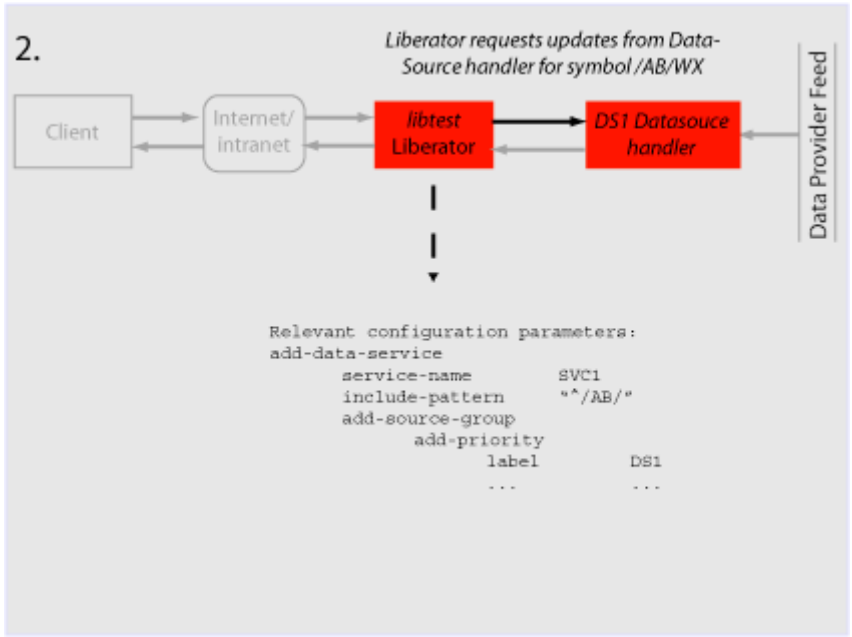
4.7 Step-by-step examples

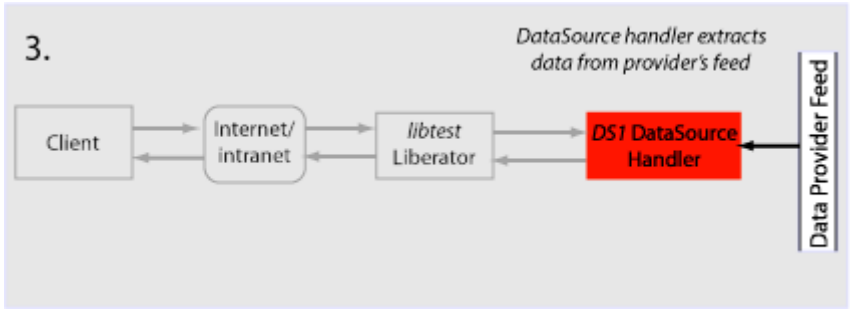
The following pages contain diagrammatic step-by-step examples of scenarios in which Liberator might be used and how it processes client requests for real-time data. Each example lists the major configuration options that must be set in the configuration file *rttpd.conf* in order to achieve the illustrated functionality.

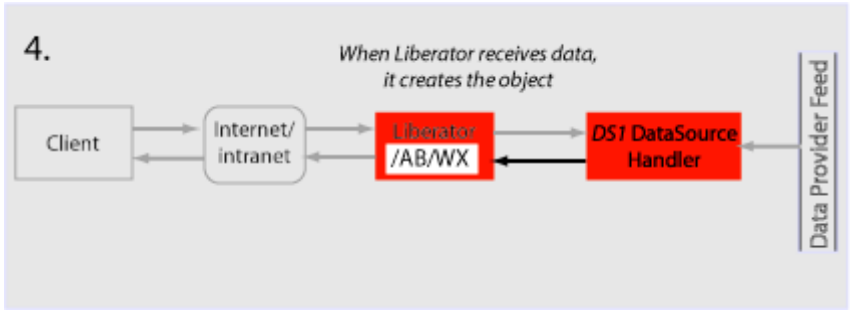
The examples in this section show how Liberator fits into a real-time system with the following functionality:

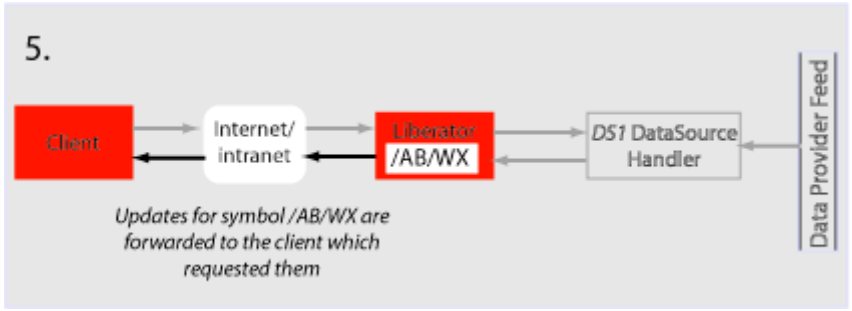
Basic active request

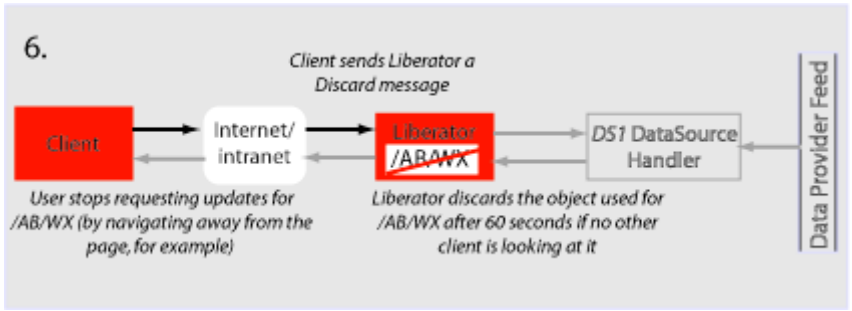


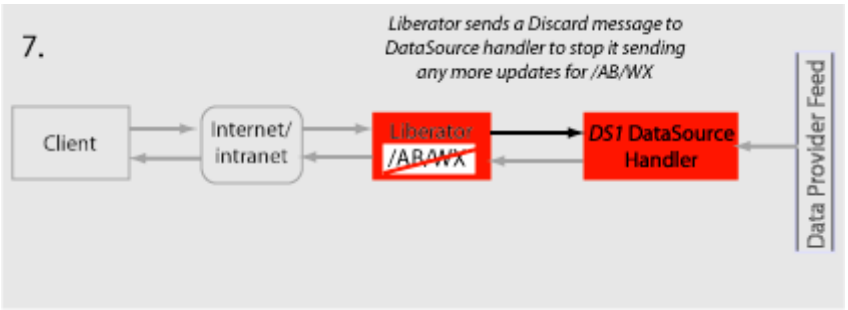




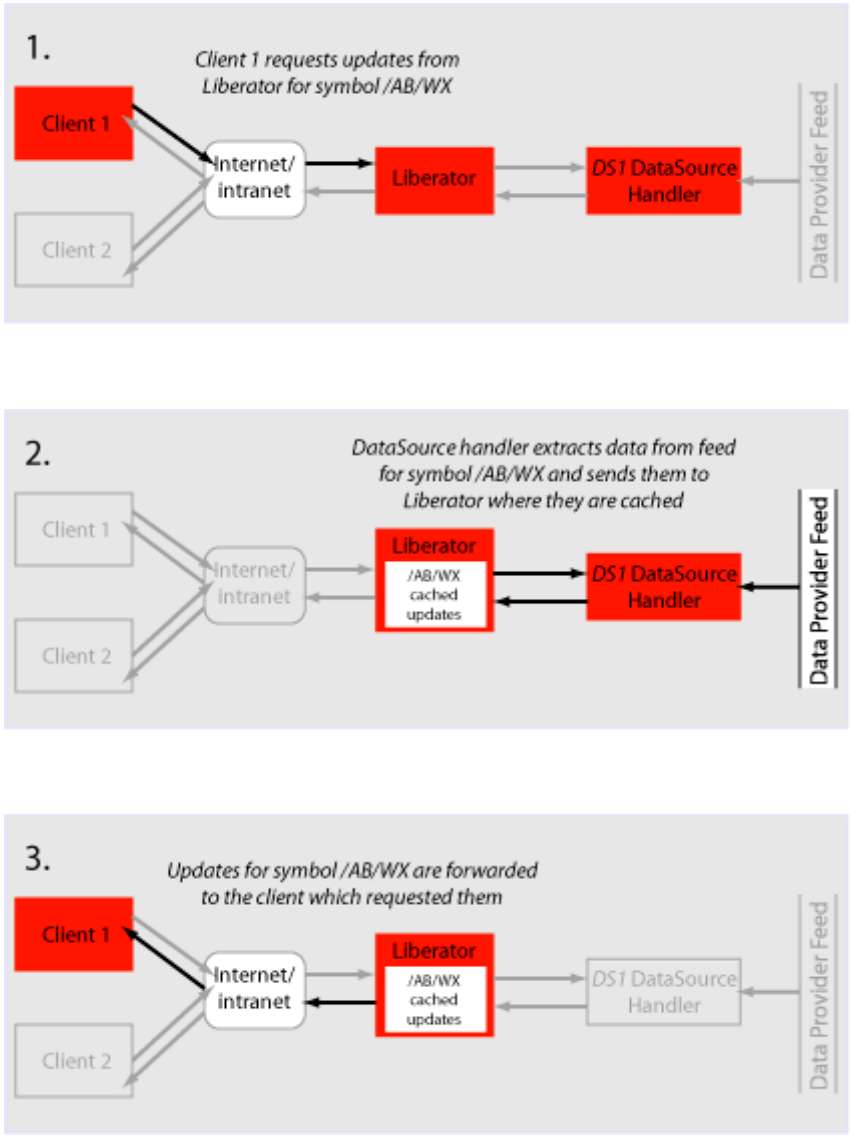


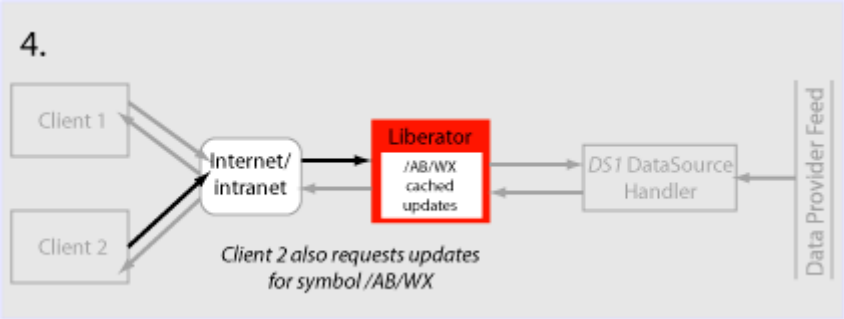


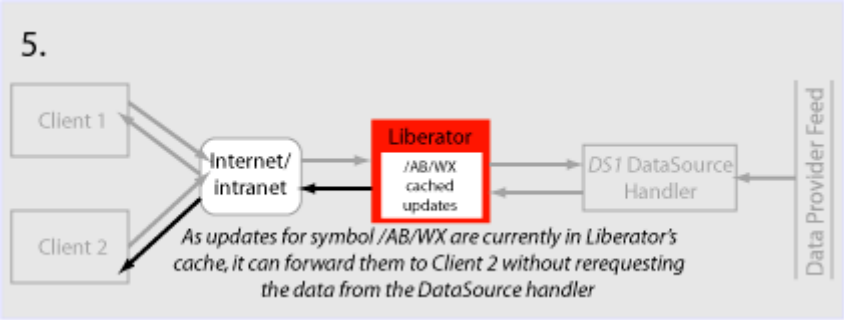




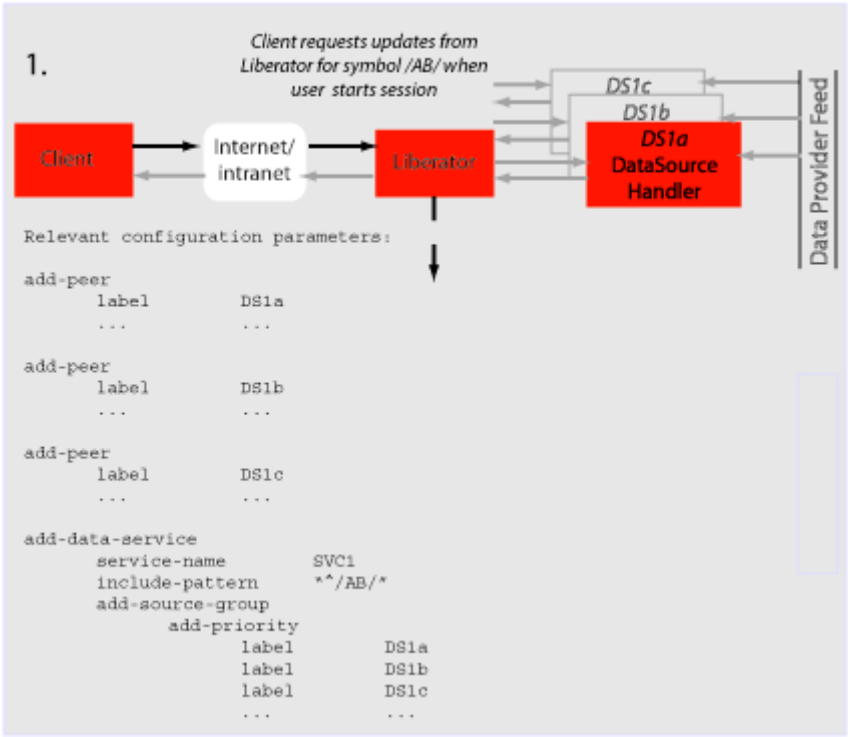
Two clients actively request same data

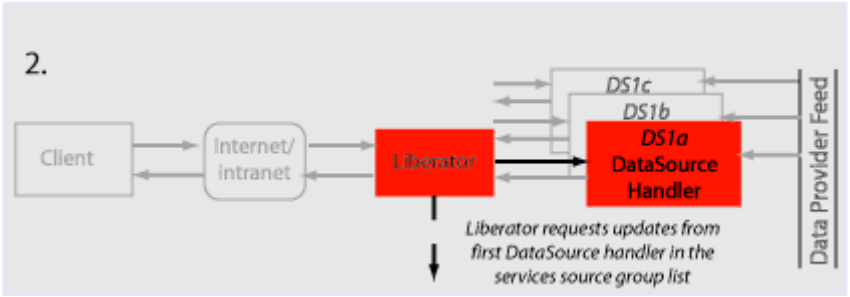


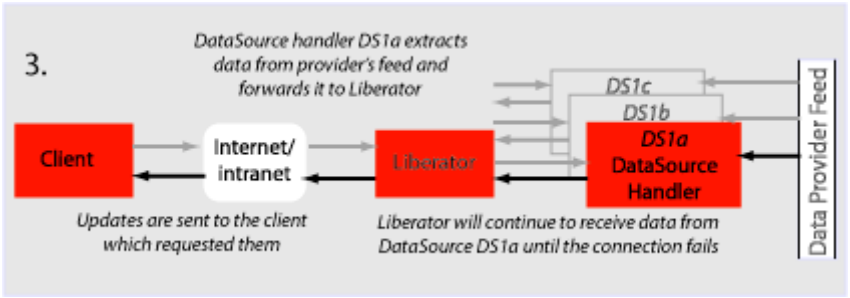


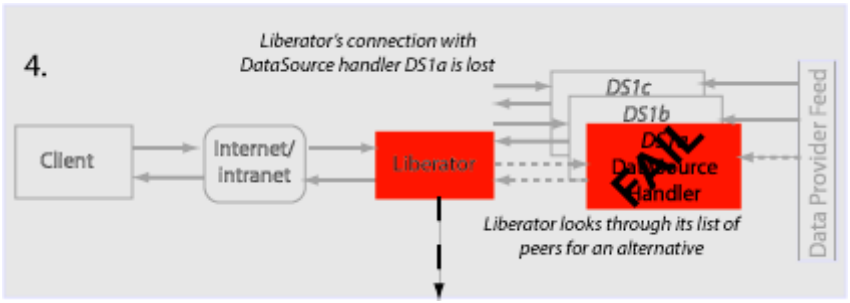


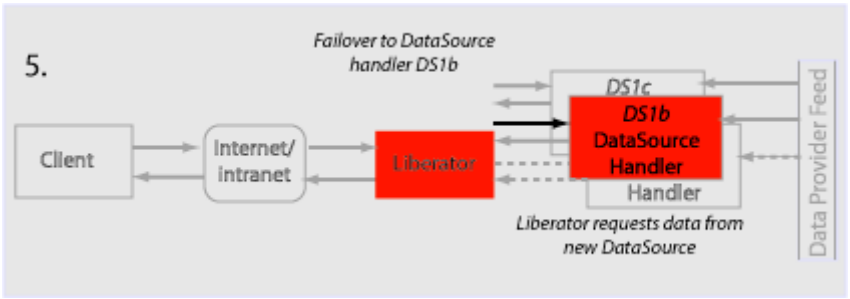
Active request with peer ('DataSource Handler') failover/load balancing

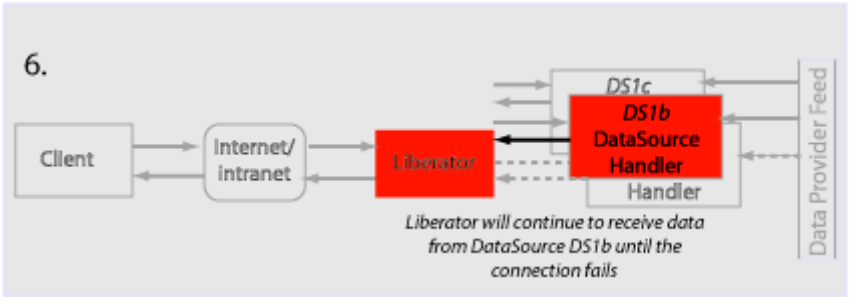


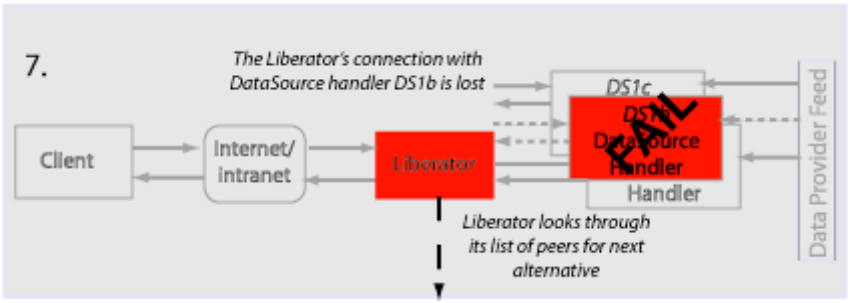


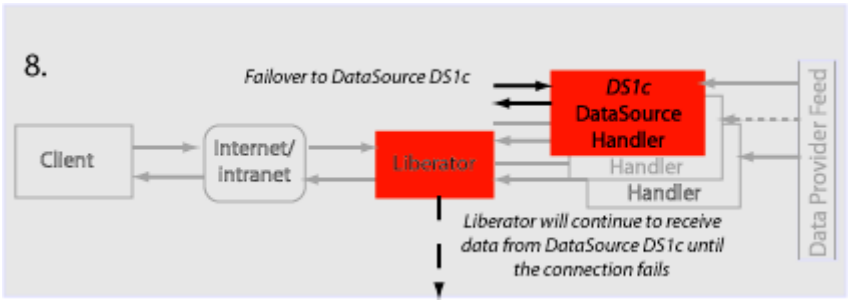


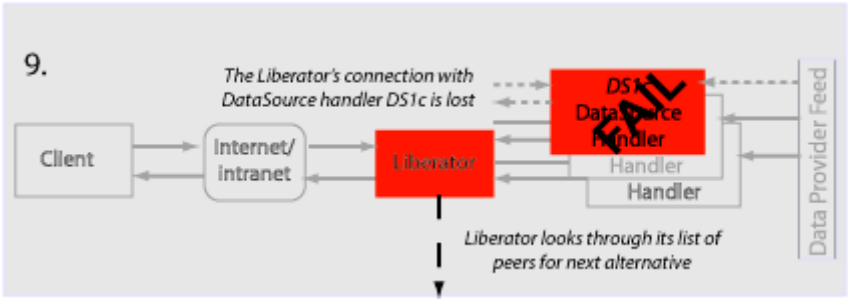


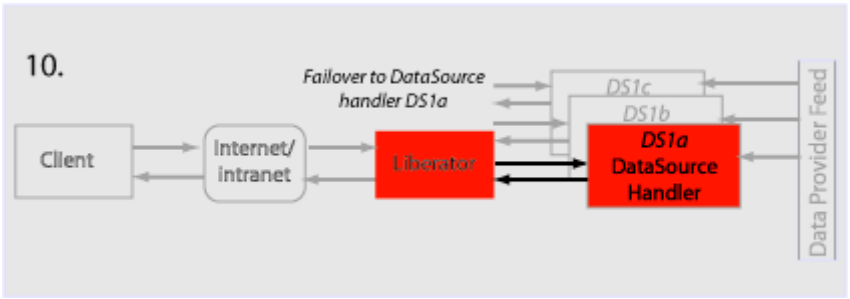




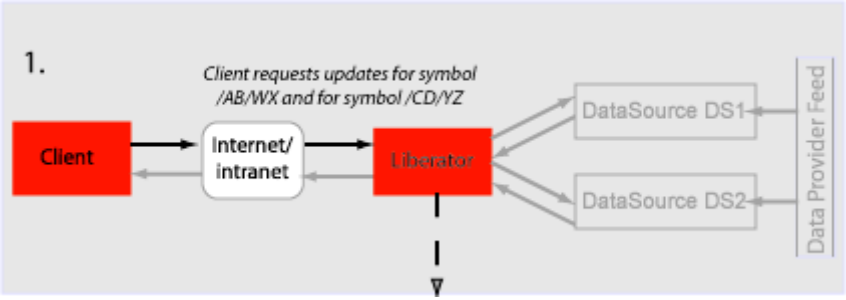


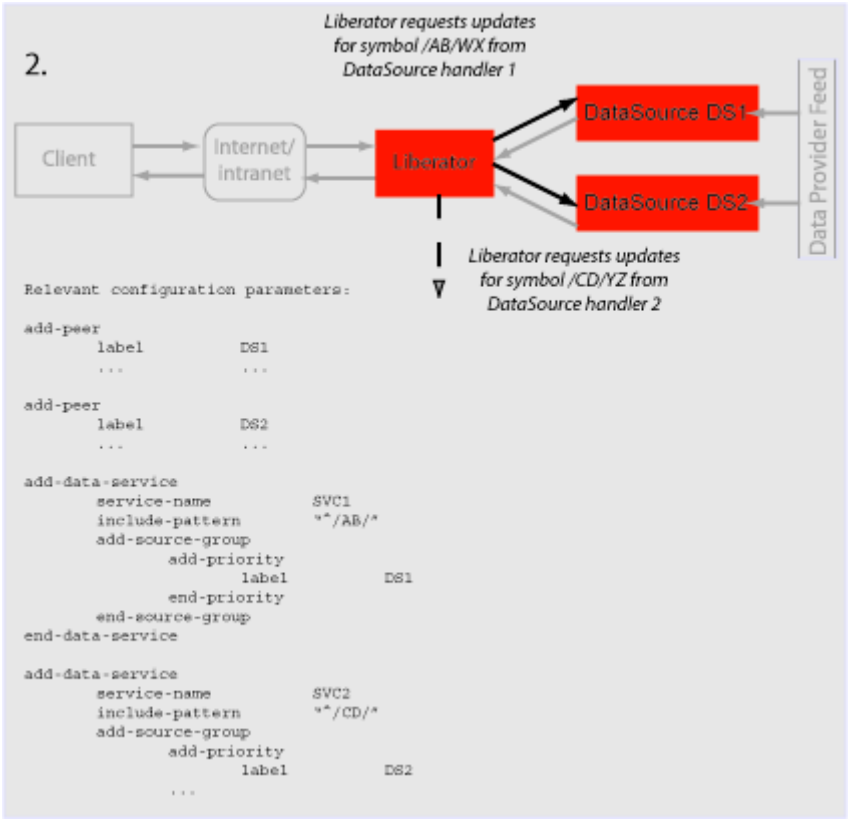


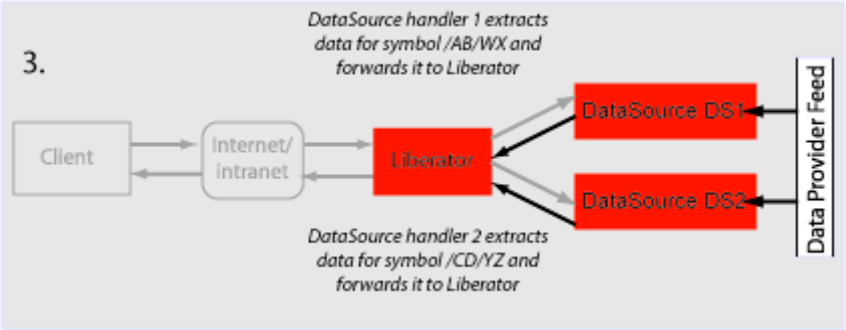


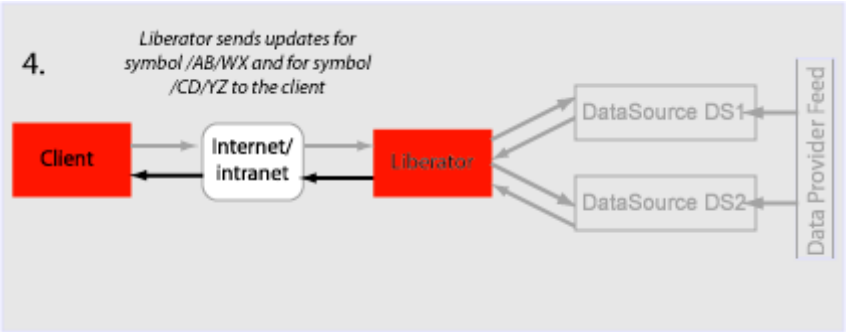


Active requests for data from 2 sources

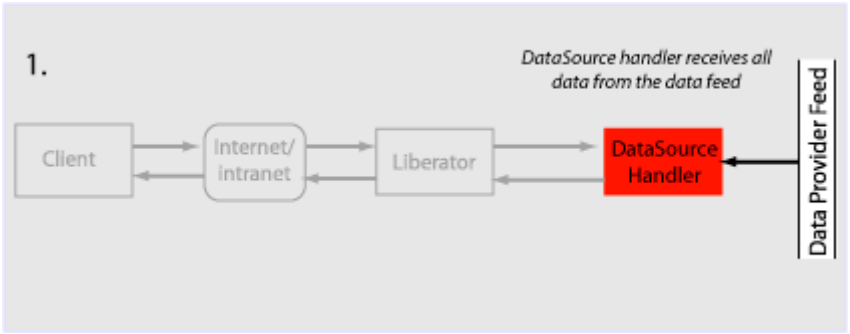


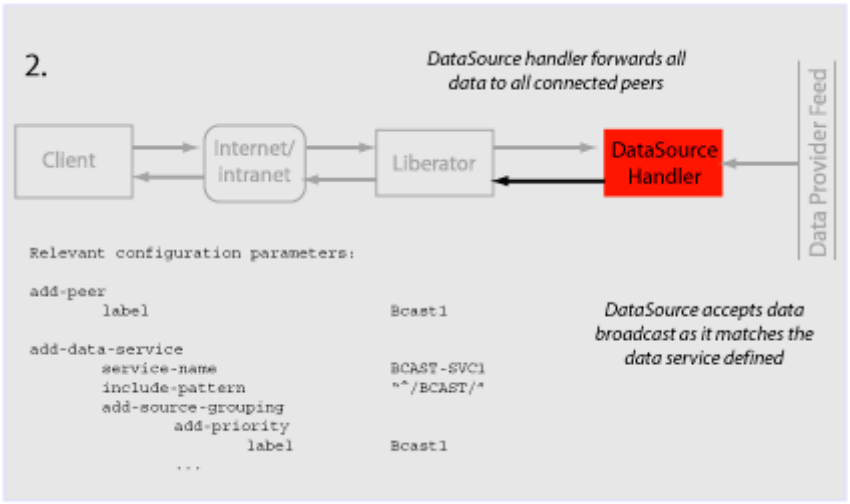


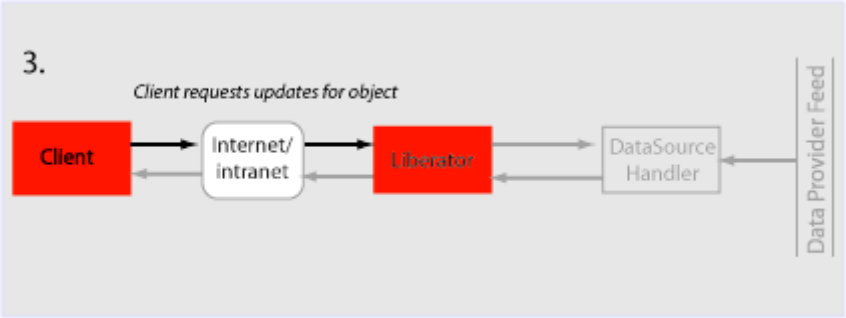


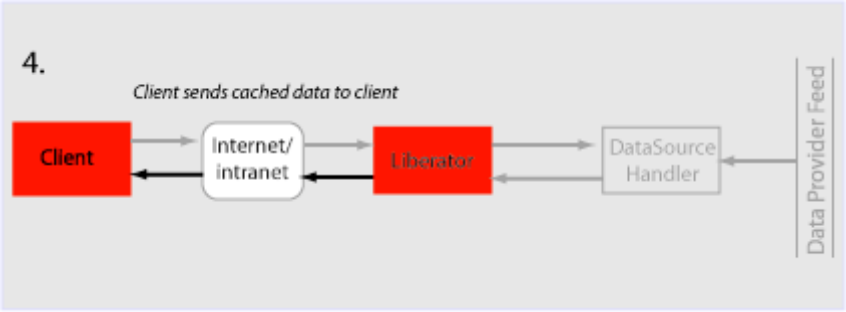


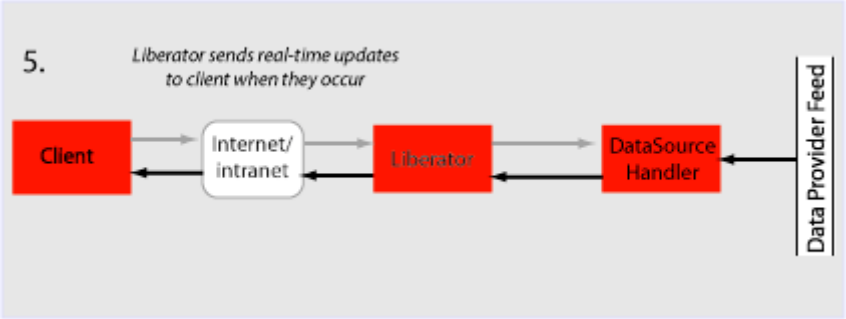
Passive source-broadcast data



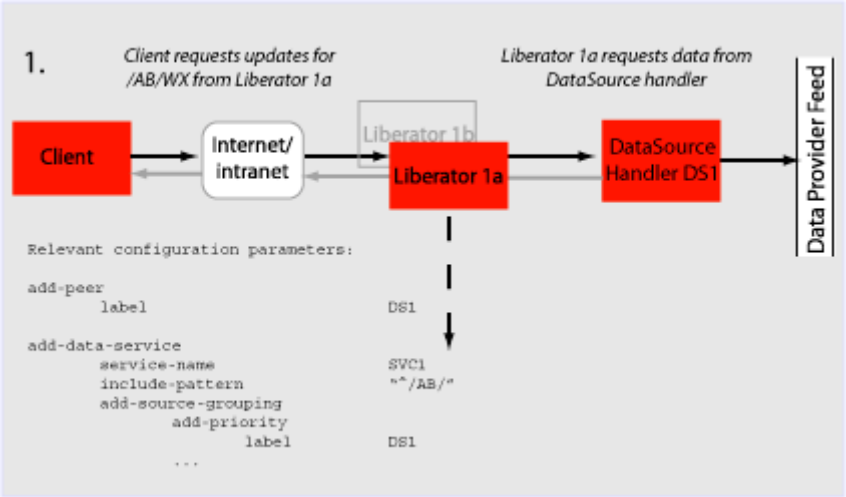


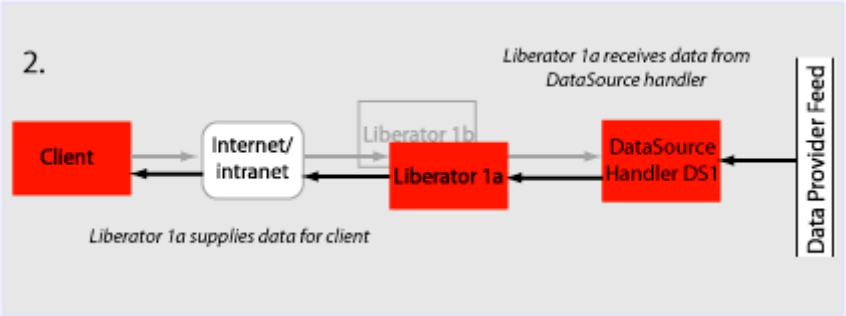


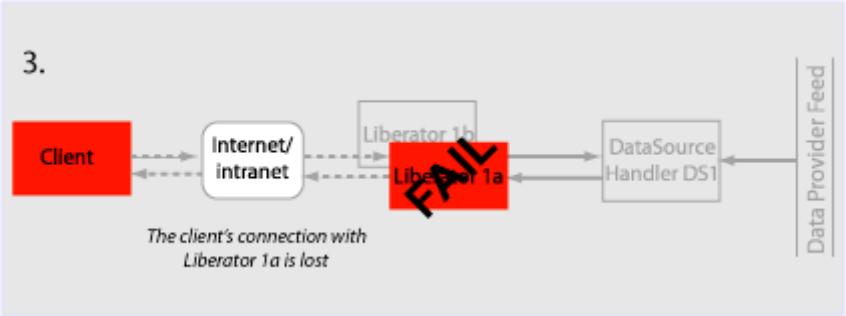


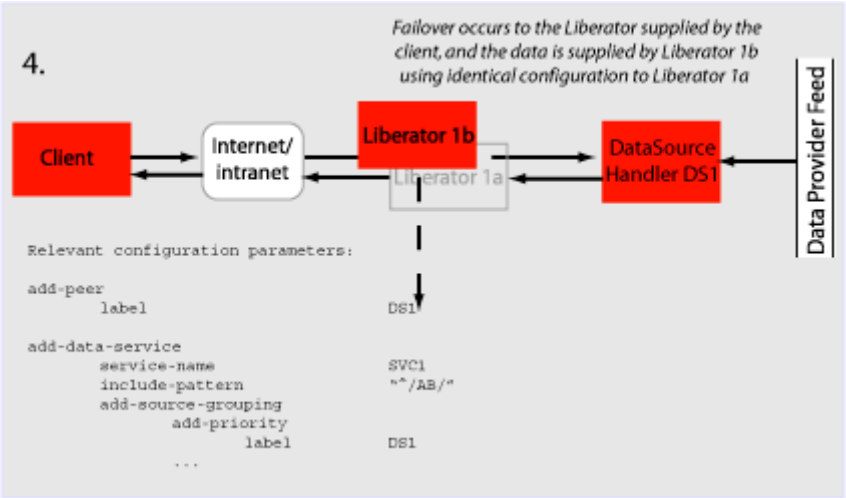


Liberator failover





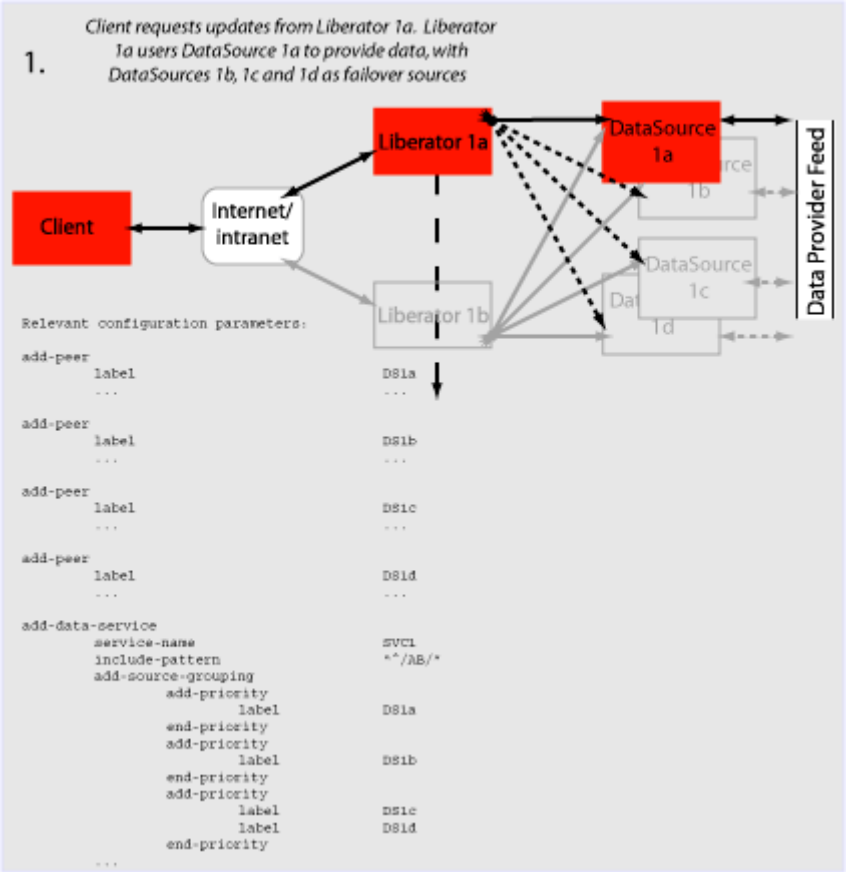


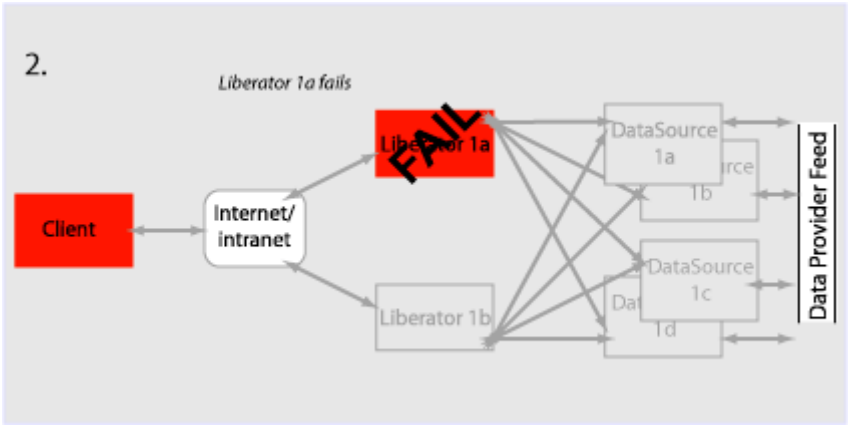


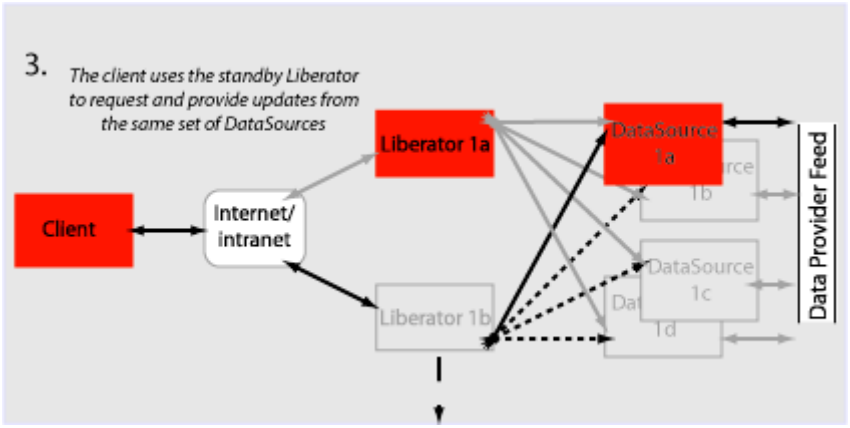
Liberator and DataSource peer failover

The following example assumes that the components are installed on different machines to provide extra resilience in case of failure:

Host A	Liberator 1a
Host B	Liberator 1b
Host C	DataSource 1a DataSource 1b
Host D	DataSource 1c DataSource 1d

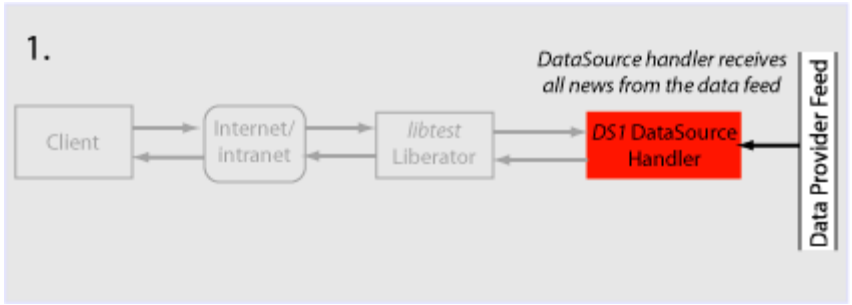

1.
Client requests updates from Liberator 1a. Liberator 1a users DataSource 1a to provide data, with DataSources 1b, 1c and 1d as failover sources
Client
Internet/ intranet
Liberator 1a
Liberator 1b
DataSource 1a
1b
DataSource 1c
1d
Data Provider Feed
Relevant configuration parameters:
add-peer
label DS1a
...
add-peer
label DS1b
...
add-peer
label DS1c
...
add-peer
label DS1d
...
add-data-service
service-name SVC1
include-pattern "^/AB/*
add-source-grouping
add-priority
label DS1a
end-priority
add-priority
label DS1b
end-priority
add-priority
label DS1c
label DS1d
end-priority
...

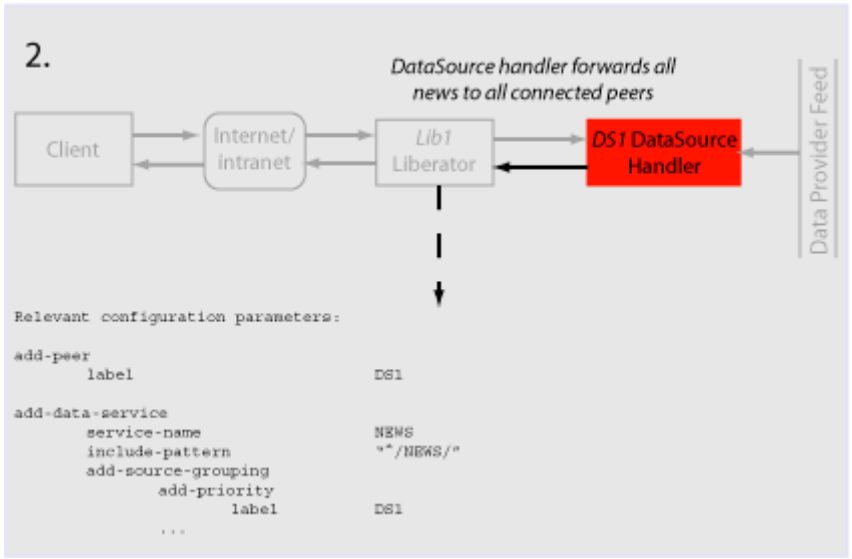


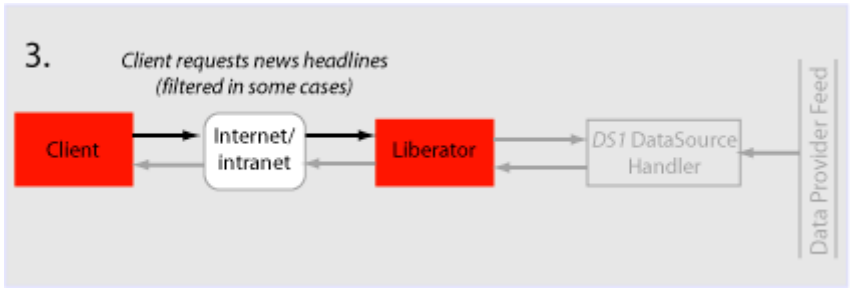


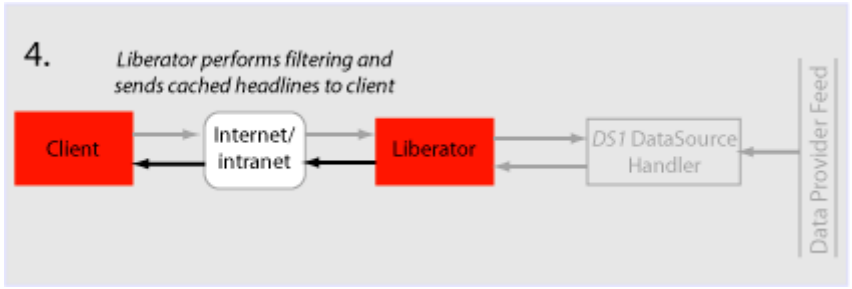
Requesting news headlines

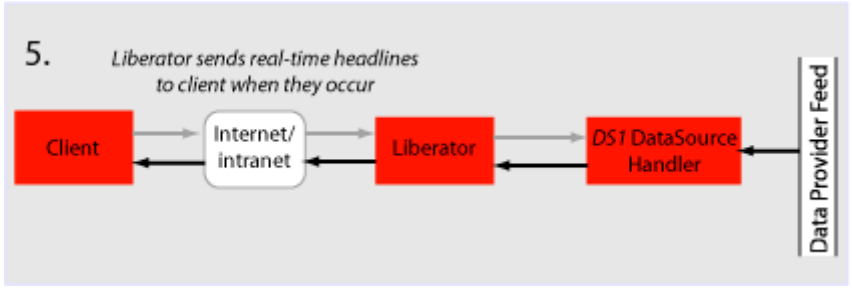
News headlines are delivered to Liberator as a broadcast feed—see page 48.



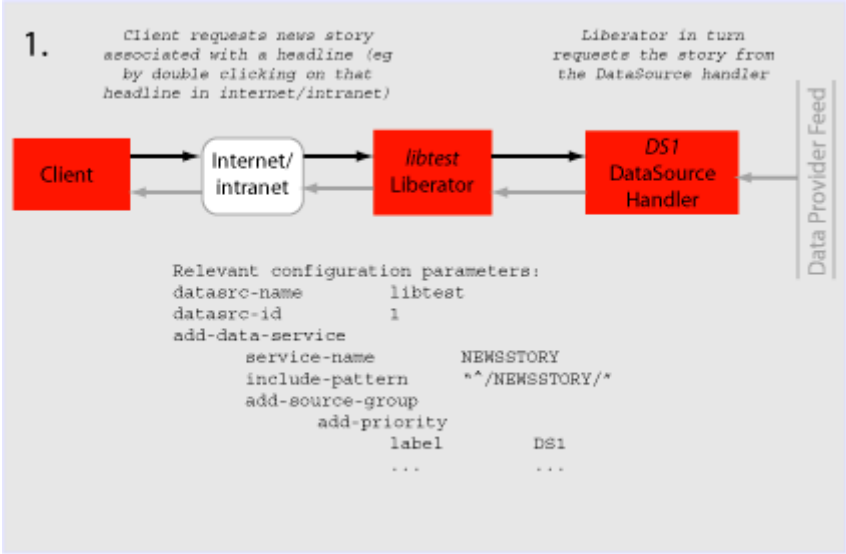


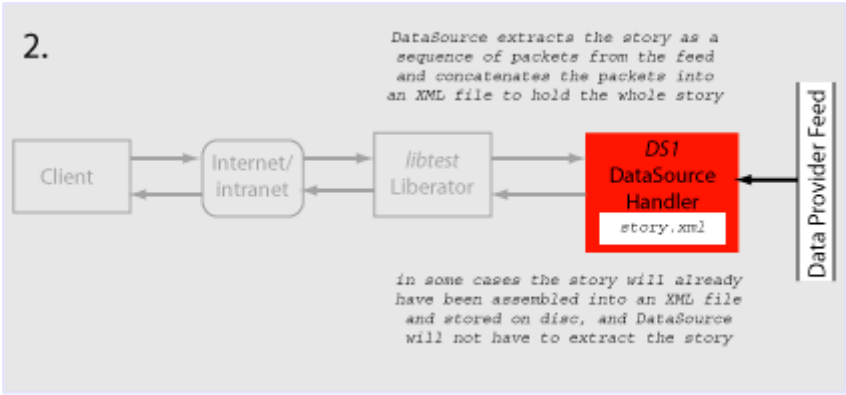


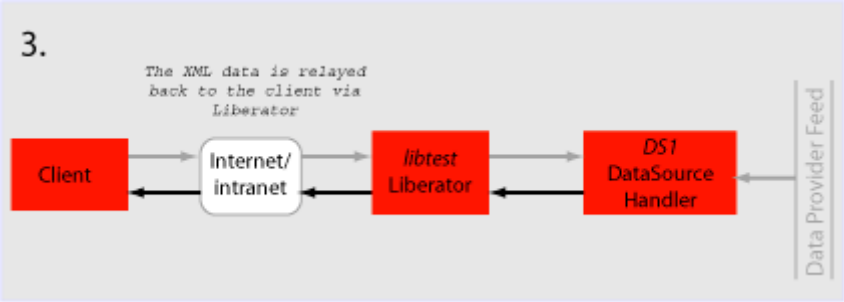




Requesting news stories

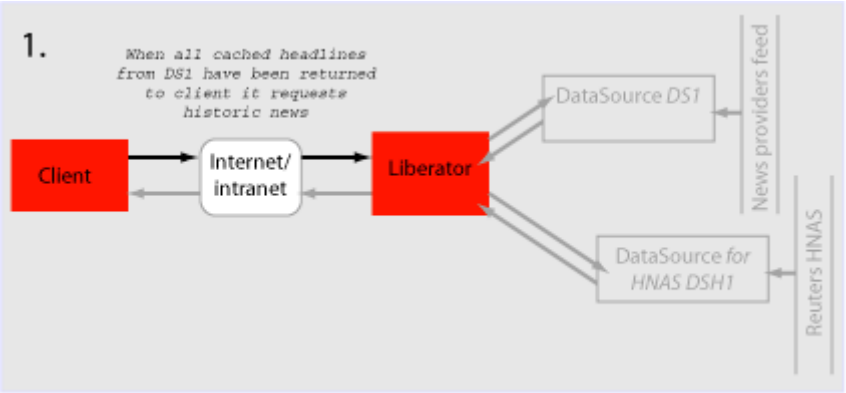


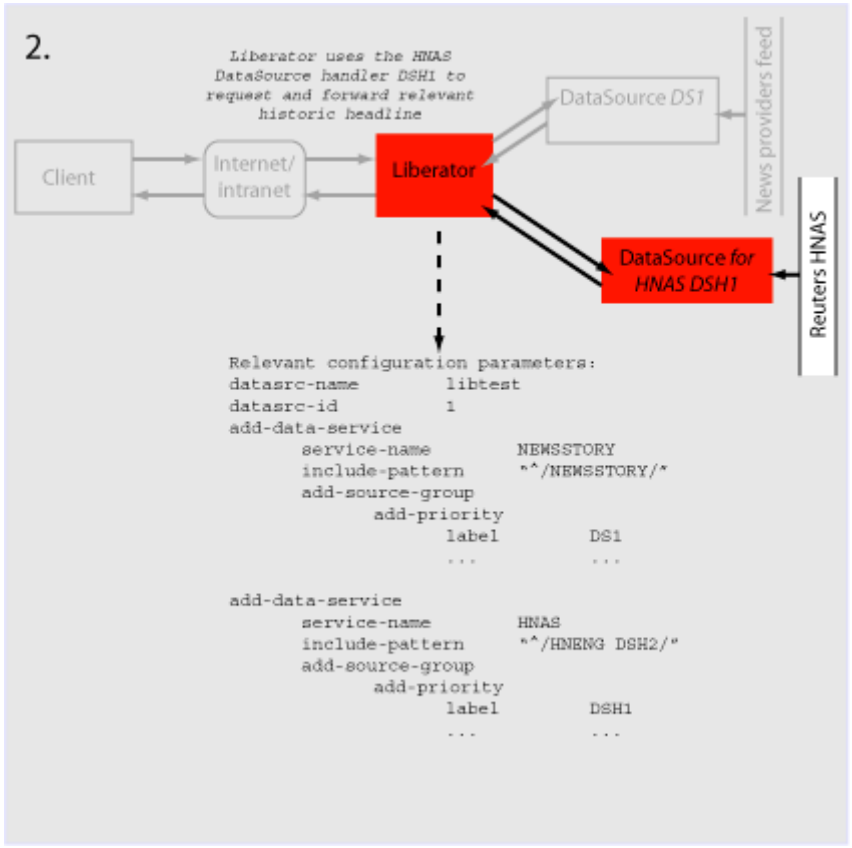


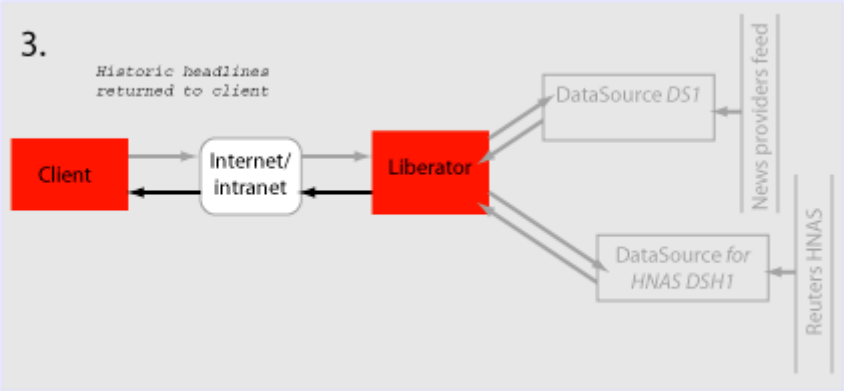


Requesting historic news headlines

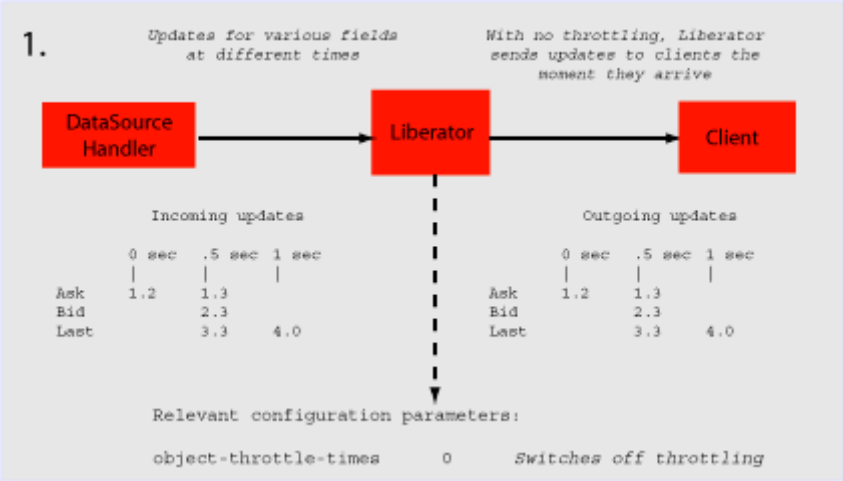
The following steps take place after Liberator has supplied all its cached real-time headlines—see page 55.

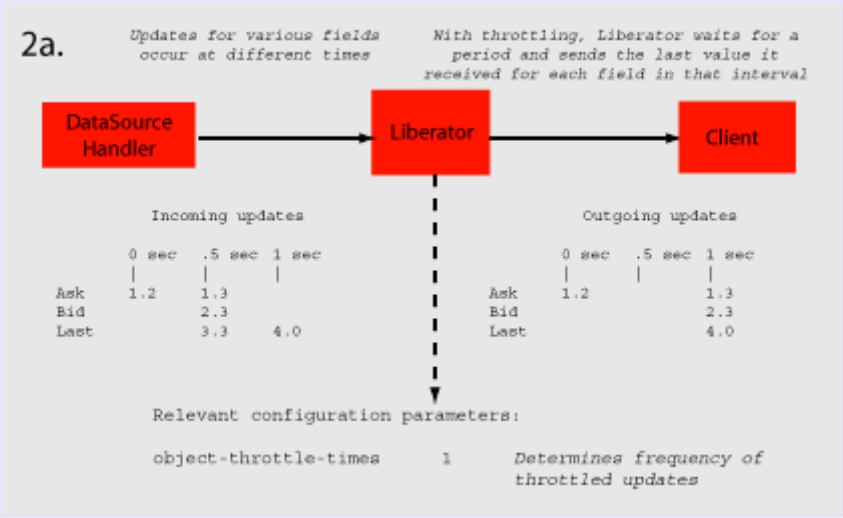


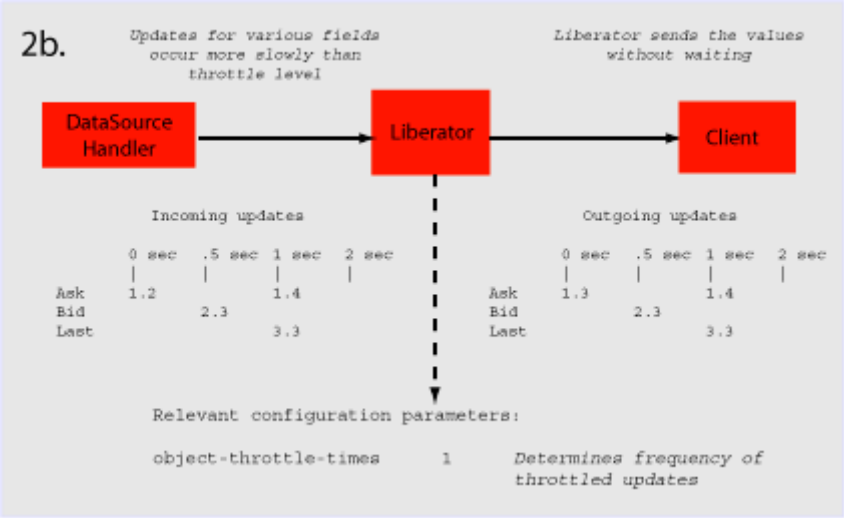




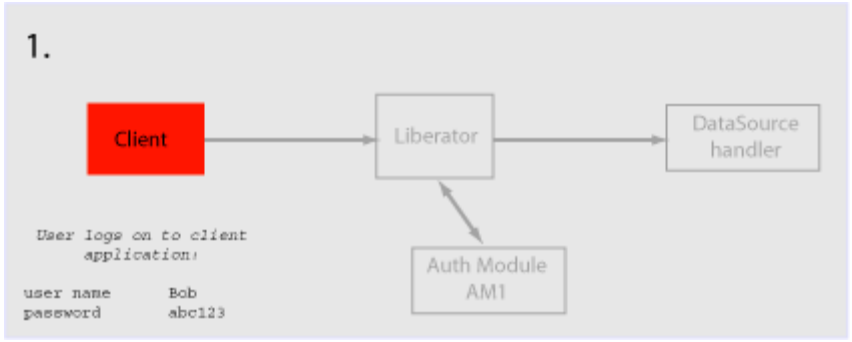
Throttling updates

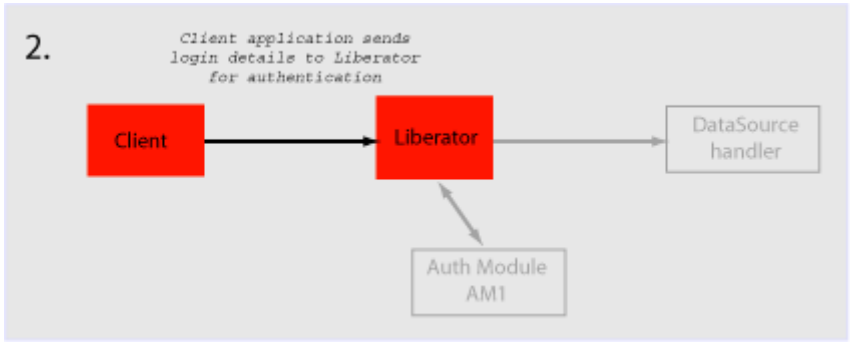


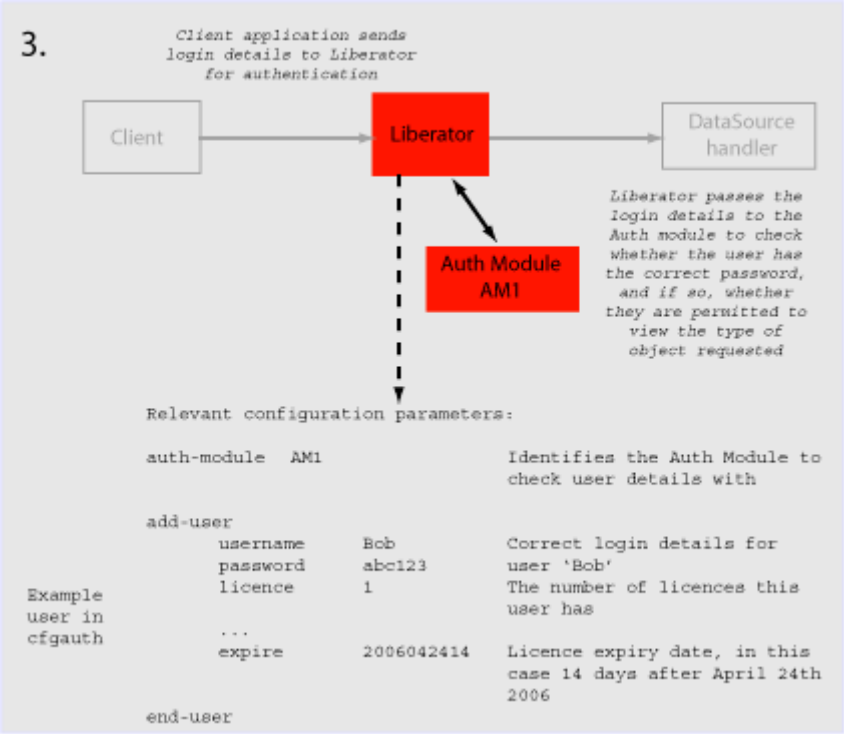


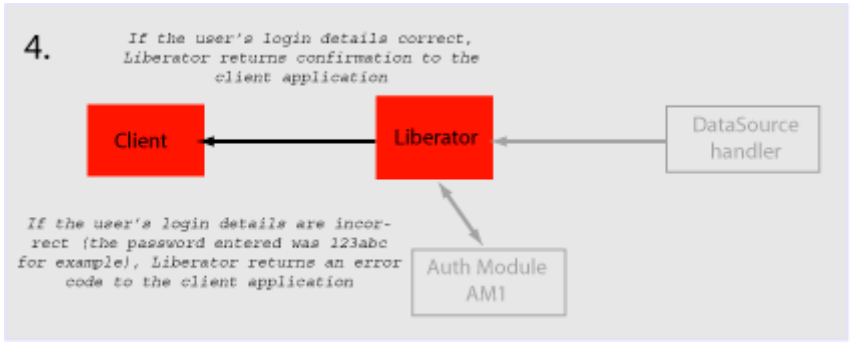


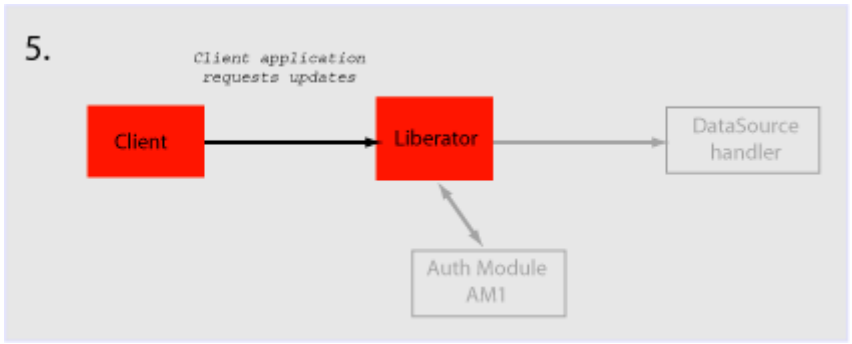
Authentication and authorization of users using Auth Modules

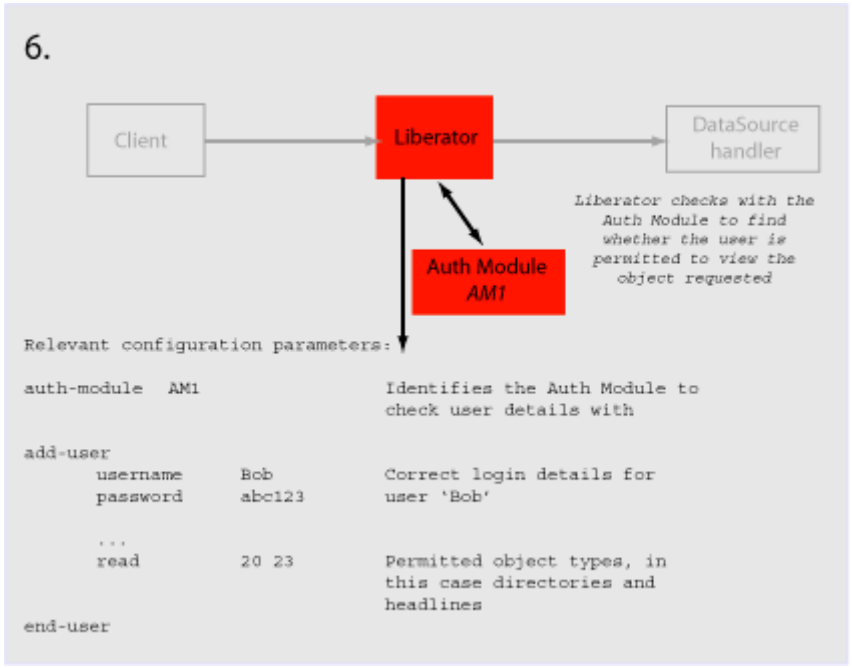


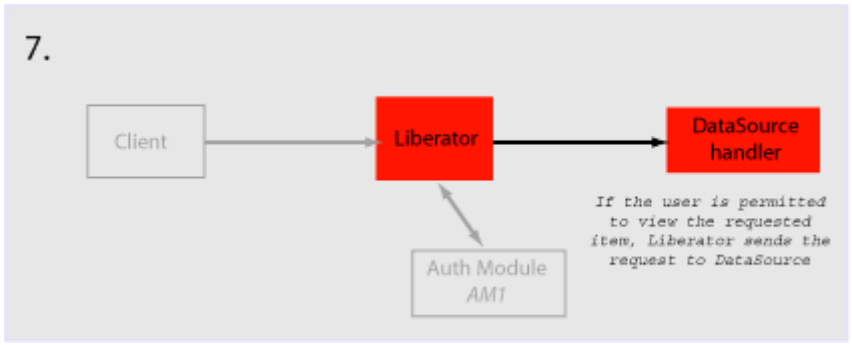


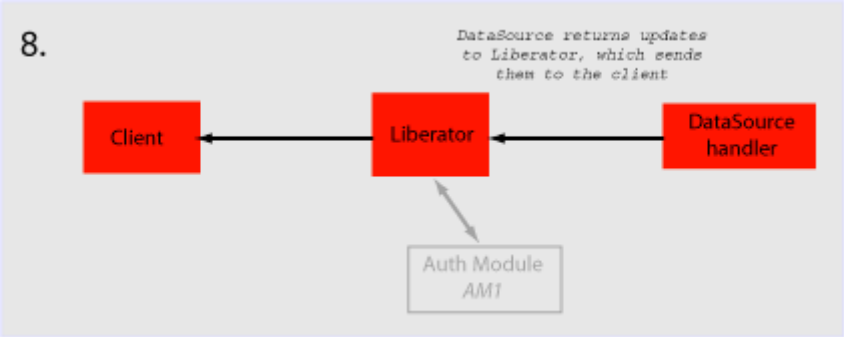












5 About the data

5.1 What is RTTP?

RTTP (Real Time Text Protocol) is a protocol developed by Caplin Systems that implements advanced real-time streaming for almost all types of textual information, including logical records, news and free-format pages. RTTP has been used by financial institutions for mission-critical data since mid-1997.

RTTP is an object-oriented server-push protocol for the distribution of streaming market data over internet-protocol networks. It supports both client-server and peer-to-peer publish/subscribe models.

RTTP builds on the functional experience of historic market data protocols, but removes many of the restrictions inherent in these protocols whilst taking advantage of advances in objected-oriented techniques and internet concepts. It can reliably publish to thousands of simultaneous users over the public internet and can also be used as a simple point-to-point protocol over a LAN.

The need to be able to communicate without hindrance across the whole of the internet along with the need to support sophisticated event-driven server-side technology have been the two primary driving forces behind the evolution of RTTP. It supports the widest range of market data instruments and activities, by providing a comprehensive set of standard data types as built-in objects, allowing user customisation of these, and finally permitting completely user-defined objects. This design philosophy has allowed RTTP to become a ready-to-use mechanism for Internet delivery with the capacity to mature over time.

RTTP ensures high data quality irrespective of most network obstacles using persistent virtual connections with smart/secure tunnelling and data health checking.

5.2 Key features of RTTP

Smart tunnelling

Web browsers are able to make HTTP connections over the internet because the proxy servers and firewalls which separate them from the web servers are specifically designed to pass on HTTP. Special protocols such as RTTP are normally not recognised by these proxy servers and firewalls, which have to be specially modified to let them pass.

Liberator avoids this problem by intelligently detecting the presence of such obstacles and employing the RTTP Smart Tunnelling technology where necessary to tunnel through them in a safe and secure manner.

Persistent Virtual Connection (PVC)

The PVC mechanism allows the Liberator to maintain a continuous virtual connection to every client irrespective of the activity of the Tunnelling Engine and transient loss of the actual connection.

In the event that a client connection is prematurely terminated (because of excessive packet loss or a proxy timeout, for example) the client RTTP layer immediately reopens the session. Liberator uses a unique session identifier to resume the previous RTTP session with no loss of context. If the delay in reconnection is excessive, this is automatically signalled to the client via the Data Health Check mechanism.

Data Status

In the event of a physical network failure, a link in the chain may fail and that updates intended for a particular client may be delayed, or may not arrive at all. In such circumstances, it is essential that the client is alerted instantly to the fact that the data may be stale.

Liberator and Integration Adapters keep track of the status of all data objects amd signal to the client if an object may contain stale data. Heartbeats between Client and Liberator, and between Liberator and Integration Adapters can be configured so loss of connections can always be handled even when the operating system does not close the connection.

Please see “Monitoring system health using heartbeats” on page 144 for further details.

5.3 About RTTP objects

Throughout this document you will find references to "RTTP objects". There are several types of RTTP object, and each type is identified by a two digit number, as described in Table 5-1.

Object Type	Description
20	Directory
21	Page
22	Record
23	News headline
24	News story
27	Chat object
28	Container object
29	Auto Subscription Directory

Table 5-1: Object types

Directory

A directory is both an object and a container for other objects. Directories can be used as a means of organising information into groups and hierarchies. Users of data streamed on RTTP can subscribe to a directory and receive updates when objects are created or deleted within that directory.

Page

A page is a free format piece of text made up of rows. RTTP supports any size of page up to 128 rows of 256 characters (typical sizes are 14 rows of 64 characters and 25 rows of 80 characters).

Record

A record (or “logical record”) is a means of storing and displaying information. Records are composed of fields which may not be of the same type: for example, a record containing equity data could have several price fields (e.g. the last traded prices) together with time and date fields, whereas an index record would have a price field but no bid or ask values.

For more information on fields, see “About RTTP fields” on page 73.

News headline and news story

Generally, news stories do not get streamed on RTTP since these do not benefit from being real-time enabled. The news headline, however, must be RTTP-enabled, so that if the user wants to read the story they can select that particular news item and use a more standard subscription mechanism to request the story.

A request for a news headline object may contain a filter string which allows a client to limit the updates it receives based on a simple logical syntax.

Chat objects

RTTP chat objects allow users logged into Liberator to chat in real-time. Each chat object represents a virtual chat room for 2 or more users.

To send a message to the channel, users contribute to chat objects.

Container

Container objects store references to other objects. A client requesting a container object will receive both changes to the container object (called structure updates), and will also be automatically subscribed to any objects that are held in the container.

As item references are added or removed from a container object, subscribed clients will receive notification of the structure changes and will automatically request or discard the relevant objects.

Auto Subscription Directory

This is a specialised directory object that allows the subscriber to the directory to be automatically subscribed to all of the contents of the directory, in a manner similar to the container object.

When combined with a filter, all objects within the directory will be subscribed to with the filter. This applies to both record and news filtering.

Auto Subscription Directories also provide the option to monitor filtering, which allows a client to distinguish easily between an infrequently updating record and a record for which many updates have been filtered out. As records' field values transit from: either matching to not matching; or not matching to matching the filter, a notification is sent.

Symbols and parameters

Most real time data handled by RTTP is identified by combinations of symbols and parameters. The symbol is stored as the name of an object on the Liberator.

- A symbol is a letter or sequence of letters used to identify a security. Symbols should always start with a "/". For example, "/DCX" is used for Daimler Chrysler Corporation, "/LO/VOD" for Vodafone trading on the London Stock Exchange, and "/MSFT" for Microsoft.

  The symbol you choose depends on the "symbology" being used by the data source. If you are running your own Liberator, this will by default be the same as the symbology of the data source to which it is connected. If you are using a third-party RTTP source, you should obtain a symbol directory from its owner.

- A parameter is a certain piece of information relating to the symbol. Typical parameters are "Bid" (the bid price), "Ask" (the asking price) or "Cls" (for the previous day's closing price).

The range of parameters available for a particular financial instrument also depends on the data source to which you are connected.

5.4 About RTTP fields

The Liberator uses fields to represent data within an object. Standard record objects are simply made up of a set of fields. Examples of these types are Bid (the bid price), Ask (the ask price), Time (Time of the last trade in seconds) and Currency (the currency in which the price is quoted).

Data comes into the Liberator via the DataSource protocol, which uses field numbers to identify fields. However, data sent to RTTP clients over the Internet uses field names to identify fields.

Record objects are probably the most important and widely used in RTTP due to the simple generic nature of the "symbol" container and "field" structure. However, within the market data arena it is important to be able to provide specific functionality to help address the needs of particular client applications and displays. This has brought about the need for a sub-classification of record field data, which is illustrated in the following pages.

Type 1 data

The majority of record based data is considered to be Type 1. This means that there is only one level of fields under the main container. Figure 5-1 shows an example field structure for a simple full quote display for the IBM stock on NYSE.

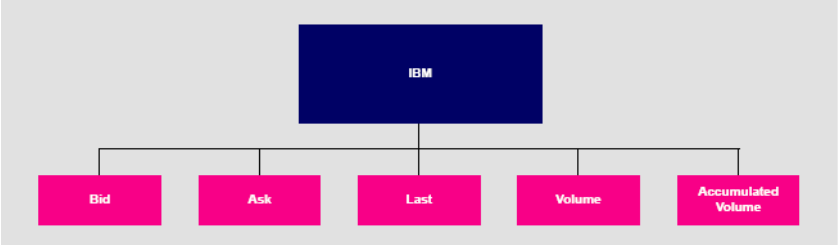


Figure 5-1: Example of Type 1 data within a record

Here, the single container IBM has one level of five fields, Bid, Ask, Last, Volume and Accumulated Volume. Whenever an update comes in to the Liberator for any of these fields the value is over-written. A user newly subscribing to IBM would then see this new value; the previous value would not be available.

Type 2 data

Type 2 data is often referred to as "level 2" data, as it is mostly used for level 2 quote data. Level 2 quote data enables several price quotes per symbol (coming from different market makers or traders) to be available at all times.

The field structure shown in Figure 5-2 might be applicable for a simple level 2 display for IBM, where there are three or more active market makers.

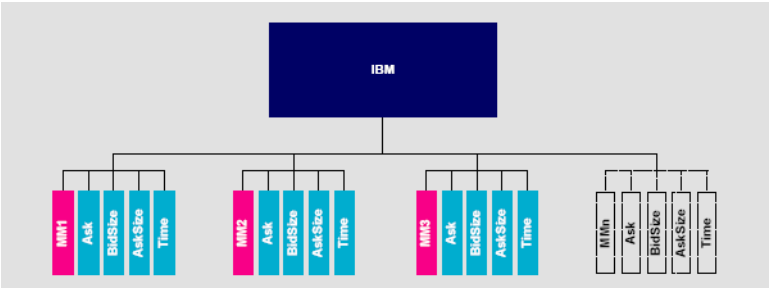


Figure 5-2: Example of Type 2 data within a record

In this case the IBM container (primary key) has a secondary key of Market Maker. This allows a new subscriber to see the full set of quotes in the market by enabling them to view each set of quotes from each market maker.

A quote update in this example will always have a market maker associated with it, causing only a specific sub-set of fields to be overwritten.

Type 3 data

Type 3 data allows for the storage of update history by keeping all updates of this type and not overwriting the symbol/field pair. A common use for Type 3 record data is for holding and viewing daily trade activity where, typically, this mechanism will only be used for a day at a time before the cache is deleted and the update list starts again.

Figure 5-3 shows a Type 3 field structure, which is similar to a Type 1 field structure but with many instances.

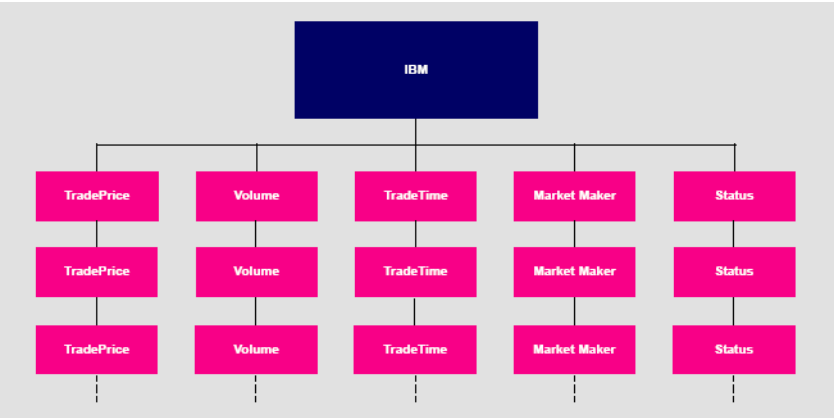


Figure 5-3: Example of Type 3 data within a record

Each new update is placed as the first (most recent) item on the list. Subscribers would receive the whole list as part of the initial subscribe response. The size and purging frequency of this list is configurable separately to the size and purging frequency of the fields themselves.

6 Communicating with clients

6.1 Enabling clients to connect using RTTP (over HTTP)

Note: *Liberator can write to a log file information about clients accessing it through HTTP. For more information on this and other logging facilities, see "Monitoring performance" on page 125*

Making an HTTP connection

- Use the following parameters in the configuration file *rttpd.conf* to enable clients to connect to Liberator using RTTP over HTTP connections.

http-interface	Space-separated list of interface IP addresses to listen on for HTTP connections. See page 174.
http-port	Network port to listen for HTTP connections. When the Liberator is running in production, this should usually be set to port 80. The Liberator will have to be started as 'root' on UNIX systems to allow binding to port 80. See page 174.
add-thread	Configures the interfaces and ports settings for additional threads. add-thread entries are optional, and the default values will be used for those threads that do not have an associated add-thread entry. See page 237.

Configuring the HTTP Keep Alive feature

- Use the following parameters in the configuration file *rttpd.conf* to enable the HTTP Keep Alive feature.

http-keepalive-max	Number of requests per connection. See page 174
http-keepalive-timeout	Timeout in seconds of HTTP Keep Alive connections. See page 175

Using cookies to aid HTTP connection

Liberator can use cookies to indicate which RTTP & MIME type was used to successfully connect, so that on subsequent attempts the client knows which connection type to try first.

- Use the following parameters in the configuration file *rttpd.conf* to enable Liberator to save cookies on client machines.

http-connection-cookie-enable	If set, the server will set a cookie in the client when the client connects over HTTP. See page 178
http-connection-cookie-expires	Number of days before the cookie expires. See page 179

6.2 Enabling clients to connect using HTTPS

Making an HTTPS connection

The Secure Sockets Layer (SSL) is a commonly-used protocol for managing the security of a message transmission on the Internet, and offers a greater level of protection than standard HTTP transmission.

Liberator can run as an HTTPS web server like most common web servers. Web pages, Java™ applets and other standard HTTP traffic can be sent over HTTPS. If an RTTP Applet is downloaded over HTTPS then all RTTP data will be over an HTTPS connection too.

Liberator supports standard SSL server-side certificates to authenticate the server to the client. They must be generated and signed by a certificate authority.

Liberator is also capable of communicating with its data sources over SSL, providing an encrypted channel over which the data sources can publish their data.

Using HTTPS on Linux

- Use the following parameters in the configuration file *rttpd.conf* to enable clients to connect using HTTPS.

https-enable	This option switches on support for HTTPS connections. See page 181
https-interface	Space-separated list of interface IP addresses to listen on for HTTPS connections. See page 181.
https-port	Network port to listen on for HTTPS connections. When the Liberator is running in production, this should usually be set to port 443. See page 182.

Virtual hosting

Virtual hosts allow a single Liberator to serve independent websites.

Note: *Each virtual host is based on the IP address the client connects to and not HTTP 1.1 name-based virtual hosts.*

There are two things that can be configured as virtual hosts:

- ❖ the directory to use as the root directory for the website;
- ❖ the SSL certificates to use for HTTPS connections.

■ Use the following parameter to use a virtual host.

add-virtual-host Identifies a virtual host that Liberator will serve. If a client connects via an ip address identified by add-virtual-host it will use the options configured. Any other IP addresses will use the global options.

Example:

```
add-virtual-host
      name                  service2
      addr                  192.168.123.123
      wwwroot               /Liberator/service2/docs
      https-certificate     cert2.pem
      https-privatekey      cert2.pem
      https-passwordfile    rttpd.https.service2.pass
end-virtual-host
```

Configuring the HTTPS connection

To setup Liberator to use HTTPS you can use the test certificate provided for the SSL sample configuration (*etc/certs/rttpd.pem* and *rttpd.key*). For more information on the Using SSL with the demonstration feed on page 167.

This certificate requires a pass phrase which is contained in the file identified by *https-passwordfile*. You will find the necessary configuration option commented out at the end of *rttpd.conf*.

- Use the following parameters in the file *rttpd.conf* to configure the HTTPS connection.

https-interface	Configures the network interface to listen on for HTTPS connections. See page 181
https-port	Configures which network port to listen on for HTTPS connections. See page 182
ssl-random-seed	Configures the seeding of the OpenSSL random number generator, which the Liberator uses for session IDs and HTTPS and DataSource SSL connections. See page 182 On Linux OpenSSL is seeded by a hardware device so using ssl-random-seed may be unnecessary.

Example:

```
https-interface 192.168.150.150 192.168.150.151

ssl-random-seed  builtin
ssl-random-seed  file  etc/randomdata
ssl-random-seed  file  etc/randomdata  1024
ssl-random-seed  exec  etc/random.sh
ssl-random-seed  exec  etc/random.sh   512
```

Applying the security policy

- Use the following parameters in the file *rttpd.conf* to determine how SSL certificates are to be used.

https-certificate	Filename of the SSL certificate. This file should be in PEM format. See page 182
https-privatekey	Filename of the SSL private key. This file should be in PEM format. See page 182

Note: *The default filename for the private by is the same as the certificate because both the certificate and the private key can be contained in the same file.*

https-passwordfile This option identifies the file containing the SSL certificate passphrase.
See page 182

Sample certificates and certificate authorities

The sample HTTPS configuration uses certificates and certificate authorities which are already set up in the Liberator kit in the directories *etc/certs* and *etc/demosrcCA*. These were created using the OpenSSL toolkit (for more information see www.openssl.org).

Note: *As this is only a sample setup you will need to tell your browser to accept the certificate even though it does not recognise the authority and the certificate is not for that server. For production you must obtain a real certificate.*

The certificate and certificate authority use the following passphrase:

Liberator certificate: `rttpdcert`

By default the Liberator will look for passphrases in the file *etc/.rttpd.https.pass*. If this file is not present a password prompt will be given when the Liberator starts. It is therefore possible to echo the password into the application on startup: to achieve this the standard startup script should be changed.

Configuring hardware devices

OpenSSL has built-in support for cryptographic acceleration. In newer versions of OpenSSL an application can get a reference to a specific representation, often a hardware device. These representations are referred to as Engines.

- Use the following parameters in the file *rttpd.conf* to configure SSL hardware.

ssl-engine-id The SSL hardware or software engine to support.
See page 184

The hardware and software engines that the Liberator supports are listed in Table 6-1 below. If you are using a different engine please contact Caplin.

ssl-engine-id option	Engine
openssl	The engine uses the normal built-in software functions

aep	Uses the Aep acceleration hardware
atalla	Uses the Compaq Atalla acceleration hardware
chil	Uses the nCipher CHIL acceleration hardware
cswift	Uses the CryptoSwift acceleration hardware
nuron	Uses the Nuron acceleration hardware
ubsec	Uses the Broadcom uBSec acceleration hardware
sureware	Uses the SureWare acceleration hardware

Table 6-1: Supported hardware and software engines

ssl-engine-flags Flags to be passed to the engine implementation.
See page 184

The available flags to use are listed in Table 6-2 below. These flags may be ORed together using the "|" operator to represent multiple flags: for example "dsa|rsa" equates to using only DSA and RSA operations.

Flag	Description
dh	Limit engine usage to only DH operations
dsa	Limit engine usage to only DSA operations
rand	Limit engine usage to only random operations
rsa	Limit engine usage to only RSA operations
all	Allow OpenSSL to use any of the above implementations

Table 6-2: ssl-engine-flags flags

6.3 Enabling clients to connect using RTTP (direct connection)

RTTP direct connection is also known as a type 1 connection. The RTTP protocol is described in more detail in the chapter entitled “About the data” on page 69.

- Use the following parameters in the configuration file *rttpd.conf* to enable clients to connect to Liberator using an RTTP direct connection.

direct-interface — Network interfaces to listen for RTTP connections.
See page 186.

direct-port — Network port to listen for RTTP connections.
See page 186.

add-thread — Configures the interfaces and ports settings for additional threads. add-thread entries are optional, and the default values will be used for those threads that do not have an associated add-thread entry.
See page 237.

6.4 Enabling clients to connect using RTTP (direct SSL connection)

Liberator can also accept direct (type 1) RTTP connections that use the Secure Sockets Layer (SSL) to provide greater security. The configuration options that specify direct connections using SSL are defined on page 187. These options are similar to the options that configure HTTPS (secure HTTP) connections; see “Enabling clients to connect using HTTPS” on page 77. The following table lists the equivalent HTTPS configuration options:

Direct SSL configuration option	Equivalent HTTPS configuration option
directssl-enable	https-enable
directssl-interface	https-interface
directssl-port	https-port
directssl-ssl-options	https-ssl-options
directssl-certificate	https-certificate
directssl-privatekey	https-privatekey

Direct SSL configuration option	Equivalent HTTPS configuration option
directssl-passwordfile	https-passwordfile
directssl-cipher-list	https-cipher-list
ssl-random-seed	ssl-random-seed
ssl-engine-id	ssl-engine-id
ssl-engine-flags	ssl-engine-flags

6.5 Configuring objects

It is possible to configure certain objects and directories that will be created on startup. This may be to make sure they are there before a broadcast source alerts updating the object, or to configure throttling for all objects in a directory.

- Use the following parameters in the file *rttpd.conf* to identify any object to be created on start-up.

add-object
: Adds an object to Liberator and defines the object's characteristics. This configuration option can also specify throttle times that are specific to this object, and override any global values that have been set. If the object is used as a directory, all objects that are subequently subscribed to under that directory will inherit the configuration options that were defined in ***add-object***.
See page 191.

object-map
: Defines an object mapping. Object mapping changes the internal name of an object when a user requests it. This allows a username to be included in the object name in order for each user to get a unique object. For example if a user called 'userX' requests /HN/NEWSSTORY/1234, the object could be mapped to /HN/NEWSSTORY/userX/1234.
See page 197.
Example:

```
object-map  "/MYCHANNELS/%1"  "/CHANNELS/%u/%1"
object-map  "/ABC/%1/%2"      "/DEF/%2/%1"
```

where %u is the Liberator login name of the requesting user, and %1 and %2 are strings to be matched in the pattern. Each object-map entry can identify up to 9 strings (%1 to %9).

default-type
: Sets the default sub-type parameter for all objects.
See page 197.

add-type-mapping
: Adds a sub-type mapping, which changes the sub-type of an object when a user requests it. You can have any number of entries. Object names are matched in the order given. Asterisk "*" is used as a wildcard character.
See page 197.

Purging objects

add-object entries enable you to specify different purging times for different objects (purging being deleting the object from the Liberator's cache). These are configured using the following parameters within add-object:

The examples below show how these options can be used to configure object purging.

purge-time	Number of minutes after midnight on Sunday to start purging.
purge-period	Number of minutes between purges.
purge-age	A multiplier on purge-period. Defines how old an object should be before it is purged.

Purging example 1

Given the following add-object entry, Liberator will recursively purge all objects under */I/CHARTS* at 2am on Monday morning, unless someone is looking at them:

```
add-object
      name /I/CHARTS
      type 20
      throttle-times 0
      purge-time 120
      purge-period 1440
end-object
```

- If purge-time = 0 and purge-period = 1440, purging would at midnight every day.
- If purge-time = 180 and purge-period = 720, purging would occur at 3am and 3pm every day.
- If purge-time = 0 and purge-period = 60, purging would occur every hour.

Purging example 2

Given the following add-object entry, Liberator will purge all objects under /I/CHARTS at midnight, unless someone is looking at them:

```
add-object
      name /I/CHARTS
      type 20
      throttle-times 0
      purge-time 0
      purge-period 1440
      purge-age 0
end-object
```

- If purge-age = 1, only objects which had not been updated for 1440 minutes (1 day) would be purged.
- If purge-age = 7, only objects which had not been updated for a week would be purged.
- If purge-period = 60 (i.e. purging every hour) and purge-age = 6, only objects 6 hours old would get purged.

Purging example 3

This example shows how to configure a weekly purge at 2am every Sunday morning.

```
add-object
      name              /DIR1
      type              20
      purge-time        8760
      purge-period      10080
end-object
```

Sending only changed fields

This feature makes the Liberator compare each update received from it's Integration Adapters with the previous update for a given symbol. If any of the fields are the same as previously received, those fields are not sent out to the client. If no fields have changed in an update, no message will be sent to the client

Where there are many fields that are infrequently updated, the size of the message transferred to client is reduced. This feature might require increased server resources and may not be suitable where there the majority of fields are frequently updated.

This feature applies to record types (including type 2 records) only.

The Liberator can be configured so that all updates to a certain symbol are processed, or so that every update to a symbol in that directory and below are processed. This feature can alternatively be implemented directly in a custom Integration Adapter (please refer to the relevant DataSource and Integration Adapter documentation; see “DataSource documentation” on page 18).

- Use the following parameters within the add-object to configure sending only changed fields.

only-changed-fields Configures an object to only forward the changed fields in an update.

Sending only changed fields example 1

A single object can be configured with this option

```
add-object
  name   /B/Object1
  type   22
  only-changed-fields
end-object
```

Sending only changed fields example 2

A whole directory and it’s descendants can be configured with this option

```
add-object
  name   /A
  type   20
  only-changed-fields
end-object
```

6.6 Identifying the fields clients can request

- Use the following parameters in the file *rttpd.conf* to identify any field that might be requested.

add-field — Defines which fields can be used within the Liberator. It configures the field name and field number, as well as setting various flags which can customise the characteristics of the field before being sent to clients.
See page 208

Flags are used for:

a) setting the number of decimal places;

b) setting the data to be Type 2 or Type 3 (for an explanation of Type 2 and Type 3 data types, see “About RTTP fields” on page 73).

You can configure multiple field numbers to be translated to the same field name if necessary, but not vice versa.

fields-file — Name of a file containing configuration for fields, to be used as an alternative to those listed in *rttpd.conf*. This file can contain a list of add-field entries and list all required fields, so that Liberator can read in the fields on startup in order to gain an up-to-date list without its own configuration being changed.
See page 208

Setting the number of decimal places

If the FieldFlags parameter of the add-field entry is set to 256, it can be used to define how many decimal places the value of a field should have. When this flag is set, a fourth argument to add-field is needed to set the number of decimal places. This fourth argument is FieldFlagsData—see page 208

- Set the FieldFlags parameter of add-field to 256
- Set the FieldFlagsData parameter of add-field to the required number of decimal places

For example:

```
add-field   Last   6   256   3
```

This would make all updates to the Last field be formatted to 3 decimal places.

Setting the record data to Type 2

Type 2 data allows updates to a record to be stored using a second index (see page 73). This means a record can contain a set of fields for each unique value of a specified field, giving a two dimensional table of data instead of the flat field/value-based arrangement used for type 1 data.

To achieve record Type 2 data, any field which is to be used as a Type 2 index must have Bit 1 set in FieldFlags, and any fields which should be within a Type 2 update should have Bit 2 set in FieldFlags.

- Set *FieldFlags* to 1 or 2

For example:

```
add-field   MarketMaker 212   3
add-field   Bid         22    2
add-field   Ask         25    2
```

Note: *Record Type 2 updates must contain the Type 2 index as the first field in the update.*

With the above configuration a record object could contain the following data:

MarketMaker	Bid	Ask
AA	123	125
BB	122	124
CC	123	126

If an update then came in with MarketMaker=BB Bid=121 Ask=125 it would replace the values in the BB row.

- Use the following parameter in the configuration file *rttpd.conf* to improve the caching of Type 2 data.

record-type2-hash-size — Size of hashtable which holds Type 2 data.
See page 199

Setting the record data to Type 3

Record Type 3 data keeps updates as sets of fields in a similar way to Type 2 data; however, updates are not replaced but added to the list. Updates are discarded when the number of updates reaches a configured limit.

Type 3 data is more analogous to trade history updates. Fields with Bit 4 set in FieldFlags are defined as Type 3 data.

- Set *FieldFlags* to 3 or 4

For example:

```
add-field TradePrice    6     4
add-field TradeTime     379   4
add-field TradeVol      178   4
```

- Use the following parameter in the configuration file *rttpd.conf* to set the number of updates to type 3 data in each record that Liberator keeps in cache.

record-type3-history-size — Maximum number of updates to keep for each record containing record type 3 data.
See page 191.

- Alternatively, you can set different cache sizes for updates to record type 3 data by specifying the size for specific objects or object hierarchies.
  Use a **record-type3-history-size** entry in an **add-object** item for each hierarchy.
See page 195.

6.7 Handling requests for news headlines

A client can request updates from news streams, and set certain filtering criteria using special codes for topics such as industries or countries.

Identifying news codes users can search for

- Use the following parameters in the configuration file *rttpd.conf* to identify valid codes that clients can use as filters.

add-newscodes	If there are permissible exceptions to newscode-max-length, this parameter should include an array of codes listing the permitted exceptions. See page 240
newscodes-valid-chars	A list of characters that are valid in a news code. The default of "/." means a news code can be any uppercase characters and the characters "/" or "." (for example "FIN" or "BT.L"). See page 241
newscode-max-length	Users can request news stories by either sending a code (for example "AFN" is African Domestic News Service; "BASK" is basketball and "CHE" will return chemical industry stories) or by entering a search string. Liberator identifies the request as being a search string rather than a code if it is over a certain length. newscode-max-length determines the maximum length of a news code. Anything longer is considered to be a search string, unless it has been identified as an exception using newscode-exceptions and add-newscodes. Only strings in upper case are considered to be codes. See page 240
newscode-exceptions	Boolean parameter that determines whether there are any exceptions to the newscode-max-length rule (i.e. whether there are any news codes that are longer than newscode-max-length). EUROPE, for example, is a news code, but is longer than the default maximum code length of 4, and would therefore need to be added to the exception list. If set to TRUE, list the exceptions in add-newscodes. See page 240
newscode-hash-size	Default number of entries in the newscode exceptions hashtable. See page 241

6.8 Adjusting the update rate

Using throttling

Liberator can send updates every fraction of a second, but in most situations this is unnecessary and at times may overload the system. When this happens, Liberator can improve performance by using its throttling feature. This is sometimes known as conflation. This means that the Liberator will wait to publish an update if it occurs less than a certain time after the previous update. This gives the Liberator a chance to publish all outstanding updates and let the system catch up.

The Liberator can supply the same object to multiple users at different throttle levels. This provides per-object per user throttling instead of just per object. This allows users viewing lots of objects, with slow network connections to the server or on low specification computers to receive data at a speed that suits their environment.

A user application can change the level of throttling for specified objects, groups of objects or all objects globally. Each object has a set of throttle levels which defines the time delay of the throttling. This set can include special cases which represent no throttling and also a stopped state in which the user will receive no updates until it asks for them.

For example an object may have five throttle levels:

1 no throttling

2 throttling at 0.5 seconds

3 throttling at 1 second

4 throttling at 2 seconds

5 the stopped state.

Your Liberator can have a default throttle level at which each object starts on login. This is typically the lowest level, but it could be set to one of the other levels. A user will start at the default throttling level when he logs in and requests objects, and may subsequently ask to go up or down a level, go to the minimum or maximum level, or stop or start updates.

- Use the following parameters in the configuration file *rttpd.conf* to configure throttle levels.

object-throttle-times — An array of throttle times in seconds.
See page 190

Acceptable values are positive numbers, 0 and "stopped" or "paused". Client applications select one of these throttle times by choosing a throttle level; each level corresponds to an entry in the array, with level 0 being the first, level 1 being the second and so on.

Setting the level to "'stopped" or "paused' means that clients are allowed to pause objects, therefore receiving no updates until the object is unpaused.

Note: *The array must be in ascending order of throttle times, and if you use "stopped" or "paused" it must be the last entry in the array.*

Example:

```
object-throttle-times  0 0.5 1 2 3 4 stopped
```

This will result in all objects having a minimum setting of 0 seconds (no throttling) and a maximum of 4 seconds.

object-throttle-default-level The throttle level that all users start at on login. The value defines the throttle level, not the throttle time. The time of each throttle level is defined in the object-throttle-times array.
See page 190

Given the example above:
object-throttle-times

0	0.5	1	2	3	4	stopped

Throttle level

0	1	2	3	4	5

If **object-throttle-default-level** is 0 (the default level), throttling will start at 0 seconds.

object-throttle-off Turns the throttling capability off.
See page 190

Configuring "bursts"

The efficiency of the Liberator can be increased by writing user output in defined "bursts", or “batches”. However, employing bursts can result in screen updates occurring in obvious pulses.

- Adjust the following parameters in the configuration file *rttpd.conf* to achieve an acceptable level of both performance and display.

burst-min Starting point in seconds of client update buffering (i.e. start of burst).
See page 236

burst-max Maximum time in seconds of client update buffering. Benchmark testing has shown that a burst-max of 0.5 seconds provides the best compromise between performance and display.
See page 236

Configuring buffering

Adjusting the way memory is pre-allocated enables you to adjust the speed at which the cache is read, and so control the trade-off between memory and performance.

- Use the following parameters in the configuration file *rttpd.conf* to set buffering levels.

buf-cache-size Overall size of the buffer cache in megabytes. On top of this the Liberator will use about 15Mb for core memory, and this memory requirement will increase as the amount of users and data increase. The suggested maximum is 512Mb.
See page 236

buf-elem-len Length of standard buffer element, in bytes. See page 236

output-queue-size The number of update messages the Liberator will store per client (maximum is 4096). The main use for this parameter is when you reconnect, as Liberator stores any messages that might have been missed.

The queue size could be increased if there are lots of reconnects or if your data updates fast and the queue fills quickly. See page 236

newsitems-saved Maximum number of news items (headlines) that Liberator stores in memory. See page 240

Returning news to clients

- Use the following parameters in the configuration file *rttpd.conf* to configure how Liberator returns news headlines to clients.

newsitems-max Maximum number of news items that the Liberator will send to any particular client for any one request.
See page 240

news-datetime-format The time string format used for news headline items (for further information please refer to strftime-within your Unix manual).
See page 241

Note: *Some data sources may override this by sending their own datetime string.*

6.9 Configuring write failure actions

If either the Liberator's output buffer is full or the RTTP client cannot read updates fast enough, updates for that client will fail. Liberator will continue to attempt to write to the client, using up system resources.

You can control this by adjusting how large the output queue can get before the Liberator stops trying to update that client and either kicks them out or checks its buffer.

- Use the following parameters in the configuration file *rttpd.conf* to configure how Liberator will check for write failures.

session-max-queue-length The size the queue in the server waiting to be sent to the client must reach before the server starts counting consecutive increases to the queue length.
See page 235

session-max-queue-count This is the number of consecutive times the queue length in the server has to increase after the session-max-queue-length has been reached before the connection is dropped.
See page 235

7 Authentication and entitlement

7.1 Overview

Liberator supports a modular system for handling authentication of users and entitlement of objects. This allows users to be authenticated, objects to have permissions loaded, read and write permissions for a user to be checked and object name mappings to be performed.

- Authentication is the process of determining whether someone is who they say they are. In networks such as the Internet, authentication is commonly done through the use of logon passwords: knowledge of the password is assumed to guarantee that the user is authentic. The user must know and use the declared password.

- authorization or entitlement is the process of giving someone permission to do or have something. A system administrator defines which users are allowed access to which files. authorization is sometimes seen as both the preliminary setting up of permissions by a system administrator and the actual checking of the permission values that have been set up when a user is getting access.

For details on how to create your own Auth Modules, refer to the companion document **Liberator Auth Module SDK Developer's Guide**.

7.2 Using auth modules

An Auth Module provides a means performing authentication and authorization.

Specifying the Auth Module to use

- Use the following parameters in the configuration file *rttpd.conf* to identify the location of Auth Modules.

auth-moddir Directory from where authentication modules are loaded.
See page 200

auth-module Name of authentication module to use.
See page 200

add-authdir An HTTP-authenticated directory. Using HTTP authentication realms is a way of naming an area of the website. If a client tries to enter a different part of the site which is protected by the same realm they will be let in automatically, but you can configure different directories with different realms for different users.
See page 176

Example:

```
add-authdir
      name        /status
      realm       Liberator Admin
      username    admin admin2
      password    admin admin2
      username    admin3
      password    admin3
end-authdir
```

In this example, the /status folder can only be used by people with the following login details:

Username	Password
admin	admin
admin2	admin2
admin3	admin3

Configuring user numbers

- Use the following parameters in the configuration file *rttpd.conf* to configure the numbers of users allowed.

max-user-limit Number of users allowed on the Liberator. This enables you to set a maximum at a level less than the license allows if desired. The default setting of 0 means there is no limit.
See page 201

max-user-warn Specifies the number of users at which a warning about the number of users approaching the maximum (set by max-user-limit) will be logged to the event log (see page 201). A warning will only be logged again if the number of users drops below the max-user-ok level.
See page 200

max-user-ok Specifies the number of users at which a message confirming that the user level is acceptable will be logged to the event log. The default setting of 0 corresponds to 90% of max-user-warn.
See page 200

A message will only be logged if a warning about the number of users has previously been logged.

Waiting times for authentication

- Use the following parameters in the configuration file *rttpd.conf* to configure how long Liberator should wait for a authenticated message from an Auth Module when there is a delay.

auth-login-timeout Timeout period in seconds when logging in and auth_new_user returns AUTH_DELAYED, which means that there is no blocking while a database is accessed or any other other blocking call is made. (auth_new_user is a function in the Liberator Auth Module SDK which authenticates a user).
See page 201

auth-map-timeout Timeout period in seconds when requesting a mapped object and auth_map_object returns AUTH_DELAYED (auth_map_object is a function in the Liberator Auth Module SDK used to deliver renamed objects to users without them seeing the new name).
See page 202

session-timeout Sets the time in seconds for which the Liberator will maintain a session if a user has connected but not managed to log in.
See page 205

Reconnecting

By default, Liberator uses the Auth Module to check a user's authentication when they attempt to reconnect, but this functionality can be disabled.

- Use the following parameters in the configuration file *rttpd.conf* to configure how to authenticate users who are reconnecting to Liberator after a connection failure.

noauth-reconnect Set to TRUE for Liberator to compare the user's username and password with those used on the previous session and not request authentication from the Auth module.
See page 204

session-reconnect-timeout Sets the time the Liberator will maintain a session for after a disconnection, to enable the user to reconnect without a new authentication request being sent to the Auth module.
See page 205

7.3 Liberator's standard auth modules

Liberator is equipped with three standard Auth Modules, openauth, cfgauth and xmlauth. Additionally, the javaauth module (which can be purchased separately) allows the Liberator to connect to authentication modules which can be build using the java auth SDK.

XMLauth

This module enables programmers and system administrators to use XML to create their own permissioning structures and control entitlement to objects held on the Liberator.

As XMLauth is more complex than the other standard modules, there is an accompanying document XML Auth Module User Guide which must be referred to for instructions on how to use this module.

openauth

This is the simplest Auth Module possible and is used for systems where no authentication or authorization is needed.

openauth will allow any username to enter the system and with any password. It can also specify whether all users have either or both read and write access to any object in the system.

To use openauth:

- Set auth-module to openauth (see “Auth modules” on page 200).

openauth uses its own configuration file *openauth.conf* to set the users' permissions. There are two configuration options in this file.

The default values for these options are used if no configuration file is present.

read-access Determines all users' read access to objects.
See page 247

write-access Determines all users' permission to write to or create any object.
See page 247

Example *openauth.conf* file

```
read-access      1
write-access     1
```

cfgauth

This module allows the number of users and the types of objects they can read to be configured.

This module is intended for relatively low numbers of users where the usernames and other details do not need to be changed often.

To use cfgauth:

- Set **auth-module** to cfgauth (see **auth-module** on page 200).

cfgauth uses its own configuration file *cfgauth.conf* to set up the users. There are two main configuration options in this file.

add-user Identifies a user and their required password and permissions.
See page 248

encrypted-passwords A global option to determine whether a password is encrypted or not.
See page 250

Note: *This changes the way the passwords are read from the configuration file, not the way they are transmitted across the network.*

Example *cfgauth.conf* file

```
encrypted-passwords     0

add-user
      username          user1
      password          pass1
      read              0 20 21 22
      licenses          2
end-user
```

javaauth This optional module allows the Liberator to connect to user defined authentication modules created using Caplin's java auth SDK - see "Appendix C: Javaauth configuration" on page 277 for details on how to configure **javaauth** to be able to connect to your module.

7.4 Signature authentication

- Use the following parameters in the configuration file *rttpd.conf* to configure how to authenticate users' signatures.

signature-validtime How long a generated signature is valid for, in seconds.
See page 243

signature-hashsize Size of hashtable for storing signature keys.
See page 243

add-sigkey Adds a signature checking key to the configuration file.
See page 243

These only come into play if an Auth Module is using Liberator's signature checking system. Liberator can check signatures produced by the Caplin KeyMaster product, which integrates with single sign on systems.

7.5 External authorization using permissions objects

Standard user permissioning as defined in Auth Modules allows you to determine a user's read and write access to objects. Liberator also supports the use of permissions objects.

As an alternative, or additionally to, controlling user permissioning through the standard Auth module configuration, an external Integration Adapter can authorize access to objects in real-time by sending permissions objects to a customized auth module in the Liberator.

A permissions object can contain structured authorization information ("permissioning" data) that is available to the custom Liberator auth module. Such an object is usually generated by a custom Integration Adapter, and the format and meaning of its contents are determined by this Integration Adapter. Updates to the object are sent to the custom Liberator auth module, which must be coded to understand the contents of the object and act on them accordingly, for example by updating the permissions for a user.

Client applications can also make use of permissions objects. A client can subscribe to particular permissions objects and receive updates to them from the Liberator, through the standard update mechanism. The client can then use the permission information to control the way the application behaves.

As an example, a back-end trading system could generate information that authorizes users to trade on objects using particular trading models, such as ESP (Executable Streaming Protocol) or RFS (Request for Stream). The custom Integration Adapter sends this authorization data to the Liberator as updates to permissions objects. The custom auth module in Liberator receives the permissions objects and uses them to manage changes to the trading permissions for each user. It also passes the changes on to the subscribing client. The client application alters the appearance and behaviour of the user's trading interface according to the changes in the permission object; for example it might need to disable the button that allows the user to trade using ESP.

The meaning of a permissions object and the actions that are taken on it are not predefined. To implement authorization using permissions objects you must design and write custom code. You would typically need to do the following:

- Define the permissions objects you require and the format and meaning of their content.
- Write a new Integration Adapter (or modify an existing one) to generate the permissions objects and updates to them.
- Write a custom Liberator auth module that can interpret updates to the permissions objects and can change authorizations accordingly.
- You may also want to write client code that subscribes to the permissions objects, interprets updates to them and changes the application behaviour accordingly.

8 Communicating with sources of data

The Liberator is capable of requesting and retrieving data from any application using the dataSource protocol which enables most Caplin and RTTP-related products to communicate with each other. These products are called DataSource peers (they are typically Integration Adapters).

8.1 What is a DataSource peer?

A DataSource peer is an application or feed handler, installed remotely, which another DataSource peer can receive data from and send to. Liberator incorporates a DataSource peer in order to request data from other DataSource applications, such as Integration Adapters .

As well as being a source of data, DataSource can act as a destination for data sent from other DataSource applications. This means the link between peers is bidirectional, as shown in Figure 8-1 below.

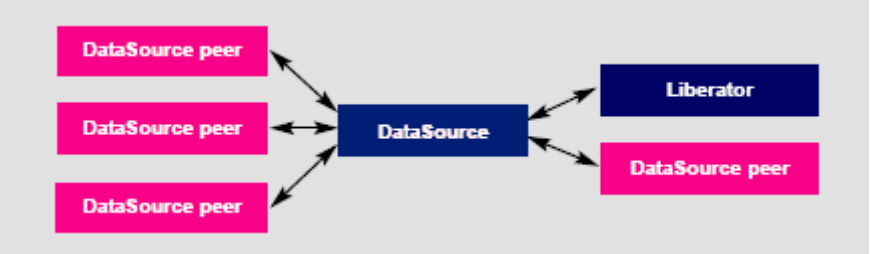


Figure 8-1: DataSource acting as a data source and data sink

There are two types of DataSource peer:

- Active DataSources, which will accept requests for objects. Active sources keep track of which objects have been requested and send updates for those objects only.

  In Liberator, an object that has been obtained by requesting it from an active DataSource is called an **active object**.

  Objects may be discarded as well as requested. This tells the source that we no longer wish to receive updates for this object.

  When a user requests an object, and the Liberator does not already have it, it will request it

from one or more of its active sources. If another user requests that object Caplin Liberator will already have all the information it needs, and will respond to the user immediately.

When a user logs out or discards an object, Liberator will send a discard message to the active DataSource (as long as no other user is viewing that object). This discard will actually take place a configurable time after the user discarded the object; this prevents objects being requested and discarded from the source unnecessarily. For more information on the relevant configuration options, see “Discarding objects” on page 120.

- Broadcast DataSources, which simply send all objects and updates to any connected peers.

8.2 Configuring Liberator to be a DataSource peer

You need to give Liberator an identifier in order for any connected peers to know which updates should be sent to it.

- Use the following parameters in the configuration file *rttpd.conf* to give a unique identifier for your Liberator.

datasrc-name The name of the Liberator, and how DataSource peers will identify it.
See page 210.

This name can be overridden by putting a value in the local-name option of the add-peer entry (see add-peer on page 211). %a represents the application name, %h the name of the host machine.

Example:

```
datasrc-name      testsrchost8
```

datasrc-id ID number of this Liberator.
See page 210

This ID can be overridden by putting a value in the local-id option of the add-peer entry (see add-peer on page 211), in which case it must match the remote-id given in the add-peer entry in the remote DataSource peer's configuration.

8.3 Connecting to DataSource peers

- Use the following parameters in the configuration file *rttpd.conf* to identify peers and configure how they connect.

datasrc-interface Network interfaces to listen for connections from DataSource peers.
See page 210

datasrc-port Network port to listen for connections from DataSource peers. The default of 0 means that no connections can be made to the Liberator.
See page 211

datasrc-sslport Network port to listen for SSL connections from DataSource peers. The default of 0 means that no SSL connections can be made to the Liberator.
See page 211 and “Making SSL connections with DataSource peers” on page 123

add-peer Identifies a DataSource peer which can be communicated with. This entry includes the ID number and name of the DataSource peer, and the ID number and name of Liberator, which is sent to the DataSource peer in order to identify your Liberator.
See “Defining datasource peer connections” below, and the options in “add-peer” on page 211.

Defining datasource peer connections

For each DataSource peer that communicates with the Liberator specify an **add-peer** entry in *rttpd.conf*. If the DataSource peer initiates the connection (so the Liberator accepts the connection request), the entry must include a **remote-id** option and optionally a **remote-name** option, as in the following example.

```
add-peer
      remote-name       DataSource_1
      remote-id         1
      ...
end-peer
```

The DataSource peer's configuration should include:

- A **datasrc-id** that matches the **remote-id** in the Liberator's **add-peer** configuration entry, and an optional **datasrc-name**.
- An **add-peer** entry containing **addr** and **port** options. These define the Liberator address and port to which the DataSource peer should send connection requests.

```
datasrc-name            DataSource_1
datasrc-id              1
...
add-peer
      addr              <<Liberator addr>>
      port              <<Liberator port>>
      ...
end-peer
```

When DataSource connects to the Liberator, the **datasrc-name** defined for the DataSource peer will override the **remote-name** defined in the Liberator's **add-peer** section.

If the Liberator initiates the connection to the DataSource peer, then specify the configuration the other way round. The **addr** and **port** options must be in the Liberator configuration and specify the connection address and port for the DataSource peer. The DataSource configuration contains **remote-id** and **remote-name** settings corresponding to the Liberator's **datasrc-id** and **datasrc-name**.

Changing the Liberator's identity in peer connections

When a connection is made between a DataSource peer and a Liberator, they exchange ids and names. The Liberator's id and name, as defined in **datasrc-id** and **datasrc-name**, are sent to the DataSource peer. Using the **local-id** and **local-name** options of the **add-peer** entry you can override the Liberator's id and name for that particular peer, as in the following example.

```
datasrc-name            Liberator_A
datasrc-id              2
...
add-peer
      local-name        Liberator_A1
      local-id          3
      remote-name       DataSource_1
      remote-id         1
      ...
end-peer
```

When a connection is made to the DataSource peer, it is sent the **local-id** and **local-name** rather than the Liberator's **datasrc-id** and **datasrc-name**. This allows you to give the Liberator different identities as seen by different DataSource peers.

Multiple connections to a DataSource peer

You may want to configure more than one connection to a single DataSource peer, for example to improve performance by utilizing multiple DataSource threads (see "Improving performance using threads", "DataSource threads" on page 152). To do this you must modify both the Liberator configuration in *rttpd.conf* and the DataSource configuration.

Assuming the DataSource peer initiates the connection to the Liberator (this is usually the case):

DataSource configuration for multiple peer connections

For each connection to the Liberator specify an **add-peer** entry with a **local-id** option and an optional **local-name** option. The **local-id** setting must be different for each entry.

```
add-peer
      local-name        MyDataSource_connx_1
      local-id          1
      addr              <<Liberator addr>>
      port              <<Liberator port>>
      ...
end-peer

add-peer
      local-name        MyDataSource_connx_2
      local-id          2
      addr              <<Liberator addr>>
      port              <<Liberator port>>
      ...
end-peer
```

Note: *The **addr** and **port** options are the same in each add-peer entry, since they are the address and port on which the Liberator listens for connection requests.*

Liberator configuration

For each connection to the DataSource peer specify an **add-peer** entry with a **remote-id** option. The **remote-id** settings should correspond to those of the **local-id** options in the DataSource configuration.

```
add-peer
      remote-name       MyDataSource_connx_1
      remote-id         1
      ...
end-peer

add-peer
      remote-name       MyDataSource_connx_2
      remote-id         2
      ...
end-peer
```

As far as the Liberator is concerned this configuration is the same as that for accepting connections from two different DataSource peers. At run time the Liberator will accept connections from peers with ids 1 and 2, and will be unaware that it is the same DataSource peer at the other end of the two connections.

If the Liberator initiates the connection to the DataSource peer, specify the configuration the other way round; the **local-id** settings must be in the Liberator configuration and the **remote-id** settings must be in the DataSource configuration. In this case the values in **local-id** will override the Liberator's global id number defined in **datasrc-id**, and the **local-name** settings will override the Liberator name defined in **datasrc-name**.

Enabling failover

Liberator knows a peer is down when it loses its network connection to the peer or it fails to receive heartbeat signals from that peer (heartbeats are explained in more detail in "Monitoring system health using heartbeats" on page 144).

The **add-peer** entries can be used to set up the Liberator to allow a data source failover and enable Liberator to connect to alternative data sources when required. A single **add-peer** section can configure a set of alternative peers to connect to using the addr and port options.

This can be configured by commenting out all the **add-peer** options and using the default settings with the exception of the following options:

addr	must have at least one data source identified to failover to. If more are specified, then the Liberator will try the first source, and if that fails too, it will try the second and so on.
port	each data source identified in the addr option must have a port specified.

Liberator will connect to the first addr and port in the list and failover to the others in order if it cannot connect to the preceding peer in the list. Having established a connection with another source, it will continue to request data from it until that connection fails and it attempts to connect to the other sources in order again.

The following example allows failover to 4 data sources; the Liberator will try each identified source in turn.

```
add-peer
     addr  192.168.201.245 192.168.201.245 192.168.201.245 192.168.201.245
     port  25110           25111           25112           25113
end-peer
```

Using data services, multiple peers can be configured for failover without Liberator needing to swap connections. See “Data services” on page 113.

- Use the following parameters in the configuration file *rttpd.conf* to determine whether Liberator ignores extra connection attempts by a user.

datasrc-reject-new-peers	If a DataSource peer tries to connect to the Liberator but there is already one connected with the same id (for example, if a peer's firewall has been down and the peer is registered as connected but in fact is not), the current peer will be disconnected and the new one is allowed to connect. datasrc-reject-new-peers turns off this default behaviour so the new DataSource peer is not allowed to connect. See page 210

- To configure the timing of heartbeats between DataSource peers use the ***heartbeat-time*** and ***heartbeat-slack-time*** options of the ***add-peer*** configuration entry. See page 214.

- Use the following parameters in the configuration file *rttpd.conf* to clear specific types of data when failing over to another peer or reconnecting to the same one.
  This allows cached data to be refreshed from the new DataSource peer.

record-clear-type1-on-failover Clear Type 1 data for active objects.
See page 199.

record-clear-type2-on-failover Clear Type 2 data for active objects.
See page 199.

record-clear-type3-on-failover Clear Type 3 data for active objects.
See page 199.

8.4 Reconnecting peers using the UDP interface

Liberator includes a UDP command interface that enables you to send a UDP message to reset peer connections after failover.

- Include the following options in the file *rttpd.conf* in order to use the UDP interface.

udp-port Port to listen on for UDP messages. If not specified then udp signals are disabled.
See page 246

udp-interface Network interface to listen on for UDP messages.
See page 246

The following UDP command can be sent over a Liberator's UDP interface.

peer-reconnect An instruction to attempt to reconnect with the specified peers. If several DataSource peers have been configured to be used as alternative or failover sources, this enables your application to reconnect to previously failed peers if they are now online. By default , the first failover address is reconnected to, if no number is given:

Syntax: ***peer-reconnect peers addr-num***

Parameter:

Name	Type	Description
peer	int	Datasource peer index which should be reconnected to after failover. ***Note:*** *These are not DataSource IDs (specified by datasrc_id parameter in the configuration file), but correspond to the order of the peers' add-peer entries in the configuration file. The first add-peer is for peer 0, the next peer 1 and so on.*
addr-num	int	Which address in the failover list to reconnect to. Defaults to the first in the list.

- For how to issue the UDP command, see the section “UDP commands” on page 144.

8.5 Data services

Note: *Data services replace the old Source Mapping feature.*

You must use data services in order for Liberator to request a particular object from a particular DataSource peer or to define where broadcast data can come from. Data services allow you to define where data comes from, based on its subject name. They also allow the definition of groups of peers in a way that allows priority, failover, and load balancing.

A data service defines the following:

- a name, which is the identifier for the service;
- a regular expression pattern match on the object name, or a number of patterns - this defines which objects will come from this service;
- a DataSource peer or set of peers that the request for the object will be forwarded to.

The DataSource peers defined for a service allow a number of different structures. Each service can have a number of ‘source groups’. Within a source group a number of priority groups can be defined, and within those priority groups, lists of peers can be defined.

When an object needs to be requested from a service and Liberator first looks at the service groups, it will make a request to a peer from each group at the same time. This may be useful if you do not know which peer has the data, or if a peer is serving a different set of fields and the data needs to be merged together.

Within a source group Liberator will look at the first priority group and request from a peer in that priority. If there are multiple peers in the priority group, Liberator will send the request to the peer with the smallest number of existing subscriptions – this achieves load balancing across peers. If no peer is connected in that priority, or if the peers in that priority did not have that object, the Liberator will try the next priority group – this achieves failover.

Active data services are identified within the data service section of the *rttpd.conf* configuration file. How these are configured is detailed in “Data services” on page 222.

Specifying the object or objects

Examples of different applications of mappings are given below.

For example:

```
include-pattern "^/NA/"
```

would request any object starting with the characters /NA/

```
include-pattern "^/[A-M]"
```

would request any object starting with the characters /A to /M

```
include-pattern "ABC"
```

would request any object containing “ABC” in any part of the name.

Note: *Remember that this is a regular expression and should start with a “^” if the pattern should only match from the beginning of the object name.*

Specifying a single DataSource peer

The DataSource peers to be mapped are specified by adding them as labels (see “Data services” on page 222).

For example:

```
add-data-service
      service-name            MyService
      include-pattern         ^/NA/
      add-source-group
            required          true
            add-priority
                  label       sic2
            end-priority
      end-source-group
end-data-service
```

would request any object starting with the characters /NA/ from the DataSource peer with ID sic2.

Specifying alternative DataSource peers

By sending your requests to a sequence of DataSource peers, you can ensure that no individual peer is overloaded. This is particularly useful when a number of peers hold similar data.

Enter alternative DataSource peers within the same priority group (see “Data services” on page 222).

For example:

```
add-data-service
      service-name            MyService
      include-pattern         ^/NA/
      add-source-group
            required          true
            add-priority
                  label       src1
                  label       src2
                  label       src3
            end-priority
      end-source-group
end-data-service
```

This means each request that matches "^/NA/" will go to one of the DataSource peers src1, src2, or src3. The request is directed to the peer with the smallest number of existing subscriptions, thus spreading the the load evenly across the peers.

Specifying multiple DataSource peers

To send the same request to more than one DataSource peer, enter more than one source group (see “Data services” on page 222).

For example:

```
add-data-service
      service-name           MyService
      include-pattern        ^/NA/
      add-source-group
            required         true
            add-priority
                  label      src1
            end-priority
      end-source-group
      add-source-group
            required         true
            add-priority
                  label      src2
            end-priority
      end-source-group
end-data-service
```

This will mean any request starting "/NA/" will be sent to DataSource peer src1 and peer src2 at the same time.

Note: *If both DataSource peers reply with data then the updates will be duplicated, so this configuration should not be used if both peers have the same data. This combination is more likely to be useful when multiple peers hold different data and you are not sure which peer has what data.*

Specifying priority or failover

You can configure a data service to send to an alternative DataSource peer if your first choice of peer is down, for example:

```
add-data-service
      service-name           MyService
      include-pattern        ^/NA/
      add-source-group
            required         true
            add-priority
                  label      src1
            end-priority
            add-priority
                  label      src2
            end-priority
      end-source-group
end-data-service
```

This will only request from peer src2 if peer src1 is down.

More complex mappings

More complex combinations of DataSource peers can be defined. For example:

```
add-data-service
      service-name           MyService
      include-pattern        ^/NA/
      add-source-group
            add-priority
                  label      src1
                  label      src2
            end-priority
      end-source-group
      add-source-group
            add-priority
                  label      src3
                  label      src4
            end-priority
      end-source-group
end-data-service
```

This results in the server sending requests to two DataSource peers simultaneously, one to whichever of src1 or src2 has the smallest number of existing subscriptions, and one to whichever of src3 or src4 has the smallest number of existing subscriptions.

Waiting for responses

Use the following parameters in the configuration file *rttpd.conf* to set the timeout period to wait for responses from a peer following a request for data.

service-request-timeout	Time in seconds that the Liberator will wait for a Service to answer a request—after this time the Liberator will send a discard to all peers that have not responded.to request from another peer if the service defines a suitable alternative. A discard is sent to the DataSource peer to cancel the timed out request. This value can be overridden for an individual service by using the request-timeout option of the ***add-data-service*** entry (see page 224).
source-request-timeout	Time in seconds that the Liberator will wait for an individual DataSource peer to answer a request—after this time Liberator will attempt to request from another peer if the service defines a suitable alternative. A discard is sent to the DataSource peer to cancel the timed out request. This value can be overridden for an individual source by using the ***request-timeout*** option of the ***add-peer*** entry (see page 211).

Allowing open subscriptions

Liberator supports open subscriptions. That is, it can keep a user's subscription request open even if the DataSource peer that can satisfy the request is down when the request is made. The request is completed once the DataSource peer is available.

- To allow all subscription requests to be open, set **service-request-timeout** to a very high value, or to -1 (no timeout set). (See page 223.)

- To allow open subscription requests for just a particular data service, set the **request-timeout** option of the **add-data-service** configuration item for the data service to -1 (no timeout set). (See page 224.)

- In either case, set the **remote-type** option for the **add-peer** configuration item of the DataSource peer(s) that must satisfy the open subscription requests. Its value must match the value of the **local-type** option in the DataSource peers's **add-peer** configuration for the Liberator (which would be either "active" or "active|contrib").

  This allows the Liberator to determine that a DataSource peer is an active source in advance of it connecting to the Liberator. The Liberator can then keep subscription requests for that DataSource peer open, even if the DataSource peer has not yet connected.

Discarding objects

In the life cycle of an active object there may be a point when no users are viewing it. When this happens, Liberator will delete the object from its cache, and send a discard instruction to the DataSource peer from which it originated so as to cancel the request for the object. To prevent unecessary discarding and subsequent re-requesting of objects, there are a number of configuration parameters that can be set in *rttp.conf* to delay the discard action. These are:

active-discard-timeout	Time in seconds that the Liberator will hold on to an active object after the last user stops viewing it. After this time Liberator will also send a discard instruction to the peer to cancel the request. See page 190.
discard-timeout option of add-data-service	Behaves in the same way as ***active-discard-timeout***, but applies only to active objects obtained through a particular data service. This option overrides the value of ***active-discard-timeout***. See “add-data-service” on page 224, “discard-timeout” on page 226, and “Data services” on page 113.
discard-timeout option of add-object	Behaves in the same way as ***active-discard-timeout***, but applies only to objects in a directory that has been defined using the ***add-object*** configuration parameter. This option overrides the value of any settings of the ***discard-timeout*** option for the data service that fetches data for the object. It also overrides ***active-discard-timeout***.

8.6 Replaying data from peers into Liberator

The DataSource Auto Replay capability means that previously-sent data can be reprocessed by the Liberator stepping through its log files and replaying the data. Auto Replay is useful following a period when the Liberator was down, as replaying data can return it to the state immediately before it was shutdown.

- Use the following parameters in the configuration file *rttpd.conf* to configure how Liberator replays data to clients.

datasrc-auto-replay	Time (in minutes after midnight) that the server should load previously received messages on a restart. If the number is negative it represents the number of minutes back from the current time. See page 221. Only peers with the recvautoreplay (4) flag set in the local-flags entry of add-peer will receive the Auto Replay data (see add-peer on page 211).
datasrc-auto-replay-days	The number of whole days to go back from the time indicated by datasrc-auto-replay (if less than 1440). See page 221.
datasrc-auto-replay-files	By default DataSource will only replay the current packet log. Use datasrc-auto-replay-files to specify a list of log files to replay. See page 221. If the files are specified without an absolute pathname, the order in which they will be searched for is: 1 Liberator root directory 2 the directory containing the current packet log 3 the log root directory You must include the current packet log. The list of log files must be in order of age, with the oldest first.

Example:

```
datasrc-auto-replay-files  packet-rttpd.old packet-rttpd.log
```

Replaying news headlines

- Use the following parameters in the configuration file *rttpd.conf* to configure how Liberator replays news to clients.

news-replay	Time (in minutes after midnight) that the server should start replaying news headlines on a restart. If the number is negative it represents the number of minutes back from the current time. See page 242. You must give news-log a value to use news-replay.
news-replay-days	The number of whole days to go back from the time indicated by news-replay (if less than 1440). See page 242.
news-replay-files	An array of strings which identifies the news logs to replay. By default DataSource will only replay the current news log (as defined by news-log). See page 242. If the files are specified without an absolute pathname, the order in which they will be searched for is: 1 Liberator root directory 2 the directory containing the current news-log 3 the log root directory You must include the current news log. The list of log files must be in order of age, with the oldest first.

Example:

```
news-replay-files    news.old  news.log
```

8.7 Making SSL connections with DataSource peers

SSL certificates can be configured at either or both client and server ends of the channel—Liberator is said to be operating in server mode when accepting connections from DataSource peers, and in client mode when connecting to DataSource peers.

There is no fallback to non-SSL operation should the SSL connection fail to be established.

- Edit the following parameter in the file *rttpd.conf* to configure SSL certificates.

start-ssl	Configures the SSL connection when setting up Liberator to be both client and server ends of an SSL channel. This group is needed in the configuration file of both client and server applications. See page 217.
ssl-passwordfile	Identifies the file containing the SSL certificate passphrase. See page 220.

Server mode only configuration

- To configure Liberator for SSL when in server mode, use the datasrc-sslport option to select the network port to listen for SSL connections from DataSource peers (see page 211).

- It is possible for DataSource to accept both SSL and non-SSL connections on different ports. Non-SSL connections should be configured using the datasrc-port option (see page 211).

Client mode only configuration

- To configure Liberator for SSL when in client mode, use the ssl option in the add-peer entry for the DataSource peer that acts as server. For more information see add-peer on page 211.

Sample certificates and certificate authorities

The sample SSL configuration found commented out in *rttpd.conf* uses certificates and certificate authorities which are already set up in the Liberator kit in the directories *etc/certs*, *etc/demosrcCA* and *etc/rttpdCA*. These were created using the OpenSSL toolkit (for more information see *www.openssl.org*).

The certificates and certificate authorities use the following passphrases:

Liberator certificate:	rttpdcert
Demonstration feed certificate:	demosrccert
Liberator certificate authority:	rttpdCA (you will need this if you create new data source certificates)
Demonstration feed certificate authority:	demosrcCA (you will need this if you create a new certificate for the Liberator.)

By default Liberator will look for passphrases in the files *etc/.rttpd.ssl.pass* and *etc/.demosrc.ssl.pass*. If these files are not present a password prompt will be given when the Liberator starts. It is therefore possible to echo the password into the application on startup: to achieve this the standard startup script should be changed.

9 Monitoring performance

The status of the Liberator can be monitored in three ways:

- by using the monitoring and management subsystem;
- by viewing the contents of log files;
- by viewing the status web page, Liberator Explorer, or the Object Browser.

9.1 Monitoring and management subsystem

Liberator supports monitoring and management via a plug-in system. This is an additional licensable feature. The monitoring subsystem allows the user to monitor many different aspects of the Liberator including the objects currently requested, the users that are currently connected and the peers that are configured. There are two monitoring plug-ins available:

- **JMX Monitoring**: Uses JMX (Java Management Extensions) to provide an interface to the monitoring subsystem. This module allows any standard JSR160 JMX client to access information and operations exposed by the system. The Caplin Management Console uses this JMX monitoring plug-in. There are also provided sample Java JMX command-line applications. A number of modifications to the configuration file are needed in order to enable JMX monitoring.
- **Socket Monitoring (sockmon)**: A simple command-based socket protocol, similar to ftp, that allows access to the information and operations exposed by the system.

For more details, please refer to the **Management and Monitoring Overview** document that is provided with the Liberator kit.

9.2 Log files

Liberator creates several log files when it runs. The format and content of messages written to the log files are described in **"Appendix B: Log file messages and formats"** starting on page 262.

Liberator can produce very large amounts of log data, depending on how much traffic it is handling and what log files and log levels are enabled. If log files consume most of the available disk space, Liberator's performance can degrade badly. *Therefore Caplin recommends that you regularly monitor the disk space being used by the log files*.

Archive or delete old log files as needed, so that Liberator does not run out of disk space. Old log files should normally be archived so that they are available for diagnostic purposes.

Log file configuration

- Specify the directory where log files will be created using **log-dir** (see page 168).
- Specify the name of each log file using the configuration items listed in Table 9-1.

By convention, log filenames have the following format:

<log type>-<application name>.log.

A log filename can be specified with the following parameters, which are substituted with their actual values when the file is opened:

- **%r** can be used to represent the application-root (see page 165)
- **%a** can be used to represent the application-name (see page 165).
- For example:

```
event-log event-%a.log
```

Because Liberator's application mame is "rttpd", this configuration item names the Liberator event log file as *event-rttpd.log*.

Log type	Configuration item that defines the filename	Default file name	Log contains
Event	**event-log**	event-rttpd.log	Messages about starting up, shutting down, and connections to data sources as well as extra general and debug information.
HTTP	**http-access-log**	http-access-rttpd.log	Each HTTP request to the server.
HTTP errors	**http-error-log**	http-error-rttpd.log	Each HTTP request resulting in an Object not found error.
Packet	**datasrc-pkt-log**	packet-rttpd.log	Each packet received from a data source.
RTTP Request	**request-log**	request-rttpd.log	Each RTTP request made to Liberator.
Session	**session-log**	session-rttpd.log	Messages re client connections, disconnections, logins and logouts.
Object	**object-log**	object-rttpd.log	All request and discard commands for objects, and whether those commands were successful.
News	**news-log**	[no news headlines stored]	News headlines for replaying on startup.

Table 9-1: Configuring log files

- For information about log levels see “Debugging” on page 146.

Log file cycling

You can manage the size of Liberator's log files by configuring log file cycling. Each log file is closed and renamed on a regular basis, and a new file is opened for writing – this process is called "cycling". The cycling frequency can be configured in a number of ways:

- ❖ Define a maximum file size above which the log file is cycled.
- ❖ Define a fixed time at which the log file is cycled.
- ❖ Define a time interval after which the log file is cycled.
- ❖ Define a combination of the above – the log file is cycled when any one of the criteria is met.

By default all log files are cycled at 04:00 hours each day, so that a separate log file of each type is created for each day. These default settings are specified using the following configuration items. These items apply to all log files except those that have an **add-log** configuration item set:

```
log-maxsize       0
log-cycle-time    240
log-cycle-period  1440
log-cycle-suffix  .%u
log-cycle-offset  -1
```

Note: *Often this default configuration can create large log files if your system has lots of fast moving data. It is useful to have as much log data as possible, but this configuration should be changed if the files are too big. Please contact Caplin Support if you would like advice about configuring your log files.*

- Use the following options in the configuration file *rttpd.conf* to set the same cycling format for all logs (except those that have an **add-log** configuration item set).

log-maxsize A value of 0 means log files will cycle every time they are checked irrespective of size.
See page 168

log-cycle-time A value of 240 represents 0400, as it is defined as minutes from midnight.
See page 168

log-cycle-period A value of 1440 represents 24 hours, as it is defined as minutes from midnight.
See page 168

log-cycle-suffix The default value of .%u appends the log filenames with a number between 1-7 for each day representing Monday to Sunday. The suffix is a format string which is passed to the system function 'strftime'—please refer to your operating system manuals for more information.
See page 169

log-cycle-offset A value of -1 means the offset is the same as log-cycle-period. When the log cycles at 0400 on Tuesday, the value passed to strftime will be 0400 on Monday, making timestamps in the filenames more meaningful.
See page 169

Example log cycling configuration (all logs):

```
log-maxsize       1024000
log-cycle-time    0
log-cycle-period  30
log-cycle-suffix  .old
log-cycle-offset  -1
```

This configuration results in each log file being checked every 30 minutes and moved to *logfile.old* if it is bigger than 1,024,000 bytes.

- Use the **add-log** configuration item to set the log cycling criteria for individual logs, overriding the global settings.

  An example use for this is to define the global settings to cycle the logs every night by default, but then use **add-log** to set the news log to cycle once a week, so you can replay the news log on startup and have a week's worth of news headlines available.

  add-log is defined on page 171.

Example log cycling configuration (individual log):

```
add-log
      name        event_log
      time        240
      period      10080
      suffix      .old
end-log
```

Results in the event log cycling once a week at 0400.

Note: *These configuration options can be used to define the cycling configuration for any Liberator file, including logs generated by the javaauth and XMLauth authentication modules (see "Liberator's standard auth modules" on page 100).*

System log files (syslog)

Some important log messages are also logged to the system log files.

Example system log messages:

```
Jan 1 12:00:00 lib1 rttpd[9999]: Liberator/6.0.0-1 starting
Jan 1 12:00:00 lib1 rttpd[9999]: Logging to /opt/Liberator/var
Jan 2 12:00:00 lib1 rttpd[9999]: received signal SIGTERM
Jan 2 12:00:00 lib1 rttpd[9999]: shutting down (6)
```

The syslog priority used is LOG_INFO. The syslog facility used for log messages can be configured with the syslog-facility option. The default is "local6".

Refer to your operating system manual for instructions on how to set up syslog to receive these messages.

Logging crash details

- Use the following parameter in the configuration file *rttpd.conf* to log application crashes.

catch-crash — Boolean option which turns on catching of application crashes. If set, Liberator attempts to write a message to the default event log when the application has crashed.
See page 165

This option should not be used unless log file messages are being used for automated monitoring as it can cause problems with core files being produced.

Note: *Applies to Linux platforms only.*

Note: *This feature is not reliable, because it is not always possible to catch segmentation faults and bus errors .*

Logging RTTP traffic

You can set up Liberator to log the RTTP protocol traffic between clients and the Liberator. To do this, define the naming convention for the log files (see the *rttp.conf* configuration entry ***rttp-log*** on page 203), and then define a list of user names (Liberator login names) whose RTTP traffic is to be logged.

You can specify the users either through Liberator configuration (see the *rttp.conf* configuration entry ***rttp-log-users*** on page 204), or if JMX monitoring is enabled for the Liberator, dynamically through the Logs tab on the Caplin Management Console (CMC). Additionally, the CMC's Session tab allows you to switch RTTP traffic logging on and off for existing user sessions.

The default log file naming convention causes an RTTP traffic log file to be generated for each combination of user and RTTP session, so if a user has more than one session established concurrently you can easily analyse the traffic for the individual sessions.

The format of the log file is defined on page 273.

- For more information about logging RTTP traffic, particularly about configuring user RTTP logging using the CMC and interpreting the log entries, see the Liberator document **Server-side RTTP Logging**.

Note: *It is recommended that in a live system you only turn on RTTP traffic logging for troubleshooting purposes. RTTP traffic logs can become very large very quickly.*

Note: *In a live system you should normally turn RTTP logging on and off using the Caplin Management Console. The Liberator configuration option* ***rttp-log-users*** *should only be used for debugging test installations. It permanently enables traffic logging for the specified users and the users' traffic will be logged even after Liberator is restarted. Logging can only be turned off by stopping the Liberator and changing the rttp-log-users configuration option*

- If you configure Liberator to write its log files to a directory other than the default *var* directory, make sure that you create within the new log directory a subdirectory to receive the server-side RTTP log files. The default name for this subdirectory is *rttp*, but you can change it through the definition of the RTTP log file names.

  Example:

```
log-dir %r/my_logs
rttp-log rttp_logs/RTTP_TRAFFIC_%l.%i
```

  log-dir specifies that log files are to be located in the subdirectory *my_logs* of the application root directory. **rttp-log** specifies that RTTP traffic log files are to be located in the *rttp_logs* subdirectory of *my_logs*, hence in *%r/my_logs/rttp_logs/*.
  Before starting Liberator, you would need to create the directory *my_logs* and the subdirectory *rttp_logs*.

9.3 Viewing log files: the logcat utility

Most Liberator logs are simple text files that can be viewed using a suitable text display utility or text editor. such as the Linux commands **cat**, **more**, and **vim**.

The packet logs are in binary format and must be viewed using the **logcat** utility, which is used in the same way as the standard **cat** command. **logcat** is located in the *bin* directory of the Liberator installation.

The **logcat** utility takes the arguments listed in Table 9-2 below.

logcat argument (`short-form, long-form`)	**Description**
`-h, --help`	Detailed information on logcat options.
`-F, --print-field-names`	Print the field names.
`-f, --fields-file`	Location and name of the fields file. Default value is *fields.conf* in the current directory
`-i, --print-info`	The version, type and source of the given log.
`-l, --print-flag-names`	Prints the flags.
`-t, --type`	Forces logcat to process a particular type of file. This takes an argument, currently only 'packet' is used. eg logcat -t packet mypacket.log.
`-v, --log-version`	The version of the log
`-z, --timezone`	Sets all times in the log to the specified timezone. To find the required timezone look in the system folder zoneinfo, sometimes found at */usr/share/lib/zoneinfo* or */usr/share/zoneinfo*.

Table 9-2: logcat options

Examples

The command:

```
logcat -i packet-rttpd.log
```

outputs:

```
Logcat: Log Type 'packet' Version 4 created by 'rttpd' in timezone
‘Europe/London’
```

The command:

```
logcat packet-rttpd.log
```

outputs:

```
Logcat: Log Type 'packet' Version 4 created by 'rttpd'
2011/06/25-16:46:52.528 +0100: 192.168.201.102 < PEERINFO 1
type2src-devsun1 0
2011/06/25-16:46:52.528 +0100: 192.168.201.102 > PEERINFO 0 rttpd-
devsun2 0
2011/06/25-16:47:00.000 +0100: 192.168.201.102 > SUBJREQ 1 1 /I/
VOD.L
2011/06/25-16:47:00.000 +0100: 192.168.201.102 > SUBJREQ 1 1 /I/BP.L
```

You can also use the tail command with logcat to display the last part of the log file and update the screen when more data appears.

Note: *The timezone offset is that of the local machine that the logs were written on.*

Example:

```
tail -f packet-rttpd.log | ../bin/logcat -t packet
```

To view very large packet logs it is possible to split the log into smaller files using the standard unix command ‘split’.

```
split -b 10m packet.log
```

This will split a large packet log into separate files of 10Mb each.

Note: *This command can produce a lot of files if you are not careful with the size parameter.*

You must then tell logcat that each part is a packet log as the header will now be missing.

```
logcat -t packet packet-xab
```

9.4 Liberator status web page

Liberator is supplied with a browser-based monitoring function that displays status information within a web page and enables you to monitor the usage of the Liberator, including the volumes of type 1, type 2 and type 3 data being processed.

- To view the status web page, point your browser at `http://<hostname>:8080` (where `<hostname>` is the host name or IP address of the machine you have installed Liberator on) and select the menu option Status.

Figure 9-1, Figure 9-2, and Figure 9-3 show the parts of a typical status web page.

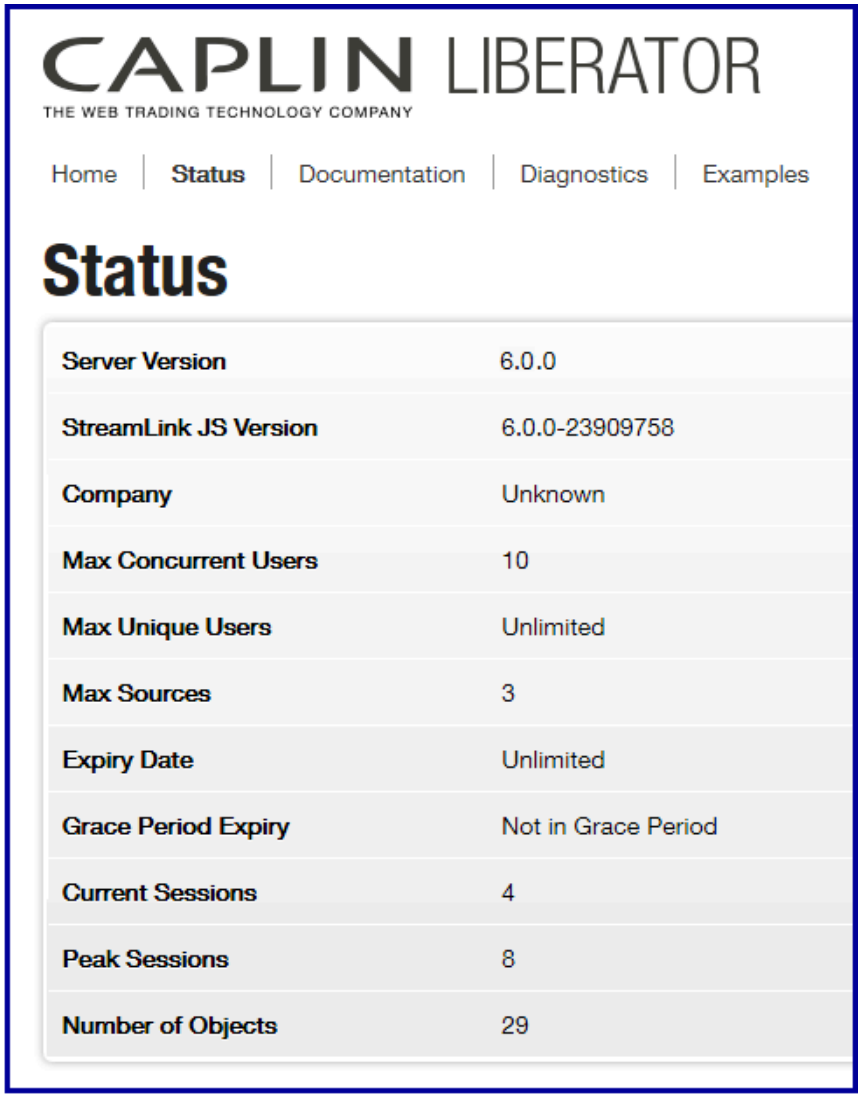


Figure 9-1: Status web page – top part

The information on the top part of the status web page is:

Liberator status information

Server Version	Release number of this version of Liberator.
StreamLink JS version	Release number of the version of StreamLink JS that the Liberator is using.
Company	The name of the company on the license agreement.
Max Concurrent Users	The maximum number of users who can be logged on to Liberator at any one time, as specified in your license agreement.
Max Unique Users	The maximum number of unique users who can log on to the Liberator over a license monitoring period (a calendar month), as specified in your license group agreement. This is the license end user limit for the Unique Users Licensing category – for more information about this, see the document **Caplin Platform: Guide to User Licensing**.
Max Sources	The maximum number of data sources that can be connected to Liberator, as specified in your license agreement.
Expiry Date	The date (month and day) when the Liberator license as a whole expires.
Grace Period Expiry	Shows whether the licence grace period has expired. In the above example, the field indicates that the license is not in the grace period. For more information about the grace period, see the document **Caplin Platform: Guide to User Licensing**.
Current Sessions	The number of user sessions currently active on the Liberator. The four columns correspond to total sessions and the number of sessions handling type 1, type 2 and type 3 data.

Peak Sessions	The maximum number of user sessions permitted on the Liberator, as specified in your license agreement. The four columns correspond to total sessions and the number of sessions handling type 1, type 2 and type 3 data.
Number of Objects	Total number of RTTP objects currently being handled.

The middle part of the status web page shows the status of the Liberator's data services and of the DataSource peers that are connected to the Liberator:

.

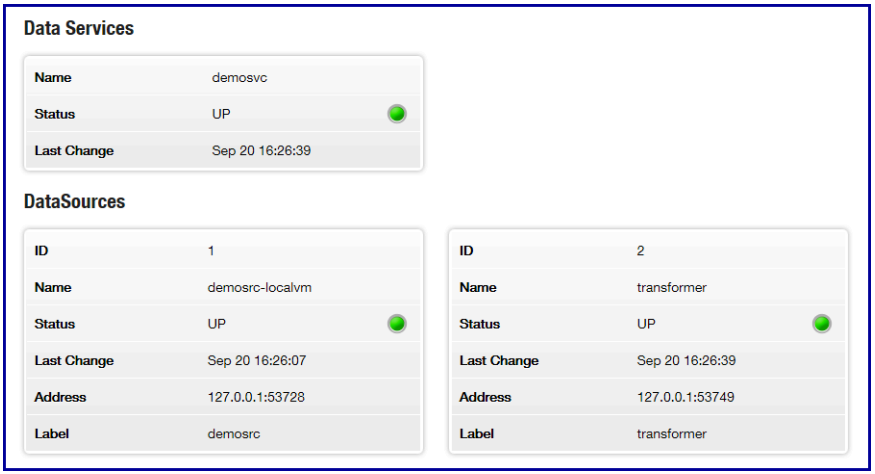


Figure 9-2: Status web page – middle part

Data services information

There is one section of information for each data service that the Liberator provides. Each section shows the following details:

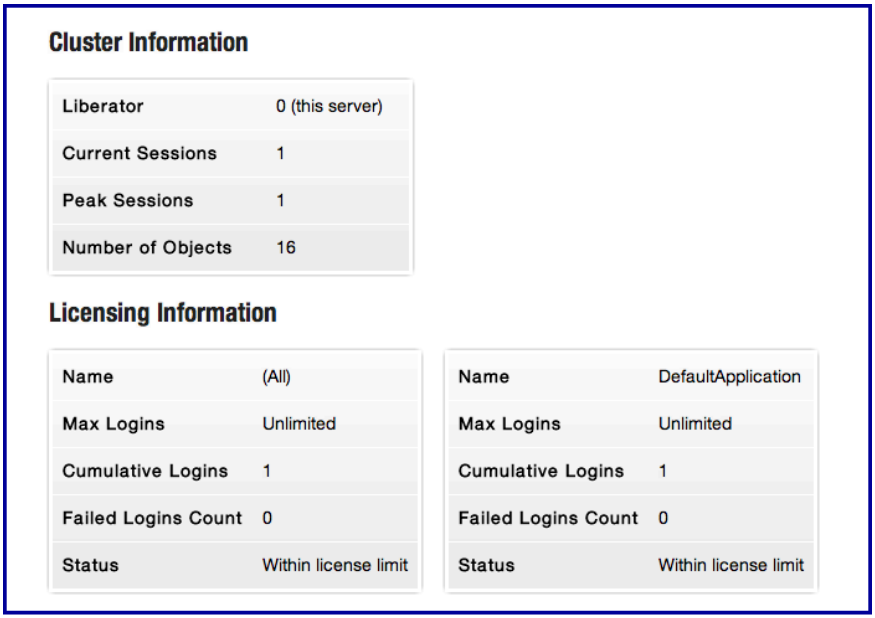
Name	Name of the data service.
Status	Status of the data service. This can be: OK if the data service is fully available (all the DataSource peers involved in providing the data service are available). LIMITED if the data service is only partially available (some of the DataSource peers involved in providing the data service are not available). DOWN if the data service is not available.
Last Change	The date and time at which the data service's status last changed.

DataSource information

There is one section of information for each DataSource peer to which the Liberator is connected. Each section shows the following details:

ID	Numerical identifier for the DataSource peer.
Name	Name of the DataSource peer.
Status	Status of the connection to the DataSource peer. This can be: UP if the connection has been established; DOWN if there was a connection, but it has been lost.
Last Change	The date and time at which the DataSource peers's status last changed.
Address	IP address and port of currently connected or most recently connected DataSource peer.
Label	The label of the DataSource peer as used in the definitions of data services that use the DataSource peer(see the ***add-priority*** configuration entry within the ***add-source-group*** configuration entry on page 227).

The bottom part of the status web page shows information about the Liberator cluster and the Liberator's license usage:



Cluster Information

Liberator	0 (this server)
Current Sessions	1
Peak Sessions	1
Number of Objects	16

Licensing Information

Name	(All)	Name	DefaultApplication
Max Logins	Unlimited	Max Logins	Unlimited
Cumulative Logins	1	Cumulative Logins	1
Failed Logins Count	0	Failed Logins Count	0
Status	Within license limit	Status	Within license limit

Figure 9-3: Status web page – bottom part

For an example and explanation of what the Licensing Information section of the status display contains, see the document **Caplin Platform: Guide to User Licensing**.

9.5 Liberator Explorer

Liberator Explorer is a diagnostic tool supplied with Liberator. It allows you to quickly and easily inspect the values of Objects in the Liberator. It is often useful to quickly look at the structure and current values of objects. For example, when you are developing an Integration Adapter, you can quickly and easily subscribe to an Object that it is producing and check that the Adapter is sending Liberator the data correctly.

- To use Lberator Explorer, navigate to Diagnostics -> Liberator Explorer and enter the name of the required object in the Subject box (for example `/EXAMPLES/BROADCAST/ABC`):

Figure 9-4 on page 142 shows an example of a Liberator Explorer page.

Liberator Explorer provides an intuitive interface, with a tree navigator on the left that you use to find the Object you wish to inspect. The selected Object is shown on the right of the page, with the Object's StreamLink callbacks for debugging purposes. A log of all events is displayed at the bottom of the page.

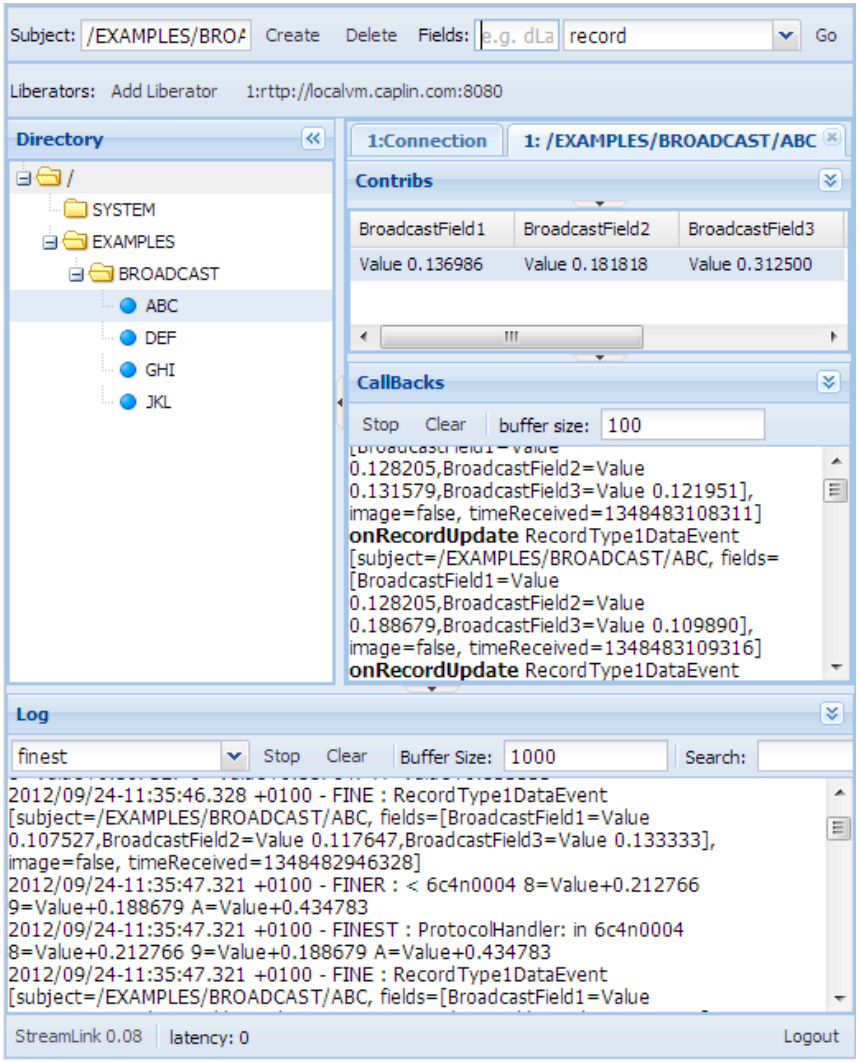


Figure 9-4: Liberator Explorer: example page

9.6 Object Browser

The Object Browser is an earlier, and simpler, equivalent of Liberator Explorer that can also be used to inspect the values of Objects in the Liberator.

- To use the Object Browser, navigate to
  Diagnostics -> Object Browsing Tool
  and enter the name of the required object in the Enter Object box
  (for example `/EXAMPLES/PRICING/TYPE1`):
  .

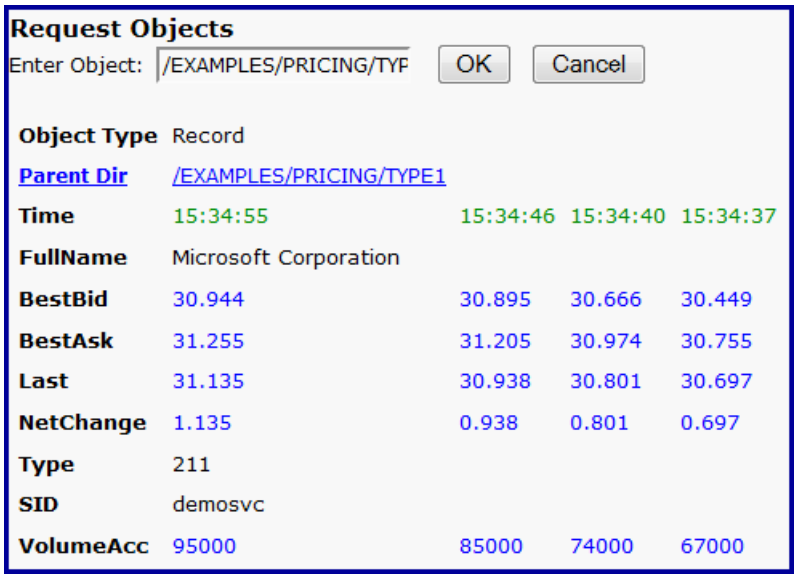


Figure 9-5: Object Browser: example page

9.7 Monitoring system health using heartbeats

- Use the following parameter in the configuration file *rttpd.conf* to configure the timing of heartbeats to client applications.

session-heartbeat — The interval in seconds between heartbeats sent from the server to the RTTP client. The value must be an integer.
See page 205

9.8 UDP commands

The Liberator includes a UDP command interface that enables you to send it UDP messages to reset peer connections after failover, and to change the verbosity of log messages.

To use this command interface you must first configure Liberator to listen for UDP messages.

- Include the following options in the configuration file *rttpd.conf*:.

udp-port — Port to listen on for UDP messages. If not specified then UDP signals are disabled.
See page 246

udp-interface — Network interface to listen on for UDP messages.
See page 246

The UDP command has the following format:

udpsend

Sends a UDP message.

Syntax: ***udpsend [-s <server-ip>] [-p <port>] message***

Parameters:

Name	Type	Default	Description
<server-ip>	string	127.0.0.1	The IP address of the machine to which the UDP message is to be sent. (This *must* be an IP address, not a host name.)
<port>	integer	10001	Port on which the Liberator listens for UDP messages. This must be the port number specified in the **udp-port** option in the Liberator's *rttpd.conf* file.
message	string	[no default]	Message to send; this can include spaces. Possible values are defined in the "Debugging" section on page 146 and the section "Reconnecting peers using the UDP interface" on page 112.

The **udpsend** utility is located in the Liberator's *bin* directory. For an example of how to use **udpsend** see the **debug** command in the "Debugging" section on page 146.

The default IP address specifies the local host, so, to send a UDP message to a Liberator located on the machine from where you are issuing the UDP command, you can just omit the *-s* parameter.

9.9 Debugging

There are several levels of verbosity of errors and events that Liberator can print to its log files. The reporting level can take any of the values shown in Table 9-3 below.

Value	Description
DEBUG	Reports all errors and events.
INFO	Reports events and information regarding normal operation and all errors included in the WARN, NOTIFY, ERROR and CRIT debug levels.
WARN	Reports minor errors and all errors included in the NOTIFY, ERROR and CRIT debug levels.
NOTIFY	Report errors regarding data corruptions and all errors included in the ERROR and CRIT debug levels.
ERROR	Reports serious errors regarding network connections and all errors included in the CRIT debug level.
CRIT	Reports critical errors that prevent Liberator from running.

Table 9-3: Debug levels

The default debugging level that is used at startup is configurable. However, when the UDP message interface is enabled (see “UDP commands” on page 144), the level can be changed dynamically while Liberator is running by using the **debug** UDP command .

- Use the following parameter in the configuration file *rttpd.conf* to set the logging level that Liberator will use at startup.

log-level Determines the errors and events that are reported to the log files when Liberator starts operating.
See page 170.

The following UDP command can be sent over a Liberator's UDP interface.

debug

Dynamically changes the level of error and event reporting. This overrides the level set using the configuration option **log-level** (see page 170).

Syntax: debug level

Parameter:

Name	Type	Default	Description
level	string	[no default]	New level of debug messages.

- For how to issue the debug UDP command, see the section “UDP commands” on page 144, and the following example.

Example:
Assuming the current directory is *$INSTALL_DIR*, the following command will change the Liberator’s event reporting level to WARN, by sending a UDP message to port 1247 on the default IP address of 127.0.0.1 (the local host).

```
./bin/udpsend -p 1247 debug WARN
```

If the command is successful the Liberator’s event log will contain entries like the following:

```
2011/06/26-15:11:55.054 +0000: INFO: Processing UDP Command 'debug'
with arguments 'WARN'
2011/06/26-15:11:55.069 +0000: NOTIFY: Attempting to change
 to WARN
2011/06/26-15:11:55.069 +0000: NOTIFY: Successfully changed debug-
level to WARN
```

9.10 Latency Measurement

Latency Measurements

Liberator can be setup to allow the latency of data updates through the system to be monitored.

Latency Chains

The latency added by each server side component in the system can be configured to add latency information as the update passes through, building up a chain of latency information. To achieve this the initial source of data must publish a millisecond timestamp to a field. Using that timestamp each DataSource component in the system will add its own delta from that timestamp when it enters and when it exits the process. Liberator will also add its own delta from the initial timestamp when a data update enters and when it is sent to a client.

Note: *This system relies on all the machines involved having their clocks synchronised. The client monitoring machine will also have to be synchronised if the last part of the journey is to be measured.*

Example Latency Chain

```
Object Name:                /VOD.L

Initial Timestamp:          LTY_INIT_TS = 1125062541880

List of Events:             LTY_LIST_EVENT = datasrc1_E,datasrc1_X,
                            transformer1_E,transformer1_X,rttpd1_E,rttpd1_X

List of Timestamp Deltas:   LTY_LIST_TS = 0,1,3,4,5,8
```

The comma separated list of deltas correspond to the event names in the list of events. Each value represents the milliseconds since the initial timestamp that the event occured. For each component there should be a Enter (E) and an Exit (X) event.

Note: *In some cases Liberator will not add an Exit event, such as when the message is a cached value and the Exit time would be very large.*

datasrc1_E	0	Time elapsed between initial timestamp and entering datasrc1
datasrc1_X	1	Time elapsed between initial timestamp and exiting datasrc1
transformer1_E	3	Time elapsed between initial timestamp and entering transformer1
transformer1_X	4	Time elapsed between initial timestamp and exiting transformer1

rttpd1_E	5	Time elapsed between initial timestamp and entering rttpd1
rttpd1_X	8	Time elapsed between initial timestamp and rttpd1

Config options:

name	type	default	description
latency-chain-enable	BOOLEAN	FALSE	Turns on latency chaining
latency-chain-name	STRING	'application-name'	The name used by this component for the latency events field.
latency-chain-init-ts-field	STRING	LTY_INIT_TS	Name of initial timestamp field
latency-chain-list-event-field	STRING	LTY_LIST_EVENT	Name of list event field
latency-chain-list-ts-field	STRING	LTY_LIST_TS	Name of list timestamp delta field

Note: *These fields must exist in the fields.conf file.*

name	type	default	description
latency-chain-base64-mode	ENUMERATED	'none'	Defines whether to base64 decode and/or encode latency fields.

Accepted values for *latency-chain-base64-mode*:

- never - Do not treat values as base64 encoded
- decode - Decode latency chain fields for all objects
- detect - Decode latency chain fields if they look encoded

- encode - Encode latency chain fields after adding local deltas if the fields were decoded

These values can be ORed together, for example, 'decode|encode' will decode the field values, add the component entries onto the end of the field values, then encode the final values.

Note: *'Encode' will only convert a value that has just been decoded into base64, it will not encode values that arrived in plain text.*

End to End Latency

The Liberator can also provide per update latency information to RTTP Clients. To achieve this RTTP Clients can be configured to calculate the offset between its own clock and the Liberators clock. This is done at regular intervals as clocks can drift overtime. With the offset available and a millisecond timestamp on each update, the RTTP Client SDKs can provide a millisecond latency figure for every update received.

The field used for the millisecond timestamp can either come from a DataSource peer or Liberator can be configured to add one itself. If a timestamp field is configured in Liberator, it will only add the timestamp to updates that do not contain that field.

Config:

name	type	default	description
timestamp-field	STRING	no default	The field name of the timestamp field.

Note: *Latency measurements will be affected by some Liberator configuration settings. The two main areas that can delay messages are object throttling (see "Using throttling" on page 92) and bursting (see "Configuring "bursts"" on page 94). Object throttling by default is set to 1 second, this means it is possible that an update gets delayed by up to 1 second by this feature. Bursting on client session output by default is set to 0.5 seconds. Again this means an update could get delayed a further 0.5 seconds on top of the throttling delay. Both these features have their benefits, throttling prevents sending out multiple updates to the same object in a short space of time, and bursting can improve overall performance in a system with a large number of clients by batching together small messages when output to a client. Throttling can be turned off if that feature is not desirable, but it is recommended to always have a burst setting, even if it is small, such as 0.1 seconds.*

10 Optimising efficiency

Adjusting the configuration parameters highlighted in this chapter can greatly improve the speed at which the Liberator performs in certain situations.

10.1 Improving performance using bursts

burst-min Recommended value: 0.1

burst-max Recommended value: 0.5
When a session starts getting more than one message in the time period it will batch those messages together and send them as a single write. The default values of 0.25 and 0.5 work well in most situations.
A ***burst-max*** setting of greater than 0.5 can give a visual effect on the client side that data is being "pulsed" instead of streamed.
See page 236

10.2 Improving performance using threads

Client session threads

threads-num This configuration option sets the number of client session threads. It defaults to 1. (See page 237.)
When Liberator needs to handle a high number of user connections, increasing ***threads-num*** can improve performance, but this must take into account the number of CPUs on the machine running the Liberator.

Note: *Liberator uses additional threads for DataSource peer connections. See “DataSource threads” on page 152.*

buf-elem-len Recommended value: 4096
Size of cached buffers. Increasing this will improve performance if using very large messages, but it will considerably affect memory usage.
See page 236.

DataSource threads

Liberator uses threads to process the communication with its DataSource peers. You can configure how Liberator allocates these threads to these peers by setting the global configuration item **peer-thread-pool-size** (see page 238) and the **thread-name** option of the **add-peer** configuration item (see page 215). This helps to improve performance when the Liberator is connected to several DataSource peers.

- Specifying the **thread-name** option for a peer creates a named thread. You can then allocate the same named thread to other peers. All the peers then share the same thread.

  For example:

```
add-peer
remote-name DataSource_1
remote-id 1
...
thread-name DS_Thread_1
end-peer

add-peer
remote-name DataSource_2
remote-id 2
...
thread-name DS_Thread_1
end-peer
```

  In this example, the connections to `DataSource_1` and `DataSource_2` are handled on a single thread called `DS_Thread_1`.

- Set **peer-thread-pool-size** to define a pool of threads that are shared between peer connections that do not have a defined **thread-name**. For example:

```
peer-thread-pool-size 2
...
add-peer
remote-name DataSource_3
remote-id 1
...
## No thread-name defined
end-peer

add-peer
remote-name DataSource_4
remote-id 2
...
## No thread-name defined
end-peer

add-peer
remote-name DataSource_5
remote-id 2
...
## No thread-name defined
end-peer
```

  In this example (continued from the previous one), a pool of two threads is allocated to three of the DataSource peer connections (`DataSource_3`, `DataSource_4`, and `DataSource_5`), so two of the connections must share a thread.

- If you need to ensure that each peer connection that is not explicitly allocated a named thread nevertheless has its own unique thread (not shared with any other connection), omit **peer-thread-pool-size** from the configuration**.**

- If neither of the **peer-thread-pool-size** or **thread-name** configuration items is defined, Liberator creates a dedicated thread for each configured peer.

Note: *This is the same behaviour as in previous versions of Liberator, which do not have these configuration items.*

- Consider allocating pool threads to DataSource peer connections with low update rates (typically 100 updates/sec or lower). In this case, there is no real performance advantage from dedicating a separate thread to each connection.

- If a particular peer connection is expected to be heavily loaded (typically when it must handle 10,000 or more updates/sec), it is recommended that you allocate a dedicated thread to that connection. For example:

```
peer-thread-pool-size 30

...

add-peer
remote-name DataSource_1
remote-id 1
...
## No thread-name defined
end-peer

...

add-peer
remote-name DataSource_90
remote-id 90
...
## No thread-name defined
end-peer

add-peer
remote-name DataSource_91
remote-id 91
...
## This peer connection is heavily loaded
## so we give it its own thread:
thread-name DS_91_Thread
end-peer
```

- If the Liberator has just one DataSource peer feeding it updates at a high rate, you may be able to improve performance by configuring more than one connection to the DataSource peer, and ensuring each connection uses a separate thread. The updates are then spread across multiple threads. For details of how to configure multiple connections to a single DataSource peer, see “Multiple connections to a DataSource peer” on page 109.

- You may need to experiment to determine the optimum configuration for the DataSource connection threads. You will need to balance the performance gains against memory usage, taking into account the number of client session threads (see “Client session threads” on page 151), and the number of CPUs on which those threads can run.

10.3 Improving performance using hashtables

Adjusting the size of the hashtables enables you to allocate memory resources and adjust performance. For example, increasing memory requirements might improve the speed of certain operations.

object-hash-size

Recommended value: Twice the maximum number of objects.
This is the size of the hashtable that holds objects. Increasing this will use extra memory, but it will benefit the speed of updates and requests if this is sufficiently high to avoid too many hash collisions.

Note: *There is an internal object for each client session, so this hash size ideally would be the maximum number of objects + maximum number of sessions. This should be approximately the number of objects the Liberator will hold. Internally there is one additional object for each logged on user, so the object hashtable should be the number of objects + number of concurrent users.*
See page 235

session-hash-size

Recommended value: Twice the max number of users
Size of session hashtable. This figure should be increased so that it is greater than the maximum concurrent users.

Note: *Increasing session-hash-size will result in more memory usage.*
See page 235

user-hash-size	Recommended value: Twice the maximum number of usernames. Size of user hashtable. This figure should be increased as an Auth module may allow more than one session per user. See page 235
record-type2-hash-size	Recommended value: Twice the maximum number of type 2 pieces of data that expected to be cached multiplied by the maximum number of objects. Size of Type 2 data hashtable. See page 199

10.4 Improving performance using TCP nodelay

direct-tcp-nodelay-off	Recommended value: FALSE Turns off the no delay feature for direct sockets. By default the Liberator turns on the TCP_NODELAY flag for direct and HTTP client sockets and gives better performance. See page 238 The no delay option will prevent TCP from buffering small amounts of data to be sent while it is waiting for an acknowledgement from a previous send.
http-tcp-nodelay-off	Recommended value: FALSE Turns off the no delay feature for HTTP sockets. See page 238
datasrc-tcp-nodelay-off	Recommended value: FALSE Turns off the no delay feature for DataSource peer sockets. See page 238

10.5 Improving performance using selected fields

By sending only the fields requested by the client, Liberator uses smaller data packets but more CPU time.

requested-fields-only — Recommended value: TRUE
Enables only fields requested by a client to be sent to that client.
See page 209

10.6 Reducing message sizes using fields.conf

Due to the way RTTP encodes field names, message sizes can be reduced slightly by configuring the most commonly used fields nearer the top of the fields.conf file.

10.7 Improving security measures

To avoid attacks on your system, Liberator includes a number of options to limit the acceptable length of RTTP messages (sent on a direct connection) and each part of an HTTP message. If Liberator receives a message longer than that configured, it will reject it instead of reading it continuously until it runs out of memory.

The following parameters configure the various maximum lengths of messages and their elements. The recommended values are the default settings for these options, but should be shortened if you experience security problems.

direct-max-line-length — Recommended value: 65536
Maximum number of bytes allowed in a single line of an RTTP message sent to Liberator through a direct connection.
See page 178

http-max-request-length — Recommended value: 1024
Maximum number of bytes allowed in a single HTTP request line (the line that contains a GET or a POST instruction).
See page 178

http-max-header-line-length	Recommended value: 65536 Maximum number of bytes allowed in a single HTTP header line. See page 178
http-max-header-lines	Recommended value: 30 Maximum number of header lines allowed in an HTTP message. See page 178
http-max-body-length	Recommended value: 65536 Maximum number of bytes allowed in the body of an HTTP message. See page 178

11 Running Liberator with many users

Liberator can normally support up to 10,000 concurrent user sessions, and up to 100,000 concurrent users on suitably specified hardware if the message rates are low.

Each connected session requires an open socket connection, which means the system needs to be able to have an open file descriptor for this socket. The operating system will typically need configuration to allow these high numbers of file descriptors.

11.1 Configuring Liberator for a high number of users

- If your license has a max-user limit then the Liberator will attempt to set a suitable file descriptor limit when it starts. If you receive the error message "Failed to set system-max-files to nnnn" when starting the Liberator, then adjust the operating system configuration as described in the Changing file descriptor limits sections below.

- If your license is for an unlimited number of users, set system-max-files (see page 165) to a suitable amount to allow the expected numbers of concurrent users to login.

Note: *Liberator uses a certain number of file descriptors internally, for log files, internal communications and handling HTTP requests. This means that if your Liberator will have 2000 users, a* ***system-max-files*** *value of 2048 will not be large enough. The safety margin that Liberator chooses when it sets* ***system-max-files*** *automatically is an extra 512.*

11.2 Changing file descriptor limits—Linux

This section describes how you can edit various Linux configuration files to adjust the file descriptor limits. Please note that the location of these files may differ according to the Linux distribution you are using. You may also want to change the settings shown in this section depending on the number of users you want to support.

- Use the following parameter in the configuration file *rttpd.conf* to set file descriptor limits.

system-max-files	Maximum file descriptors for this process. This is overridden if the license states a higher number of users. See page 165

Note: *On some systems you may also need to configure the operating system to allow a higher number of open file descriptors in order to set system-max-files.*

Note: *If your license is for an unlimited amount of users, you will need to set* ***system-max-files*** *to a number higher than your expected maximum concurrent users. See also "max-user-warn" on page 200*

Per process

The following changes allow you to change the file descriptor limit per process, from the default soft limit anywhere up to the hard limit. This will allow you to increase **system-max-files** to a suitable amount.

- In */etc/security/limits.conf* add the lines:

```
*  soft   nofile   1024
*  hard   nofile   32768
```

- In */etc/pam.d/login* add:

```
session required /lib/security/pam_limits.so
```

System-wide

The following changes configure the system-wide file descriptor limits.

- Add the following to */etc/sysctl.conf*:

```
fs.file-max = 32768
fs.inode-max = 131072
```

Configuring the range of ports

The following changes configure the range of ports to be used by the system.

- Add the following to */etc/sysctl.conf*:

```
net.ipv4.ip_local_port_range= 32768 61000
```

11.3 Liberator demonstrations

To check your Liberator is running properly some simple StreamLink JS examples are provided. . These use data from a demonstration Integration Adapter.

To view the examples:

- Start the demo feed
  (see “Starting and stopping the demo feed on Linux” on page 161).
- The Liberator’s status page should now show the Integration Adapter (called "demosrc") as being UP.
- Navigate to the Liberator’s Examples page
  (select the Examples menu item on the horizontal menu bar).

11.4 Starting and stopping the demo feed on Linux

The demo feed should be started using the demosrc script.

- Start the feed by entering:

```
$ cd /opt/Liberator
$ ./etc/demosrc start
```

- Stop the feed by entering:

```
$ cd /opt/Liberator
$ ./etc/demosrc stop
```

These commands can be issued from anywhere; the current working directory does not matter.

11.5 Using an SSL connection for the demo feed

The default *rttpd.conf* configuration file has a sample SSL section which will work with the demonstration data feed.

For instructions on how to adjust the configuration to enable this SSL connection, see “Using SSL with the demonstration feed” on page 162

11.6 Viewing the examples on the website

To view the examples web page:

- Point your browser at *http://<hostname>:8080* (where <hostname> is the host name or IP address of the machine you have installed the Liberator on);
- Click on Examples.

You will be prompted for a username and password. The default values are *admin* and *admin*.

Note: *These defaults correspond to the default username and password options in the add-authdir entry of your configuration file (see add-authdir on page 176). Any changes made to this entry will be reflected in the accessibility of the web site example pages.*

11.7 Using SSL with the demonstration feed

The default *rttpd.conf* configuration file has a sample SSL section which will work with the demonstration data feed described in this chapter.

Configuring the demonstration SSL connection

To enable the demonstration SSL connection, you must edit the configuration files for both the Liberator and the demonstration feed:

- Edit *rttpd.conf* and "uncomment" the datasrc-sslport and start-ssl options (i.e. remove the "#" characters) at the bottom of the file, as shown below.

```
## SSL ########################################

datasrc-sslport         25001

start-ssl
        enable-server
        server-authmode 1
        server-cert     certs/rttpd.pem
        server-key      certs/rttpd.key
        CAfile          rttpdCA/cacert.pem
        CApath          rttpdCA/newcerts
end-ssl
```

Edit *demosrc.conf* and comment out the first add-peer section (i.e. add a "#" character to the start of each line) and uncomment the second add-peer section and the start-ssl section, as shown below.

```
## DATASRC ########################################

#add-peer
#        port     25000
#end-peer

add-peer
        port      25001
        ssl
end-peer

...

## SSL ############################################

start-ssl
        enable-client
        client-authmode 1
        client-cert     certs/demosrc.pem
        client-key      certs/demosrc.key
        CAfile          demosrcCA/cacert.pem
        CApath          demosrcCA/newcerts
end-ssl
```

12 Appendix A: Configuration reference

Liberator is configured by editing the entries in the plain text file *rttpd.conf*. This can be found within the Liberator installation directory (see “Installing Liberator” on page 20).

Some of the more advanced configuration items are described in “Optimising efficiency” on page 151.

rttpd.conf is split into different sections, each concentrating on a different area of functionality. Each section and the parameters within them are described below.

For an explanation of the syntax of the configuration language, and how to make use of variables, conditionals, and macros when defining configuration, see the document **DataSource For C Configuration Syntax Reference**.

12.1 Main

This section of *rttpd.conf* configures the main system settings.

application-root

Specifies the root directory of the application installation.

Type: string

Default value: [current working directory]

application-name

Distinguishes this application from other applications.

Type: string

Default value: [set by application]

event-log

Filename of the event log.

Type: string

Default value: event-rttpd.log (event-%a.log)

system-max-files

Maximum file descriptors for this process.

Type: integer

Default value: 1024

runtime-user

This specifies a user to run the server as (UNIX only).

Type: string

Default value: [no default]

catch-crash

Turns on catching of application crashes (Linux platforms only).

Type: boolean

Default value: FALSE

include-file

Imports configuration parameters from another file. These parameters will be overwritten if the same parameter occurs later in the main configuration with a different value.

%a is replaced by application-name (see page 165) and %h is replaced by the host name of the machine. This enables application or host-specific configuration to be used.

Type: string

Default: [no default]

Example:

```
include-file myfile-%a-%h.conf
```

The * wildcard character matches any string. In the following example, configuration parameters are imported from all files that have the *.conf* file extension.

Example:

```
include-file *.conf
```

pid-filename

Allows the location of the pid file to be defined uses the usual %a,%n,%r expansion options. Additionally %u is available which is the users home directory.

Type: string

Default value: %r/var/%a.pid

license-file

This is the filename of the license file for the application. The standard Liberator kit uses *license-rttpd.conf* which is specified in the default *rttpd.conf*.

Type: string

Default value: license.conf

Note: *For information about other configuration items relating to licensing, refer to the document* ***Caplin Platform: Guide to User Licensing****.*

syslog-facility

This is the syslog facility to use when logging to the unix based syslog - See “System log files (syslog)” on page 130.

Type: string

Default value: local6

ssl-config-name

This is the name of the section to load from the OpenSSL configuration file when OpenSSL is initialized. Liberator uses OpenSSL to support:

- HTTPS connections.
- Connections to DataSource peers through Secure Sockets Layer.
- Validation of signatures in KeyMaster user credentials tokens.

Type: string

Default value: openssl_conf
This is the system default section defined by OpenSSL.

By default, Liberator uses the OpenSSL configuration file *$LIBERATOR_ROOT/openssl.cnf*. This can be overridden by defining the filename and path through the environment variable **OPENSSL_CONF**.

12.2 Logging

This section of *rttpd.conf* configures the logging of events. You can set global settings to specify the cycling of all log files (see the following configuration items), or configure the cycling of each log file individually (see “Advanced log file settings” on page 171).

log-dir

Default directory in which to store log files.

Type: string

Default value: application-root/var (%r/var)

log-maxsize

Maximum log file size in bytes.

Type: integer

Default value: 0

log-max-history

Maximum number of log lines to retain for monitoring

Type: Integer

Default value: 10

log-cycle-time

Time at which logs will cycle, in minutes from midnight.

Type: integer

Default value: 240 (i.e. 0400 hours).

Note: *If the time is greater than 1440 it is taken from the start of the week (Midnight Sunday night). This allows weekly log cycling on a specific day if the period is set accordingly as well.*

log-cycle-period

Interval between cycling logs, in minutes.

Type: integer

Default value: 1440 (i.e. daily)

log-cycle-suffix

Suffix for cycled logs. See the UNIX manual page for strftime to see the possible format strings that can be used here.

Type: string

Default value: %u

log-cycle-offset

Specifies how many minutes to take off the current time when creating the suffix.

Type: integer

Default value: [The same as log-cycle-period. For example, if cycling at 0400 hours, the time passed into strftime to create the suffix will be 0400 hours the previous day.]

log-level

Determines the errors and events that are reported to the log files when Liberator is operating. Acceptable values are shown in Table A.1 below.

Note: *A list of all error messages and their associated logging level can be found as "Appendix B: Log file messages and formats" on page 262*

Type: string

Default value: info

Value	Description
DEBUG	Reports all errors and events.
INFO	Reports events and information regarding normal operation and all errors included in the WARN, NOTIFY, ERROR and CRIT debug levels.
WARN	Reports minor errors and all errors included in the NOTIFY, ERROR and CRIT debug levels.
NOTIFY	Report errors regarding data corruptions and all errors included in the ERROR and CRIT debug levels.
ERROR	Reports serious errors regarding network connections and all errors included in the CRIT debug level.
CRIT	Reports critical errors that prevent Liberator running.

Table 12-1: Debug levels

12.3 Advanced log file settings

As well as the global configure options for log file cycling in the Logging section, individual log files can be cycled.

add-log

Overrides the global default for a particular log file.

Syntax:

```
add-log
      name              [value]
      maxsize           [value]
      time              [value]
      period            [value]
      suffix            [value]
      offset            [value]
      level             [value]
      monitor-level     [value]
end-log
```

The options in this entry are:

Name	Type	Default	Description
name	string	[no default]	Name of the log to cycle. If no value is entered the global settings are used. Acceptable values are: event_log — the event log file (see page 165). http_access_log — the HTTP access log file (see page 176). javaauth_log — the Javaauth log file. news_log — the log file to store news headlines (see page 241). object_log — the object log file (see page 203). request_log — the request log file (see page 203). packet_log — the packet log file (see page 210). session_log — the session log file (see page 203). xmlauth_log — the XMLauth log file.
maxsize	integer	0	Maximum log file size in bytes. The log files will be cycled if they exceed the size specified here, therefore a value of 0 means log files will cycle every time they are checked.
time	integer	240 (i.e. 0400 hours)	Time at which logs will cycle, in minutes from midnight.
period	integer	1440 (i.e. daily)	Interval between cycling logs, in minutes.
suffix	string	%u	Suffix for cycled logs. This is passed through strftime (refer to your Unix manual for further information on strftime). The default value of %u results in a file being created for each day of the week.

Name	Type	Default	Description
offset	integer	log-cycle-period	Specifies how many minutes to take off the current time when creating the suffix. For example, if cycling at 0400 hours, the time passed into strftime to create the suffix will be 0400 hours the previous day.]
level	string	INFO (this defaults to the global option log-level).	Debug level for the log. ***Note:*** *This is only valid for the event log.*
monitor-level	string	NOTIFY (this defaults to the global option monitor-level).	Debug level to send messages to monitoring. ***Note:*** *This is only valid for the event log.*

12.4 HTTP

This section of *rttpd.conf* configures the HTTP connection and type of contents.

http-wwwroot The root directory of the html files.

Type: string

Default value: application-root/htdocs (%r/htdocs)

http-interface Network interfaces to listen on for HTTP connections.

Default value: [all available interfaces]

Syntax: **http-interface IPaddresses**

The option in this entry is:

Name	Type	Default	Description
IPaddresses	array	[all available interfaces]	Space-separated list of interface IP addresses to listen on for HTTP connections.

http-port Network port to listen for HTTP connections.

Type: integer

Default value: 8080

http-keepalive-max Number of requests per connection (HTTP Keep Alive feature).

Type: integer

Default value: 10000

http-keepalive-timeout Timeout period in seconds of HTTP Keep Alive connections.

Type: integer

Default value: 120

http-refuse-time Time in seconds to refuse new connections if no sockets are available.

Type: float

Default value: 5.0

http-server-name This is used in the HTTP response headers. This option is to change the default, which is sometimes advised for security reasons so the type of server is not advertised.

Type: string

Default value:

http-indexfile List of files to attempt to use when a directory is requested.

Type: string

Default value: index.html, index.js

http-rttp-content-type Default RTTP stream content type.

Type: string

Default value: application/octet-stream

http-def-content-type Default HTTP content type.

Type: string

Default value: text/plain

http-err-content-type Error message content type.

Type: string

Default value: text/html

http-idx-content-type Index page content type.

Type: string

Default value: text/html

http-access-log Name of the HTTP access log file.

Type: string

Default value: http-access-rttpd.log (http-access-%a.log)

http-error-log Name of the HTTP error log file. This file logs all HTTP requests that result in an Object not found error.

Type: string

Default value: http-error-rttpd.log (http-error-%a.log)

add-authdir Defines an HTTP-authenticated directory.

Syntax:

```
add-authdir
      name              [value]
      realm             [value]
      username          [values]
      password          [values]
      check-module
end-authdir
```

The options in this entry are:

Name	Type	Default	Description
name	string	/status	The full HTTP directory name.
realm	string	Liberator Admin	The HTTP basic authentication realm.
username	array of strings	admin	Username or names. See below.
password	array of strings	admin	Password or passwords (to match the users in username list. See below
check-module	boolean	FALSE	Determines whether this directory will ask the Auth Module to authenticate the user instead of using the list of usernames and passwords given above.

Multiple usernames and passwords can be entered in the following ways: either as space-separated lists, as individual entries, or a combination of the two.

Examples:

```
add-authdir
      username    Alf Bill Carl Dave
      password    pwA pwB pwC pwD
end-authdir
```

or

```
add-authdir
      username    Alf
      password    pwA
      username    Bill
      password    pwB
      username    Carl Dave
      password    pwC pwD
end-authdir
```

direct-max-line-length

Maximum number of bytes allowed in a single line of an RTTP message sent to Liberator through a direct connection.

Default value: 65536

http-max-request-length

Maximum number of bytes allowed in a single HTTP request line (the line that contains a GET or a POST).

Default value: 1024

http-max-header-line-length

Maximum number of bytes allowed in a single HTTP header line.

Default value: 65536

http-max-header-lines

Maximum number of header lines allowed in an HTTP message.

Default value: 30

http-max-body-length

Maximum number of bytes allowed in the body of an HTTP message.

Default value: 65536

http-connection-cookie-enable

If set, the server will set a cookie in the client when the client connects over HTTP.

Type: boolean

Default value: FALSE

http-connection-cookie-expires

Number of days before the cookie expires.

Type: integer

Default value: 1

12.5 RTTP

This section of *rttpd.conf* configures the RTTP connection.

rttp-type5-js

RTTP Type 5 Javascript filename

Type: String

Default value: /sl4b/javascript-rttp-provider/streaming-type5.js

rttp-type5-pad-length

RTTP Type 5 header padding in bytes

Type: Integer

Default value: 4096

rttp-type3-timeout

RTTP Type 3 timeout in seconds

Type: Float

Default value: 10.0

rttp-hostname

RTTP Hostname Override

Type: String

Default value: [NO DEFAULT]

12.6 HTTPS

This section of *rttpd.conf* configures HTTPS connections.

https-enable

This option switches on support for HTTPS connections.

Type: boolean

Default value: FALSE

https-interface

This option configures the network interface to listen on for HTTPS connections.

Syntax: https-interface IPaddresses

The option in this entry is:

Name	Type	Default	Description
IPaddresses	array	[all available interfaces]	Space-separated list of interface IP addresses to listen on for HTTPS connections.

https-ssl-options

This option defines the levels of the SSL protocol that are supported for HTTPS connections.

Type: istring

Permitted values:	SSL_OP_ALL	All SSL protocol levels are supported.
	SSL_OP_NO_SSLv3	The SSLv3 protocol is not supported.
	SSL_OP_NO_SSLv2	The SSLv2 protocol is not supported.
	SSL_OP_NO_TLSv1	The SSLv1 protocol is not supported.
Default value:	SSL_OP_NO_SSLv2	

https-port

This option configures what network port to listen on for HTTPS connections.

Type: integer

Default value: 4443

https-certificate

This option configures the filename of the SSL certificate. This file should be in PEM format.

Type: string

Default value: cert.pem

https-privatekey

This option configures the filename of the SSL private key. This file should be in PEM format.

Type: string

Default value: cert.pem

https-passwordfile

This option identifies the file containing the SSL certificate passphrase.

Type: string

Default value: .rttpd.https.pass

https-cipher-list

Accesses the Openssl function SSL_CTX_set_cipher_list which allows different SSL ciphers to be configured. Please refer to the OpenSSL documentation for more information on this feature (http://www.openssl.org).

Type: string

Default value: DEFAULT

ssl-random-seed

Configures the seeding of the OpenSSL random number generator, which the Liberator uses for session IDs and HTTPS and DataSource SSL connections.

Syntax: ***ssl-random-seed type arg1 arg2***

The options in this entry are:

Name	Default	Description
type	[no default]	Type of random number generation. Must be one of the following: builtin This takes no arguments and uses various system commands to produce random output. file Uses the data in the file to seed the random number generator. exec Uses the output of the command to seed the random number generator.
arg1	[no default]	If type is file, this is a filename (relative to the Liberator Root Directory). If type is exec, this is a command line (relative to the Liberator Root Directory)
arg2	[no default]	If type is file, this specifies how many bytes of the file to use. If type if exec, this specifies how many bytes of the output to use.

Any number of **ssl-random-seed** entries can be given.

Examples:

```
ssl-random-seed  builtin
ssl-random-seed  file  etc/randomdata
ssl-random-seed  file  etc/randomdata  1024
ssl-random-seed  exec  etc/random.sh
ssl-random-seed  exec  etc/random.sh   512
```

Note: *On Linux OpenSSL is seeded by a hardware device so using ssl-random-seed may be unnecessary.*

ssl-engine-id

The SSL hardware or software engine to support. The default value of 'openssl' or 'software' will stop the server attempting to use an SSL card. If a value of 'all' is provided then the server will attempt to find and use any SSL cards available on the machine. Any other value will be considered to be a specific SSL card - please refer to the OpenSSL documentation for a full list of what is supported.

Type: string

Default value: openssl

ssl-engine-flags

Flags to be passed to the engine implementation.

Type: string

Default value: All (see page 81 for a list of flags)

add-virtual-host

Identifies a virtual host that Liberator will serve.

Syntax:

```
add-virtual-host
      name                    [value]
      addr                    [value]
      wwwroot                 [value]
      https-certificate       [value]
      https-privatekey        [value]
      https-passwordfile      [value]
end-virtual-host
```

The options in this entry are:

Name	Type	Default	Description
name	string	addr	Name for this virtual host.
addr	string	[no default]	Local ip address or hostname.
wwwroot	string	http-wwwroot	Root directory of the HTML files. Overrides the global setting ***http-wwwroot*** (see page 174)

Name	Type	Default	Description
https-certificate	string	https-certificate	Filename of the SSL certificate. Overrides the global setting ***https-certificate*** (see page 182).
https-privatekey	string	https-privatekey	Filename of the SSL private key. Overrides the global setting ***https-privatekey*** (see page 182)
https-passwordfile	string	https-passwordfile	File containing the SSL certificate passphrase. Overrides the global setting ***https-passwordfile*** (see page 182)

12.7 Direct connections

This section of *rttpd.conf* configures direct RTTP connections.

Note: *Direct RTTP connections are also known as "RTTP type 1" connections.*

direct-interface Network interfaces to listen for direct (type 1) RTTP connections.

Default value: [all available interfaces]

Syntax: direct-interface IPaddresses

The option in this entry is:

Name	Type	Default	Description
IPaddresses	array	[all available interfaces]	Space-separated list of interface IP addresses to listen on for RTTP connections.

direct-port Network port to listen for direct (type 1) RTTP connections.

Type: integer

Default value: 15000

direct-refuse-time Time in seconds to refuse new connections if no sockets are available

Type: float

Default value: 5.0

12.8 Direct connections using SSL

This section of *rttpd.conf* configures direct RTTP connections to Liberator that use the Secure Sockets Layer (SSL) to provide greater security.

Note: *Direct RTTP connections are also known as "RTTP type 1" connections.*

directssl-enable

This option switches on support for direct connections using SSL.

Type: boolean

Default value: FALSE

directssl-interface

This option configures the network interface to listen on for direct connections using SSL.

Syntax: directssl-interface IPaddresses

The option in this entry is:

Name	Type	Default	Description
IPaddresses	array	[all available interfaces]	Space-separated list of interface IP addresses to listen on for direct connections using SSL.

directssl-port

This option configures what network port to listen on for direct connections using SSL.

Type: integer

Default value: 15001

directssl-ssl-options

This option defines the levels of the SSL protocol that are supported for direct connections using SSL.

Type: string

Permitted values:		
	SSL_OP_ALL	All SSL protocol levels are supported.
	SSL_OP_NO_SSLv3	The SSLv3 protocol is not supported (all other levels are).
	SSL_OP_NO_SSLv2	The SSLv2 protocol is not supported. (all other levels are)
	SSL_OP_NO_SSLv1	The SSLv1 protocol is not supported. (all other levels are)

Default value: SSL_OP_NO_SSLv2

directssl-certificate

This option configures the filename of the SSL certificate used for direct connections using SSL. This file should be in PEM format.

Type: string

Default value: cert.pem

directssl-privatekey

This option configures the filename of the SSL private key used for direct connections using SSL. This file should be in PEM format.

Type: string

Default value: cert.pem

directssl-passwordfile

This option identifies the file containing the SSL certificate passphrase used for direct connections using SSL.

Type: string

Default value: .rttpd.directssl.pass

directssl-cipher-list

Accesses the Openssl function SSL_CTX_set_cipher_list which allows different SSL ciphers to be configured for direct connections using SSL. Please refer to the OpenSSL documentation for more information on this feature (http://www.openssl.org).

Type: string

Default value: DEFAULT

Other options

The following configuration options also apply to direct connections using SSL:

- “ssl-random-seed” on page 182.
- “ssl-engine-id” on page 184.
- “ssl-engine-flags” on page 184.

12.9 Objects

This section of *rttpd.conf* configures the way the Liberator deals with RTTP objects and the mapping of RTTP object types. See “About RTTP objects” on page 71. for more information.

object-throttle-times

An array of throttle times in seconds. For more information see “Using throttling” on page 92.

Type: float array

Default value: 1.0

object-throttle-default-level

The throttle level that all users start at on login..

Type: integer

Default value: 0

object-throttle-off

Turns the throttling capability off.

Type: boolean

Default value: FALSE

active-discard-timeout

Time in seconds that the Liberator will hold on to an active object after the last user stops viewing it. If the object is a directory, the timeout will not begin until the last user has stopped viewing the last object in the directory.

The value of ***active-discard-timeout*** is overidden for specific objects by any data service level discard timeouts (see ***discard-timeout*** option of ***add-data-service*** on page 226), and by the ***discard-timeout*** option of ***add-object*** (see page 191).

See also “Discarding objects” on page 120.

Type: integer

Default value: 60

record-type3-history-size

Maximum number of updates to keep for each record containing type 3 data.
This is a global setting, but it can be overridden for a specific object or object hierarchy by a **record-type3-history-size** option in an **add-object** configuration entry (see page 195).

Type: integer

Default value: 10

record-max-cache

Maximum number of type 3 record data to keep.

Note: *This configuration item is deprecated. Use* ***record-type3-history-size*** *instead.*

Type: integer

Default value: 10

add-object

Adds an object to Liberator and defines the object's characteristics.

add-object can be used to configure a directory with specific characteristics. For example, to create a directory called `/TRADE`, define an ***add-object*** entry with the ***name*** option set to `/TRADE` and the ***type*** option set to `20`, and with the other options set as required. Then define a data service with an ***include-pattern*** of `^/TRADE` (see "add-data-service" on page 224). When users subscribe to `/TRADE` they will receive objects, under the directory `/TRADE`, that the DataSource peer has sent to the Liberator (such as `/TRADE/ABC`, `/TRADE/DEF`, and so on).

Because the directory was created using ***add-object***, the objects created in the directory *inherit* the directory's characteristics (***discard-timeout***, ***throttle-times***, and so on) .

Syntax:

```
add-object
      name              [value]
      type              [value]
      flags             [value]
      init              [value]
      discard-timeout   [value]
      throttle-times    [value]
      purge-time        [value]
      purge-period      [value]
      purge-age         [value]
      record-type3-history-size [value]
      no-batching
      only-changed-fields
end-object
```

The options in this entry are:

Name	Type	Default	Description
name	string	[no default]	The name of the object. Must be set.
type	string or integer	[no default]	The RTTP object type, as either the object type name or the object type number. Must be set. Values are shown in Table A.2 below.
flags	integer	0	Flags used when creating the object.
init	string	NULL	Object type-specific initialisation string.

Name	Type	Default	Description
discard-timeout	integer	See description.	Only applies to directory objects. The time in seconds for which the Liberator will hold on to an object in the directory after the last user stops viewing it. After this time the object is deleted from the Liberator's cache and Liberator sends a discard instruction to the DataSource peer to cancel the subscription to the object. When this option is specified, its value overrides any settings of the ***discard-timeout*** for the data service that fetches data for the object (see page 226). It also overrides the global option ***active-discard-timeout*** (see page 190 and "Discarding objects" on page 120). The value defaults to the ***discard-timeout*** for the data service that fetches data for the object (see page 226).

Name	Type	Default	Description
throttle-times	float array	[no default]	An array of throttle times in seconds (same as object-throttle-times; see page 190). Acceptable values are positive numbers, 0, "high", "priority" and "stopped" or "paused". ***Note:*** *The array must be in ascending order of throttle times, and if you use "stopped" or "paused" it must be the last entry in the array.* ***Note:*** *If the only throttle time is "high" or "priority" it means the object is a high priority object and updates for it will jump the user's output queue on the server. This should only be used for objects sending important and infrequent messages.*
purge-time	integer	-1 (no purge)	Only applies to directory objects. Number of minutes from midnight to start purging the objects in the directory (deleting them from the Liberator's cache). See page 84 for a description and examples of how this option can be used to configure object purging.
purge-period	integer	1440	Only applies to directory objects. Number of minutes between purges. See page 84 for a description and examples of how this option can be used to configure object purging.

Name	Type	Default	Description
purge-age	integer	0	Only applies to directory objects. A multiplier on purge-period. Defines how old an object should be before it is purged. See page 84 for a description and examples of how this option can be used to configure object purging.
record-type3-history-size	integer	See description.	Only applies to record objects containing type 3 data. Maximum number of type 3 data updates to keep for this object / object hierarchy. This option overrides (for this object or object hierarchy only) the setting of the global configuration item **record-type3-history-size** (see page 191). The default value is the setting of the global configuration item **record-type3-history-size**.

Name	Type	Default	Description
no-batching	boolean	FALSE	Updates to objects that match the ***name*** option are *not* batched. When an update to a matching object is received, the update message is added to the queue of batched messages, and all messages in the queue are immediately sent to the client. The ***no-batching*** option overrides the ***burst-min*** and ***burst-max*** parameter settings (see page 94), and is intended for trade messages where latency is an issue. For example, if the ***name*** option of ***add-object*** is set to /TRADE, updates to objects that start with /TRADE/ are immediately sent to the client.
only-changed-fields	boolean	FALSE	Configures an object to only forward the fields changed from the last update

Table A.2: Object types

Object Type Name	Object Type Number	Description
directory	20	Directory
page	21	Page
record	22	Record
news	23	News headline
story	24	News story
chat	27	Chat object
container	28	Container object
permission	30	Permission object

default-type

Sets the default sub-type of objects.

Type: integer

Default value: 211

add-type-mapping

Adds a sub-type mapping.

Type: string, integer

Default value: [no default]

object-map

Adds an object mapping.

This maps an object name to a different object name..

Syntax: ***object-map [name of object to be changed] [new name for object]***

Type: string

Default value: [no default]

The strings defining ***[name of object to be changed]*** and ***[new name for object]*** can contain parameters of the form ***%n***, ***%u***, and ***%U***.

%n is the number of a string to be matched in the pattern. Each object-map entry can specify up to 9 strings to match (***%1*** to ***%9***).

%u represents the Liberator login name of the requesting user, to be included in the new object name; for example, "alice".

%U represents the Liberator session username of the requesting user, to be included in the new object name. If a user is allowed to have multiple concurrent logins on the Liberator, each login is identified by a unique session username. For example, "alice-0", "alice-1", ...

lExample 1:

```
object-map  "/ABC/%1/%2       "/DEF/%2/%1"
```

This mapping changes the object **/ABC/EUR/FX** to an object with the name **/DEF/FX/EUR**

Example 2:

```
object-map  "/MYCHANNELS/%1   "/CHANNELS/%u/%1"
```

If a user called “alice” requests an object called **/MYCHANNELS/ABC**, the above object mapping would change this name to **/CHANNELS/alice/ABC**

Example 3:

```
object-map  "/MYCHANNELS/%1   "/CHANNELS/%U/%1"
```

Asume a user called “alice” is logged on to the Liberator twice, and the Liberator has generated a unique login name of “alice-1” for the first session, and “alice-2” for the second session. In the second session alice requests an object called **/MYCHANNELS/ABC**. The above object mapping would change this name to **/CHANNELS/alice-2/ABC**.

object-precache-enable

Enables the caching of objects before they are requested.

Type: boolean

Default value: FALSE

record-clear-type1-on-failover

Clears all type 1 data for active objects when failing over to a new DataSource peer or reconnecting to the same one. This can allow cached data to be refreshed from the new DataSource peer.

Type: boolean

Default value: FALSE

record-clear-type2-on-failover

Clears all type 2 data for active objects when failing over to a new DataSource peer or reconnecting to the same one. This can allow cached data to be refreshed from the new DataSource peer.

Type: boolean

Default value: FALSE

record-clear-type3-on-failover

Clears all type 3 data for active objects when failing over to a new DataSource peer or reconnecting to the same one. This can allow cached data to be refreshed from the new DataSource peer.

Type: boolean

Default value: FALSE

record-type2-hash-size

Size of the Type 2 data hashtable.

Type: integer

Default value: 65536

12.10 Auth modules

This section of *rttpd.conf* configures the authentication and entitlement processing. For more information on Liberator permissioning, please refer to “Authentication and entitlement” on page 97.

auth-moddir

Directory from where authentication modules are loaded.

Type: string

Default value: application-root/lib (%r/lib)

auth-module

Name of authentication module.

Type: string

Default value: xmlauth

max-user-warn

Specifies the number of users at which a warning about the number of users approaching the maximum (set by max-user-limit) will be logged to the event log (see “max-user-limit” on page 201

Type: integer

Default value: 0 [no warning]

max-user-ok

Specifies the number of users at which a message confirming that the user level is acceptable will be logged to the event log.

Type: integer

Default value: 0

max-user-limit

Number of users allowed on the Liberator.

Type: integer

Default value: 0

Note: *The Liberator license may impose stricter limits on the number of user logins allowed on the Liberator than those defined by* ***max-user-limit****..*
For more information, see the document ***Caplin Platform: Guide to User Licensing****.*
For information about the particular user login limits set by your license, please contact Caplin Support.

auth-eject-users

Liberator imposes limits on the number of users who can be logged in at any one time. This may be through the setting of **max-user-limit**, or the limits are more likely determined by the Liberator license (for more about this, see the document **Caplin Platform: Guide to User Licensing**). **auth-eject-users** determines Liberator's behavior when a user login limit is exceeded.

When **auth-eject-users** is FALSE, Liberator rejects any further attempts to log in.

When **auth-eject-users** is TRUE, Liberator allows a new user to log in by first ejecting the oldest existing user session that matches the username and application-id of the new user. If there is no match on username and application-id, Liberator attempts a match on username alone. If the Liberator is in a cluster it will look across all the Liberators in the cluster for a matching session to eject.

Type: boolean

Default value: FALSE

For information about application-ids, see the section "More about asset class licensing" in the ***Caplin Platform: Guide to User Licensing****.*

auth-login-timeout

Timeout period in seconds when logging in and auth_new_user returns AUTH_DELAYED (see auth_new_user in the companion document **Liberator Auth Modules Developer's Guide**).

Type: integer

Default value: 30

auth-map-timeout

Timeout period in seconds when requesting a mapped object and auth_map_object returns AUTH_DELAYED (see auth_map_object in the companion document **Liberator Auth Modules Developer's Guide**).

Type: integer

Default value: 30

write-users-period

Time period in seconds for writing users file

Type: integer

Default value: 3600

write-users-time

Time in minutes from midnight before writing users file. -1 means never

Type: integer

Default value: -1

12.11 Sessions

This section of *rttpd.conf* configures the user session.

session-log

Name of the session log file.

Type: string

Default value: session-rttpd.log (session-%a.log)

request-log

Name of the request log file.

Type: string

Default value: request-rttpd.log (request-%a.log)

object-log

Name of the object log file that keeps a record of all request and discard commands for objects, and whether those commands were successful.

Type: string

Default value: object-rttpd.log (object-%a.log)

rttp-log

Name of the RTTP traffic log file that records the RTTP traffic between a client and the Liberator.

Type: string

Default value: rttp/%c_%l.%i
where:
%c is the client application id (for example, “SL4B”),
%l is the user name (Liberator login name) associated with the RTTP session,
%i is the RTTP session id.
For example: *rttp/SL4B_JSmith.0x-ab-9*

It is strongly recommended that the name of the RTTP traffic log file contains at least the %l (user name) and %i (RTTP session id) markers, so that a separate log file is generated for each session for each user named in ***rttp-log-users***. If these markers are absent, the log entries for all RTTP sessions will be mixed together in the same file, making it difficult to determine which messages came from which sessions and users.

If you configure Liberator to write its log files to a directory other than the default *var* directory (see "log-dir" on page 168), make sure that you create within the new log directory a subdirectory (as specified by **rttp_log** or its default setting) to receive the server-side RTTP log files.

rttp-log-users

A list of Liberator users (Liberator login names) for whom RTTP traffic logs are to be generated. If this configuration entry is absent or empty, only RTTP traffic logs that have been specified using the Caplin Management Console will be generated (see "Logging RTTP traffic" on page 131). The names of the traffic log files are defined by ***rttp-log*** (see page 203).

Type: array of strings

Default value: Empty string (No RTTP traffic logged)

Note: *The Liberator configuration option **rttp-log-users** should only be used for debugging test installations. It permanently enables traffic logging for the specified users and the users' traffic will be logged even after Liberator is restarted. Logging can only be turned off by stopping the Liberator and changing the rttp-log-users configuration option. In a live system you should normally turn RTTP logging on and off using the Caplin Management Console. See "Logging RTTP traffic" on page 131.*

The user names can be entered in the following ways: either as space separated lists, as individual entries, or a combination of the two.

Examples:

```
rttp-log-users    Alf Bill Carl
```

or

```
rttp-log-users    Alf
rttp-log-users    Bill
rttp-log-users    Carl
```

noauth-reconnect

Determines whether the Liberator uses the Auth Module to check a user's authentication when the user attempts to reconnect.

Type: boolean

Default value: FALSE

session-id-len

Defines the length in characters of the unique identifier for a session.

Type: integer (max 255)

Default value: 12

For added security use this option to increase the size of the session id. However, please note the following regarding compatibility with client libraries:

Note: *StreamLink for Browsers (SL4B) compatibility.*
Versions of SL4B prior to 4.3.0 are only compatible with Liberator 4.3.0 upwards if ***session-id-len*** *is set to* ***6****.*

StreamLink for Java (SL4J) compatibility.
Previous versions of SL4J are compatible with Liberator 4.3.0 upwards, regardless of the length of the session id. Similarly, SL4J 4.3.0 upwards maintains backwards compatibility with earlier Liberator versions.

session-timeout

Sets the time in seconds for which the Liberator will maintain a session if a user has connected but not managed to log in.

Type: integer

Default value: 60

session-reconnect-timeout

Sets the time in seconds for which the Liberator will maintain a session following a disconnection.

Type: integer

Default value: 30

session-heartbeat

The interval in seconds between heartbeats sent from the server to the RTTP client.

Type: integer

Default value: 0 (no heartbeats)

12.12 Clustering

This section of *rttpd.conf* configures the clustering of multiple Liberators.

cluster-index

The index number of this cluster node.

Type: integer

Default value: 0

cluster-cache-request-objects

Determines whether to request objects when other Liberators do.

Type: boolean

Default value: FALSE

cluster-cache-source-routing

Determines whether to request objects from the same source as other Liberators.

Type: boolean

Default value: FALSE

cluster-addr

Network interface of this node.

Type: boolean

Default value: FALSE

cluster-port

Network port of this node.

Type: boolean

Default value: FALSE

type1-host

RTTP type 1 host.

Type: boolean

Default value: FALSE

type1-port

RTTP type 1 port.

Type: boolean

Default value: FALSE

type2-url

RTTP type 2 url.

Type: boolean

Default value: FALSE

cluster-cache-source-routing

Determines whether to request objects from the same source as other Liberators.

Type: boolean

Default value: FALSE

priority

Adds a cluster node.

Syntax:

```
add-cluster-node
      cluster-addr      [value]
      cluster-port      [value]
end-cluster-node
```

The options in this entry are:

Name	Type	Default	Description
cluster-addr	string	127.0.0.1	Network interface of this node.
cluster-port	integer	2333	Network port of this node.

12.13 Fields

This section of *rttpd.conf* configures the fields contained in objects. For more information see “About RTTP objects” on page 71

fields-file

Name of an alternative file for fields configuration.

Type: string

Default value: fields.conf

add-field

Adds a field.

Syntax:

```
add-field   FieldName   FieldNumber    FieldFlags   [FieldFlagsData]
```

The options in this entry are:

Name	Type	Default	Description
FieldName	string	[no default]	The name of the field.
FieldNumber	integer	[no default]	The number of the field. Must be between -65535 and 65535 inclusive.
FieldFlags	integer	0	The flags passed by the field (see the section entitled “Identifying the fields clients can request” on page 88 for more information).
FieldFlagsData	integer	-1	Number of decimal places the field uses when FieldFlags is set to 256 (see the section entitled “Identifying the fields clients can request” on page 88 for more information).

FieldFlags (text)	FieldFlags (integer)	Description	FieldFlagsData	FieldFormat
type2_index index	1	Identifies field as Type 2 index	Not used	Not used
type2	2	Identifies field as Type 2 field	Not used	Not used
type3	4	Identifies field as Type 3 field	Not used	Not used
dp decimal_precision	256	Decimal point precision mode	Number of decimal places	Not used

Table 12-2: Acceptable values of FieldFlags option

Note: *Due to the way RTTP encodes field names, message sizes can be reduced slightly by configuring the most commonly used fields nearer the top of the fields.conf file.*

ignore-unknown-fields

Suppresses the generation of error log messages when Liberator receives unknown fields. By default, Liberator logs an error when it receives a field that it does not recognize.

Type: boolean

Default value: FALSE

requested-fields-only

Enables only the fields requested by the client to be sent by Liberator.

Type: boolean

Default value: FALSE

numeric-locale

Locale for numeric field formatting

12.14 DataSource peers

This section of *rttpd.conf* is used to configure the connections between the Liberator and its sources of data, called DataSource peers. A DataSource peer is a remote application which uses DataSource messaging to send real time data to the Liberator.

datasrc-name

The name of the Liberator, and how DataSource peers will identify it.

Type: string

Default value: %a-%h

datasrc-id

ID number of this Liberator.

Type: integer

Default value: 0

datasrc-reject-new-peers

Determines whether a DataSource peer trying to connect to the Liberator when there is already one connected with the same id, is forbidden to connect.

Type: boolean

Default value: FALSE

datasrc-pkt-log

Name of the Liberator packet log file.

Type: string

Default value: packet-rttpd.log (packet-%a.log)

datasrc-interface

Network interface to listen for connections from DataSource peers.

Type: integer

Default value: [all available interfaces]

datasrc-port

Network port to listen for connections from DataSource peers. The default of 0 means that no connections can be made to Liberator.

Type: integer

Default value: 0

datasrc-sslport

Network port to listen for SSL connections from DataSource peers.

Type: integer

Default value: 0 (no SSL connections can be made)

datasrc-default-obj-hash-size

Default number of entries in the active object hashtable.This size can be overridden by putting a value in the ***obj-hash-size*** option of the ***add-peer entry***.

Type: integer

Default value: 16384

datasrc-rerequest-timeout

The time in seconds that the Liberator waits for a DataSource peer to respond when rerequesting an object that the DataSource peer was previously sending to Liberator.
A re-request happens when a DataSource peer goes down and comes back up.

Default value: 30

add-peer

Adds a DataSource peer. You can have a maximum of 1023 add-peer entries in your configuration file.

Syntax:

```
add-peer
      <option>                [value]
      ...
end-peer
```

The options in this entry are:

Name	Type	Default	Description
remote-id	integer	1	ID number of DataSource peer.
remote-name	string	scr-0	Name of DataSource peer. Gets overridden by the startup packet when the peer connection is made.
remote-flags	integer	0	DataSource peer flags. Gets overridden by the startup packet when the peer connection is made.
remote-type	integer	0	DataSource peer type. Gets overridden by the startup packet when the peer connection is made. Possible values are: "`none`" or "`broadcast` or 0 (broadcast, no contributions) "`active`" or 1 (active, no contributions.) "`contrib`" or 2 (broadcast, with contributions) "`active|contrib`" or 3 (active, with contributions). See the Note 2 for further information.
local-id	integer	datasrc-id	ID number of the Liberator. Sent to the DataSource peer.
local-name	string	datasrc-name	Name of the Liberator. Sent to the DataSource peer.

Name	Type	Default	Description
local-flags	integer	0	Flags determining restart and reconnection behaviour. The flags can be ORed together (for example "`sendfromseq|recvautoreplay`"). Possible values: "`none`" or 0 No special restart/reconnection behaviour. "`sendfromseq`" or 1 When reconnecting, missed packets should be requested based on sequence number. "`recvautoreplay`" or 4 When restarting, this peer should accept replay updates.
addr	array of strings	localhost	Space-separated list of addresses to connect to if making the connection and not listening/ accepting the connection. See the Note 1 for further information.
port	array of integers	[no default]	Space-separated list of ports to connect to if making the connection and not listening/ accepting the connection. See the Note 1 for further information.
queue-size	integer	50	Message queue size.
obj-hash-size	integer	***datasrc-default-obj-hash-size***	Number of entries in active object hashtable.

Name	Type	Default	Description
ssl	boolean	False	Determines whether this connection should be made using SSL. For more information on SSL connections, see “Making SSL connections with DataSource peers” on page 123.
request-timeout	float	-1 (no timeout)	Time in seconds that the Liberator will wait for this DataSource peer to answer a request. When set to a positive value it overrides the global request timeout option ***source-request-timeout***.
heartbeat-time	integer	[disabled]	Time in seconds between DataSource heartbeats. The two peers involved in a DataSource connection compare their ***heartbeat-time*** values and use the lowest.
heartbeat-slack-time	integer	2	When the Liberator does not receive an expected DataSource heartbeat it waits ***heartbeat-slack-time*** seconds before disconnecting from the peer and trying to reconnect to it. (This value is not compared by peers.)

Name	Type	Default	Description
thread-name	string	an empty string	Defines a named thread on which communications with the peer are processed. If the same thread name is defined for more than one DataSource peer, all these peers share this same thread. If this option is not defined, or is an empty string, but the global configuration option **peer-thread-pool-size** (see page 238) is more than zero, a thread from the global pool of peer connection threads is allocated to the peer. If neither **thread-name** nor **peer-thread-pool-size** is defined, a dedicated thread is allocated to the peer. **thread-name** can be a maximum of 15 characters long. It is recommended that the first 10 characters be unique within the Liberator so that peer connection thread names can be easily distinguished when appearing in Liberator logs and the output of Linux commands such as `top`. **thread-name** must not be set to a string beginning with “peer” or “ptpt”, as these prefixes are used for internally generated thread names.

Name	Type	Default	Description
label	string	peer[int]	There must be a label set for each ***label*** used in the ***add-priority*** section of the ***add-data-service*** option (see page 228).

Note 1: ***addr*** *and* ***port*** *should only be included if the connection is to be made to the peer as opposed to listening for a connection. If additional addr and port combinations are given they will be used as failover addresses if the first fails to connect (the peer must be configured to accept connections—this is done through the* ***datasrc-port*** *entry in the peer's configuration file).*

Note 2: *The* ***remote-type*** *can appear in log files and is available to monitoring utilities. In addition, if a client application requests data provided by a DataSource peer of this type, but the remote DataSource peer has not yet connnected to the Liberator, Liberator uses this information to inform the client application that the service is not available because the DataSource peer is down.*

Note 3: ***thread-name:*** *For more information on configuring DataSource peer connection threads, see "Improving performance using threads" on page 151.*

start-ssl

Configures the SSL connection when setting up the Liberator to be both client and server ends of a Secure Sockets Layer channel.

Note: *Not all options listed in this group are appropriate for both server and client modes.*

Syntax:

```
start-ssl
      enable-server
      enable-client
      server-authmode   [value]
      client-authmode   [value]
      server-cert       [value]
      client-cert       [value]
      server-key        [value]
      client-key        [value]
      cipher            [value]
      ssl2
      ssl3
      CApath            [value]
      CAfile            [value]
      ssl-info
end-ssl
```

The options in this entry are:

Name	Type	Default	Description
enable-server	boolean	FALSE	Enables server-side SSL.
enable-client	boolean	FALSE	Enables client-side SSL.
server-authmode	integer	0	A logical OR of the flags described in Table 12-3 on page 220. Exactly one of the mode flags SSL_VERIFY_NONE and SSL_VERIFY_PEER must be set at any time.

Name	Type	Default	Description
client-authmode	integer	0	A logical OR of the flags described in Table 12-3 on page 220. Exactly one of the mode flags SSL_VERIFY_NONE and SSL_VERIFY_PEER must be set at any time.
server-cert	string	server.pem	Filename of the location of the server-side certificate.
server-key	string	server-cert	Filename of the location of the server-side private key.
client-cert	string	NULL	Filename of the location of the client-side certificate.
client-key	string	client-cert	Filename of the location of the client-side private key.
cipher	string	[strongest common cipher]	Sets the cipher to be used for the connection (usually] defined on client side). Cipher types can be identified using the https-cipher-list option (see page 182).
ssl2	boolean	FALSE	Sets the SSL protocol level to Level 2.
ssl3	boolean	FALSE	Sets the SSL protocol level to Level 3.
CApath	string	System CApath	Sets the directory name of the directory where the trusted certificates are held.

Name	Type	Default	Description
CAfile	string	NULL	Sets the filename of the file where all trusted certificates are held.
ssl-info	boolean	FALSE	Enables SSL connection negotiation debugging.

client-authmode and server-authmode flags

Table 12-3 below describes the flags to be used for the *client-authmode* and *server-authmode* options within the start-ssl group.

Name	Value	server-authmode description	client-authmode description
SSL_VERIFY_NONE	0	Liberator will not send a client certificate request to the client, so the client will not send a certificate.	If not using an anonymous cipher (disabled by default), the Liberator will send a certificate which will be checked. The handshake will be continued regardless of the verification result.
SSL_VERIFY_PEER	1	Liberator sends a client certificate request to the client. The certificate returned (if any) is checked. If the verification process fails, the TLS/SSL handshake is immediately terminated, with an alert message containing the reason for the verification failure. The behaviour can be controlled by the additional SSL_VERIFY_FAIL and SSL_VERIFY_CLIENT_ONCE flags.	The server certificate is verified. If the verification process fails, the TLS/SSL handshake is immediately terminated with an alert message containing the reason for the verification failure. If no server certificate is sent, because an anonymous cipher is used, SSL_VERIFY_PEER is ignored.

Name	Value	server-authmode description	client-authmode description
SSL_VERIFY_FAIL	2	If the client did not return a certificate, the TLS/SSL handshake is immediately terminated with a "handshake failure" alert. This flag must be used together with SSL_VERIFY_PEER.	Ignored
SSL_VERIFY_CLIENT_ONCE	4	Only request a client certificate on the initial TLS/SSL handshake. Do not ask for a client certificate again in case of a renegotiation. This flag must be used together with SSL_VERIFY_PEER.	Ignored

Table 12-3: client-authmode and server-authmode flags

ssl-passwordfile Identifies the file containing the SSL certificate passphrase.

Type: string

Default value: .rttpd.ssl.pass

12.15 Data replay

This section of *rttpd.conf* configures how Liberator replays data.

datasrc-auto-replay

Time (in minutes after midnight) that the server should load previously received messages on a restart.

Type: integer

Default value: 1440 (i.e. no replay)

datasrc-auto-replay-days

The number of whole days to go back from the time indicated by datasrc-auto-replay (if less than 1440).

Type: integer

Default value: 7

datasrc-auto-replay-files

Specifies a list of log files to replay.

Type: string

Default value: 0

12.16 Data services

This section of *rttpd.conf* configures Liberator's data services.

What is a data service?

See "Data services" on page 113 for an overview of what data services are and how they are used.

The service name

This is an identifier which can be used in status messages. RTTP objects have a field called SID which is the service name.

Note: *When picking a service for an object, the first defined match takes priority. As such you should ensure that each object is associated with one and only one service.*

The subject patterns

These are regular expression strings to accept or deny for this service. A service will allow multiple patterns including patterns to deny. Exclude patterns can help to define the namespace used.

Note: *When checking pattern matches within a service definition, the first match takes priority whether it is an include or an exclude.*

The source groups

The main part of the service definition is the source groups. This is one or more sets of sources, plus certain attributes which define the behaviour of the group. In most cases only a single group is defined. When multiple groups are defined for a service it means that a request will attempt to get the object from a source from each group. Multiple groups allow an object to have different sets of fields coming from different sources, for example.

Priorities

Priorities are defined within each source group and are taken in the order in which they are defined. Multiple labels can be defined within each priority. Within a priority, the peer selected is the one with the smallest number of existing subscriptions.

Timeouts

There are several timeouts associated with data services:

- ***service-request-timeout*** (see page 223)
- ***request-timeout*** option of ***add-data-service*** (see page 224)
- ***retry-time*** option of ***add-source-group*** in ***add-data-service*** (see page 227)
- ***source-request-timeout*** (see page 223)
- ***request-timeout*** option of ***add-peer*** (see page 214)
- ***cleanup-stale-timeout*** (see page 223)

- ***discard-timeout*** of ***add-data-services*** (see page 226)

Use the following options to configure data services.

service-request-timeout

Global request timeout in seconds for all data services.

Type: float

Default value: 10

The ***service-request-timeout*** applies to a request for an object via a service. Should no response be received within this time from the peers providing a service, the object will be assumed to be not available.

service-request-timeout is a global setting applied to all data services. You can override this timeout for individual DataServices by setting the ***request-timeout*** option of ***add-data-service*** – see page 224.

source-request-timeout

Global request timeout in seconds for all DataSource peers.

Type: float

Default value: -1 (no timeout set)

The ***source-request-timeout*** applies to individual DataSource peers within data services. It is the time that the Liberator will wait for a DataSource peer to answer a request.

source-request-timeout is a global timeout that applies to all DataSource peers. You can override this timeout for individual peers by setting the ***request-timeout*** option of ***add-peer*** – see page 214.

cleanup-stale-timeout

When all the DataSource peers in a service have been down for ***cleanup-stale-timeout*** seconds, stale objects will be deleted from the Liberator cache .

Type: float

Default value: -1 (no timeout set)

add-data-service

Starts the definition of a data service. Syntax:

```
add-data-service
      service-name           [value]
      request-timeout        [value]
      exclude-pattern        [value]
      include-pattern        [value]
      required-state         [value]
      discard-timeout        [value]
      add-source-group
            required         [boolean]
            retry-time       [value]
            add-priority
                  label      [value]
                  ...
            end-priority
            ...
      end-source-group
      ...
end-data-service
```

service-name

Name of the service group.

Type: string

Default value: none

request-timeout

This option defines the timeout in seconds for all requests for this service. Should no response be received within this time from the peers providing the service, the object is assumed to be unavailable. The ***request-timeout*** is the master timeout, it overrides the ***retry-time*** option and the timeouts on individual peer requests (see ***retry-time*** on page 227).

Type: float

Default value: -1

The default value of -1 means that requests will never time out.

exclude-pattern

Patterns to exclude, defined as regular expressions. For examples of the specification format see “include-pattern” on page 225 .

Type: string array

Default value: none

include-pattern

Patterns to include, defined as regular expressions.

Type: string array

Default value: none

Example 1:

```
include-pattern    ^/A/ ^/B/ ^/C/
```

This example specifies three patterns to include: ‘^/A’, ‘^/B’ and ‘^/C’.

Example 2:

```
include-pattern    ^/A/
include-pattern    ^/B/
include-pattern    ^/C/
```

This example specifies the same three patterns as in example 1, but as separate ***include-pattern*** statements.

required-state

Specifies the state that the service must be in when the Liberator starts up, in order for the Liberator to accept client connections.

Type: String with one of the values:
`down`
`limited`
`ok`

Default value: `down`

When the Liberator starts up it responds to the setting of the ***required-state*** option as follows:

required-state value	Liberator startup behaviour
`down`	Liberator accepts client connections.
`limited`	Liberator only accepts client connections if the service has status “limited” or “ok”.
`ok`	Liberator only accepts client connections if the service has status “ok”.

The Liberator will only accept client connections if the status of each specified service matches the corresponding ***required-state*** option according to the above table.

The default value of `down` means that, if no ***required-state*** options are specified for any services, the Liberator will accept client connections at start up, regardless of the state of the services.

The Liberator checks service status against the ***required-state*** option only at start up. For example, if ***required-state*** is `ok` and the service status is “ok” at start up, then client connections are accepted, but If the service status subsequently changes to “down”, the Liberator will continue to accept client connections.

discard-timeout

Specifies the time in seconds for which the Liberator will hold on to an active object obtained through this data service after the last user stops viewing it. After this time the object is deleted from the Liberator’s cache and Liberator sends a discard instruction to the DataSource peer to cancel the subscription to the object.

This option overrides the setting of the global time out for active objects, ***active-discard-timeout*** (see page 190 and “Discarding objects” on page 120). If an object obtained by this data service has been specified through ***add-object***, the ***discard-timeout*** option of the ***add-object*** (if any) overrides *this* ***discard-timeout*** (see page 191).

Type integer

Default value The setting of the global time out for active objects: ***active-discard-timeout*** (see page 190).

add-source-group

Add a source group.

Syntax:

```
add-data-service
...
      add-source-group
            required          [boolean]
            retry-time        [value]
            add-priority
                  label       [value]
                  label       [value]
                  ...
            end-priority
            ...
      end-source-group
...
end-data-service
```

See also the examples in “Data services” on page 113.

required

When set to true, a status stale message or status information message is generated if a DataSource peer within the source group goes down.

Type: boolean

Default value: false

retry-time

After finding that none of the peers (defined by ***label*** options) in the source group have responded to a request, the Liberator waits ***retry-time*** seconds before reissuing the request to the group.

Type: float

Default value: 30

The Liberator will issue the request to each of the peers in the source group in turn. Each request is timed out according to the setting of ***request-timeout*** in the ***add-peer*** option (see page 211). If none of the peers in the group replies, the Liberator waits ***retry-time*** and then again tries each connection in turn. It will repeat this sequence until the master timeout for the service, defined by the ***request-timeout option*** of ***add-data-service***, expires (see page 224). If the

add-data-service request-timeout out is set to -1, or is not defined, the Liberator will continue to reissue the request indefinitely.

add-priority

Start a priority group. Specifies one or more DataSource peers to be assigned to the data service, in a group of the same priority. For examples of how to use this configuration option see “Data services” on page 113.

Syntax:

```
add-data-service
...
      add-source-group
            ...
            add-priority
                  label       [value]
                  label       [value]
                  ...
            end-priority
      ...
      end-source-group
...
end-data-service
```

label

Peer label identifying a peer that provides the data service. The label is inserted in an ***add-priority*** option.

Type: string array

Default value: none

The label is defined using a ***label*** option within the ***add-peer*** configuration option – see “add-peer” on page 211.

Example 1:

```
            add-priority
                  label       src1 src2
            end-priority
```

This example specifies two peer labels, `src1` and `src2`.

Example 2:

```
add-priority
      label       src1
      label       src2
end-priority
```

This example specifies the same two peer labels as in example 1, but as separate ***label*** statements.

Below is an example section of *rttpd.conf* illustrating how data services are configured. See also the examples in “Data services” on page 113.

```
add-data-service

      service-name           FX

      exclude-pattern        ^/I/X/
      include-pattern        ^/I/
      include-pattern        ^/B/

      add-source-group
            required         true
            retry-time       45
            add-priority
                  label      sourceA
            end-priority
            add-priority
                  label      sourceB
                  label      sourceC
            end-priority
      end-source-group

      add-source-group
            required         false
            add-priority
                  label      source1
                  label      source2
            end-priority
      end-source-group

end-data-service
```

Default behaviour

If no data service is defined in *rttpd.conf* then the application will act as if the following data service configuration was defined:

```
add-data-service
      service-name default
      include-pattern         ^/
      required-state          down
      add-source-group
            required          false
            add-priority
                  label       source1
            end-priority
      end-source-group
      add-source-group
            required          false
            add-priority
                  label       source2
            end-priority
      end-source-group
      .
      .
      .
      add-source-group
            required          false
            add-priority
                  label       sourceN
            end-priority
      end-source-group
end-data-service
```

This means that a request will be sent to all active DataSource peers at once. The default ***required-state*** value of down means that the Liberator will accept client connections at start up, regardless of the state of the services.

Conversion of pre-version 4.0 source mapping

Pre-version 4.0 source mapping should be converted to version 4.0 data services, as shown in the following examples.

Note that all peers should have a label defined in the ***add-peer*** configuration section. In the examples the label is 'src' appended with the the remote-id.

add-source-mapping /A/* 1 should be converted to:

```
include-pattern         ^/A/
add-source-group
      required          true
      add-priority
            label       src1
      end-priority
end-source-group
```

add-source-mapping /A/* 1,2 should be converted to:

```
include-pattern         ^/A/
add-source-group
      required          true
      add-priority
            label       src1
            label       src2
      end-priority
end-source-group
```

add-source-mapping /A/* 1 2 should be converted to:

```
include-pattern         ^/A/
add-source-group
      required          true
      add-priority
            label       src1
      end-priority
end-source-group
add-source-group
      required          true
      add-priority
            label       src2
      end-priority
end-source-group
```

add-source-mapping /A/* 1,2 3,4 should be converted to:

```
include-pattern         ^/A/
add-source-group
      required          true
      add-priority
            label       src1
            label       src2
      end-priority
end-source-group
add-source-group
      required          true
      add-priority
            label       src3
            label       src4
      end-priority
end-source-group
```

12.17 Latency

latency-chain-enable Enable Latency Chaining.

Type:	Boolean
Default:	FALSE

latency-chain-name Latency Chain Name used for event list field.

Type:	String
Default:	%a

latency-chain-init-ts-field Latency Chain Init Timestamp Field Name.

Type:	String
Default:	LTY_INIT_TS

latency-chain-list-event-field

Latency Chain Event List Field Name.

Type: String

Default: LTY_LIST_EVENT

latency-chain-list-ts-field

Latency Chain Timestamp List Field Name.

Type: String

Default: LTY_LIST_TS

latency-chain-base64-mode

This option defines how latency chain field values will be processed with respect to base64 encoding. The options can be ORed together, for example 'decode|encode' will decode the field values, add the component entries onto the end of the field values, then encode the final values.

Type: Integer

Default: 0

Note: *'Encode' will only convert a value to base64 that has just been decoded, it will not encode values that have arrived in plain text.*

Acceptable Values:

Name	Value	Desc
never	0	Default - do not treat values as base64 encoded
decode	1	Decode latency chain fields before processing
detect	2	Decode latency chain fields if they are encoded
encode	4	Encode latency chain fields after processing

12.18 Tuning

This section of *rttpd.conf* configures the more advanced options available in Liberator. These are dealt with in more depth in Optimising efficiency on page 151

object-hash-size

Size of RTTP object hashtable. This should be approximately the number of objects the Liberator will hold.

Type: integer

Default value: 5000

user-hash-size

Size of user hashtable.

Type: integer

Default value: 8192

session-hash-size

Size of session hashtable.

Type: integer

Default value: 8192

session-max-queue-length

The size the queue in the server waiting to be sent to the client must reach before the server starts counting consecutive increases to the queue length.

Type: integer

Default value: 5242880

session-max-queue-count

This is the number of consecutive times the queue length in the server has to increase after the session-max-queue-length has been reached before the connection is dropped.

Type: integer

Default value: 10

burst-min

Starting point in seconds of client update buffering (i.e. start of burst).

Type: float

Default value: 0.1

burst-max

Maximum time in seconds of client update buffering.

Type: float

Default value: 0.5

burst-increment

Burst buffer delay increment in seconds

Type: float

Default value: 0.05

buf-cache-size

Overall size of the buffer cache in megabytes. On top of this the Liberator will use about 15Mb for core memory, and this memory requirement will increase as the amount of users and data increase.

Type: integer

Default value: 16

buf-elem-len

Length of standard buffer element, in bytes.

Type: integer

Default value: 4096

output-queue-size

The number of update messages the Liberator will store per client.

Type: integer

Default value: 64 (maximum is 4096)

threads-num

Number of session threads to run.

Type: integer

Default value: 1

Note: ***The maximum permitted number of session threads is 10.***
If you set the value of threads-num to more than 10, Liberator reduces it to 10, and this is logged on the console and in the event log when Liberator starts up.

add-thread

Configures the options for each session thread.

Syntax:

```
add-thread
      http-interface    [value]
      http-port         [value]
      direct-interface  [value]
      direct-port       [value]
end-thread
```

The options in this entry are:

Name	Type	Default	Description
http-interface	string	[All available interfaces]	Network interface to listen for HTTP connections.
http-port	integer	8080	Network port to listen for HTTP connections.
direct-interface	string	[All available interfaces]	Network interface to listen for direct (type1) RTTP connections.
direct-port	integer	15000	Network port to listen for direct (type1) RTTP connections.

peer-thread-pool-size The number of pooled DataSource peer connection threads.

Type: integer

Default value: 0 (No pool of DataSource peer connection threads is created.)

If a DataSource peer does not have a named thread configured for it (see the **thread-name** option of **add-peer** on page 215), communication with the peer is handled on one of the pool threads, unless **peer-thread-pool-size** is 0 or not defined, in which case a thread is created and dedicated to that peer. If **peer-thread-pool-size** is less than the number of peer connections with no explicitlynamed thread, these connections will share threads from the pool.

direct-tcp-nodelay-off Turns off the no delay feature for direct sockets.

Type: boolean

Default value: FALSE

http-tcp-nodelay-off Turns off the no delay feature for HTTP sockets.

Type: boolean

Default value: FALSE

datasrc-tcp-nodelay-off Turns off the no delay feature for DataSource peer sockets.

Type: boolean

Default value: FALSE

batch-discard-time Batch time for active discards

Type: float

Default value: 2.0

object-delete-batchtime Time for batching up deletes

Type: float

Default value: 0.5

object-delete-time Time delay for deleting a group of objects

Type: float

Default value: 0.5

12.19 News

This section of *rttpd.conf* configures the way in which the Liberator handles requests for news headlines.

newsitems-saved

Maximum number of news items (headlines) that Liberator stores in memory.

Type: integer

Default value: 500

newsitems-max

Maximum number of news items that the Liberator will send to any particular client for any one request.

Type: integer

Default value: 500

newscode-max-length

Determines the maximum length of a news code.

Type: integer

Default value: 4

newscode-exceptions

Determines whether there are any exceptions to the newscode-max-length rule (i.e. whether there are any news codes that are longer than newscode-max-length).

Type: boolean

Default value: FALSE

add-newscodes

If there are permissible exceptions to newscode-max-length, this parameter should include an array of codes listing the permitted exceptions.

Type: string array

Default value: [no default]

newscode-hash-size Default number of entries in the newscode exceptions hashtable.

Type: integer

Default value: 191

news-purge-time This represents the number of minutes from midnight that the purge of news headlines (i.e. deletion from the Liberator's cache) should take place.

Type: integer

Default value: -1 (no purge, in which case newsitems-max will limit the number of headlines stored.)

news-purge-days Number of days-worth of headlines to keep when purging.

Type: integer

Default value: 0

news-datetime-format Time string used for news headline items (UNIX users should refer to strftime within your Unix manual for further information).

Type: YY mm HH:MM:SS

Default value: "%b [int] %H:%M:[str]" (i.e. current year, month, hours, minutes and seconds)

newscodes-valid-chars A list of characters that are valid in a news code.

Type: string

Default value: "/." (any uppercase characters and the characters "/" or "." (for example "FIN" or "BT.L").

news-log Filename of log file to store news headlines for replaying on startup.

Type: string

Default value: [no news headlines stored]

news-replay

Time (in minutes after midnight) that the server should start replaying news headlines on a restart.

Type: integer

Default value: OFF

news-replay-days

The number of whole days to go back from the time indicated by news-replay (if news-replay less than 1440).

Type: integer

Default value: 0

news-replay-files

The news logs to replay.

Type: string array

Default value: news-log

newsitems-hash-size

Size of the news items hashtable

Type: integer

Default value: 191

12.20 KeyMaster

This section of *rttpd.conf* configures the way in which user signatures are authenticated. For more information on how the Liberator authenticates users, see “Authentication and entitlement” on page 97.

signature-validtime

How long a generated signature is valid for, in seconds.

Type: integer

Default value: 600

signature-hashsize

Size of hashtable for storing signature keys.

Type: integer

Default value: 8192

add-sigkey

Adds a signature checking key to the configuration file.

Syntax:

```
add-sigkey
      key-id           [values]
      timeout          [values]
      keyfile          [values]
      hashing-algorithm [values]
end-sigkey
```

The options in this entry are:

Name	Type	Default	Description
key-id	string	[no default]	The identifier of this signature key. When Liberator uses the cfgauth authorization module, key-id must be the same as the ***siguser*** option of an ***add-user*** entry in the cfgauth configuration file (see ***add-user*** for *cfgauth.conf* on page 248).
timeout	float	600	How long a generated signature is valid for, in seconds. Overrides ***signature-validtime*** (see page 243).
keyfile	string	[no default]	Filename of public key.

Name	Type	Default	Description
hashing-algorithm	string	md5	Used to change the hashing algorithm used in KeyMaster authentication. Permitted values for Java-based KeyMaster are “md5” and “sha256”. “md5” means the MD5withRSA algorithm, sha256 means the SHA256withRSA algorithm. For Java-based KeyMaster the set value must match the setting of the KeyMaster *web.xml* parameter ***encrypting.generator.signature.algorithm*** Permitted values for KeyMaster.NET are ”md5”, “sha1”, “sha256”, “sha384”, “sha512”, and “ripemd160”, For more information, see the **KeyMaster Administration Guide** (KeyMaster 4.4 May 2009, or later), and the ***KeyMasterHashingAlgorithm*** enumeration in the **KeyMaster.NET API Reference**.
signing-algorithm	string	md5	Deprecated – use ***hashing-algorithm***.

12.21 UDP interface

udp-port

Port to listen on for UDP messages. If not specified then udp signals are disabled.

Type: integer

Default value: [no default]

udp-interface

Network interface to listen on for UDP messages.

Type: integer

Default value: [all available interfaces]

12.22 Openauth.conf configuration

read-access Determines all users' read access to objects.

If set to 0 no user can view any objects;

If set to 1 all users can view all objects.

Default value: 1

write-access Determines all users' permission to write to or create any object.

If set to 0 no user can write to any object;

If set to 1 all users can write to any object.

Default value: 0

12.23 Cfgauth.conf configuration

add-user

Adds a user to the cfgauth configuration file.

The entry must use the following syntax:

```
add-user
      username    [values]
      password    [values]
      licenses    [values]
      read        [values]
      write       [values]
      prefix      [values]
      sigcheck
      siguser     [values]
      http
      expire      [value]
end-user
```

The options in this entry are:

Name	Type	Default	Description
username	string	[no default]	The username for this user.
password	string	[no default]	The password for this user. If encrypted-passwords is set to 1 then this should be an encrypted password as produced by the cfgpass utility (see encrypted-passwords on page 250).
licenses	integer	1	The number of licenses this user has.
read	integer array	none	Space-separated list of object types this user can read. The types are listed in the default cfgauth.sample file.
write	integer array	none	Space-separated list of object types this user can write to.

Name	Type	Default	Description
prefix	string	none	This is an optional prefix that will be added to all requests by this user.
sigcheck	boolean	FALSE	If set to TRUE ignores password and uses the signature checker to authenticate the user. The signature checker works by having a public key specified by an add-sigkey entry in *rttpd.conf* (see Signature Authentication on page 103). add-sigkey includes a user parameter, which must be the same as the siguser parameter below to authenticate a user.
siguser	string	username	If sigcheck is set to TRUE, this option is used to identify the user for the purpose of checking signatures. This means you can have several users pointing to the same signature checking key—siguser must be the same as the user parameter in an add-sigkey entry in *rttpd.conf*. “Signature authentication” on page 103for more information.
http	boolean	FALSE	If set to TRUE allows HTTP authentication for this user. See auth_http_request function in the accompanying document **Liberator Auth Module Developer's Guide**.
expire	YYYYMMDDNN	NULL	If set, defines a start date and number of days this user is valid for. The format of the string should be YYYYMMDDNN, where NN is the number of days the user is valid for.

encrypted-passwords Determines whether a password is encrypted or not.

If set to 0	password is set as clear text;
If set to 1	password is encrypted.
Default value:	0

12.24 Licensing

UUPP

The uupp database is the license usage database in which Liberator records information for checking license compliance. UUPP means **U**nique **U**sers **P**er (license monitoring) **P**eriod.

Configuration options:

Name	Type	Default	Description
uupp-qdbm-name	STRING	%r/users/uupp-%a.conf [%a will be replaced by the application name, eg rttpd]	Location of the database.
uupp-delimiter	CHAR	':'	Used to separate application name/user id - it may need to be changed if users/apps can have the delimiter in the name
uupp-sync-time	FLOAT	300 (secs)	Time to synchronise the database to disc

Note: *The license file can be found in the etc directory of the root of your Liberator installation.*

Note: *For more information about configuration items relating to licensing, refer to the document* ***Caplin Platform: Guide to User Licensing****.*

12.25 Java Configuration

All java configuration options are now held in an external file from *rttpd.conf*.

java-file

Name of an alternative file for java configuration.

Type: string

Default value: java.conf

12.26 Java.conf configuration

A Java Virtual Machine (JVM) is required to execute Java modules created using the Java Auth SDK and to enable JMX Monitoring.

Note: *When using Linux, LD_LIBRARY_PATH must be set to /usr/java/jre/lib/i386:/usr/java/jre/lib/i386:/server where the java runtime environment is installed in /usr/java/jre.*

jvm-location

Location of the Java Virtual Machine file libjvm.so. Should contain the complete path and include the .so suffix.

Type: string

Default value: [no default]

jvm-global-classpath

Location of the global classpath. There must be a separate **jvm-global-classpath** entry for each classpath.

Type: string

Default value: %r/lib/java

add-javaclass

Identifies the Auth SDK Java module to be loaded and is also used to specify the JMX monitoring console.

syntax:
add-javaclass
class-name
class-id
classpath
end-javaclass

The options in this entry are:

Name	Type	Default	Description
class-name	string	[no default]	The fully-qualified class name of the Java module to load.

Name	Type	Default	Description
class-id	string	0	Short identifier of the Java class.
classpath	string array		Adds a Java classpath.

jvm-options

Adds a standard startup option for the JVM. Multiple configuration lines may be specified.

Type: string

Default value: no default

For example, to enable socket debugging on port 9955 the following configuration options could be added:

jvm-options -Xdebug

jvm-options -Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=9955

You can also change the size of the heap in the Java Virtual Machine from the default settings (you may need to do this for performance reasons):

jvm-options -Xms${JVM_MIN_HEAP_SIZE} -Xmx${JVM_MAX_HEAP_SIZE}

{JVM_MIN_HEAP_SIZE} is the initial heap size in megabytes.

{JVM_MAX_HEAP_SIZE} is the maximum heap size in megabytes.

Note: *It is recommended to set {JVM_MIN_HEAP_SIZE} and {JVM_MAX_HEAP_SIZE} to the same value.*

12.27 Monitoring configuration

When the monitoring module is JMX, add configuration to *java.conf*, as in the following example (you can just uncomment these lines in the *java.conf* file supplied with Liberator):

```
add-javaclass
    class-name  com.caplin.management.jmx.JMXController
    class-id    jmx
    classpath   %r/lib/java/jmx-child-classloader.jar
    classpath   %r/lib/java/common-jmx.jar
end-javaclass
```

The following configuration options are specified in in *rttpd.conf*.

monitor-plugin

Loads the monitoring module into the Liberator

To load the sockmon (socket-based) monitoring module.

syntax: ***monitor-plugin sockmon***

To load the JMX monitoring module

syntax: ***monitor-plugin jmx***

add-monuser

Specifies the credentials that allow a monitoring client application to log into the Liberator. There are several ways to specify these credentials:

syntax (alternative 1a):

```
add-monuser
        user  username
        pass  password
        addr  127.0.0.1
end-monuser
```

syntax (alternative 1b):

```
add-monuser
    user        username
    key-id      akeyidentifier
    addr        127.0.0.1
end-monuser
```

Note: *The addr option can only be specified when the monitoring module is sockmon (socket-based monitoring). Do not specify an addr option when the monitoring module is JMX, as it not possible to restrict the network address from which a JMX enabled monitoring client can connect to Liberator.*

The options for these entries are:

Name	Type	Default	Description
user	string	[any]	The username that the monitoring client application will use to log in to the Liberator. If the ***user*** and ***pass*** options, or the ***user*** and ***key-id*** options, are not specified, then the Liberator will accept monitoring login requests from *any* user with the IP address specified in the addr option.
pass	string	[any]	The password that the monitoring client application will use to log in to the Liberator.
key-id	string	[any]	The identifier of a signature key, as specified in the ***key-id*** option of an ***add-sigkey*** configuration entry (see “KeyMaster” on page 243). This option should be used instead of the ***pass*** option when the monitoring client application will log in using a KeyMaster user credentials token. See “Use of the key-id option” on page 259.

Name	Type	Default	Description
addr	string	[any]	An IP address in dot notation (n.n.n.n). The Liberator will accept monitoring login requests from this IP address only. Just ONE address may be specified per *add-monuser* block. If multiple addresses are required then define each one in a separate *add-monuser* block. If the addr option is not specified then the Liberator will accept monitoring login requests from *any* IP address. ***Note:*** This option can only be specified when the monitoring module is sockmon (socket-based monitoring). *Do not specify an addr option when the monitoring module is JMX, as it not possible to restrict the network address from which a JMX enabled monitoring client can connect to Liberator.*

Example:

```
add-monuser
      user  admin
      pass  admin
      addr  192.168.201.107
end-monuser
```

In this example the Liberator will accept monitoring login requests from the IP address 192.168.201.207, where the user login name is “admin” and the user’s password is also “admin”.

For the sockmon monitoring module only, you can also specify the network access credentials using network and netmask specifications, instead of a specific IP address. This will allow login requests from multiple addresses within a network, without needing to define multiple *add-monuser* blocks.

syntax (alternative 2a):

```
add-monuser
    user        admin
    pass        admin
    network     192.168.201.0
    netmask     255.255.255.0
end-monuser
```

syntax (alternative 2b):

```
add-monuser
    user        admin
    key-id      akeyidentifier
    network     192.168.201.0
    netmask     255.255.255.0
end-monuser
```

The network specification options for this entry are:

Name	Type	Default	Description
network	string	[no default]	An IP network specification in dot notation (n.n.n.n).
netmask	string	[no default]	An IP netmask specification in dot notation (n.n.n.n).

Note: *The network and netmask options can only be specified when the monitoring module is sockmon (socket-based monitoring). Do not specify these options when the monitoring module is JMX, as it not possible to restrict the network address from which a JMX enabled monitoring client can connect to Liberator.*

Example:

```
add-monuser
      user        admin
      pass        admin
      network     192.168.201.0
      netmask     255.255.255.0
end-monuser
```

In this example the Liberator will accept monitoring login requests from any host address on the 192.168.201.0 network.

Note: *It is recommended that you specify an add-monuser entry with explicit user, pass or key-id, and addr options (or network and netmask options instead of addr).*
If the add-monuser entry has no options explicitly defined, or is missing altogether, the Liberator will by default accept monitoring login requests from any user on any IP address.

The default Liberator configuration file in the install kit contains the following monitoring client login credentials:

```
add-monuser
      user        admin
      pass        admin
      addr        127.0.0.1
end-monuser
```

This configuration assumes that you are using sockmon (socket-based) monitoring, and will only permit the monitoring client to access the Liberator from the local machine. This may mean the monitoring console will not connect, so you may need to change these options.

Note: *If you are using JMX monitoring, you must remove the addr option from the default configuration shown above, otherwise the JMX-based monitoring client will not be able to connect to the Liberator.*

The following configurations will allow a monitoring client console to log in to the Liberator from *any* network address:

```
add-monuser
      user        admin
      pass        admin
end-monuser
```

or (sockmon monitoring only)

```
add-monuser
      user        admin
      pass        admin
      network     0.0.0.0
      netmask     0.0.0.0
end-monuser
```

Use of the key-id option

A monitoring client application can be designed so that it logs in to the Liberator through KeyMaster, supplying a digitally signed user credentials token instead of a password.

When this is the case, specify the ***add-monuser*** options ***user*** and ***key-id***, rather than ***user*** and ***pass***. The key-id option should correspond to the ***key-id*** value in an ***add-sigkey*** configuration item (see page 243).

The ***add-sigkey*** item specifes a signature checking key. When the JMX enabled client application logs in, Liberator will look for an ***add-monuser*** entry with a matching username and then find the ***add-sigkey*** item that has a ***key-id*** value matching the ***key-id*** in ***add-monuser***. It uses the ***keyfile*** option of the ***add-sigkey*** item to locate the file containing the user's public encryption key. The Liberator's auth module can then use this public key to validate the digital signature in the user credentials token.

log-monitor-level

Specifies the log level for the monitoring log file. This file will be located with the other Liberator log files in the *var* directory. The file name will depend on the mode the user is running the Liberator in. For JMX monitoring, the file will always be prefixed with jmx-. So for example if the Liberator was running in SSL mode then the file would be *jmx-rttpd_ssl.log*. Log wrapping will be applied to this file if wrapping is enabled.

syntax: log-monitor-level LEVEL

Please see “Appendix C: Javaauth configuration” on page 277 for valid values of LEVEL.

monitor-moddir

Monitor module directory

If the first two characters are %r then this will be expanded as relative to the liberator application root directory.

Type: string

Default value: %r/lib

session-monitoring-interval

Session monitoring interval in seconds (set to -1 to disable)

Type: float

Default value: -1.0

object-monitoring-interval

Object monitoring interval in seconds (set to -1 to disable)

Type: float

Default value: -1.0

process-usage-period

Defines the time interval in seconds at which the Liberator’s CPU time counters ***user-cputime-total*** and ***system-cputime-total*** are updated. These counters can be viewed through the monitoring and management subsystem. If you are using JMX monitoring, these counters are available to the Caplin Management Console; they can be viewed in the Explorer tab under rttpd.server.system.

Type: float

Default value: 10 seconds.

12.28 Javaauth configuration

This configuration should be added to java.conf as in the following example:

```
add-javaclass
      class-name  examples/DelayedLoginAuthenticator
      class-id    authenticator
      class-path  /home/dom/src/rttpd/src/lib/java/examples.jar
end-javaclass
```

Where there is an entry:

```
javauth-classid          authenticator
```

in *javaauth.conf*.

debug-level

This sets the debug level.

Type: string

Default value: DEBUG

javaauth-classid

This specifies the class-id to load.

Type: string

Default value: javaauth

13 Appendix B: Log file messages and formats

13.1 Session log

The session log records actions and events regarding Liberator sessions and connections.

Session log messages

Table 13-1 lists the possible messages that will be written to the session log.

Message type	Description	EXTRA values
OPEN	New session opened. Client has connected.	[none]
CLOSE	Session closed.	Reason for session closing: CHUCKOUT The server has terminated its session with the client because the server is unable to write data to the client. This can happen when the server is not consuming the data sent to it from Liberator quickly enough. The Liberator's internal request queue fills up, causing the Liberator to terminate the session with the client. This situation could arise if the client sends a high volume of requests to the Liberator and is unable to handle the Liberator's responses in a timely fashion (with or without network delays). CHUCKOUT2 (This reason code is no longer generated) (cont'd...)

CLOSE (...cont'd)		CHUCKOUT3 The server has terminated its session with the client because the client has written a message that is too long (this is probably because of a fault in the StreamLink library being used by the client). RECONNECTED Reconnected on new session, so this one closed TIMEOUT Timeout after lost connection TIMEOUT2 Timeout due to not logging in succssfully. CLOSE_TYPE1 Lost type 1 connection CLOSE_TYPE2 Lost type 2 connection CLOSE_TYPE3 Lost type 3 connection LOGOUT Logout from client
LOST	Session connection lost.	Reason for loss of connection: values as for CLOSE
LOGIN_OK	Session logged in successfully.	[none]
RECON_OK	Session reconnected successfully.	[none]

LOGIN_FAIL	Session failed to login.	Reason for login failure: ACCOUNT_EXPIRED Account expired AUTH_ERROR Auth error (auth module did something unexpected) INCORRECT_PASSWORD Invalid password INVALID_IP_ADDRESS Invalid IP addres SITE_LICENSE_EXCEEDED Site licence exceeded USER_LICENSE_EXCEEDED Userlicence exceeded USER_UNKNOWN Invalid username
LOGOUT_OK	Session logged out.	[none]

Table 13-1: Session log messages

Note: *Additional information about licenses is also written to the Liberator event log; for details see the* ***Caplin Platform: Guide to User Licensing****.*

Session log format

All session log messages have the same format:

TIMESTAMP IP-ADDRESS SESSION-TYPE MSG-TYPE USERNAME APPLICATION-ID SESSION-ID REASON [EXTRA]

For RECON_OK the last field gives the previous session ID (i.e. the session ID of the session that has been reconnected to).

The REASON field is only used by CLOSE, LOST and LOGIN_FAIL. Otherwise it is just LOGIN_OK or LOGOUT_OK.

Example:

```
2011/06/25-04:55:54.101 +0100: 192.168.201.210 LOGIN_OK maggie 1001RK 0
2011/06/25-05:05:59.301 +0100: 192.168.201.210 LOGOUT_OK maggie 1001RK 0
2011/06/25-05:05:59.201 +0100: 192.168.201.210 CLOSE - 1001RK 1
2011/06/25-05:07:01.201 +0100: 192.168.201.210 OPEN - 1007zX 0
2011/06/25-05:07:01.201 +0100: 192.168.201.210 LOGIN_OK maggie 1007zX 0
2011/06/25-05:14:18.201 +0100: 192.168.201.104 LOST - 0006IW 4
2011/06/25-05:14:18.201 +0100: 192.168.201.104 OPEN - 000346 0
2011/06/25-05:14:18.201 +0100: 192.168.201.104 RECONNECT_OK livedemos
000346 0 0006IW
2011/06/25-05:14:18.201 +0100: 192.168.201.104 CLOSE - 0006IW 64
2011/06/25-05:17:06.201 +0100: 192.168.201.210 LOGOUT_OK maggie 1007zX 0
2011/06/25-05:17:06.201 +0100: 192.168.201.210 CLOSE - 1007zX 1
2011/06/25-05:18:08.201 +0100: 192.168.201.210 OPEN - 00002C 0
2011/06/25-05:18:08.201 +0100: 192.168.201.210 LOGIN_OK maggie 00002C 0
2011/06/25-05:21:08.201 +0100: 192.168.201.121 LOST - 0004dG 4
2011/06/25-05:21:08.201 +0100: 192.168.201.121 OPEN - 0003L- 0
2011/06/25-05:21:09.201 +0100: 192.168.201.121 RECONNECT_OK livedemos
0003L- 0 0004dG
2011/06/25-05:21:09.201 +0100: 192.168.201.121 CLOSE - 0004dG 64
2011/06/25-05:28:14.201 +0100: 192.168.201.210 LOGOUT_OK maggie 00002C 0
2011/06/25-05:28:14.201 +0100: 192.168.201.210 CLOSE - 00002C 1
2011/06/25-05:29:15.201 +0100: 192.168.201.210 OPEN - 1003gD 0
2011/06/25-05:29:15.201 +0100: 192.168.201.210 LOGIN_OK maggie 1003gD 0
2011/06/25-05:39:21.201 +0100: 192.168.201.210 LOGOUT_OK maggie 1003gD 0
2011/06/25-05:39:21.201 +0100: 192.168.201.210 CLOSE - 1003gD 1
2011/06/25-05:40:22.201 +0100: 192.168.201.210 OPEN - 1007-- 0
2011/06/25-05:40:22.201 +0100: 192.168.201.210 LOGIN_OK maggie 1007-- 0
2011/06/25-05:50:28.201 +0100: 192.168.201.210 LOGOUT_OK maggie 1007-- 0
```

13.2 Request log

The request log shows the raw RTTP messages sent by each client. This is before the message is parsed so could contain anything in that field.

Request log format

All request log essages have the same format:

TIMESTAMP IP-ADDRESS USERNAME SESSION-ID MESSAGE
Example:

```
2011/06/26-15:04:42.201 +0100: 192.168.201.16 - 2ADgSL "2ADgSL LOGIN 000000
CLIENT/CLEAR RTTP/2.0 demouser demopass"
2011/06/26-15:04:46.201 +0100: 192.168.201.16 demouser 2ADgSL "2ADgSL LOGOUT"
2011/06/25-04:11:27.201 +0100: 192.168.201.210 maggie 0004p_ "0004p_ REQUEST /
EQUITIES/MSFT"
2011/06/25-04:11:27.201 +0100: 192.168.201.210 maggie 0004p_ "0004p_ REQUEST /
FX/GBP"
2011/06/25-04:11:28.201 +0100: 192.168.201.210 maggie 0004p_ "0004p_ REQUEST /
IPE/IPE/HB"
2011/06/25-04:14:17.201 +0100: 192.168.201.104 - 0006IW "0006IW NOOP"
2011/06/25-04:14:17.201 +0100: 192.168.201.104 - 0006IW "0006IW LOGIN 0003U-
CLIENT/CLEAR RTTP/0.2 bobby11 mypassw11"
2011/06/25-04:21:08.201 +0100: 192.168.201.121 - 0004dG "0004dG NOOP"
2011/06/25-04:21:08.201 +0100: 192.168.201.121 - 0004dG "0004dG LOGIN 00077o
CLIENT/CLEAR RTTP/0.2 bobby11 mypassw11"
2011/06/25-04:21:32.201 +0100: 192.168.201.210 maggie 0004p_ "0004p_ LOGOUT "
2011/06/25-04:22:34.201 +0100: 192.168.201.210 - 1002i0 "1002i0 LOGIN 000000
CLIENT/CLEAR RTTP/0.2 maggie thatcher"
2011/06/25-04:22:34.201 +0100: 192.168.201.210 maggie 1002i0 "1002i0 REQUEST /
EQUITIES/MSFT"
2011/06/25-04:22:34.201 +0100: 192.168.201.210 maggie 1002i0 "1002i0 REQUEST /
FX/GBP"
2011/06/25-04:22:34.201 +0100: 192.168.201.210 maggie 1002i0 "1002i0 REQUEST /
IPE/IPE/HB"
2011/06/25-04:32:39.201 +0100: 192.168.201.210 maggie 1002i0 "1002i0 LOGOUT "
```

13.3 Object log

The object log shows which objects are successfully requested and discarded by each RTTP client. This is after processing client requests and one line per object instead of the unprocessed request log.

Object log format

All object log messages have the same format:

TIMESTAMP SESSION-ID TYPE OBJECT
Where TYPE is either REQUEST, DISCARD, or MAP.

Example:

```
2009/09/07-11:57:33.810 +0100:9bjLYd MAP /HBT/snpsrc-frac
(/snpsrc-frac)
2011/06/25-08:27:06.201 +0100: 0000Fw REQUEST /HBT/snpsrc-frac
(/snpsrc-frac)
2009/06/25-08:27:07.201 +0100: 0000Fw MAP /HBT/snpsrc-deci
(/snpsrc-deci)
2009/06/25-08:27:07.201 +0100: 0000Fw REQUEST /HBT/snpsrc-deci
(/snpsrc-deci)
2009/06/25-08:37:10.201 +0100: 0000Fw DISCARD /HBT/snpsrc-frac
(/snpsrc-frac)
```

Log entries of type MAP show how user object names are mapped to Liberator object names. For example:

```
MAP /HBT/snpsrc-frac (/snpsrc-frac)
```

This log entry shows that the user object name `/snpsrc-frac`, as shown in parentheses, has been mapped to the Liberator object `/HBT/snpsrc-frac`.

Log entries of type REQUEST show user requests for object subscriptions, and entries of type DISCARD show discard requests. In each case the object name supplied by the user is shown in parentheses. For example:

```
REQUEST /HBT/snpsrc-frac (/snpsrc-frac)
```

This log entry shows that the user requested `/snpsrc-frac`, which refers in the Liberator to the (previously mapped) object `/HBT/snpsrc-frac`

13.4 Packet log

The packet log shows all messages sent between Liberator and its data sources.

Packet log messages

Table 13-2 lists the possible messages that can be written to the packet log.

Message	Description	EXTRA arguments
PEERINFO	These messages are sent and received when two DataSource applications connect. A PEERINFO can also be sent at other times indicating the status of a DataSource peer.	DATASRC-NAME [MSG-ID] [MSG-STR]
DATAUPDATE	This is the most common type of message in a packet log. These messages are actual data being sent from a DataSource peer to a Liberator.	SEQUENCE-NUMBER FLAGS OBJECT-NAME NUM-FIELDS [FIELD-DATA]
SUBJREQ	This message shows when a request for objects is made to a DataSource peer.	NUM-OBJECTS [OBJECT-NAMES]
SUBJDSC	This message shows when a discard message is sent to a DataSource peer, informing the peer that the Liberator no longer wants to receive updates for that object.	Identical format to SUBJREQ.
DOWN	This shows when a DataSource connection goes down. More detailed information on connections can be found in the event log.	

NODATA	This message is sent when a DataSource peer does not have the object being requested or wishes to inform the Liberator at any point of a change in this condition.	OBJECT-NAME FLAGS
STATUS	This shows object status messages from a DataSource peer.	OBJECT-NAME FLAGS MSG-ID MSG-STR

Table 13-2: Packet log messages

Table 13-3 lists the possible value for the FLAGS field when used in a NODATA message.

EXTRA argument	Description
PEERINFO MSG-ID MSG-STR	Status conditions.
DATAUPDATE FIELD-DATA	A space separated list of field/value pairs. These are given as field numbers as that is what is sent on the Datasource protocol, these would have to be matched up with the Liberator's fields.conf to find the names.
SUBJREQ SUBJDSC OBJECT-NAMES	A space-separated list of object names.

NODATA FLAGS	1 NOT FOUNDThe object does not exist 2 READ DENIEDAccess to this object is denied 4 DELETE Delete the object 5 UNAVAILABLEObject may exist but is not available at the moment
STATUS OBJECT-NAME FLAGS	The object in question. 0x0000 Status Info 0x0001 Object is Stale (won't receive updates) 0x0002 Object is Stale and should be removed from any watch lists 0x0004 Object is not stale. 0x0100 Wait for update. When used with Stale indicates that a data update will clear the Stale status. When used with Not Stale indicates that the object is only not stale once a data update is received. 0x1101 Failover. Object is stale and Liberator should failover to another DataSource peer if possible. MSG-ID MSG-STRUser definable

Table 13-3: NODATA message flags

Packet log format

All packet log messages have the same format:

TIMESTAMP IP-ADDRESS DIRECTION TYPE PEER-ID [EXTRA]

The DIRECTION field is either "<" or ">". "<" means a message is received, and ">" is a message sent. With sent messages the PEER-ID is the ID of the DataSource peer the message is being sent to, and with received messages it is the ID of the DataSource peer the message is from.

Packet log examples

PEERINFO messages:

```
2011/06/26-15:04:03.201 +0100: 127.0.0.1 < PEERINFO 1 demosrc-
bigsun 0
2011/06/26-15:04:03.201 +0100: 127.0.0.1 > PEERINFO 0 rttpd-bigsun
0
2011/06/26-15:37:36.201 +0100: 127.0.0.1 < PEERINFO 3 testsrc-
mtserv1 6 Warning
```

DATAUPDATE messages:

```
2011/06/26-15:04:03.201 +0100: 127.0.0.1 < DATAUPDATE 1 1 48 /DEMO/
AAPL 8 10003=Apple 10436=21.332 10441=22.203 10006=21.767
10005=09:04 10032=2000000 10011=0.887 10005=09:04
2011/06/26-15:04:03.201 +0100: 127.0.0.1 < DATAUPDATE 1 2 48 /DEMO/
AMZN 8 10003=Amazon 10436=8.165 10441=8.499 10006=8.332 10005=09:04
10032=1700000 10011=-0.388 10005=09:04
2011/06/26-15:04:03.201 +0100: 127.0.0.1 < DATAUPDATE 1 3 48 /DEMO/
CSCO 8 10003=Cisco 10436=13.932 10441=14.501 10006=14.217
10005=09:04 10032=45700000 10011=0.347 10005=09:04
```

SUBJREQ messages:

```
2011/06/26-15:22:37.201 +0100: 127.0.0.1 > SUBJREQ 3 2 /I/VOD.L /I/
ANL.L
```

SUBJDSC messages:

```
2011/06/26-15:23:45.201 +0100: 127.0.0.1 > SUBJDSC 3 2 /I/VOD.L /I/
ANL.L
```

DOWN messages:

```
2011/06/26-15:24:09.201 +0100: 127.0.0.1 < DOWN 3
```

NODATA messages:

```
2011/06/26-15:28:53.201 +0100: 127.0.0.1 < NODATA 3 /I/VOD.L 1
2011/06/26-15:28:53.201 +0100: 127.0.0.1 < NODATA 3 /I/ANL.L 1
```

STATUS messages:

```
2011/06/26-15:40:48.201 +0100: 127.0.0.1 < STATUS /I/VOD.L 0x0001 8
Data may be stale
2011/06/26-15:40:53.201 +0100: 127.0.0.1 < STATUS /I/VOD.L 0x0104 6
Data may be ok now
2011/06/26-15:40:58.201 +0100: 127.0.0.1 < STATUS /I/VOD.L 0x0004 4
Data is ok now
2011/06/26-15:41:03.201 +0100: 127.0.0.1 < STATUS /I/VOD.L 0x0000 9
Everything is fine
2011/06/26-15:41:20.201 +0100: 127.0.0.1 < STATUS /I/VOD.L 0x1101 3
Try somewhere else
```

13.5 HTTP access log

This logs all HTTP requests made to the Liberator. This is similar to most web servers log files.

HTTP access log format

The format is as follows:

TIMESTAMP IP-ADDRESS REQUEST HTTP-RESPONSE-CODE RESPONSE-SIZE-IN-BYTES PORT-NUMBER

Example:

```
192.168.201.16 - - [26/Jul/2011:15:04:34 +0100] "GET /demos/rtml/
rtml.html HTTP/1.1" 200 2192
192.168.201.16 - - [26/Jul/2011:15:04:34 +0100] "GET /demos/rtml/
common.css HTTP/1.1" 200 522
192.168.201.16 - - [26/Jul/2011:15:04:34 +0100] "GET /rtml/ HTTP/
1.1" 200 9570
192.168.201.16 - - [26/Jul/2011:15:04:35 +0100] "GET /rtml/lib/
formatting.js HTTP/1.1" 200 3769
192.168.201.16 - - [26/Jul/2011:15:04:35 +0100] "GET /rtml/lib/
stale.js HTTP/1.1" 200 1167
192.168.201.16 - - [26/Jul/2011:15:04:35 +0100] "GET /rtml/
w3clibrary.js HTTP/1.1"200 3122
```

13.6 HTTP error log

Example:

```
[26/Jun/2011:11:47:09.123 +0000] [error] [client 127.0.0.1] File
does not exist: /opt/Liberator/htdocs/notfound
```

13.7 RTTP traffic log

The RTTP traffic log records the RTTP traffic between a client and the Liberator. It is intended to be used for troubleshooting purposes. RTTP traffic logging can be enabled by configuration (see “rttp-log” on page 203 and “rttp-log-users” on page 204) or through the Caplin Management Console.

RTTP traffic log format

RTTP traffic log entries have the format:

>>>TIMESTAMP
<RTTP message as text>>

or

<<<TIMESTAMP
<RTTP message as text>>

where:

- **>>>** indicates that the RTTP message has been sent *from* the Liberator to the client
- **<<<** indicates that the RTTP message has been sent *to* the Liberator from the client
- ***TIMESTAMP*** has the format ***dd_mon hh:mm:ss.ss*** (for example 23_Aug 15:22:14.07)

Example:

```
>>> 16 Aug 23:08:05.77
a("1l RECONNECT+OK");
a("7_ 1 demosrc-devlinux1 demosrc-devlinux1+IS+UP");
a("7_ 2 demosrc2-devlinux1 demosrc2-devlinux1+IS+UP");
a("83 service1 service1+IS+OK");
a("83 demosvc demosvc+IS+OK");
z();
</script>

<script>
<<< 16 Aug 23:08:10.78
1Ay3pS NOOP
>>> 16 Aug 23:08:10.78
a("4n NOOP+OK");
z();
</script>

<script>
<<< 16 Aug 23:08:15.78
1Ay3pS NOOP
>>> 16 Aug 23:08:15.78
a("4n NOOP+OK");
z();
</script>
<script>
```

13.8 Event log

The event log is a text log file which can be viewed with normal commands. It contains information about starting up, shutting down, connections to DataSource peers, and license usage.

Note: *For a list of the event log messages related to licensing, refer to the document* ***Caplin Platform: Guide to User Licensing****.*

Example:

```
2011/06/25-13:52:17.420 +0100: CONFIG: UDP Message port not configured
2011/06/25-13:52:17.420 +0100: NOTIFY: Liberator/6.0.0 starting
2011/06/25-13:52:17.420 +0100: NOTIFY: Logging to /opt/caplin/Liberator/var
2011/06/25-13:52:17.421 +0100: NOTIFY: Licence will expire on Wed Dec 28 00:00:00 2011
2011/06/25-13:52:17.421 +0100: NOTIFY: system-max-files set to 1024
2011/06/25-13:52:17.422 +0100: INFO: Loaded auth module <openauth>
2011/06/25-13:52:17.423 +0100: INFO: Next cycle of UUPP database (/opt/caplin/Liberator/
users/uupp-rttpd.db) scheduled for Wed Aug 31 23:59:59 2011
2011/06/25-13:52:17.426 +0100: INFO: Read in 101 unique users from database
2011/06/25-13:52:17.455 +0100: INFO: Created object /(220) [0x8c37578/0]
2011/06/25-13:52:17.455 +0100: NOTIFY: Field CONTRIB_USER not known, setting unique user
fieldnumber to 20000
2011/06/25-13:52:17.459 +0100: INFO: 2 CPUs CONFIGURED
2011/06/25-13:52:17.459 +0100: INFO: 2 CPUs ONLINE
2011/06/25-13:52:17.493 +0100: INFO: Created object /SYSTEM(200) [0x9f41478/1]
2011/06/25-13:52:17.493 +0100: INFO: Created object /SYSTEM/NODE-0(200) [0x9f41628/2]
2011/06/25-13:52:17.493 +0100: INFO: Created object /SYSTEM/NODE-0/INFO(200) [0x9f417d0/3]
2011/06/25-13:52:17.493 +0100: INFO: Changing type of /SYSTEM/NODE-0/INFO from 200 to 201
[0x9f417d0/3]
2011/06/25-13:52:17.493 +0100: INFO: Created object /SYSTEM/INFO(200) [0x9f41a58/4]
2011/06/25-13:52:17.493 +0100: INFO: Changing type of /SYSTEM/INFO from 200 to 201
[0x9f41a58/4]
2011/06/25-13:52:17.493 +0100: INFO: Created object /SYSTEM/LICENSE(200) [0x9f41d10/5]
2011/06/25-13:52:17.493 +0100: INFO: Changing type of /SYSTEM/LICENSE from 200 to 201
[0x9f41d10/5]
2011/06/25-13:52:17.493 +0100: INFO: Created object /SYSTEM/NODE-0/SRC-0(200) [0x9f41fd8/6]
2011/06/25-13:52:17.493 +0100: INFO: Changing type of /SYSTEM/NODE-0/SRC-0 from 200 to 201
[0x9f41fd8/6]
2011/06/25-13:52:17.493 +0100: INFO: Created object /SYSTEM/NODE-0/SRC-1(200) [0x9f42248/7]
2011/06/25-13:52:17.493 +0100: INFO: Changing type of /SYSTEM/NODE-0/SRC-1 from 200 to 201
[0x9f42248/7]
2011/06/25-13:52:17.493 +0100: INFO: Created object /SYSTEM/NODE-0/SRC-2(200) [0x9f425a8/8]
2011/06/25-13:52:17.493 +0100: INFO: Changing type of /SYSTEM/NODE-0/SRC-2 from 200 to 201
[0x9f425a8/8]
2011/06/25-13:52:17.493 +0100: INFO: Created object /SYSTEM/NODE-0/SRC-3(200) [0x9f42890/9]
2011/06/25-13:52:17.493 +0100: INFO: Changing type of /SYSTEM/NODE-0/SRC-3 from 200 to 201
[0x9f42890/9]
2011/06/25-13:52:17.494 +0100: INFO: Created object /SYSTEM/NODE-0/SRC-4(200) [0x9f42ba0/
10]
2011/06/25-13:52:17.494 +0100: INFO: Changing type of /SYSTEM/NODE-0/SRC-4 from 200 to 201
[0x9f42ba0/10]
2011/06/25-13:52:17.494 +0100: INFO: Created object /SYSTEM/NODE-0/SERVICE(200) [0x9f42eb0/
11]
2011/06/25-13:52:17.494 +0100: INFO: Created object /SYSTEM/NODE-0/SERVICE/svc1(200)
[0x9f43048/12]
2011/06/25-13:52:17.494 +0100: INFO: Changing type of /SYSTEM/NODE-0/SERVICE/svc1 from 200
to 201 [0x9f43048/12]
2011/06/25-13:52:17.494 +0100: INFO: Created object /MT1(200) [0x9f43428/13]
2011/06/25-13:52:17.494 +0100: INFO: Changing type of /MT1 from 200 to 222 [0x9f43428/13]
2011/06/25-13:52:23.675 +0100: INFO: Accepted connection from 127.0.0.1 42371
2011/06/25-13:52:23.676 +0100: NOTIFY: Accepting peer id 1 on 127.0.0.1 42371
```

```
2011/06/25-13:52:33.642 +0100: INFO: Created object /SYSTEM/USERS(200) [0x9f45b80/14]
2011/06/25-13:52:33.642 +0100: INFO: Created object /SYSTEM/USERS/demouser-0(200)
[0x9f45ce8/15]
2011/06/25-13:52:33.642 +0100: INFO: Changing type of /SYSTEM/USERS/demouser-0 from 200 to
202 [0x9f45ce8/15]
2011/06/25-13:52:37.114 +0100: INFO: Removed object /SYSTEM/USERS/demouser-0(202)
[0x9f45ce8/15]
2011/06/25-13:52:37.114 +0100: INFO: Adding to batch-delete timer for /SYSTEM/USERS/
demouser-0(202) [0x9f45ce8/15]
2011/06/25-13:52:39.619 +0100: NOTIFY: Lost connection to peer 1 on 127.0.0.1 42371
2011/06/25-13:52:42.616 +0100: INFO: Deleted object /SYSTEM/USERS/demouser-0(202)
[0x9f45ce8/15]
2011/06/25-13:52:42.863 +0100: NOTIFY: Received signal SIGINT (2)
2011/06/25-13:52:42.864 +0100: NOTIFY: Shutting down - SIGNAL (6)
```

14 Appendix C: Javaauth configuration

Follow the steps below to configure the javaauth module. The example given configures the included *examples.OpenAuthenticator* module.

- Ensure java authentication has been specified in the Liberator License (see example license.conf below). Please contact Caplin Systems Ltd if the module is not present.

```
start-license
      signature   XXXXXXXXXXXXXXXXXXX
      company     Caplin Systems
      hostname    hostname1
      max-users   500
      expire      2011060330
      https       1
      module      cfgauth auth
      module      openauth auth
      module      xmlauth auth
      module      javaauth auth
end-license
```

- Ensure there is an Oracle JVM version 1.4 or higher installed. The **jvm-location** configuration option in *java.conf* should point to the installed location of the libjvm.so library, for example, */usr/local/jdk/jre/lib/sparc/server/libjvm.so.*

- Ensure the Liberator is not running.

- Create or edit the configuration file *javaauth.conf* in the etc directory. It must contain the option **javaauth-classid** that refers to the class-id of the Java Auth module to be loaded. The Java Auth debug level is also set here. For example:

```
javaauth-classid  authenticator
debug-level       DEBUG
```

- Edit the configuration file *java.conf* within the etc directory. The auth-module option should be set to javaauth, the jvm-location should point to the installed JVM and the **jvm-global-classpath** option should point to *javaauth.jar* within the *lib.java* directory.

  To configure the specific Java Authenticator class to load, create an add-javaclass section and insert the classpath and class-name details for the authentication module, with a class-id which matches the class-id of the javaauth module to be loaded. For example:

```
auth-module           javaauth
jvm-location          /usr/local/jdk/jre/lib/i386/server/libjvm.so
jvm-global-classpath  %r/lib/java/javaauth.jar

add-javaclass
    class-name        examples.OpenAuthenticator
    class-id          authenticator
    classpath         %r/lib/java/javaauth-examples.jar
end-javaclass
```

Please see “Java.conf configuration” on page 252 for details on the above parameters.

- Start the Liberator.

A

B

C

D

E

F

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

W

CAPLIN

THE WEB TRADING TECHNOLOGY COMPANY

Contact Us

Caplin Systems Ltd
Cutlers Court
115 Houndsditch
London EC3A 7BR
Telephone:+44 20 7826 9600
www.caplin.com