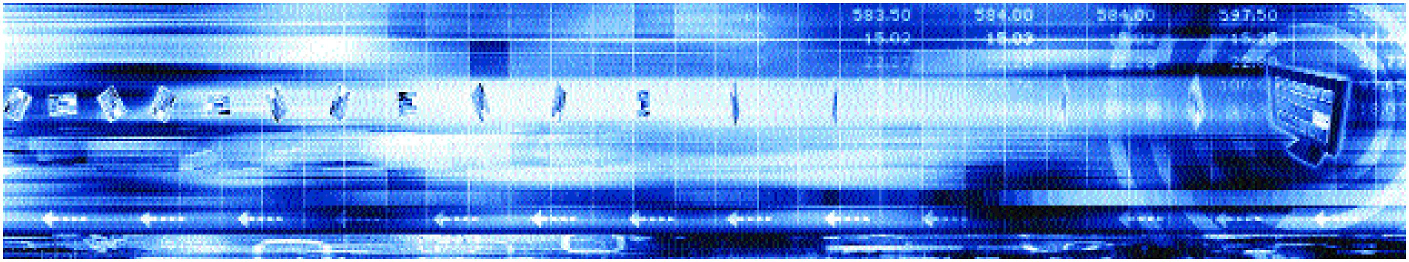


CAPLIN LIBERATOR 4.2

Administration Guide

June 2006

Preface 15

What this document contains 15

Who should read this document 15

Typographical conventions 15

Acronyms and glossary 16

Feedback 16

Acknowledgments 16

Overview 17

What is the Liberator? 17

Caplin’s Platform architecture 18

What’s new in version 4.2? 20

Architectural examples 21

Example 1—internal network 21

Example 2—Internet 21

Functions and features of the Liberator 22

Operational features 22

Permissioning and security features 24

The Liberator web site 25

Restricting data 25

Liberator’s data sources 25

Data sources 25

Data source features 26

Data features 27

Getting started 30

Installing Liberator 30

Introduction 30

Conventions and Assumptions 30

Step-by-Step Standard Install 30

Upgrading Liberator 32

Starting Liberator. 32
Introduction *32*
Step-By-Step Start-up *33*
About your Liberator licence 38
Licence limits *38*
Full secure set up on Linux and Solaris 38
Running multiple Liberators from the same install location. 40
Clustering and intelligent source routing 41
Intelligent Source Routing *42*
Step-by-step examples 42
Basic active request. *44*
Two clients actively request same data *47*
Active request with DataSource failover/load balancing *49*
Active requests for data from 2 sources *53*
Passive source-broadcast data *56*
Liberator failover *58*
Liberator and DataSource failover *60*
Requesting news headlines. *63*
Requesting news stories *65*
Requesting historic news headlines *67*
Throttling updates *70*
Authentication and authorisation of users using Auth Modules *72*

About the data 77

What is RTTP? 77
Key features of RTTP 77
Smart tunnelling. *77*
Persistent Virtual Connection (PVC) *78*
Data Status *78*
About RTTP objects 78
Directory *79*
Page *79*
Record *79*

News headline and news story *79*
Chat objects *79*
Container *79*
Auto Subscription Directory *80*
Symbols and parameters *80*

About RTTP fields 80
Type 1 data *81*
Type 2 data *81*
Type 3 data *82*

Communicating with clients 84

Enabling clients to connect using RTTP (over HTTP). 84
Specifying the Liberator URL *84*
Configuring the HTTP Keep Alive feature *84*
Using cookies to aid HTTP connection *84*

Enabling clients to connect using HTTPS 85
Making an HTTPS connection *85*
Virtual hosting *85*
Configuring the HTTPS connection *86*
Applying the security policy *87*
Sample certificates and certificate authorities *88*
Configuring hardware devices *88*

Enabling clients to connect using RTTP (direct connection). 89

Configuring objects 90
Purging objects *91*
Sending only changed fields *93*

Identifying the fields clients can request 95
Setting the number of decimal places *95*
Setting the record data to Type 2 . . . . *96*
Setting the record data to Type 3 . . . . *97*

Handling requests for news headlines 97
Identifying news codes users can search for *98*

Adjusting the update rate 98
Using throttling. *98*
Configuring "bursts" *101*
Configuring buffering *101*
Returning news to clients. *102*
Configuring write failure actions 103

Authentication and entitlement 104

Overview 104
Using auth modules 104
Specifying the Auth Module to use *104*
Configuring user numbers *106*
Waiting times for authentication. *106*
Reconnecting. *107*
Liberator's standard auth modules 107
XMLauth *107*
openauth *108*
cfgauth. *108*
javaauth *109*
Signature authentication 109

Communicating with sources of data 111

What is a DataSource peer?. 111
Configuring Liberator to be a DataSource peer 112
Connecting to DataSource peers 113
Enabling failover *113*
Reconnecting peers using the UDP interface 115
peer-reconnect. *116*
Data services 116
Specifying the object or objects. *117*
Specifying a single DataSource peer. *118*

Specifying alternative DataSource peers. 118

Specifying multiple datasource peers 119

Specifying priority or failover 120

More complex mappings 121

Waiting for responses 121

Waiting for requests. 122

Replaying data from peers into Liberator 122

Replaying news headlines. 124

Making SSL connections with DataSources 124

Sample certificates and certificate authorities 125

Monitoring performance 127

Monitoring and management subsystem 127

Log files. 127

Log file configuration 127

Log file cycling 129

System log files (syslog) 131

Logging crash details. 131

The logcat utility 132

Examples. 132

Status web page 134

Data source information. 135

Object Browser 136

Debugging. 137

debug. 138

Latency Measurement. 138

Latency Measurements 138

End to End Latency 141

Optimising efficiency 142

Improving performance using bursts. 142

Improving performance using threads. 142
Improving performance using hashtables 143
Improving performance using TCP nodelay. 144
Improving performance using selected fields 144
Reducing message sizes using fields.conf 144
Improving security measures 145

Running Liberator with many users. 146

Configuring Liberator for a high number of users 146
Changing file descriptor limits—Linux 146
Per process 147
System-wide 147
Configuring the range of ports 147
Changing file descriptor limits—Solaris. 148

Liberator demonstrations 149

Starting the demo feed—Linux and Solaris. 149
Using an SSL connection for the demo feed. 150
Viewing the examples on the website 150
Using SSL with the demonstration feed 151
Configuring the demonstration SSL connection 151

Appendix A: Configuration reference 153

Main 154
application-root 154
application-name 154
event-log 154
system-max-files 154
runtime-user. 154
catch-crash 154

include-file 155
pid-filename 155
license-file 155
syslog-facility 155

Logging 156
log-dir 156
log-maxsize 156
log-max-history 156
log-cycle-time 156
log-cycle-period 156
log-cycle-suffix 157
log-cycle-offset 157
debug-level 157

Advanced log file settings 159
add-log 159

HTTP 162
http-wwwroot 162
http-interface 162
http-port 162
http-keepalive-max 162
http-keepalive-timeout 163
http-refuse-time 163
http-server-name 163
http-indexfile 163
http-rttp-content-type 163
http-def-content-type 163
http-err-content-type 164
http-idx-content-type 164
http-access-log 164
http-error-log 164
add-authdir 164
direct-max-line-length 166
http-max-request-length 166
http-max-header-line-length 166

http-max-header-lines *166*
http-max-body-length *166*
http-connection-cookie-enable *166*
http-connection-cookie-expires *167*

RTTP 168
rttp-type5-js *168*
rttp-type5-pad-length *168*
rttp-type3-timeout *168*
rttp-hostname *168*

Enable HTTPS 169
https-enable *169*
https-interface *169*
https-port *169*
https-certificate *169*
https-privatekey *169*
https-passwordfile *170*
https-cipher-list *170*
ssl-random-seed *170*
ssl-engine-id *171*
ssl-engine-flags *172*
add-virtual-host *172*

Direct connections 174
direct-interface *174*
direct-port *174*
direct-refuse-time *174*

Objects 175
object-throttle-times *175*
object-throttle-default-level *175*
object-throttle-off *175*
active-discard-timeout *175*
record-max-cache *175*
add-object *175*
default-type *178*
add-type-mapping *178*

object-map *179*
object-precache-enable *179*
record-type1-clear-on-failover *179*
record-type2-clear-on-failover *179*
record-type3-clear-on-failover *179*
record-type2-hash-size *180*
Auth modules 181
auth-moddir *181*
auth-module *181*
max-user-warn *181*
max-user-ok *181*
max-user-limit *181*
auth-login-timeout *182*
auth-map-timeout *182*
write-users-period *182*
write-users-time *182*
Sessions 183
session-log *183*
request-log *183*
object-log *183*
noauth-reconnect *183*
session-timeout *183*
session-reconnect-timeout *184*
session-heartbeat *184*
Clustering 185
cluster-index *185*
cluster-cache-request-objects *185*
cluster-cache-source-routing *185*
cluster-addr *185*
cluster-port *185*
type1-host *185*
type1-port *186*
type2-url *186*
cluster-cache-source-routing *186*

priority 186

Fields. 187

fields-file. 187

add-field 187

requested-fields-only 188

numeric-locale 188

DataSource peers 189

datasrc-name. 189

datasrc-id 189

datasrc-reject-new-peers. 189

datasrc-pkt-log. 189

datasrc-interface 189

datasrc-port 190

datasrc-sslport 190

datasrc-default-obj-hash-size 190

datasrc-rerequest-timeout 190

add-peer 190

start-ssl 195

ssl-passwordfile 198

Data replay 199

datasrc-auto-replay 199

datasrc-auto-replay-days. 199

datasrc-auto-replay-files 199

Data services 200

New in version 4.0 200

The service name 200

The subject patterns 200

The source groups. 200

Priorities. 200

Timeouts 201

service-request-timeout 201

source-request-timeout 201

cleanup-stale-timeout 201

. . . . 201

datasrc-default-obj-hash-size *201*
add-data-service *202*
service-name *202*
request-timeout *202*
exclude-pattern *202*
include-pattern *203*
add-source-group *203*
required *203*
retry-time *203*
add-priority *203*
label *203*
Default behaviour. *205*
Conversion. *205*
latency-chain-enable *208*
latency-chain-name *208*
latency-chain-init-ts-field *208*
latency-chain-list-event-field *208*
latency-chain-list-ts-field *208*
latency-chain-base64-mode *208*

Tuning 210
object-hash-size. *210*
user-hash-size *210*
session-hash-size *210*
session-max-queue-length *210*
session-max-queue-count *210*
burst-min *211*
burst-max. *211*
burst-increment *211*
buf-cache-size *211*
buf-elem-len. *211*
output-queue-size *211*
threads-num. *212*
add-thread *212*
direct-tcp-nodelay-off. *212*

http-tcp-nodelay-off *213*
datasrc-tcp-nodelay-off *213*
batch-discard-time *213*
object-delete-batchtime *213*
object-delete-time *213*
News 214
newsitems-saved *214*
newsitems-max *214*
newscode-max-length *214*
newscode-exceptions *214*
add-newscodes *214*
newscode-hash-size *215*
news-purge-time *215*
news-purge-days *215*
news-datetime-format *215*
newscodes-valid-chars *215*
news-log *216*
news-replay *216*
news-replay-days. *216*
news-replay-files *216*
newsitems-hash-size *216*
KeyMaster 217
signature-validtime. *217*
signature-hashsize. *217*
add-sigkey *217*
UDP interface 219
udp-port *219*
udp-interface *219*
Openauth.conf configuration 220
read-access *220*
write-access. *220*
Cfgauth.conf configuration 221
add-user. *221*
encrypted-passwords. *223*

Licencing 224
UUPP *224*

Java Configuration 224
java-file *224*

Java.conf configuration 224
jvm-location *225*
jvm-global-classpath *225*
add-javaclass *225*
jvm-options *226*

Monitoring configuration 226
monitor-plugin *226*
add-monuser *226*
log-monitor-level *227*
monitor-moddir *227*
session-monitoring-interval *228*
object-monitoring-interval *228*

Javaauth configuration 228
debug-level *229*
javaauth-classid *229*

Container configuration 229
container-request-timeout *229*

Appendix B: Log file messages and formats . . . 230

Session log 230
Session log messages *230*
Session log format *231*

Request log 232
Request log format *232*

Object log 233
Object log format *233*

Packet log 234
Packet log messages *234*

Packet log format *236*
Packet log examples *236*

HTTP access log 238
HTTP access log format *238*

HTTP error log 239

Event log 239

Appendix C: Debug Levels and Messages 241

Appendix D: Javaauth configuration 242

Appendix E: Performance benchmark 244

Benchmark test methodology 244
Hardware *244*

Message content 245

Results 245

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 authorising 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 Solaris and Linux procedures.

1.3 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

1.4 Acronyms and glossary

Please refer to the glossary available on Caplin's Client Portal at demo.caplin.com/clientportal for an explanation of common terms used in this guide.

1.5 Feedback

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

Please email your thoughts to documentation@caplin.com.

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*).

2 Overview

2.1 What is the Liberator?

Liberator is a complete connectivity and subscription management system for streaming market data and trade messages over intranets, extranets and the Internet.

It is capable of handling up to tens of thousands of concurrent users. It handles both HTTP and RTTP traffic (please see What is RTTP? on page 77), but 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 104 for further details.

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

Caplin's Platform architecture

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

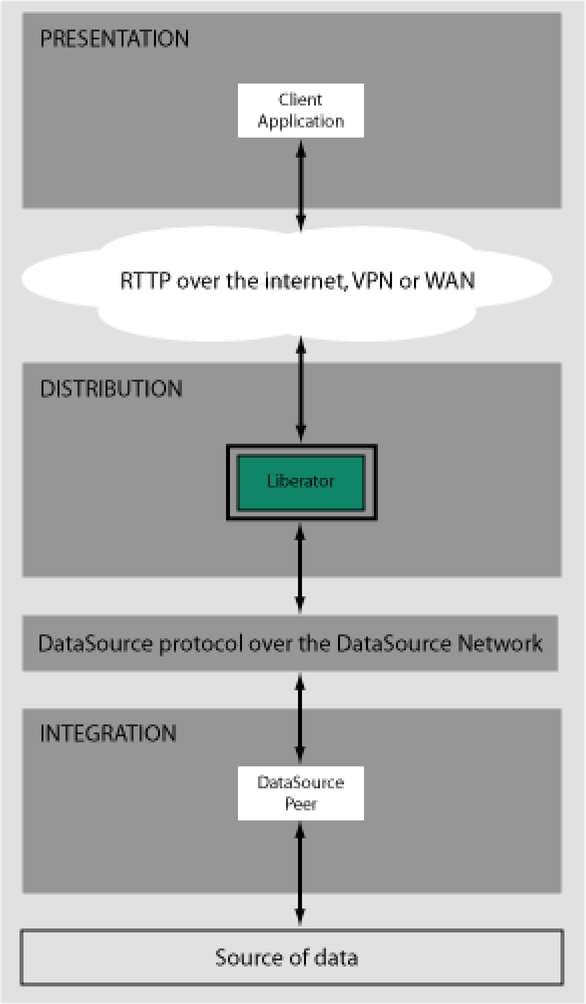


Figure 2-1: Liberator's place in Caplin's architectureInternal architecture

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 Data sources on page 25). Liberator then aggregates the data and publishes it over the Internet using RTTP, where it can be integrated into web pages or custom applications.

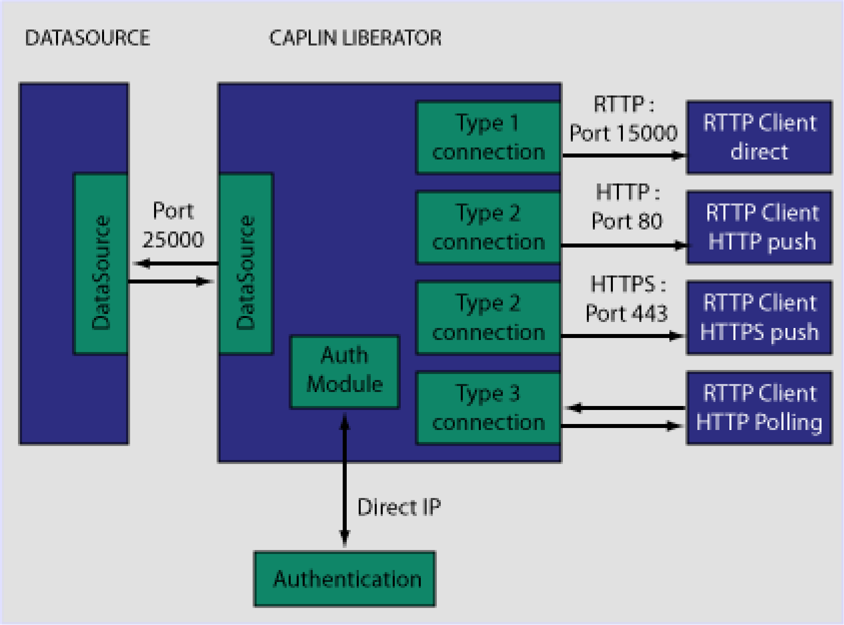


Figure 2-2: Liberator internal architecture

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.

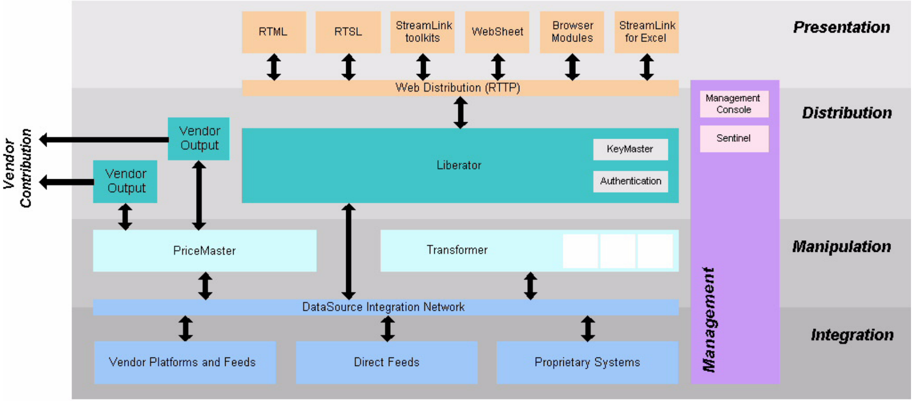


Figure 2-3: Caplin's architecture

2.2 What's new in version 4.2?

Version 4.2 of the Liberator platform incorporates several enhancements to the major new features added in 4.0:

- Monitoring - enhanced ability to view the users, objects and peers involved in the platform using either JMX or a simple socket-based protocol - please refer to the section Monitoring and management subsystem on page 127 for further details.
- Data Services - allow greater control over the prioritizing and failover of peers than was available in version 3.6 of the Liberator platform with the source mapping feature - please refer to the section Data services on page 200 for further details of how to specify this.
- Auto Directories - allow clients subscribing to a directory to automatically be subscribed to all objects within it - please refer to the section Auto Subscription Directory on page 80 for further details.
- Containers - allow clients to be automatically subscribed to an arbitrary collection of objects through references - please refer to the section Container on page 79 for further details.

- Intelligent Source Routing - allows greater efficiency when using clustered Liberators. For example, if multiple Liberators are clustered and all require an object which exists on a certain data source, then the intelligent source routing functionality will ensure that only that one data source is serving up the objects and in only one update stream to all of the Liberators. Please refer to Clustering and intelligent source routing on page 41 for further details.

- Latency Measurement, enhanced ability to monitor and measure the latency of messages across the system and through to client applications. Please refer to Latency Measurement on page 138 for further details.

2.3 Architectural examples

Below are two examples of Liberator installations. There are many possible configurations that include more intricate load balancing, fault-tolerant and firewall protected environments.

Example 1—internal network

Figure 2-4 shows a simple internal network environment for redistributing real time data. The Liberator is connected to the Triarch network via Caplin's DataSource for Triarch feed handler.

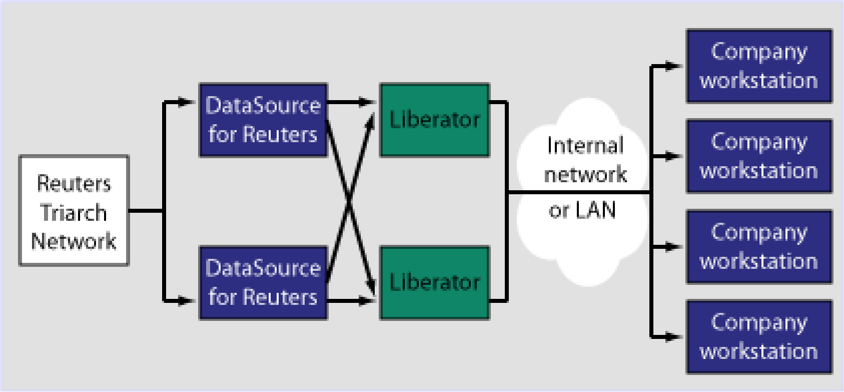


Figure 2-4: Liberator in a simple internal network

Example 2—Internet

Figure 2-5 shows a full Internet environment with two Liberators for load balancing and fault-tolerance purposes. In this case both Liberators are receiving data from a DataSource handler positioned on the other side of an internal firewall. This diagram also shows that the users are able to contribute data back to the data source via the liberator.

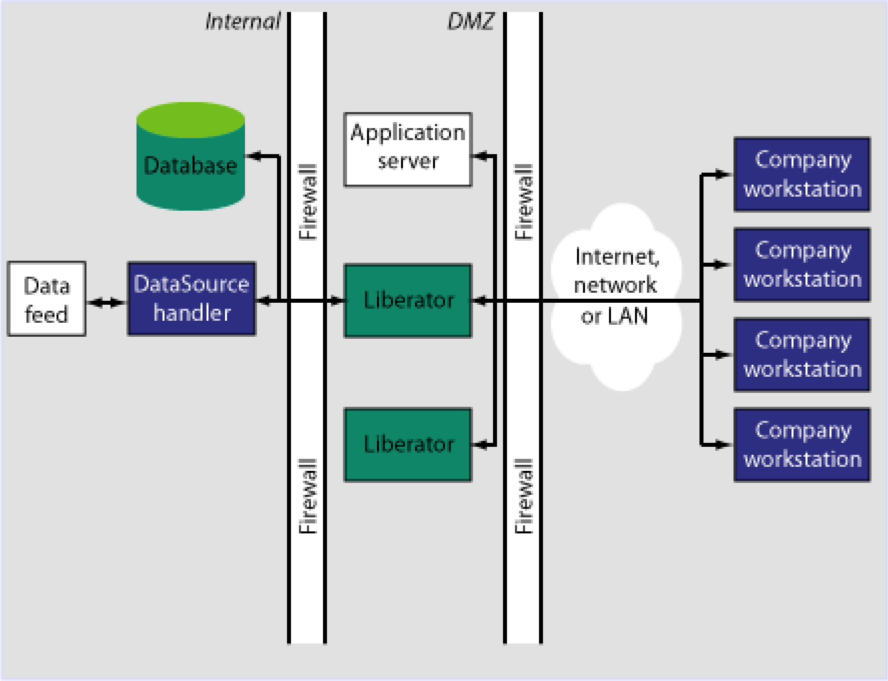


Figure 2-5: Liberator in a full Internet environment

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 77.

Clustering

Each Liberator can be configured as a node within a cluster, in order to share information about the number of licences, 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 40.

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 to request it from.

- For details on how to configure global caching, see Running multiple Liberators from the same install location on page 40.

Multi-threading

Liberator is a multi-threaded application. It uses one main thread on the DataSource side, and a configurable number of threads on the client session side. This number should ideally match the number of CPUs on a multi-processor machine.

- For details on how to configure optimal threading, see Improving performance using threads on page 142.

Note: *(Linux only) As the Liberator uses a number of threads, one Liberator can be listed as a number of processes when you view a process status list.*

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 98.

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 101.

Permissioning and security features

Authentication and permissioning modules

Liberator supports a modular system for handling authentication and authorisation. 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 104.

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 164.

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 accompanying document **KeyMaster 4.0 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, see the accompanying document **Liberator Auth Module Developer's Guide**.

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 the DataSources
- ❖ view or download documentation for the Liberator and associated components
- ❖ view demonstration applications written using the SL4B SDK

■ 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 Status web page on page 134 and Liberator demonstrations on page 149

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 data source feeding the Liberator.

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

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 98.

2.5 Liberator’s data sources

Data sources

Liberator is capable of retrieving data from any application that uses the DataSource protocol, a protocol that enables most Caplin and RTTP-related products to communicate with each other.

The DataSource API handles data from a variety of sources, such as Triarch, RMDS, Comstock and TIB, please refer to the DataSource documentation for further details.

You can write your own data source application to connect to other DataSource applications using the DataSource SDK. A DataSource-enabled application can contribute any logical record, page update or other type of data by using a straightforward API.

The DataSource SDKs available include:

- DataSource SDK for C (Solaris, Linux, Windows);
- DataSource SDK for Java

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

Please see the DataSource documentation for further details.

Data source 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. Liberator can also use SSL for HTTPS.

For details on how to configure Liberator for SSL, see Enabling clients to connect using HTTPS on page 85.

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 116.

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—a group of data sources are regarded as one data source and requests will be sent to them in a round-robin fashion.

- For details on how to implement active data services, see page 116.

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 122.

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 190.

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 on how to reset connections with UDP commands, see Reconnecting peers using the UDP interface on page 115.
- For details on how to adjust logging levels with UDP commands, see Debugging on page 137.
-

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 178.

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 on page 175.

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 175. The purging of news headlines is set using the news-purge-days on page 215 and news-purge-time on page 215.

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 97. For a description of RTTP news objects, see News headline and news story on page 79.

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 80). 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 95.

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 by Liberator and its DataSources.

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 90 and auth_map_object in the companion document **Liberator Auth Module Developer's Guide**.

News headlines

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

For details on how to configure active news headlines, see the document **DataSource for HNAS Administrator's Guide**.

3 Getting started

3.1 Installing Liberator

Introduction

The standard install procedure described below shows how to install Liberator in a flexible way to allow easy changes in the future.

Conventions and Assumptions

In the following procedure, (and in Caplin installation guides generally), 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. Requests can be made to Caplin Support (support@caplin.com).

It will also be assumed that the Liberator will be installed to */apps/caplin*. This path will be referred to as $INSTALL_DIR. All paths given below will be relative to */apps/caplin* i.e. */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-4.2.0-1-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-4.2.0-1-i686-pc-linux-gnu.tar.gz
```

   Solaris

```
$ uncompress Liberator-4.2.0-1-sparc-sun-solaris2.8.tar.Z
$ tar xf Liberator-4.2.0-1-sparc-sun-solaris2.8.tar
```

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-4.2.0-1 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 config files that should not be overwritten by an upgrade.*

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.

3.2 Starting Liberator

Introduction

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

Liberator receives data from DataSources. In this installation we are going to connect to an example DataSource 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.

   Edit this to change the variable LIBERATOR_ROOT to point to the root instance install directory (/liberator1). The extract below shows an example

```
if [ -z "$LIBERATOR_ROOT" ]; then
LIBERATOR_ROOT=/apps/caplin/liberator1
export LIBERATOR_ROOT
fi
```

2. The Liberator has a number of ports that need to be defined. Please see below for details:

   `http-port` This is the port that listens for HTTP requests. The Liberator has an inbuilt webserver which hosts pages to check status info etc.

   `udp-port` This is the port which listens for UDP messages. UDP messages can be used to issue commands to the Liberator while it is running.

   `direct-port` This is the port which listens for direct (type1) RTTP connections.

   `datasrc-port` This is the port which listens for connections from DataSource peers.

   `https-port` This is the port used to listen for HTTPS connections. This is only used if https is enabled in the Liberator configuration.

   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 ports can be modified and set according to your requirements. An example config 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
      local-type  none
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.

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:

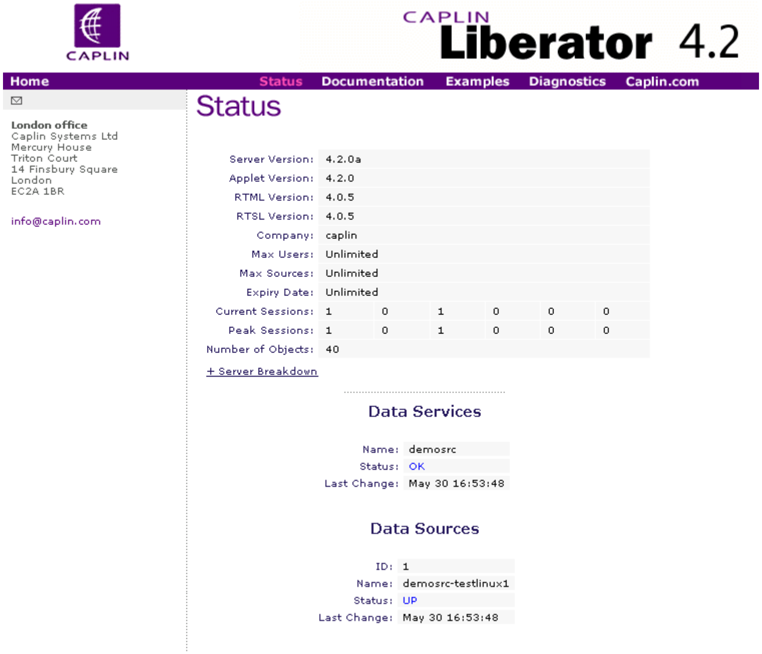
http://hostname:8080   8080 is the http-port specified in the Liberator configuration file.

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.

An example status page is shown below:



London office
Caplin Systems Ltd
Mercury House
Triton Court
14 Finsbury Square
London
EC2A 1BR

info@caplin.com

Status

Server Version:	4.2.0a					
Applet Version:	4.2.0					
RTML Version:	4.0.5					
RTSL Version:	4.0.5					
Company:	caplin					
Max Users:	Unlimited					
Max Sources:	Unlimited					
Expiry Date:	Unlimited					
Current Sessions:	1	0	1	0	0	0
Peak Sessions:	1	0	1	0	0	0
Number of Objects:	40					

+ Server Breakdown

Data Services

Name:	demosrc
Status:	OK
Last Change:	May 30 16:53:48

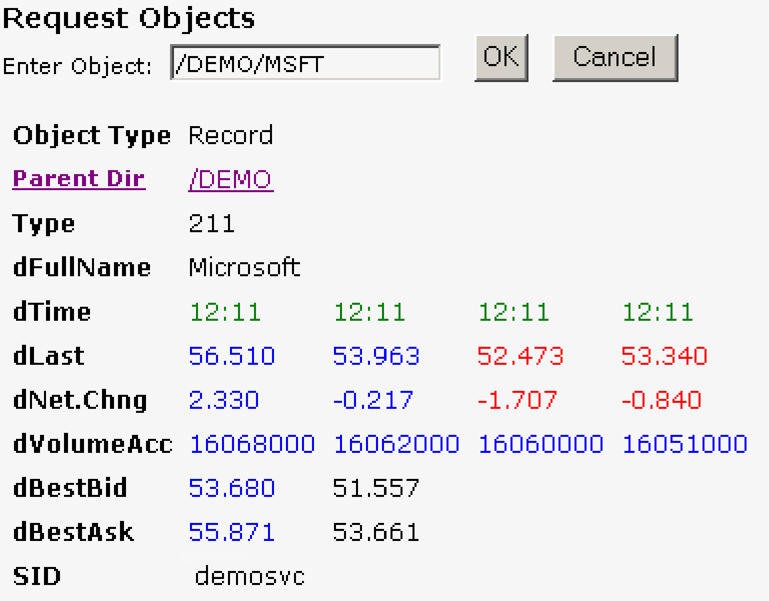
Data Sources

ID:	1
Name:	demosrc-testlinux1
Status:	UP
Last Change:	May 30 16:53:48

Figure 3-1: Example Liberator status page after initial step-by-step set up

The important part to note is the Data Sources section, which will tell us whether demosrc has status UP or DOWN. The usual cause of the status showing down is incorrect ports being specified in *demosrc.conf*.

7. The Object Browsing Tool, which ships with the Liberator, can be used to request some data from the demo data source. Browse to Examples -> Object Browsing Tool and request /DEMO/MSFT.



Request Objects

Enter Object: /DEMO/MSFT

Object Type	Record			
Parent Dir	/DEMO			
Type	211			
dFullName	Microsoft			
dTime	12:11	12:11	12:11	12:11
dLast	56.510	53.963	52.473	53.340
dNet.Chng	2.330	-0.217	-1.707	-0.840
dVolumeAcc	16068000	16062000	16060000	16051000
dBestBid	53.680	51.557		
dBestAsk	55.871	53.661		
SID	demosvc			

Figure 3-2: Object Browser Tool from the Examples page

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
```

3.3 About your Liberator licence

Liberator licence 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 licences which simplifies deployment by enabling one file to control several installations.

If you need to change, upgrade or renew your licence agreement, this file will be forwarded to you by Caplin along with the appropriate installation instructions.

Licence limits

- The default licence causes Liberator to shut down after 30 minutes of operation. Contact Caplin for a full licence to replace this.
- Liberator licences can limit users to a maximum number of unique users per month. A unique user is an individual person. Liberator can either limit the number of unique users or create an audit log for billing purposes.
- When a number of Liberators are clustered (see page 40 for details on how to configure clusters), if the connection with one is lost, for a set time the remaining Liberators will not count the disconnected Liberator's users in relation to the total licence allowance.

  After this period, the total number of permitted users will be reduced by an amount proportional to the failed Liberator's load. This proportion is calculated by assuming that the users are shared equally between Liberators.

  The licence contains another timeout setting that determines how long this reduced user count will last, whether the failed Liberator reconnects or not, before the maximum number of users are permitted again.

3.4 Full secure set up on Linux and Solaris

If you want to use port 80, 443 or any other restricted port, or if your licence 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-4.0.0.1.tar.Z | tar xf -
$ ln -s Liberator-4.0.0.1 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 *run* directory.

```
# vi etc/rttpd    [Edit the LIBERATOR_ROOT line to point to /opt/
caplin/Liberator]
```

8. Configure ports and set any other required parameters.

9. Log in as root.

10. 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.

3.5 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 seperate 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*

3.6 Clustering and intelligent source routing

A cluster enables a group of Liberators to act as one, in order to monitor licence 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 DataSource it requested the object 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 DataSource. This can have two advantages, firstly it minimises the load on the DataSources as they are not all serving up the same objects. and secondly it minimises the bandwidth used on the DataSource to Liberator network as each update is only sent by a single DataSource. This can be a significant advantage if the DataSources 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 190 and add-data-service on page 202), so each Liberator must use the same labels for the relevant DataSources.

3.7 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:

Active requests

Basic active request page 44
Two clients actively request same data page 47
Active request with DataSource failover page 60
Active requests for data from 2 sources page 53

Passive source—broadcast data page 56

Failover

Liberator failover page 58
Liberator and DataSource failover page 60

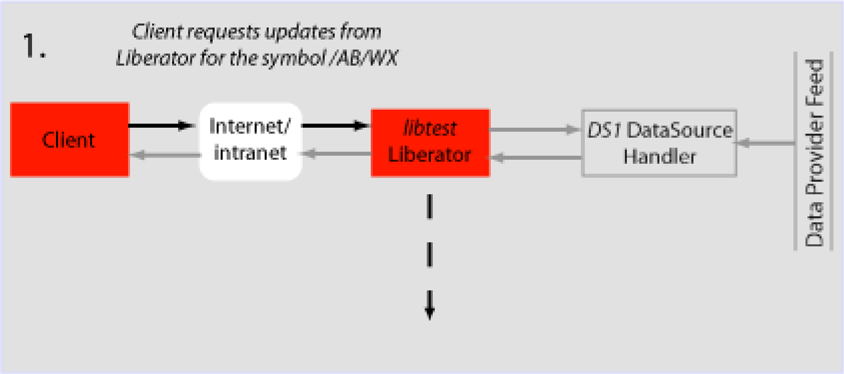
Requesting news

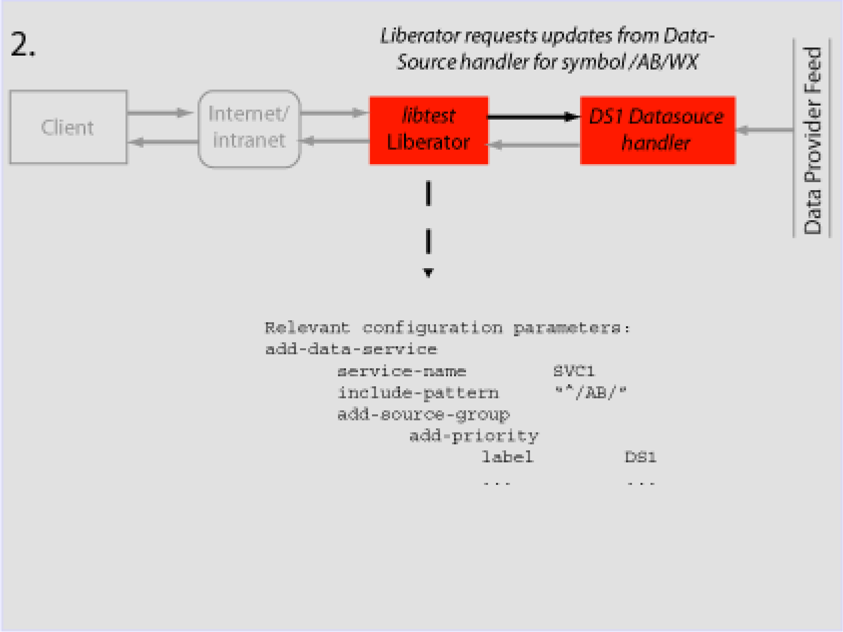
Requesting news headlines page 63
Requesting historic news headlines page 67
Requesting news stories page 65

Throttling updates page 70

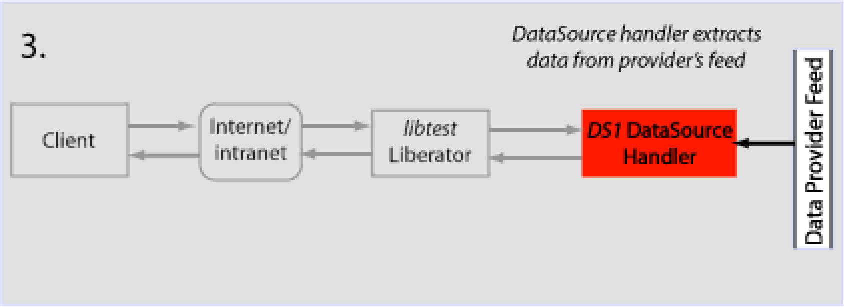
Authentication and authorisation of users using Auth Modules page 72

Basic active request

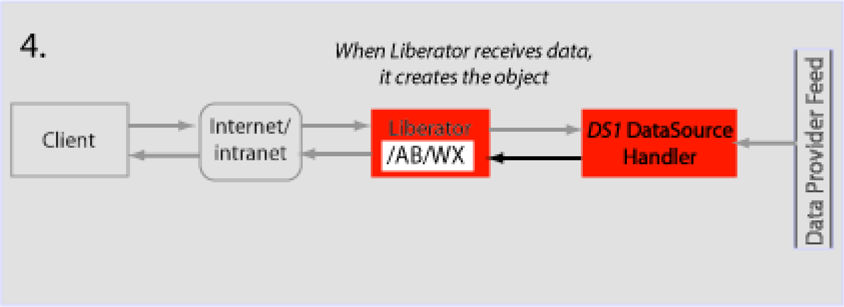




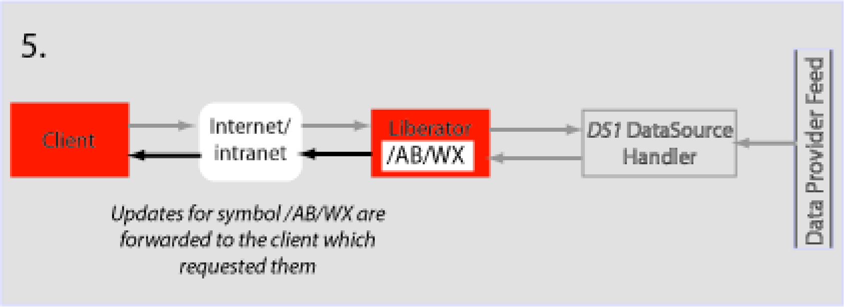
```
Relevant configuration parameters:
add-data-service
       service-name        SVC1
       include-pattern     "^/AB/"
       add-source-group
              add-priority
                    label        DS1
                    ...          ...
```



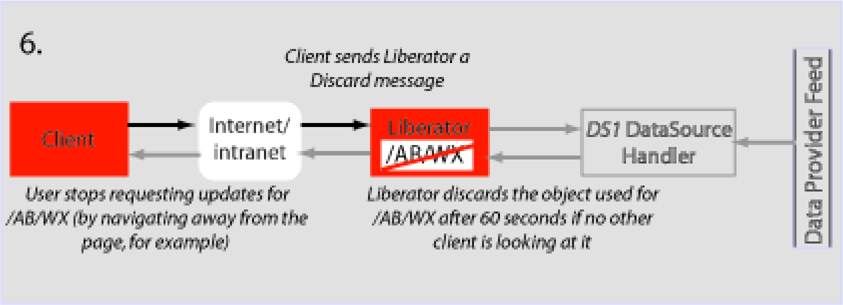
DataSource handler extracts data from provider's feed

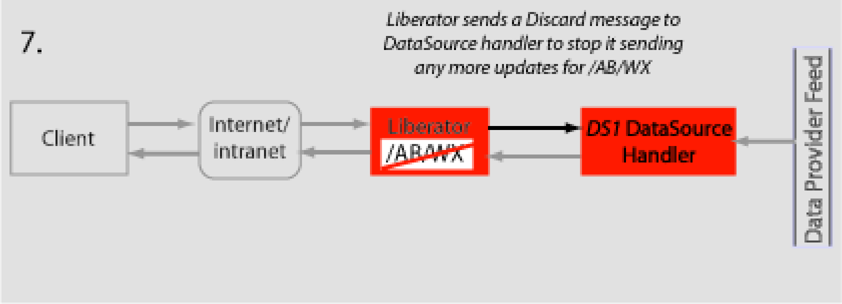


When Liberator receives data, it creates the object

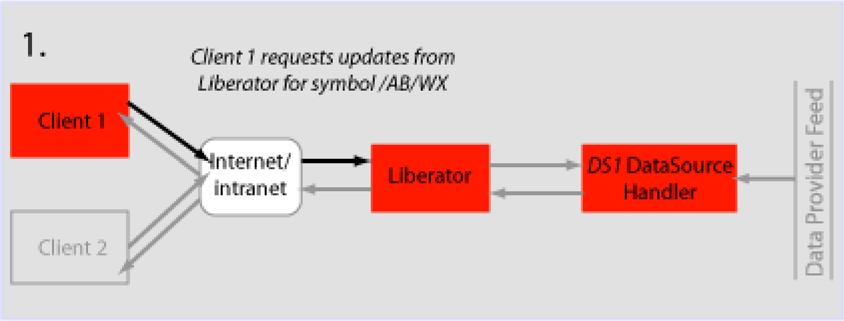


Updates for symbol /AB/WX are forwarded to the client which requested them

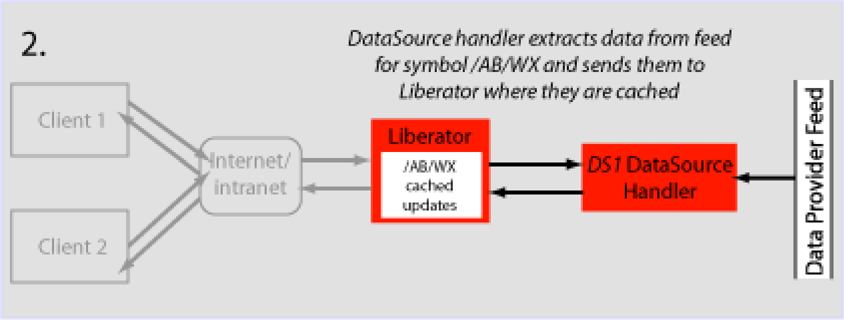

6.
Client sends Liberator a Discard message
Client
Internet/ intranet
Liberator
/AB/WX
DS1 DataSource Handler
Data Provider Feed
User stops requesting updates for /AB/WX (by navigating away from the page, for example)
Liberator discards the object used for /AB/WX after 60 seconds if no other client is looking at it


7.
Liberator sends a Discard message to DataSource handler to stop it sending any more updates for /AB/WX
Client
Internet/ intranet
Liberator
/AB/WX
DS1 DataSource Handler
Data Provider Feed

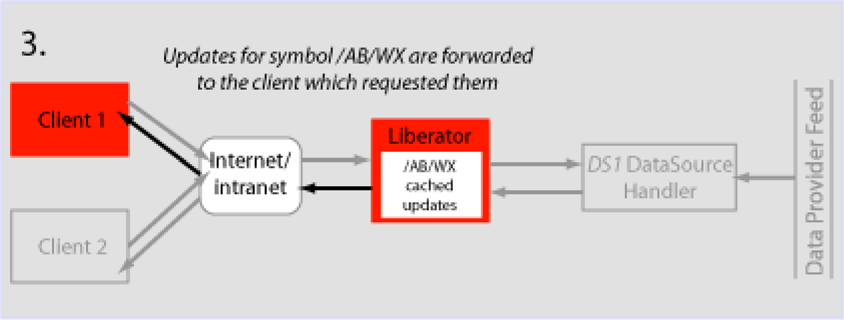
Two clients actively request same data



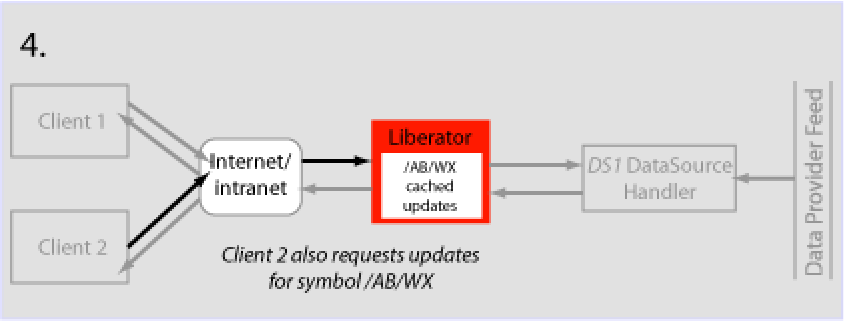
Client 1 requests updates from Liberator for symbol /AB/WX



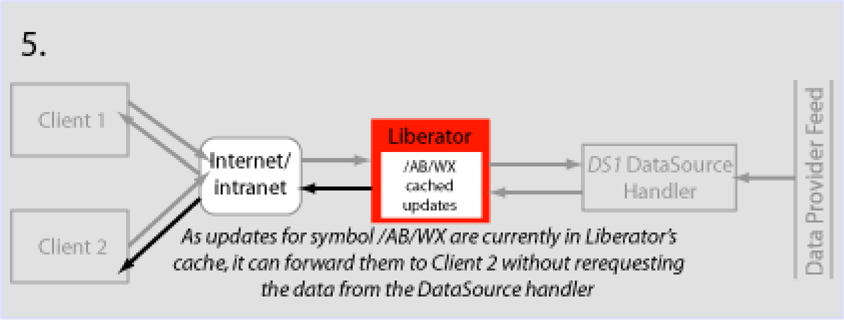
DataSource handler extracts data from feed for symbol /AB/WX and sends them to Liberator where they are cached



Updates for symbol /AB/WX are forwarded to the client which requested them

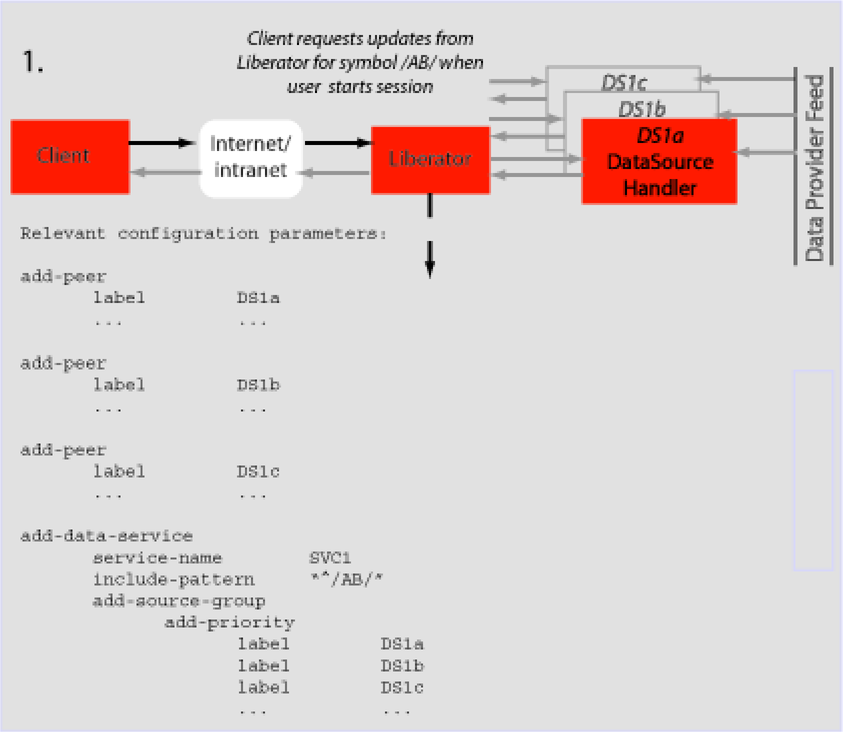


Client 2 also requests updates for symbol /AB/WX

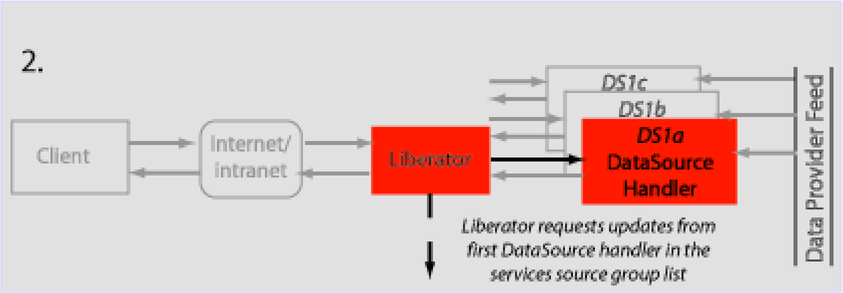


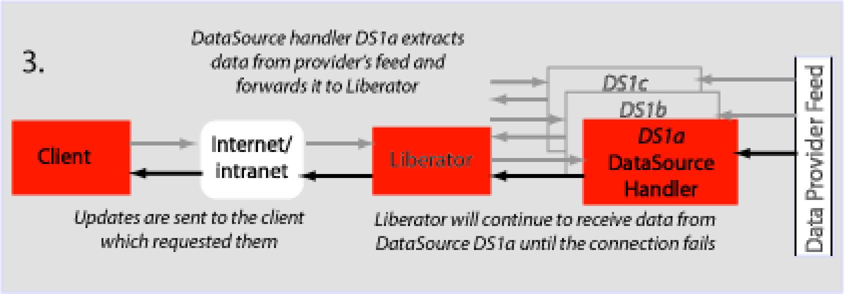
As updates for symbol /AB/WX are currently in Liberator's cache, it can forward them to Client 2 without rerequesting the data from the DataSource handler

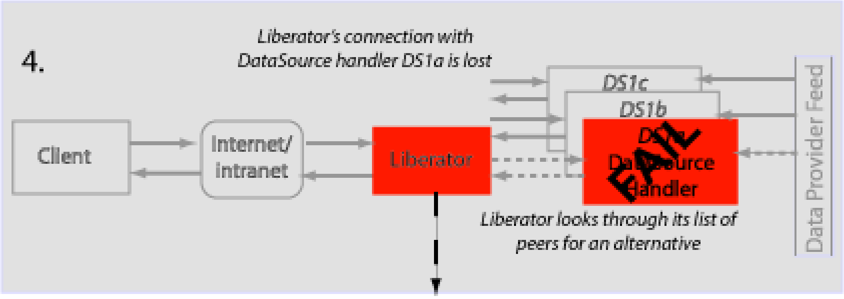
Active request with DataSource failover/load balancing

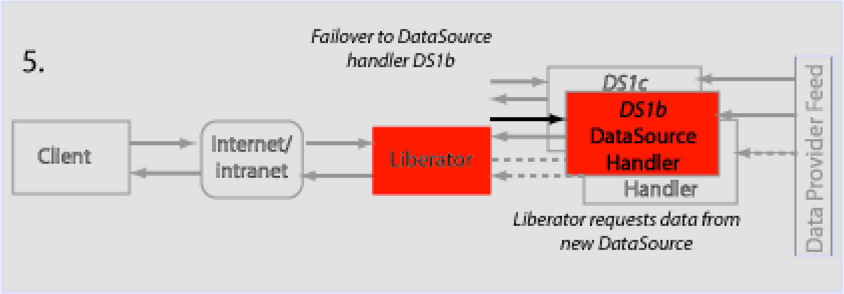


```
Relevant configuration parameters:

add-peer
        label           DS1a
        ...             ...

add-peer
        label           DS1b
        ...             ...

add-peer
        label           DS1c
        ...             ...

add-data-service
        service-name            SVC1
        include-pattern         "^/AB/"
        add-source-group
                add-priority
                        label           DS1a
                        label           DS1b
                        label           DS1c
                        ...             ...
```

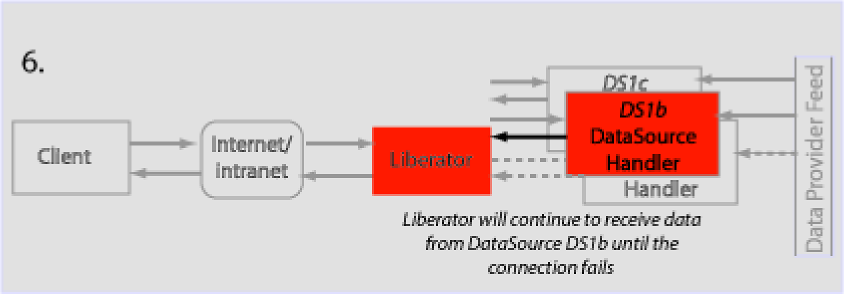

2.
Client
Internet/ intranet
Liberator
DS1c
DS1b
DS1a DataSource Handler
Data Provider Feed
Liberator requests updates from first DataSource handler in the services source group list


3.
DataSource handler DS1a extracts data from provider's feed and forwards it to Liberator
Client
Internet/ intranet
Liberator
DS1c
DS1b
DS1a DataSource Handler
Data Provider Feed
Updates are sent to the client which requested them
Liberator will continue to receive data from DataSource DS1a until the connection fails


4.
Liberator's connection with DataSource handler DS1a is lost
Client
Internet/ intranet
Liberator
DS1c
DS1b
DS1a DataSource Handler
FAIL
Data Provider Feed
Liberator looks through its list of peers for an alternative

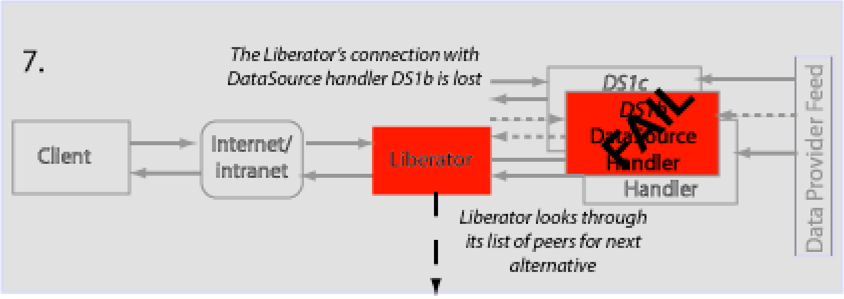

5.
Failover to DataSource handler DS1b
Client
Internet/ intranet
Liberator
DS1c
DS1b DataSource Handler
Handler
Data Provider Feed
Liberator requests data from new DataSource

6.



Liberator will continue to receive data from DataSource DS1b until the connection fails

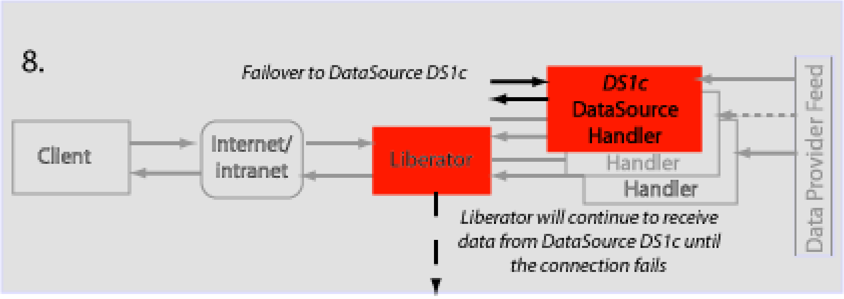
7.



The Liberator's connection with DataSource handler DS1b is lost

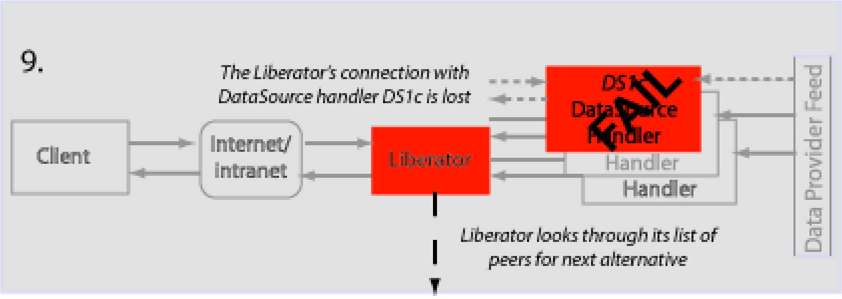
Liberator looks through its list of peers for next alternative

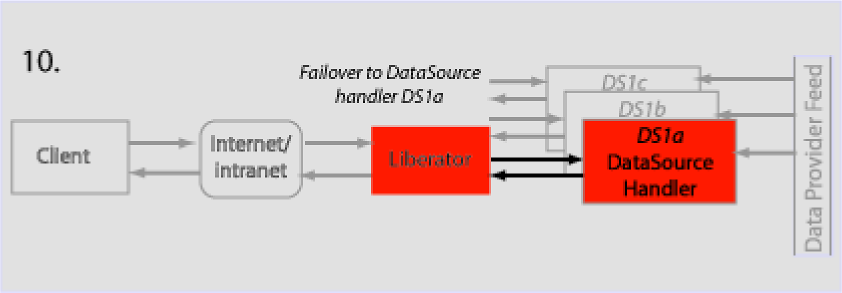
8.



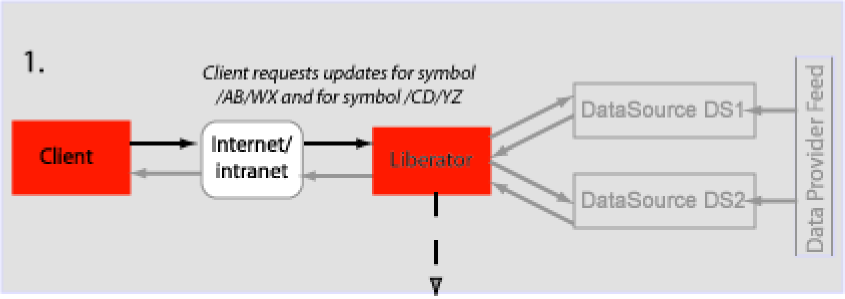
Failover to DataSource DS1c

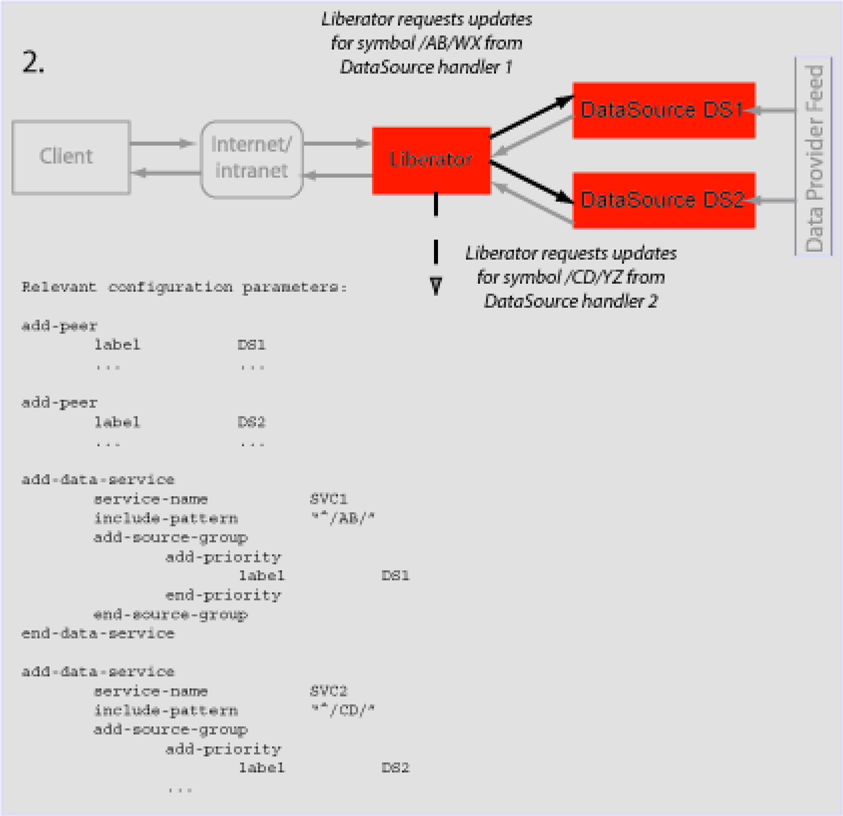
Liberator will continue to receive data from DataSource DS1c until the connection fails


9.
The Liberator's connection with DataSource handler DS1c is lost
Client
Internet/ intranet
Liberator
DS1c DataSource Handler
FAIL
Handler
Handler
Liberator looks through its list of peers for next alternative
Data Provider Feed


10.
Failover to DataSource handler DS1a
Client
Internet/ intranet
Liberator
DS1c
DS1b
DS1a DataSource Handler
Data Provider Feed

Active requests for data from 2 sources

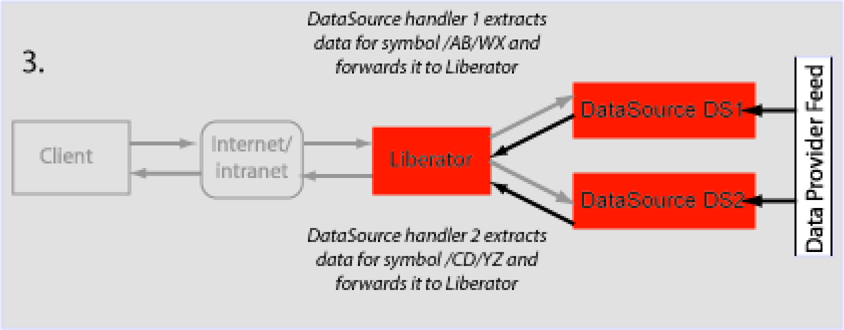


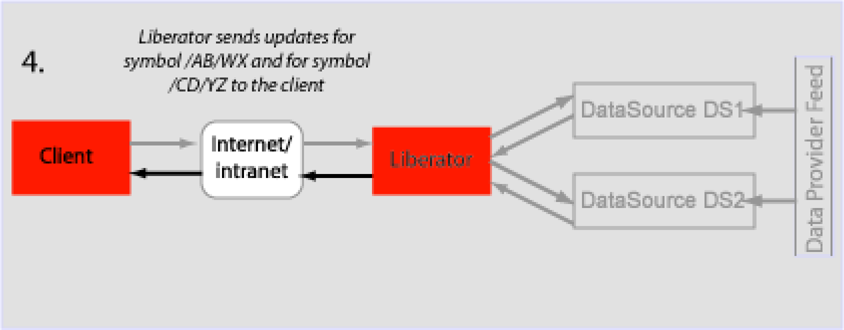


Liberator requests updates for symbol /AB/WX from DataSource handler 1

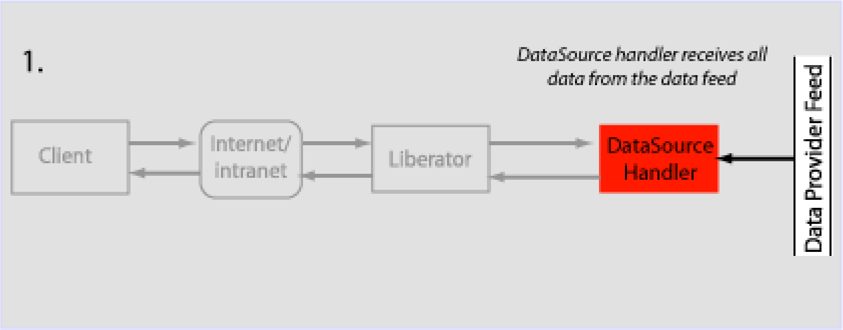
Liberator requests updates for symbol /CD/YZ from DataSource handler 2

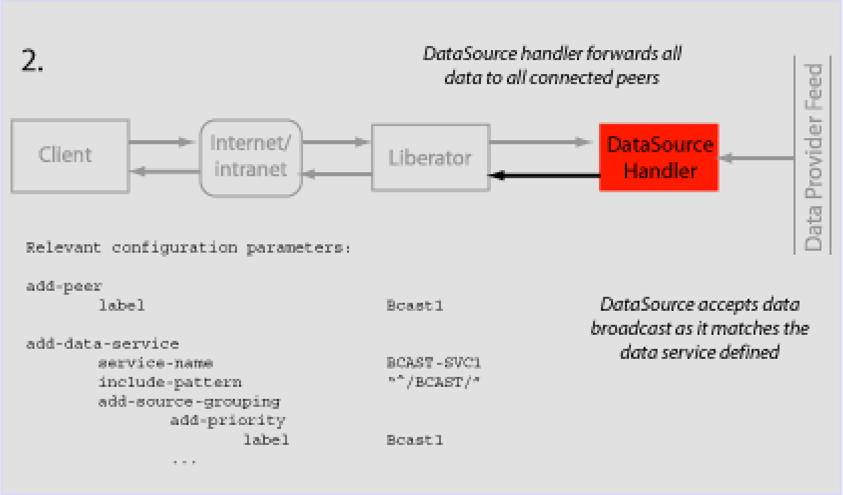
```
Relevant configuration parameters:

add-peer
        label                   DS1
        ...                     ...

add-peer
        label                   DS2
        ...                     ...

add-data-service
        service-name                    SVC1
        include-pattern                 "^/AB/"
        add-source-group
                add-priority
                        label                   DS1
                end-priority
        end-source-group
end-data-service

add-data-service
        service-name                    SVC2
        include-pattern                 "^/CD/"
        add-source-group
                add-priority
                        label                   DS2
                ...
```


3.
DataSource handler 1 extracts data for symbol /AB/WX and forwards it to Liberator
Client
Internet/ intranet
Liberator
DataSource DS1
DataSource DS2
Data Provider Feed
DataSource handler 2 extracts data for symbol /CD/YZ and forwards it to Liberator

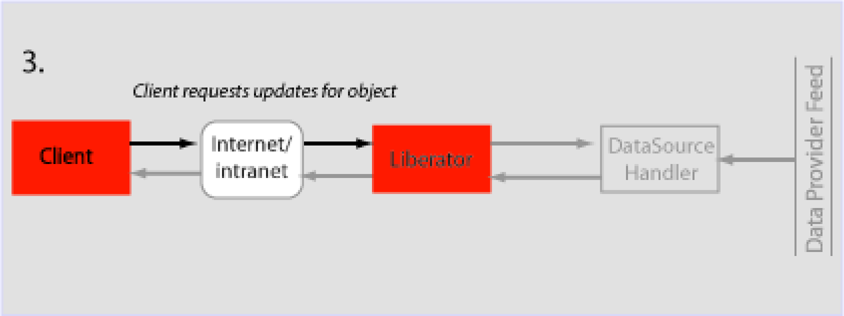

4.
Liberator sends updates for symbol /AB/WX and for symbol /CD/YZ to the client
Client
Internet/ intranet
Liberator
DataSource DS1
DataSource DS2
Data Provider Feed

Passive source-broadcast data



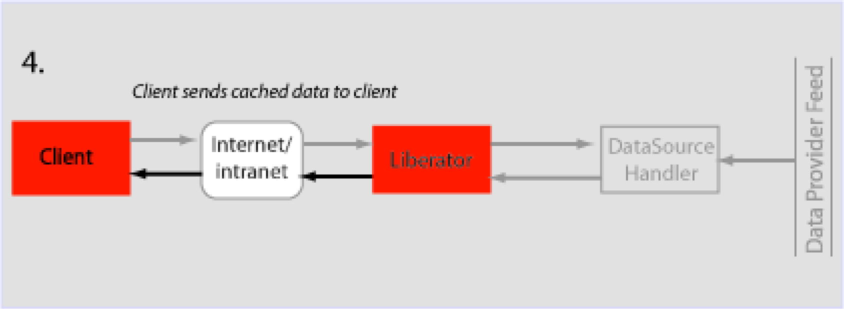

2.
DataSource handler forwards all data to all connected peers
Client
Internet/ intranet
Liberator
DataSource Handler
Data Provider Feed
DataSource accepts data broadcast as it matches the data service defined

```
Relevant configuration parameters:

add-peer
        label                               Bcast1

add-data-service
        service-name                        BCAST-SVC1
        include-pattern                     "^/BCAST/"
        add-source-grouping
                add-priority
                        label               Bcast1
                ...
```



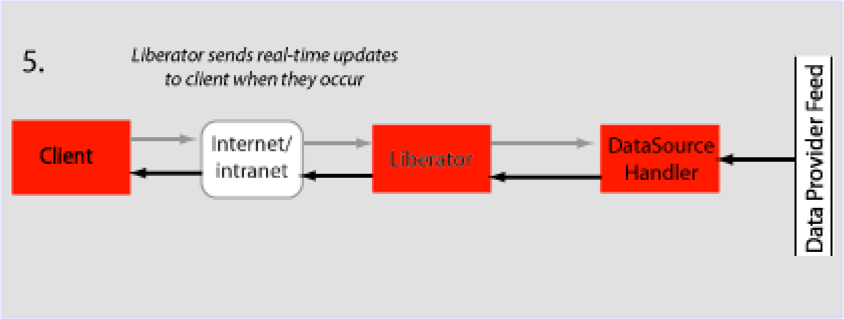
3.

Client requests updates for object



4.

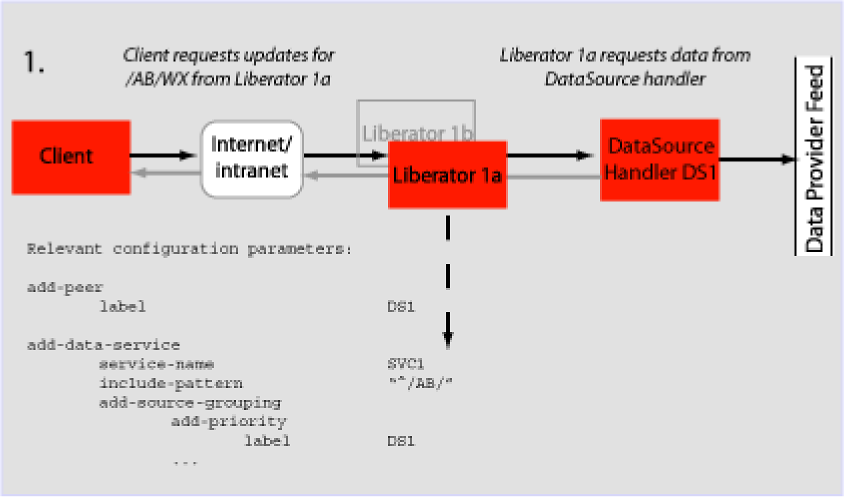
Client sends cached data to client



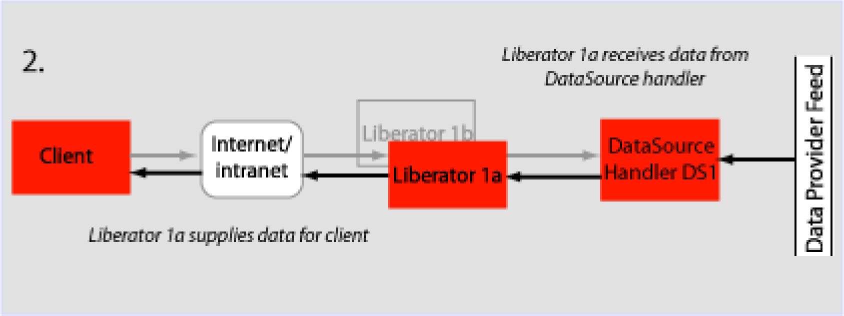
5.

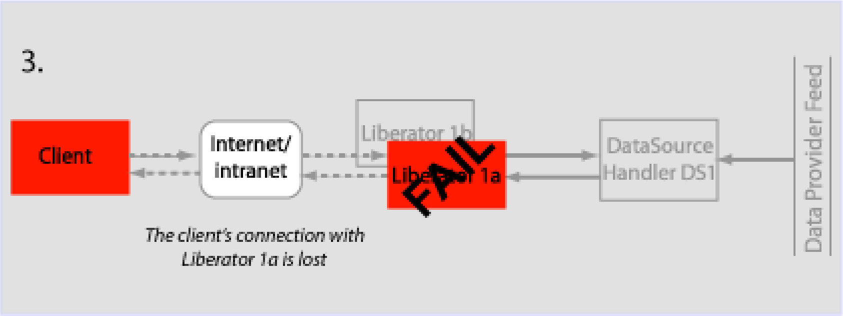
Liberator sends real-time updates to client when they occur

Liberator failover

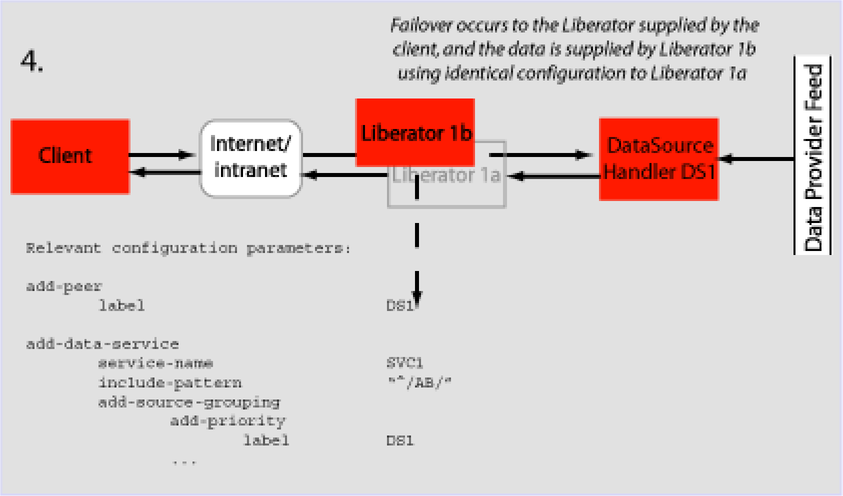

1.
Client requests updates for /AB/WX from Liberator 1a
Liberator 1a requests data from DataSource handler
Client
Internet/ intranet
Liberator 1b
Liberator 1a
DataSource Handler DS1
Data Provider Feed

```
Relevant configuration parameters:

add-peer
        label                          DS1

add-data-service
        service-name                   SVC1
        include-pattern                "^/AB/"
        add-source-grouping
                add-priority
                        label          DS1
                ...
```





The client's connection with Liberator 1a is lost



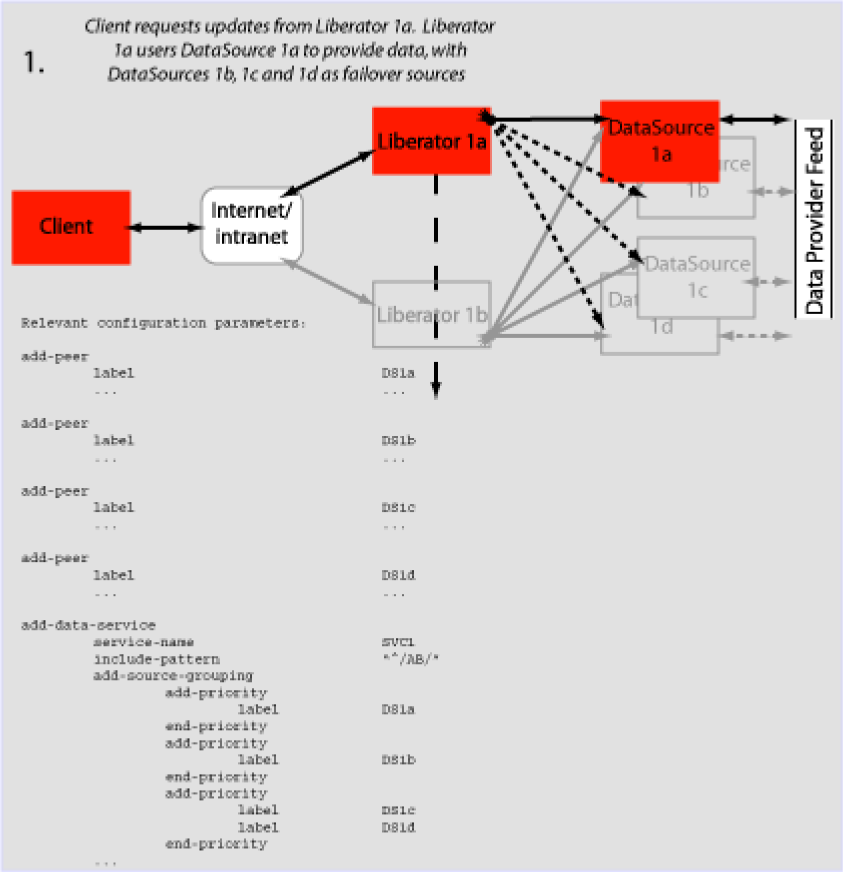
Failover occurs to the Liberator supplied by the client, and the data is supplied by Liberator 1b using identical configuration to Liberator 1a

```
Relevant configuration parameters:

add-peer
        label                               DS1

add-data-service
        service-name                        SVC1
        include-pattern                     "^/AB/"
        add-source-grouping
                add-priority
                        label               DS1
                ...
```

Liberator and DataSource failover

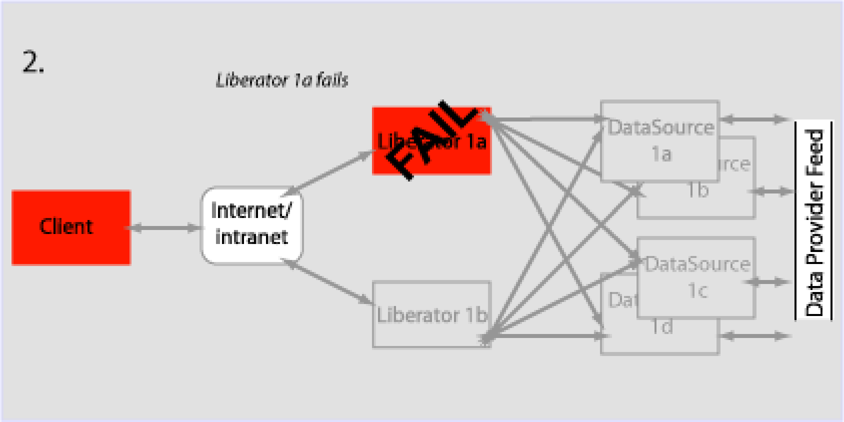
The following example assumes that the components are installed on different machines to provide extra resilience in case of failure:

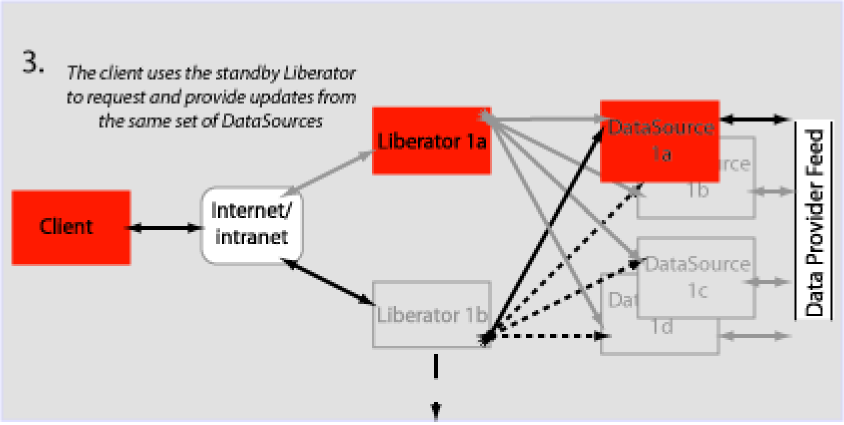
Host A	Liberator 1a
Host B	Liberator 1b
Host C	DataSource 1a DataSource 1b
Host D	DataSource 1c DataSource 1d



1. *Client requests updates from Liberator 1a. Liberator 1a users DataSource 1a to provide data, with DataSources 1b, 1c and 1d as failover sources*

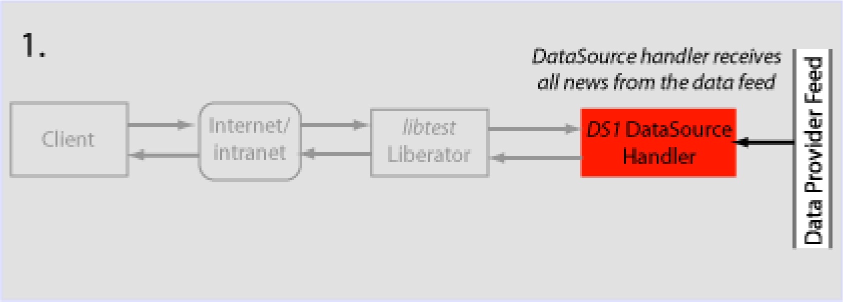
```
Relevant configuration parameters:

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
          ...
```

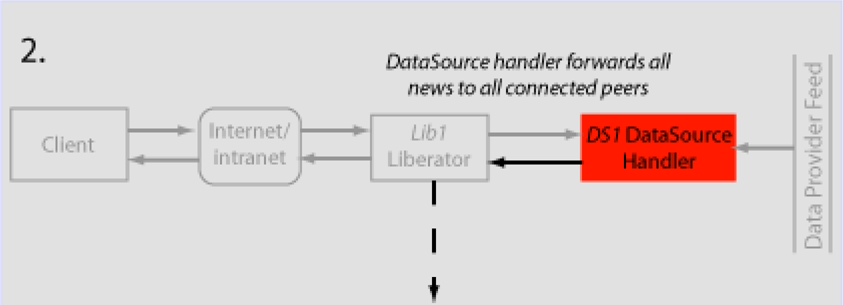

2.
Liberator 1a fails
Liberator 1a
FAIL
Client
Internet/
intranet
Liberator 1b
DataSource
1a
DataSource
1b
DataSource
1c
DataSource
1d
Data Provider Feed


3.
The client uses the standby Liberator
to request and provide updates from
the same set of DataSources
Liberator 1a
Client
Internet/
intranet
Liberator 1b
DataSource
1a
DataSource
1b
DataSource
1c
DataSource
1d
Data Provider Feed

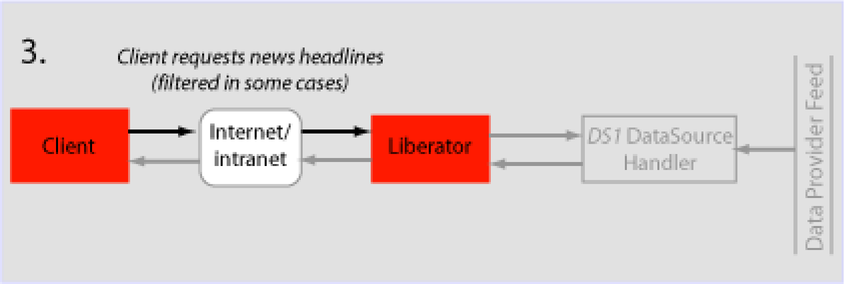
Requesting news headlines

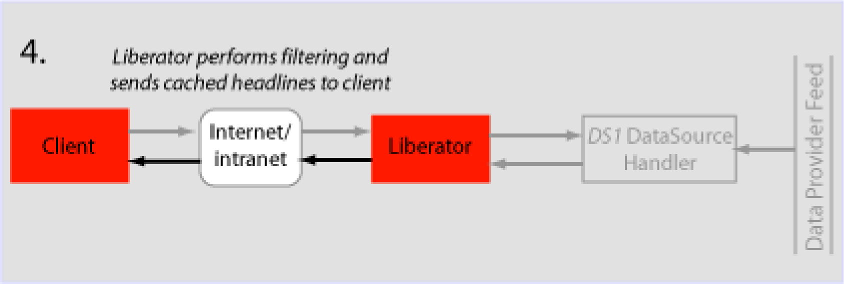
News headlines are delivered to Liberator as a broadcast feed—see page 56.

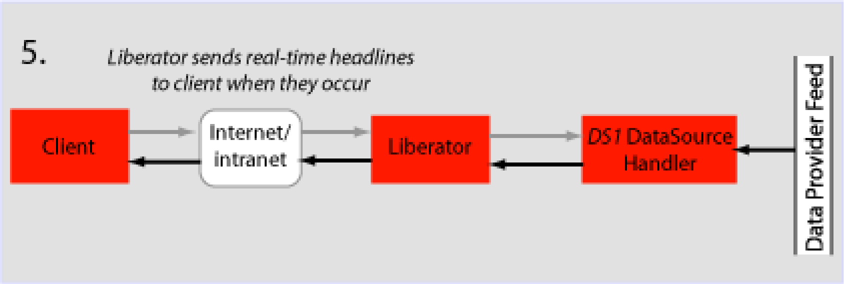




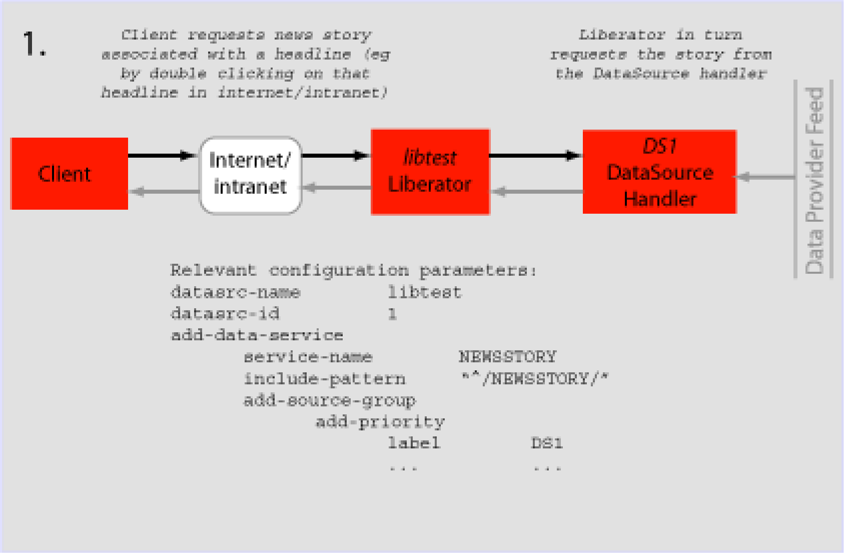
```
Relevant configuration parameters:

add-peer
        label                        DS1

add-data-service
        service-name                 NEWS
        include-pattern              "^/NEWS/"
        add-source-grouping
                add-priority
                        label        DS1
               ...
```


3.
Client requests news headlines
(filtered in some cases)
Client
Internet/
intranet
Liberator
DS1 DataSource
Handler
Data Provider Feed


4.
Liberator performs filtering and
sends cached headlines to client
Client
Internet/
intranet
Liberator
DS1 DataSource
Handler
Data Provider Feed


5.
Liberator sends real-time headlines
to client when they occur
Client
Internet/
intranet
Liberator
DS1 DataSource
Handler
Data Provider Feed

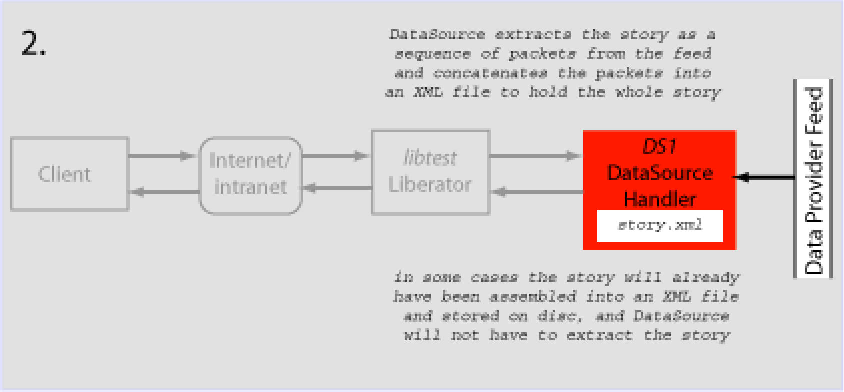
Requesting news stories



Client requests news story associated with a headline (eg by double clicking on that headline in internet/intranet)

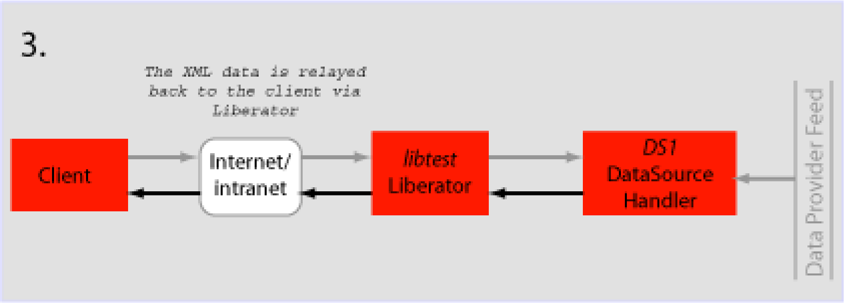
Liberator in turn requests the story from the DataSource handler

```
Relevant configuration parameters:
datasrc-name          libtest
datasrc-id            1
add-data-service
        service-name          NEWSSTORY
        include-pattern       "^/NEWSSTORY/"
        add-source-group
                add-priority
                        label         DS1
                        ...           ...
```



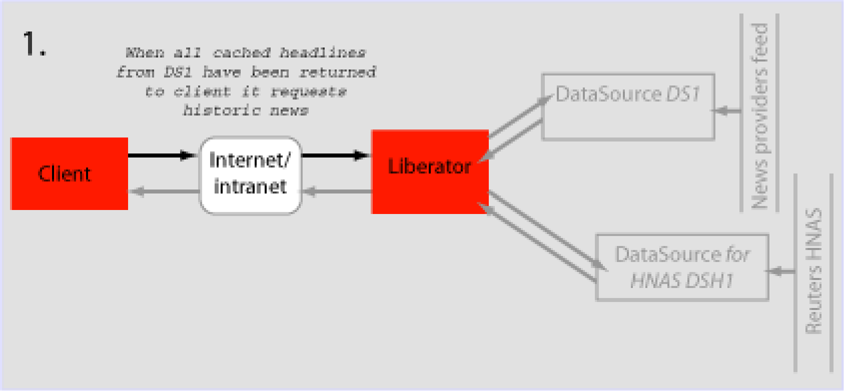
DataSource extracts the story as a sequence of packets from the feed and concatenates the packets into an XML file to hold the whole story

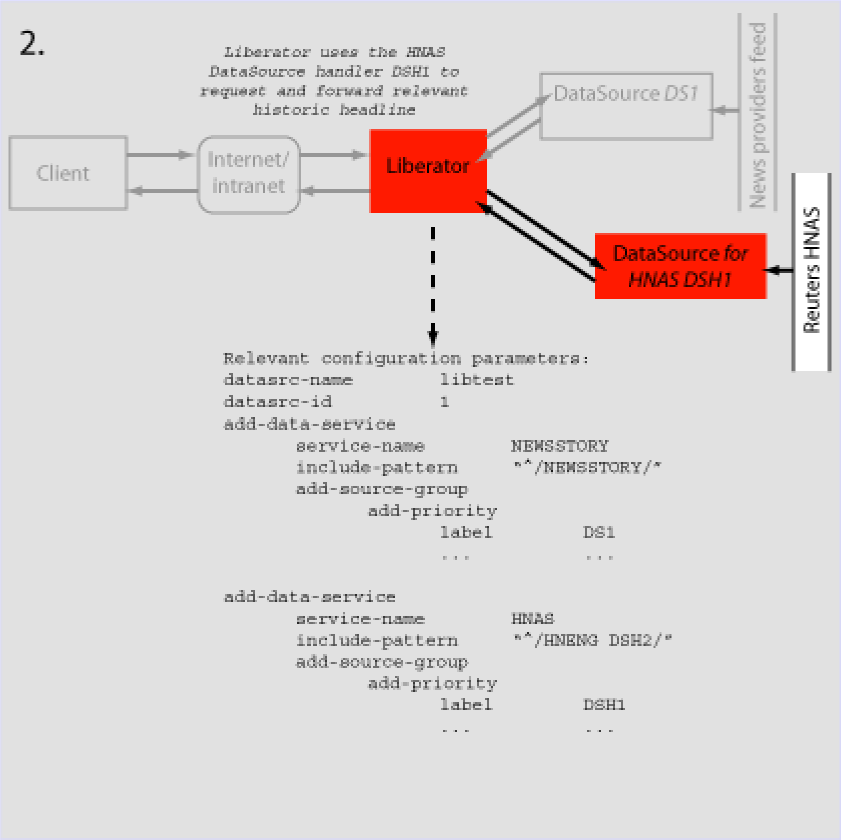
in some cases the story will already have been assembled into an XML file and stored on disc, and DataSource will not have to extract the story


3.
The XML data is relayed back to the client via Liberator
Client
Internet/ intranet
libtest Liberator
DS1 DataSource Handler
Data Provider Feed

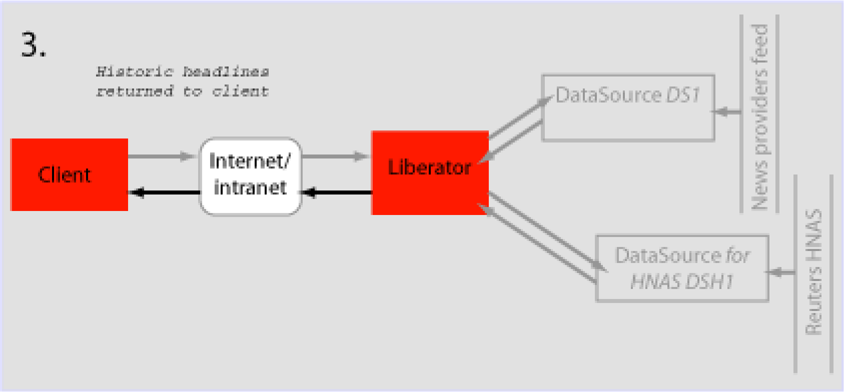
Requesting historic news headlines

The following steps take place after Liberator has supplied all its cached real-time headlines—see page 63.

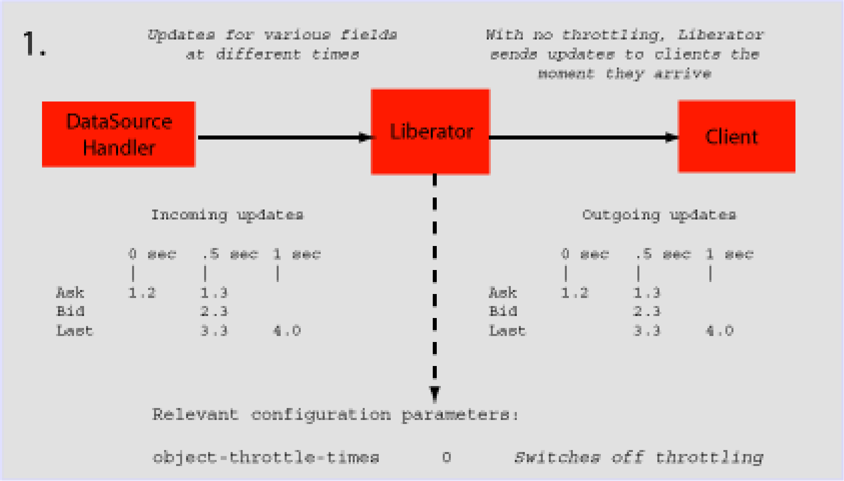




```
Relevant configuration parameters:
datasrc-name        libtest
datasrc-id          1
add-data-service
       service-name       NEWSSTORY
       include-pattern    "^/NEWSSTORY/"
       add-source-group
             add-priority
                    label        DS1
                    ...          ...

add-data-service
       service-name       HNAS
       include-pattern    "^/HNENG DSH2/"
       add-source-group
             add-priority
                    label        DSH1
                    ...          ...
```


3.
Historic headlines
returned to client
Client
Internet/
intranet
Liberator
DataSource DS1
News providers feed
DataSource for
HNAS DSH1
Reuters HNAS

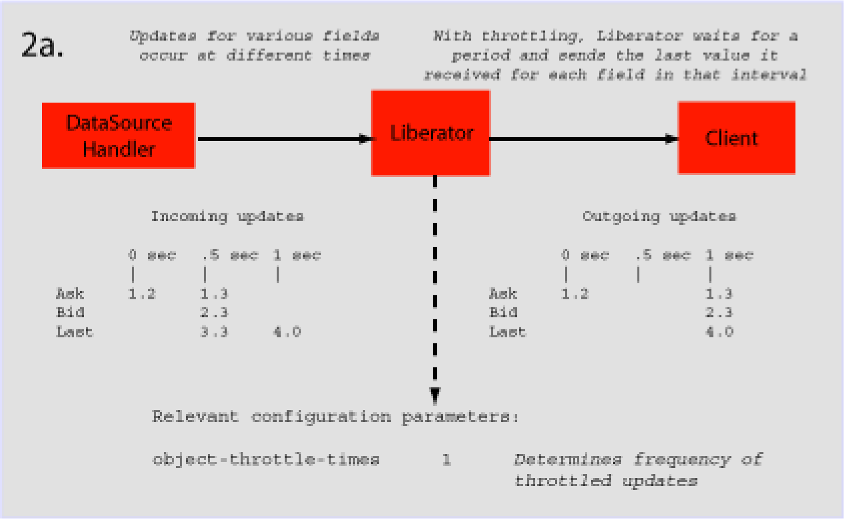
Throttling updates


1.
Updates for various fields at different times
With no throttling, Liberator sends updates to clients the moment they arrive
DataSource Handler
Liberator
Client

```
Incoming updates

        0 sec   .5 sec  1 sec
        |       |       |
Ask     1.2     1.3
Bid             2.3
Last            3.3     4.0
```

```
Outgoing updates

        0 sec   .5 sec  1 sec
        |       |       |
Ask     1.2     1.3
Bid             2.3
Last            3.3     4.0
```

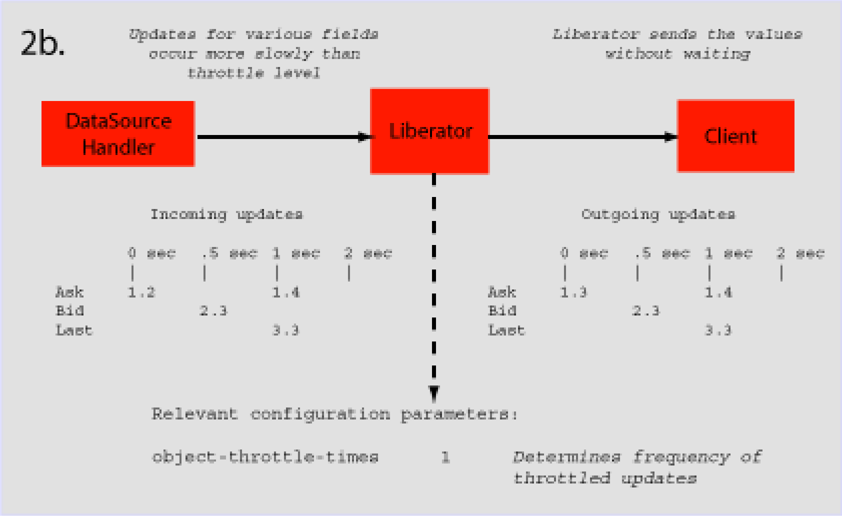
```
Relevant configuration parameters:

object-throttle-times      0      Switches off throttling
```


2a.
Updates for various fields occur at different times
With throttling, Liberator waits for a period and sends the last value it received for each field in that interval
DataSource Handler
Liberator
Client

```
Incoming updates

        0 sec   .5 sec  1 sec
        |       |       |
Ask     1.2     1.3
Bid             2.3
Last            3.3     4.0
```

```
Outgoing updates

        0 sec   .5 sec  1 sec
        |       |       |
Ask     1.2             1.3
Bid                     2.3
Last                    4.0
```

```
Relevant configuration parameters:

object-throttle-times      1      Determines frequency of
                                  throttled updates
```


2b.
Updates for various fields occur more slowly than throttle level
Liberator sends the values without waiting
DataSource Handler
Liberator
Client
Incoming updates
Outgoing updates
Relevant configuration parameters:

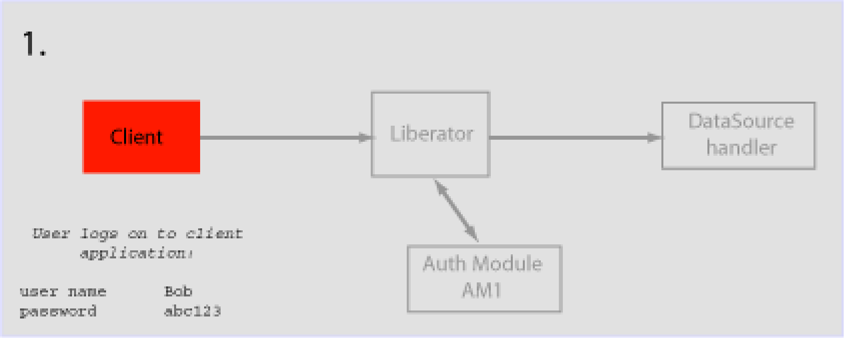
```
Incoming updates

        0 sec   .5 sec  1 sec   2 sec
        |       |       |       |
Ask     1.2             1.4
Bid             2.3
Last                    3.3
```

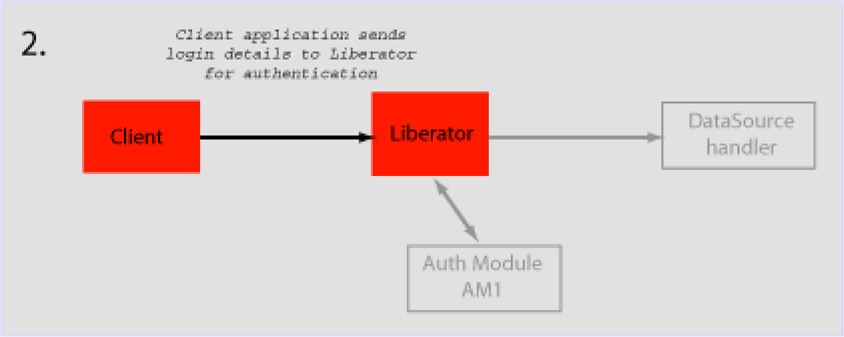
```
Outgoing updates

        0 sec   .5 sec  1 sec   2 sec
        |       |       |       |
Ask     1.3             1.4
Bid             2.3
Last                    3.3
```

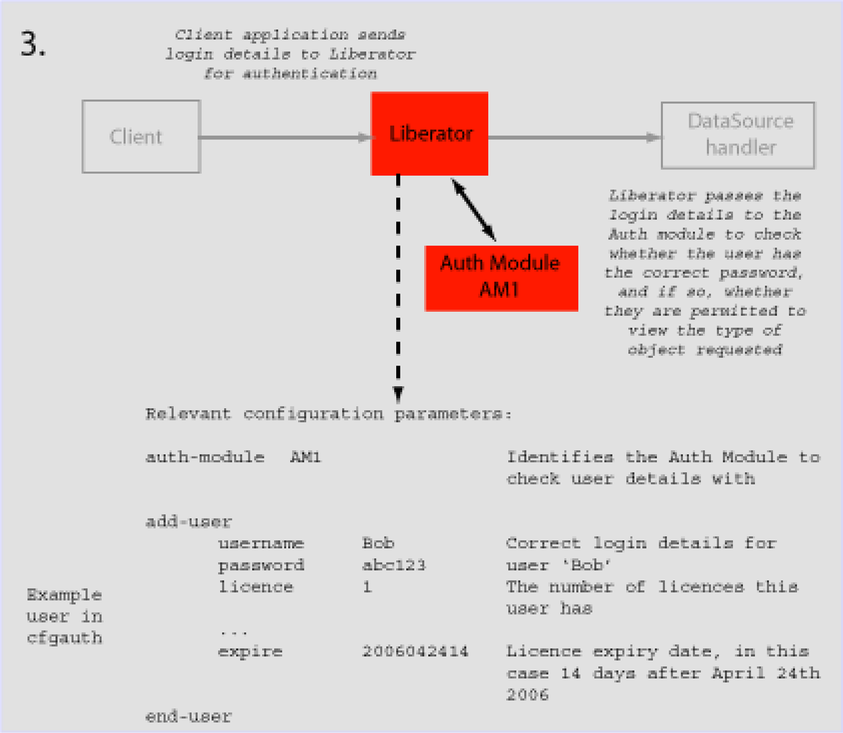
Relevant configuration parameters:

```
object-throttle-times      1      Determines frequency of
                                  throttled updates
```

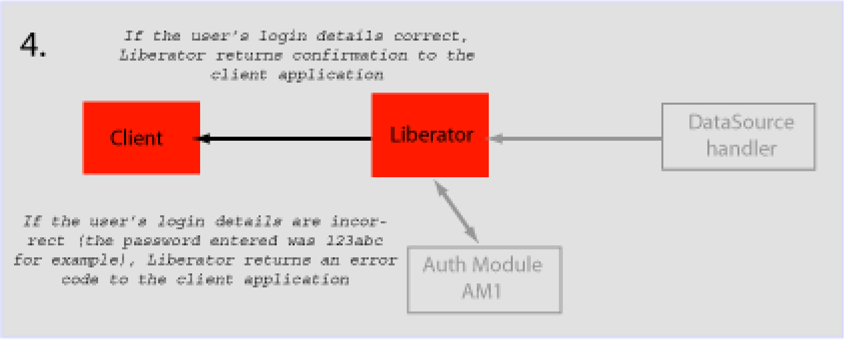
Authentication and authorisation of users using Auth Modules

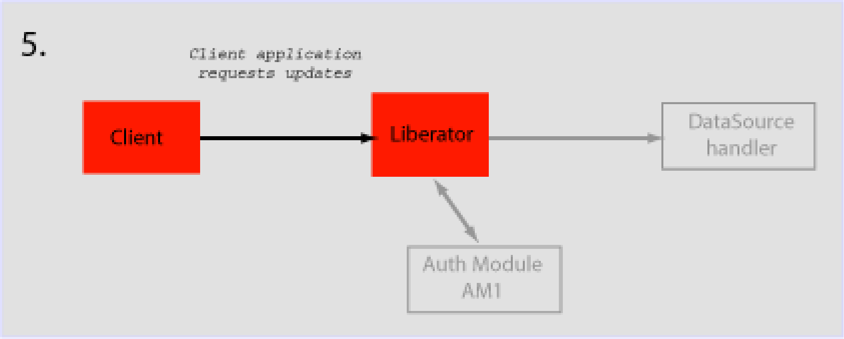


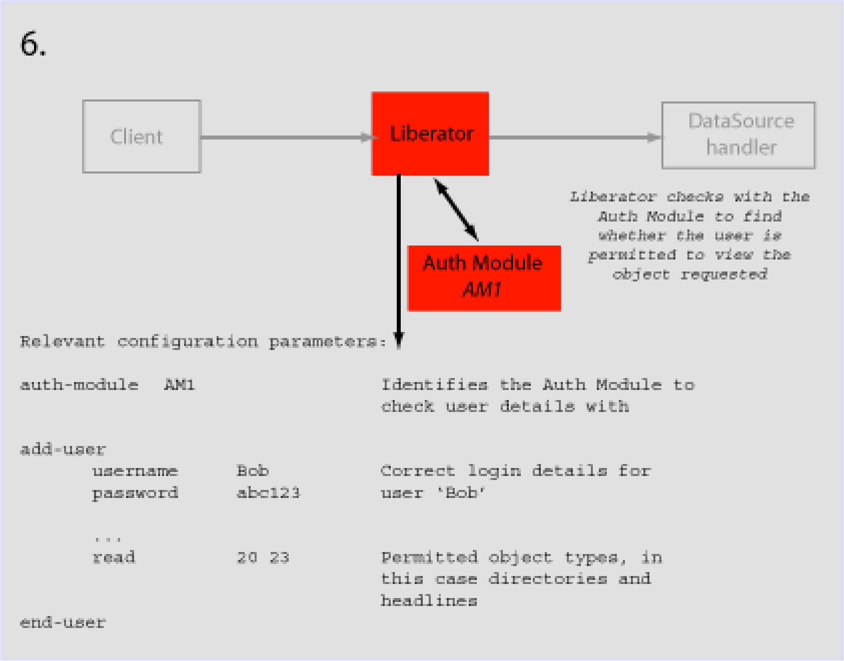

2.
Client application sends login details to Liberator for authentication
Client
Liberator
DataSource handler
Auth Module AM1


3.
Client application sends login details to Liberator for authentication
Client
Liberator
DataSource handler
Auth Module AM1
Liberator passes the login details to the Auth module to check whether the user has the correct password, and if so, whether they are permitted to view the type of object requested

```
Relevant configuration parameters:

auth-module   AM1                    Identifies the Auth Module to
                                     check user details with

add-user
        username     Bob             Correct login details for
        password     abc123          user 'Bob'
        licence      1               The number of licences this
                                     user has
Example
user in ...
cfgauth expire       2006042414      Licence expiry date, in this
                                     case 14 days after April 24th
                                     2006
end-user
```

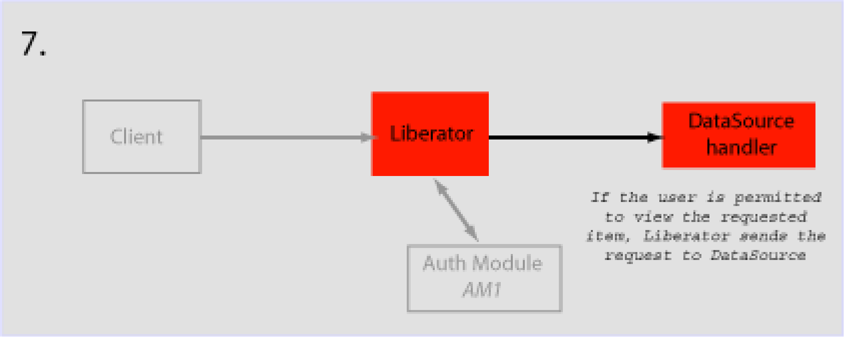

4.
If the user's login details correct, Liberator returns confirmation to the client application
Client
Liberator
DataSource handler
If the user's login details are incorrect (the password entered was 123abc for example), Liberator returns an error code to the client application
Auth Module AM1


5.
Client application requests updates
Client
Liberator
DataSource handler
Auth Module AM1

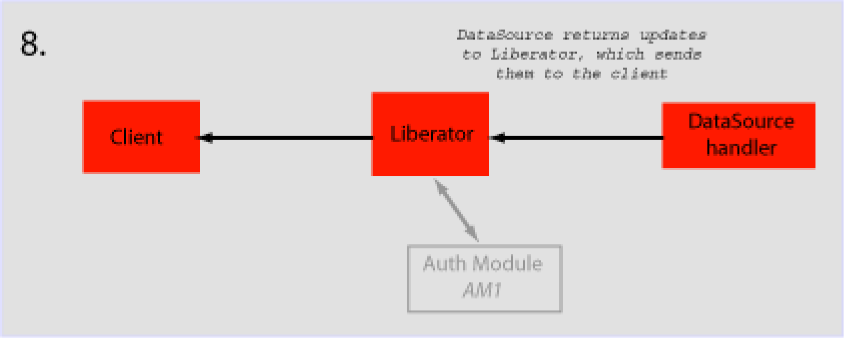


Liberator checks with the Auth Module to find whether the user is permitted to view the object requested

```
Relevant configuration parameters:

auth-module   AM1                  Identifies the Auth Module to
                                   check user details with

add-user
       username     Bob            Correct login details for
       password     abc123         user 'Bob'

       ...
       read         20 23          Permitted object types, in
                                   this case directories and
                                   headlines
end-user
```



If the user is permitted to view the requested item, Liberator sends the request to DataSource


8.
DataSource returns updates to Liberator, which sends them to the client
Client
Liberator
DataSource handler
Auth Module
AM1

4 About the data

4.1 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.

4.2 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 DataSources 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 DataSources 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 136 for further details.

4.3 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 4-1.

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 4-1: 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 80.

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.

4.4 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 4-1 shows an example field structure for a simple full quote display for the IBM stock on NYSE.

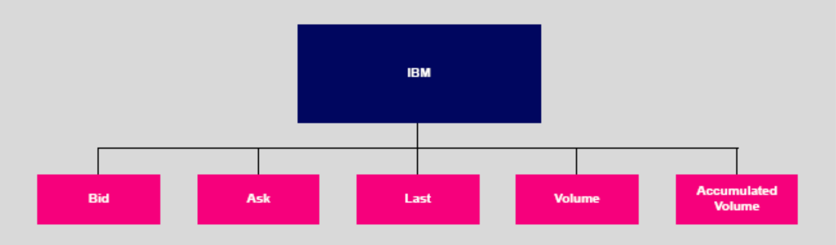


Figure 4-1: Example of Type 1 data within a record

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 4-2 might be applicable for a simple level 2 display for IBM, where there are three or more active market makers.

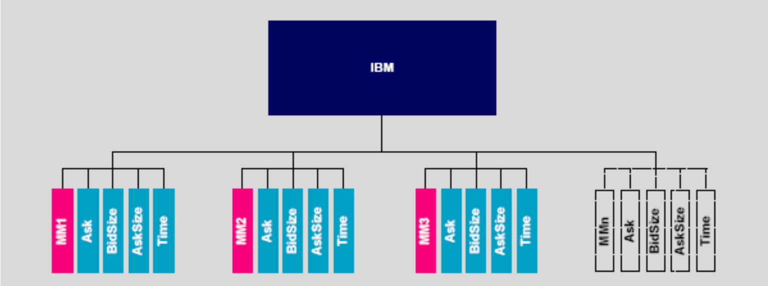


Figure 4-2: Example of Type 2 data within a record

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 4-3 shows a Type 3 field structure, which is similar to a Type 1 field structure but with many instances.

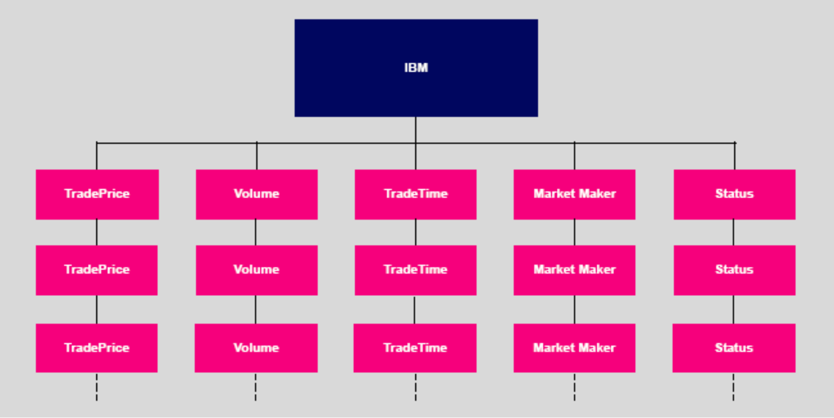


Figure 4-3: Example of Type 3 data within a record

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.

5 Communicating with clients

5.1 Enabling clients to connect using RTTP (over HTTP)

Specifying the Liberator URL

Liberator can write information regarding clients accessing it using HTTP to a log file. For more information on this and other logging facilities, see Monitoring performance on page 127

- 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 169

http-port — Network port to listen for HTTP connections. When the Liberator is running in production, this should usually be set to port 80 for HTTP or 443 for HTTPS. The Liberator will have to be started as 'root' on UNIX systems to allow binding to port 80.
See page 162

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 212

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 162

http-keepalive-timeout — Timeout in seconds of HTTP Keep Alive connections.
See page 163

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 166
http-connection-cookie-expires	Number of days before the cookie expires. See page 167

5.2 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. 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.

Liberator can run as an HTTPS server. Web pages, Java applets and other standard HTTP traffic can be sent over HTTPS. If the 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.

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

Using HTTPS—Linux and Solaris

https-enable	This option switches on support for HTTPS connections. See page 169

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_ssl.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 169

https-port
Configures which network port to listen on for HTTPS connections.
See page 169

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 170

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 169

https-privatekey
Filename of the SSL private key. This file should be in PEM format.
See page 169

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 170

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 171

The hardware and software engines that the Liberator supports are listed in Table 5-1 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 5-1: Supported hardware and software engines

ssl-engine-flags Flags to be passed to the engine implementation.
See page 172

The available flags to use are listed in Table 5-2 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 5-2: ssl-engine-flags flags

5.3 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 77.

- 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 174.

direct-port Network port to listen for RTTP connections.
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 212.

5.4 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 parameter in the file *rttpd.conf* to identify any object to be created on start-up.

add-object — Identifies an object that a client can request. This configuration option can also specify throttle times that are specific to this object, and override any global values that have been set.
See page 175

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 user X requests /HN/NEWSSTORY/1234, the object could be mapped to /HN/NEWSTORY/userX/1234.
See page 179
Example:

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

where %u is the username 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 178

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 178

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 DataSources 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 datasource (please refer to the DataSource SDK Documentation).

- 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
```

5.5 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 187

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 80).

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 187

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 187

- 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 81). 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 180

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 analagous 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 Type 3 records Liberator will keep in cache.

record-max-cache Maximum number of type 3 record data to keep.
See page 175

5.6 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 214
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 215
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 214
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 214
newscode-hash-size	Default number of entries in the newscode exceptions hashtable. See page 215

5.7 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 175

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 175

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 175

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 211

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 211

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 211

buf-elem-len Length of standard buffer element, in bytes. See page 211

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 211

newsitems-saved Maximum number of news items (headlines) that Liberator stores in memory. See page 214

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 214

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 215

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

5.8 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 210

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 210

6 Authentication and entitlement

6.1 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.

- Authorisation 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. Authorisation 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**.

6.2 Using auth modules

An Auth Module provides a means performing authentication and authorisation.

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 181

auth-module Name of authentication module to use.
See page 181

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 164

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 licence allows if desired. The default setting of 0 means there is no limit.
See page 181

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 181). A warning will only be logged again if the number of users drops below the max-user-ok level.
See page 181

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 181

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 182

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 182

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 183

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 183

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 184

6.3 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 seperately) 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 authorisation 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 181).

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 220

write-access Determines all users' permission to write to or create any object.
See page 220

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 181).

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 221

encrypted-passwords A global option to determine whether a password is encrypted or not.
See page 223

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 D: Javaauth configuration on page 242 for details on how to configure **javaauth** to be able to connect to your module.

6.4 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 217

signature-hashsize Size of hashtable for storing signature keys.
See page 217

add-sigkey Adds a signature checking key to the configuration file.
See page 217

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

7 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.

7.1 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 DataSources and feeds.

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 7-1 below.

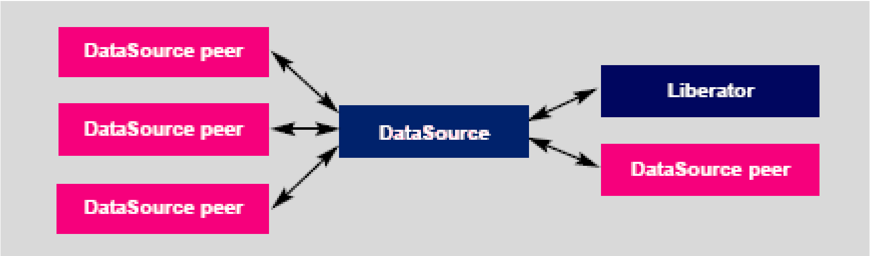


Figure 7-1: DataSource acting as a data source and data sink

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.

  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 one minute after the user discarded the object: this is prevents objects being requested and discarded from the source unnecessarily.

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

7.2 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 189.

This name can be overridden by putting a value in the local-name option of the add-peer entry (see add-peer on page 190). %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 189

This ID can be overridden by putting a value in the local-id option of the add-peer entry (see add-peer on page 190), in which case it must match the remote-id given in the add-peer entry in the remote DataSource's configuration.

7.3 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 189

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 190

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 190 and Making SSL connections with DataSources on page 124

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 page 190

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 136).

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 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
```

New in Liberator 4.0, using Data Services, multiple peers can be configured for failover without Liberator needing to swap connections. Please see Data services on page 116.

- 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 allowed to connect.

datasrc-reject-new-peers turns off this default behaviour so the new DataSource peer is not allowed to connect. See page 189

- Use the following parameters in the configuration file *rttpd.conf* to configure the timing of heartbeats between peers.

heartbeat-time	Time in seconds between DataSource heartbeats. Peers compare their heartbeat-time values and all use the lowest. See page 195
heartbeat-slack-time	Time in seconds after a heartbeat has not been received before disconnecting and trying to reconnect (this value is not compared by peers). See page 195

- 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.

record-type1-clear-on-failover	Clear Type 1 data for active objects. See page 179
record-type2-clear-on-failover	Clear Type 2 data for active objects. See page 179
record-type3-clear-on-failover	Clear Type 3 data for active objects. See page 179

7.4 Reconnecting peers using the UDP interface

Liberator includes a UDP command interface that enables you to send a UDP messages 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 219

udp-interface Network interface to listen on for UDP messages.
See page 219

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.

7.5 Data services

You must use Data Services in order for Liberator to request a particular object from a particular DataSource or to define where broadcast data can come from.

Data Services is the replacement to the old Source Mapping feature. Data Services allow an administrator to define where data comes from, based on its subject name. It also allows 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 datasouce 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 request from just one peer, keeping state so the next object will be requested from a different peer - 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 200.

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 200).

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 200.)

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 the first request that matches "^/NA/" will go to DataSource peer src1, the second to src2 and so on, wrapping round to DataSource peer 1 again, spreading the load evenly.

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 200.)

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 you are not sure which peer has which data.*

Specifying priority or failover

You can configure a data service to send to an alternative Data Source 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 from src1 and src2 on a round robin basis, and one from src3 and src4 on a round robin basis.

Waiting for responses

Use the following parameter 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 ro 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 Data services on page 200).

source-request-timeout

Time in seconds that the Liberator will wait for an individual DataSource 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 ro 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 add-peer on page 190).

Waiting for requests

Use the following parameter in the configuration file rttpd.conf to set the timeout period to wait for a request for data before discarding the data from Liberator's cache.

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 175.

7.6 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 199.

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 190).

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 199.

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 199.

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 216.

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 216.

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 216.

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
```

7.7 Making SSL connections with DataSources

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 DataSources, and in client mode when connecting to DataSources.

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 195.
ssl-passwordfile	Identifies the file containing the SSL certificate passphrase. See page 198.

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 190).

- 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 190).

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 190.

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.

8 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 or the object browser.

8.1 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 Enterprise 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. These modifications are documented in the **Enterprise Monitoring Console Getting Started Guide**.
- **Socket Monitoring** : A simple command-based socket protocol, similar to ftp, that allows access to the information and operations exposed by the system.

Please refer to the **Management and Monitoring Overview** document which is provided with the Liberator kit for more details.

8.2 Log files

Liberator creates several log files when it runs. The format and content of messages written to the request log are described in **Appendix C: Log File Messages and Message Formats** starting on page 230.

Log file configuration

To view packet logs, you must use the logcat utility (see The logcat utility on page 132). Other logs are simple text.

- Specify the directory where log files will be created using log-dir (see page 156)
- Specify the names of each log file using the options listed in Table 8-1. These take the form [log type] [file location] [application name].log and will take the form [log type] [application name].log in the specified directory.

Log filenames can use the following abbreviations:

- %r can be used to represent application-root (see page 154)
- %a can be used to represent application-name (see page 154).

Log type	Configured by	Default	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 default]	News headlines for replaying on startup.

Table 8-1: Configuring log files

Log file cycling

By default all log files are cycled at 0400 each day and a separated log file created for each day. These default settings are specified using the following configuration options:

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

Note: *Often this default config 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 config 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.

log-maxsize A value of 0 means log files will cycle every time they are checked irrespective of size.
See page 156

log-cycle-time A value of 240 represents 0400 as it is defined as minutes from midnight.
See page 156

log-cycle-period A value of 1440 represents 24 hours as it is defined as minutes from midnight.
See page 156

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 157

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 157

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
```

Results in each log file being checked every 30 minutes and moved to *logfile.old* if it is bigger than 1024000 bytes.

- Use the following parameter in the configuration file *rttpd.conf* to set different cycling formats for different logs.

add-log-cycle Overrides the global settings and defines different cycling settings for individual log files. A possible application of this is to cycle the logs every night by default, but 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.
See page 159

Example log cycling configuration (individual log):

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

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

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/4.0.0-1 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 154

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 and Solaris platforms only.*

Note: *This feature is not reliable as catching of segmentation faults and bus errors is not always possible.*

8.3 The logcat utility

The **logcat** utility is needed to process the binary packet logs and can be used in the same way as the standard **cat** command. It can be found in the bin directory of the Liberator installation.

The logcat utility can take the arguments listed in Table 8-2 below.

Argument	Description
-h, --help	Detailed information on logcat options
-v, --log-version	the version of the log
-i, --print-info	The version, type and source of the given 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.
-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.

Table 8-2: 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'
2005/08/22-16:46:52.528 +0100: 192.168.201.102 < PEERINFO 1
type2src-devsun1 0
2005/08/22-16:46:52.528 +0100: 192.168.201.102 > PEERINFO 0 rttpd-
devsun2 0
2005/08/22-16:47:00.000 +0100: 192.168.201.102 > SUBJREQ 1 1 /I/
VOD.L
2005/08/22-16:47:00.000 +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.*

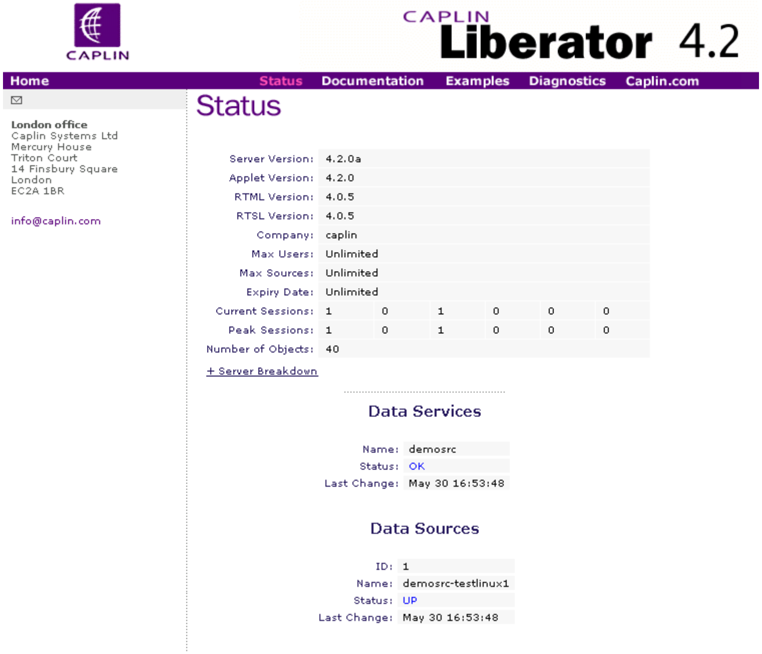
You must then tell logcat that each part is a packet log as the header will now be missing.

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

8.4 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.

Figure 8-1 shows a typical status web page.



London office
Caplin Systems Ltd
Mercury House
Triton Court
14 Finsbury Square
London
EC2A 1BR

info@caplin.com

Status

Server Version:	4.2.0a					
Applet Version:	4.2.0					
RTML Version:	4.0.5					
RTSL Version:	4.0.5					
Company:	caplin					
Max Users:	Unlimited					
Max Sources:	Unlimited					
Expiry Date:	Unlimited					
Current Sessions:	1	0	1	0	0	0
Peak Sessions:	1	0	1	0	0	0
Number of Objects:	40					

+ Server Breakdown

Data Services

Name:	demosrc
Status:	OK
Last Change:	May 30 16:53:48

Data Sources

ID:	1
Name:	demosrc-testlinux1
Status:	UP
Last Change:	May 30 16:53:48

Figure 8-1: Status web page

- 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 click on Status.

The information contained on the status web page includes the following.

Liberator status information

Server Version

Release number of this version of Liberator.

Applet Version

Release number of this version of the RTTP Applet.

RTML Version

Release number of this version of RTML.

RTSL Version

Release number of this version of RTSL.

Company

The name of the company on the licence agreement.

Max Users

The maximum number of users who can be logged on to Liberator, as specified in your licence agreement.

Max Sources

The maximum number of data sources which can be connected to Liberator, as specified in your licence agreement.

Expiry Date

Date when your Liberator licence expires.

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 licence 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.

Data source information

There is one column of information for each data source to which the Liberator is connected. Each column shows the following details:

ID

Numerical identifier for the data source.

Name

Name of the data source.

Status

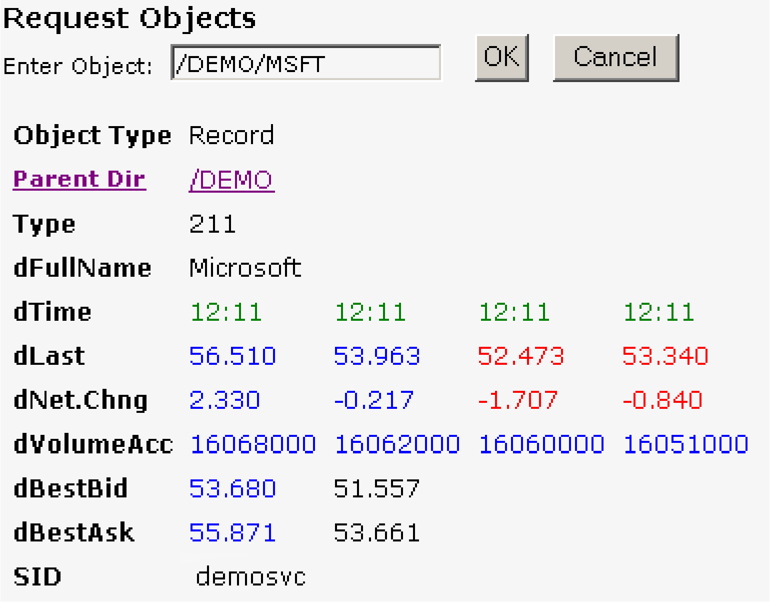
Status of the connection to the data source. This can be either:
UP if the connection has been established;
DOWN if there was a connection, but it has been lost.

Last Change The date and time at which Status last changed.

Addr IP address and port of currently connected or most recently connected DataSource.

8.5 Object Browser

The Object Browsing Tool, which ships with the Liberator, can be used to request data from the data source. Browse to Examples -> Object Browsing Tool and request (for example) /DEMO/MSFT.



Request Objects

Enter Object: /DEMO/MSFT

Object Type	Record			
Parent Dir	/DEMO			
Type	211			
dFullName	Microsoft			
dTime	12:11	12:11	12:11	12:11
dLast	56.510	53.963	52.473	53.340
dNet.Chng	2.330	-0.217	-1.707	-0.840
dVolumeAcc	16068000	16062000	16060000	16051000
dBestBid	53.680	51.557		
dBestAsk	55.871	53.661		
SID	demosvc			

Figure 8-2: Object Browser Tool from the Examples page

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 184

8.6 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 are shown in Table 8-3 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 8-3: Debug levels

Note: *A list of all error messages and their associated debug level can be found as Appendix D on page 241. In addition to these, Liberator also logs other system messages and messages from previous releases.*

The default debugging level that is used at startup is configurable. However, the level can be changed dynamically while Liberator is running by using the debug UDP command when the UDP message interface is enabled.

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

debug-level Determines the errors and events that are reported to the log files when Liberator starts operating.
See page 157

- 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 219

udp-interface Network interface to listen on for UDP messages.
See page 219

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 debug-level (see page 157).

Syntax: debug level

Parameter:

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

8.7 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 seperated 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 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 98) and bursting (see Configuring "bursts" on page 101). 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.*

9 Optimising efficiency

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

9.1 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 211

9.2 Improving performance using threads

threads-num Recommended value: At least 1 per CPU
This states the number of client side threads and defaults to 2; however, Liberator will also use a thread on its DataSource side. There must be at least one thread per CPU.
See page 212

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 211

9.3 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 210

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 210

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 210

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 180

9.4 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 212

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 213

datasrc-tcp-nodelay-off

Recommended value: FALSE
Turns off the no delay feature for datasource peer sockets.
See page 213

9.5 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 188

9.6 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.

9.7 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 166

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 166

http-max-header-line-length
Recommended value: 65536
Maximum number of bytes allowed in a single HTTP header line.
See page 166

http-max-header-lines
Recommended value: 30
Maximum number of header lines allowed in an HTTP message.
See page 166

http-max-body-length
Recommended value: 65536
Maximum number of bytes allowed in the body of an HTTP message.
See page 166

10 Running Liberator with many users

Liberator can normally support up to 10,000 concurrent user sessions, and upto 30,000 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.

10.1 Configuring Liberator for a high number of users

- If your licence 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 licence is for an unlimited number of users, set system-max-files (see page 154) 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.*

10.2 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.

- 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 licence states a higher number of users. See page 154

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 licence 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 181*

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.

In */etc/rc.d/rc.local* add:

```
echo 32768 > /proc/sys/fs/file-max
echo 131072 > /proc/sys/fs/inode-max
```

- Alternatively, on RedHat 6.2 and above you can achieve the same by adding the following into */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. Using kernel 2.4 upwards, the following is the default for systems with more than 128Mb of RAM.

- In */etc/rc.d/rc.local*, add:

```
echo "32768 61000" > /proc/sys/net/ipv4/
ip_local_port_range
```

- Alternatively for RedHat 6.2 and above add the following to */etc/sysctl.conf*:

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

10.3 Changing file descriptor limits—Solaris

The following commands change both the per process and the system-wide file descriptor limits. They also increase the size of the TCP connection hashtable.

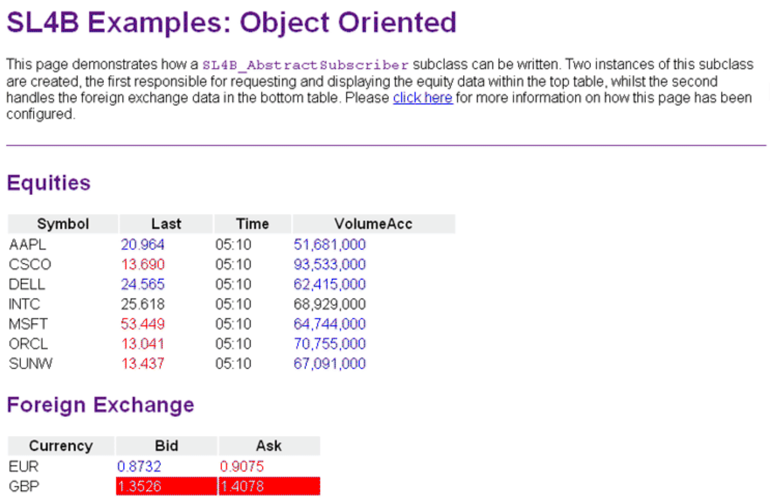
- In */etc/system* add:

```
set rlim_fd_cur = 256
set rlim_fd_max = 32768
set tcp:tcp_conn_hash_size = 65536
set ipc_tcp_conn_hash_size = 65536
```

11 Liberator demonstrations

To check your Liberator is running properly some simple examples are provided, created using the SL4B SDK.

Figure 11-1 shows one of these examples, in which values randomly generated by Liberator are updated in real time.



SL4B Examples: Object Oriented

This page demonstrates how a `SL4B_AbstractSubscriber` subclass can be written. Two instances of this subclass are created, the first responsible for requesting and displaying the equity data within the top table, whilst the second handles the foreign exchange data in the bottom table. Please click here for more information on how this page has been configured.

Equities

Symbol	Last	Time	VolumeAcc
AAPL	20.964	05:10	51,681,000
CSCO	13.690	05:10	93,533,000
DELL	24.565	05:10	62,415,000
INTC	25.618	05:10	68,929,000
MSFT	53.449	05:10	64,744,000
ORCL	13.041	05:10	70,755,000
SUNW	13.437	05:10	67,091,000

Foreign Exchange

Currency	Bid	Ask
EUR	0.8732	0.9075
GBP	1.3526	1.4078

Figure 11-1: SL4B example

In order to view this example, you must perform the following steps:

- Start the demo feed;
- Access the relevant page on the Liberator web site.

11.1 Starting the demo feed—Linux and Solaris

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.

11.2 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 151

11.3 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 164). Any changes made to this entry will be reflected in the accessibility of the web site example pages.*

11.4 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 30).

Some of the more advanced configuration options are described in Optimising efficiency on page 142.

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.

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 and Solaris platforms only).

Type: boolean

Default value: FALSE

include-file

Imports configuration parameters contained in another file. These 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 154) 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
```

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

syslog-facility

This is the syslog facility to use when logging to the unix based syslog - See “System log files (syslog)” on page 131.

Type:	string
Default value:	user6

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, or configure the cycling of each log file individually.

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 UNICX 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.]

debug-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 debug level can be found as Appendix B: Log file messages and formats on page 230*

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: http_access_log the HTTP access log file (see page 164 news_log the log file to store news headlines (see page 216 object_log the object log file (see page 183 request_log the request log file (see page 183 session_log the session log file (see page 183 event_log the event log file (see page 154 packet_log the packet log file (see page 189
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: 20

http-keepalive-timeout Timeout period in seconds of HTTP Keep Alive connections.

Type: integer

Default value: 30

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 Enable HTTPS

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 HTTP connections.

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_ssl.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 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.

Name	Default	Description
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 89 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 162)
https-certificate	string	https-certificate	Filename of the SSL certificate. Overrides the global setting https-certificate (see page 169).

Name	Type	Default	Description
https-privatekey	string	https-privatekey	Filename of the SSL private key. Overrides the global setting https-privatekey (see page 169)
https-passwordfile	string	https-passwordfile	File containing the SSL certificate passphrase. Overrides https-passwordfile (see page 170)

12.7 Direct connections

This section of rttpd.conf configures the direct (type 1) RTTP connection.

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 are:

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 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 78. for more information.

object-throttle-times

An array of throttle times in seconds. For more information see Using throttling on page 98.

Type: integer 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.

Type: integer

Default value: 60

record-max-cache

Maximum number of type 3 record data to keep.

Type: integer

Default value: 10

add-object

Adds a default object.

Syntax:

```
add-object
      name               [value]
      type               [value]
      flags              [value]
      init               [value]
      source             [value]
      throttle-times     [value]
      purge-time         [value]
      purge-period       [value]
      purge-age          [value]
      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	integer	[no default]	The RTTP object type. 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
throttle-times	float array	[no default]	An array of throttle times in seconds (same as object-throttle-times; see page 175). 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)	Number of minutes from midnight to start purging (deleting the object from the Liberator's cache). See page 90 for a description and examples of how this option can be used to configure object purging.
purge-period	integer	1440	Number of minutes between purges. See page 90 for a description and examples of how this option can be used to configure object purging.

Name	Type	Default	Description
purge-age	integer	0	A multiplier on purge-period. Defines how old an object should be before it is purged. See page 90 for a description and examples of how this option can be used to configure object purging.
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	Description
20	Directory
21	Page
22	Record
23	News headline
24	News story
27	Chat 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.

Syntax: ***object-map [name of object to be changed] [new name for object]***

Type: string

Default value: [no default]

object-precache-enable

Enables the caching of objects before they are requested.

Type: boolean

Default value: FALSE

record-type1-clear-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.

Type: boolean

Default value: FALSE

record-type2-clear-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.

Type: boolean

Default value: FALSE

record-type3-clear-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.

Type: boolean

Default value: FALSE

record-type2-hash-size	Size of the Type 2 data hashtable.	
	Type:	integer
	Default value:	65536

12.9 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 104.

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 181

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

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

12.10 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 which 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)

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-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)

12.11 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.

12.12 Fields

This section of *rttpd.conf* configures the fields contained in objects. For more information see About RTTP objects on page 78

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 95 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 95 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.*

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

12.13 DataSource peers

This section of *rttpd.conf* is used to configure the connections between the Liberator and its sources of data. 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.

Default value: 16384

datasrc-rerequest-timeout

The time in seconds that the Liberator waits for a datasource to respond when rerequesting an object that the datasrc was previously sending to Liberator. A rerequest happens when a datasrc goes down and comes back up.

Default value: 30

add-peer

Adds a DataSource peer. You can have a maximum of 63 add-peer entries in your configuration file.

Syntax:

```
add-peer
      remote-id               [value]
      remote-name             [value]
      remote-flags            [value]
      remote-type             [value]
      local-id                [value]
      local-name              [value]
      local-flags             [value]
      local-type              [value]
      addr                    [value]
      port                    [value]
      queue-size              [value]
      queue-delay             [value]
      obj-hash-size           [value]
      ssl                     [value]
      request-timeout         [value]
      label                   [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 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.

Name	Type	Default	Description
remote-type	integer	0	DataSource peer type. Gets overridden by the startup packet when the peer connection is made. Possible values are "none" (0), "active" (1) or "contrib" (2).
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.
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.
local-type	integer	0	Data source type sent to the connecting peer. Possible values are "none" (0), "active" (1) or "contrib" (2).
addr	array of strings	[no default]	Space-separated list of addresses to connect to if making the connection and not listening/ accepting the connection.*

Name	Type	Default	Description
port	array of integers	[no default]	Space-separated list of ports to connect to if making the connection and not listening/ accepting the connection.*
queue-size	integer	50	Message queue size.
queue-delay	float	0.1	Message queue delay in seconds.
obj-hash-size	integer	datasrc-default-obj-hash-size	Number of entries in active object hashtable.
ssl	boolean	False	Determines whether this connection should be made using SSL. For more information on SSL connections, see Making SSL connections with DataSources on page 124
request-timeout	float	10	Time in seconds that the Liberator will wait for this DataSource peer to answer a request. Overrides the global request timeout option.
heartbeat-time	integer	[disabled]	Time in seconds between DataSource heartbeats.
heartbeat-slack-time	integer	2	Time in seconds after a heartbeat has not been received before disconnecting and trying to reconnect.
label	string	peer[int]	There must be a label set for each label used in the **Data services** section.

Note: *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).*

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 server-authmode and client-authmode flags on page 197. 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 server-authmode and client-authmode flags on page 197. 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 170).
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.ssl.pass

12.14 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

12.15 Data services

This section of *rttpd.conf* configures how Liberator connects to its data sources.

New in version 4.0

Version 4.0 of the Liberator incorporates improved source mapping over previous versions, with the following key features.

- More intuitive configuration
- More clearly defined handling of objects recieving data from multiple sources at the same time
- Ability to define both failover and round-robin style mappings with priorities

A Source Mapping definition is called a Service. A service comprises a name, one or more subject patterns to match and one or more source groups.

The service name

This is an indentifier 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, labels are tried on a round robin basis.

Timeouts

There are several timeouts associated with Data Services. By default when an object is requested, if after 10 seconds there has been no response, the request will be cancelled and the user informed. This is the *service-request-timeout* and applies to the whole object request. It is also possible to timeout requests to individual sources, to allow the request to move on to another source. This is the *source-request-timeout*, which by default is not turned on. The Source timeout is only useful when using a priority group with multiple sources within it as it allows the system to try more than one source before giving up. These timeouts are only relevant when a source does not respond to a request, when the timeout occurs a discard message is sent to the source so it is aware that the request is cancelled.

service-request-timeout

Global request timeout for all services.

Type:	float
Default value:	10

source-request-timeout

Global request timeout for all sources.

Type:	float
Default value:	-1

cleanup-stale-timeout

Time in seconds after which stale objects will be deleted from the Liberator cache. .

Type:	float
Default value:	0 (disabled)

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. 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

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]
      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 configures the timeout for all requests for this service. Should no response be received from peers within this time, the object will be assumed to be not available.

The default value of -1 means that requests will never timeout.

Type:	float
Default value:	-1.000000

exclude-pattern

Patterns to exclude.

Type:	function
Default value:	none

include-pattern

Patterns to include.

Type:	function
Default value:	none

add-source-group

Add a source group.

Type:	boolean
Default value:	false

required

This sets whether or not this source going down generates a status stale message or a status info message.

Type:	boolean
Default value:	false

retry-time

This sets the amount of time in seconds after finding that all the labels in a group are down, before trying to connect to that group again.

Type:	float
Default value:	30

add-priority

Start a priority group.

Type:	string array
Default value:	none

label

Peer labels.

Type:	string array
Default value:	none

Below is an example section of *rttpd.conf* illustrating data services:

```
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 was defined:

```
add-data-service
      service-name default
      include-pattern        ^/
      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 DataSources at once.

Conversion

Pre-version 4.0 source mapping should be converted to version 4.0 data services in the following manner:

All peers should have a label defined in the add-peer configuration section, for these example conversion, 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
```

12.16 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

12.17 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 142

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: 2

add-thread

Configures the options for each 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.

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

12.18 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

12.19 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 104

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]
end-sigkey
```

The options in this entry are:

Name	Type	Default	Description
key-id	string	[no default]	The user name for this signature key. Must be the same as the siguser parameter of an add-user entry in an Auth Module configuration file (see add-user for cfgauth on page 221).
timeout	float	600	How long a generated signature is valid for, in seconds. Overrides signature-validtime (see page 217).
keyfile	string	[no default]	Filename of public key.

12.20 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]

12.21 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

12.22 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 223).
licenses	integer	1	The number of licences 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 109). 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 109for 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

12.23 Licencing

UUPP

The license file can be found in the *etc* directory of the root of your Liberator installation.

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

12.24 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

12.25 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.
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

12.26 Monitoring configuration

This configuration should be added to java.conf as in the following example

```
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
```

monitor-plugin

Loads the JMX monitoring module into the Liberator.

syntax: ***monitor-plugin jmx***

add-monuser

Specifies the credentials to allow a JMX enabled client application to log into the Liberator.

syntax:

```
add-monuser
    user     admin
    pass     admin
    addr     127.0.0.1
end-monuser
```

The options for this entry are:

Name	Type	Default	Description
user	string	[no default]	The username that the client application will use to log in to the Liberator.
pass	string	[no default]	The password that the client application will use to log in to the Liberator.
addr	string	Allow all addresses	Specify that the Liberator should only accept monitoring login requests from a certain IP address. Only ONE address may be specified per *add-monuser* block. If multiple addresses are required then multiple *add-monuser* blocks may be defined.

Example:

```
add-monuser
      user  admin
      pass  admin
      addr  127.0.0.1
end-monuser
```

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. 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: Debug Levels and Messages on page 241 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

12.27 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

12.28 Container configuration

container-request-timeout

TImeout in seconds for broadcast container objects

Type:	float
Default value:	5.0

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 arguments
OPEN	New session opened. Client has connected.	[none]
CLOSE	Session closed.	Reason for session closing: 1 Logout from client 2 Lost type 1 connection 4 Lost type 2 connection 8 Lost type 3 connection 16 Server kicked out as couldn't write 32 Server kicked out as couldn't write whole packets 64 Reconnected on new session, so this one closed 128 Timeout after lost connection 256 Timeout due to not logging in successfully
LOST	Session connection lost.	Reason for loss of connection: arguments 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: -1 General denial -2 Another general denial -3 Invalid username -4 Invalid password -5 Invalid IP address -6 Account expired -7 User licence exceeded -8 Site licence exceeded -9 Auth error (module did something unexpected) -91 to -100 User defined reasons
LOGOUT_OK	Session logged out.	[none]

Table 13-1: Session log messages

Session log format

All session log messages have the same format:

TIMESTAMP IP-ADDRESS MSG-TYPE USERNAME 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. For others this field will be 0.

Example:

```
2005/08/15-04:55:54.101 +0100: 192.168.201.210 LOGIN_OK maggie 1001RK 0
2005/08/15-05:05:59.301 +0100: 192.168.201.210 LOGOUT_OK maggie 1001RK 0
2005/08/15-05:05:59.201 +0100: 192.168.201.210 CLOSE - 1001RK 1
2005/08/15-05:07:01.201 +0100: 192.168.201.210 OPEN - 1007zX 0
2005/08/15-05:07:01.201 +0100: 192.168.201.210 LOGIN_OK maggie 1007zX 0
2005/08/15-05:14:18.201 +0100: 192.168.201.104 LOST - 0006IW 4
2005/08/15-05:14:18.201 +0100: 192.168.201.104 OPEN - 000346 0
2005/08/15-05:14:18.201 +0100: 192.168.201.104 RECONNECT_OK livedemos
000346 0 0006IW
2005/08/15-05:14:18.201 +0100: 192.168.201.104 CLOSE - 0006IW 64
2005/08/15-05:17:06.201 +0100: 192.168.201.210 LOGOUT_OK maggie 1007zX 0
2005/08/15-05:17:06.201 +0100: 192.168.201.210 CLOSE - 1007zX 1
2005/08/15-05:18:08.201 +0100: 192.168.201.210 OPEN - 00002C 0
2005/08/15-05:18:08.201 +0100: 192.168.201.210 LOGIN_OK maggie 00002C 0
2005/08/15-05:21:08.201 +0100: 192.168.201.121 LOST - 0004dG 4
2005/08/15-05:21:08.201 +0100: 192.168.201.121 OPEN - 0003L- 0
2005/08/15-05:21:09.201 +0100: 192.168.201.121 RECONNECT_OK livedemos
0003L- 0 0004dG
2005/08/15-05:21:09.201 +0100: 192.168.201.121 CLOSE - 0004dG 64
2005/08/15-05:28:14.201 +0100: 192.168.201.210 LOGOUT_OK maggie 00002C 0
2005/08/15-05:28:14.201 +0100: 192.168.201.210 CLOSE - 00002C 1
2005/08/15-05:29:15.201 +0100: 192.168.201.210 OPEN - 1003gD 0
2005/08/15-05:29:15.201 +0100: 192.168.201.210 LOGIN_OK maggie 1003gD 0
2005/08/15-05:39:21.201 +0100: 192.168.201.210 LOGOUT_OK maggie 1003gD 0
2005/08/15-05:39:21.201 +0100: 192.168.201.210 CLOSE - 1003gD 1
2005/08/15-05:40:22.201 +0100: 192.168.201.210 OPEN - 1007-- 0
2005/08/15-05:40:22.201 +0100: 192.168.201.210 LOGIN_OK maggie 1007-- 0
2005/08/15-05:50:28.201 +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 session log messages have the same format:

TIMESTAMP IP-ADDRESS USERNAME SESSION-D MESSAGE

Example:

```
2005/08/21-15:04:42.201 +0100: 192.168.201.16 - 2ADgSL "2ADgSL LOGIN 000000
CLIENT/CLEAR RTTP/2.0 demouser demopass"
2005/08/21-15:04:46.201 +0100: 192.168.201.16 demouser 2ADgSL "2ADgSL LOGOUT"
2005/08/15-04:11:27.201 +0100: 192.168.201.210 maggie 0004p_ "0004p_ REQUEST /
EQUITIES/MSFT"
2005/08/15-04:11:27.201 +0100: 192.168.201.210 maggie 0004p_ "0004p_ REQUEST /
FX/GBP"
2005/08/15-04:11:28.201 +0100: 192.168.201.210 maggie 0004p_ "0004p_ REQUEST /
IPE/IPE/HB"
2005/08/15-04:14:17.201 +0100: 192.168.201.104 - 0006IW "0006IW NOOP"
2005/08/15-04:14:17.201 +0100: 192.168.201.104 - 0006IW "0006IW LOGIN 0003U-
CLIENT/CLEAR RTTP/0.2 bobby11 mypassw11"
2005/08/15-04:21:08.201 +0100: 192.168.201.121 - 0004dG "0004dG NOOP"
2005/08/15-04:21:08.201 +0100: 192.168.201.121 - 0004dG "0004dG LOGIN 00077o
CLIENT/CLEAR RTTP/0.2 bobby11 mypassw11"
2005/08/15-04:21:32.201 +0100: 192.168.201.210 maggie 0004p_ "0004p_ LOGOUT "
2005/08/15-04:22:34.201 +0100: 192.168.201.210 - 1002i0 "1002i0 LOGIN 000000
CLIENT/CLEAR RTTP/0.2 maggie thatcher"
2005/08/15-04:22:34.201 +0100: 192.168.201.210 maggie 1002i0 "1002i0 REQUEST /
EQUITIES/MSFT"
2005/08/15-04:22:34.201 +0100: 192.168.201.210 maggie 1002i0 "1002i0 REQUEST /
FX/GBP"
2005/08/15-04:22:34.201 +0100: 192.168.201.210 maggie 1002i0 "1002i0 REQUEST /
IPE/IPE/HB"
2005/08/15-04:32:39.201 +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 or DISCARD.

Example:

```
2005/08/15-08:27:06.201 +0100: 0000Fw REQUEST /HBT/snpsrc-frac
2005/08/15-08:27:07.201 +0100: 0000Fw REQUEST /HBT/snpsrc-deci
2005/08/15-08:27:07.201 +0100: 0000Fw REQUEST /IPE/IPE/HB
2005/08/15-08:37:10.201 +0100: 0000Fw DISCARD /HBT/snpsrc-frac
2005/08/15-08:37:10.201 +0100: 0000Fw DISCARD /HBT/snpsrc-deci
2005/08/21-15:04:42.201 +0100: 2ADgSL DISCARD /DEMO/AMZN
2005/08/21-15:04:42.201 +0100: 2ADgSL DISCARD /DEMO/CSCO
2005/08/21-15:04:42.201 +0100: 2ADgSL DISCARD /DEMO/EBAY
2005/08/21-15:04:42.201 +0100: 2ADgSL DISCARD /DEMO/FX/DEM
2005/08/21-15:04:42.201 +0100: 2ADgSL DISCARD /DEMO/FX/EUR
2005/08/21-15:04:42.201 +0100: 2ADgSL DISCARD /DEMO/FX/GBP
2005/08/21-15:04:46.201 +0100: 2ADgSL DISCARD /DEMO/AMZN
2005/08/21-15:04:46.201 +0100: 2ADgSL DISCARD /DEMO/CSCO
2005/08/21-15:04:46.201 +0100: 2ADgSL DISCARD /DEMO/EBAY
2005/08/21-15:04:46.201 +0100: 2ADgSL DISCARD /DEMO/FX/DEM
2005/08/21-15:04:46.201 +0100: 2ADgSL DISCARD /DEMO/FX/EUR
2005/08/21-15:04:46.201 +0100: 2ADgSL DISCARD /DEMO/FX/GBP
```

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 a two datasource applications connect. A PEERINFO can also be sent at other times indicating the status of a DataSource.	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 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.	NUM-OBJECTS [OBJECT-NAMES]
SUBJDSC	This message shows when a discard message is sent to a DataSource, informing a datasource 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 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.	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 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 the message is being sent to, and with received messages it is the ID of the DataSource the message is from.

Packet log examples

PEERINFO messages:

```
2005/08/21-15:04:03.201 +0100: 127.0.0.1 < PEERINFO 1 demosrc-
bigsun 0
2005/08/21-15:04:03.201 +0100: 127.0.0.1 > PEERINFO 0 rttpd-bigsun
0
2005/08/21-15:37:36.201 +0100: 127.0.0.1 < PEERINFO 3 testsrc-
mtserv1 6 Warning
```

DATAUPDATE messages:

```
2005/08/21-15:04:03.201 +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
2005/08/21-15:04:03.201 +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
2005/08/21-15:04:03.201 +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:

```
2005/08/21-15:22:37.201 +0100: 127.0.0.1 > SUBJREQ 3 2 /I/VOD.L /I/
ANL.L
```

SUBJDSC messages:

```
2005/08/21-15:23:45.201 +0100: 127.0.0.1 > SUBJDSC 3 2 /I/VOD.L /I/
ANL.L
```

DOWN messages:

```
2005/08/21-15:24:09.201 +0100: 127.0.0.1 < DOWN 3
```

NODATA messages:

```
2005/08/21-15:28:53.201 +0100: 127.0.0.1 < NODATA 3 /I/VOD.L 1
2005/08/21-15:28:53.201 +0100: 127.0.0.1 < NODATA 3 /I/ANL.L 1
```

STATUS messages:

```
2005/08/21-15:40:48.201 +0100: 127.0.0.1 < STATUS /I/VOD.L 0x0001 8
Data may be stale
2005/08/21-15:40:53.201 +0100: 127.0.0.1 < STATUS /I/VOD.L 0x0104 6
Data may be ok now
2005/08/21-15:40:58.201 +0100: 127.0.0.1 < STATUS /I/VOD.L 0x0004 4
Data is ok now
2005/08/21-15:41:03.201 +0100: 127.0.0.1 < STATUS /I/VOD.L 0x0000 9
Everything is fine
2005/08/21-15:41:20.201 +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 - - [21/Aug/2005:15:04:34 +0100] "GET /demos/rtml/
rtml.html HTTP/1.1" 200 2192
192.168.201.16 - - [21/Aug/2005:15:04:34 +0100] "GET /demos/rtml/
common.css HTTP/1.1" 200 522
192.168.201.16 - - [21/Aug/2005:15:04:34 +0100] "GET /rtml/ HTTP/
1.1" 200 9570
192.168.201.16 - - [21/Aug/2005:15:04:35 +0100] "GET /rtml/lib/
formatting.js HTTP/1.1" 200 3769
192.168.201.16 - - [21/Aug/2005:15:04:35 +0100] "GET /rtml/lib/
stale.js HTTP/1.1" 200 1167
192.168.201.16 - - [21/Aug/2005:15:04:35 +0100] "GET /rtml/
w3clibrary.js HTTP/1.1"200 3122
```

13.6 HTTP error log

Example:

```
[01/Dec/2004:11:47:09.123 +0000] [error] [client 127.0.0.1] File
does not exist: /opt/Liberator/htdocs/notfound
```

13.7 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 and connections to datasources.

Example:

```
2005/08/25-13:52:17.420 +0100: CONFIG: UDP Message port not configured
2005/08/25-13:52:17.420 +0100: NOTIFY: Attempting to change debug-level to INFO
2005/08/25-13:52:17.420 +0100: NOTIFY: Successfully changed debug-level to INFO
2005/08/25-13:52:17.420 +0100: NOTIFY: Liberator/4.0.0 starting
2005/08/25-13:52:17.420 +0100: NOTIFY: Logging to /opt/caplin/Liberator/var
2005/08/25-13:52:17.421 +0100: NOTIFY: Licence will expire on Wed Dec 28 00:00:00 2005
2005/08/25-13:52:17.421 +0100: NOTIFY: system-max-files set to 1024
2005/08/25-13:52:17.422 +0100: INFO: Loaded auth module <openauth>
2005/08/25-13:52:17.423 +0100: INFO: Next cycle of UUPP database (/opt/caplin/Liberator/
users/uupp-rttpd.db) scheduled for Wed Aug 31 23:59:59 2005
2005/08/25-13:52:17.426 +0100: INFO: Read in 101 unique users from database
2005/08/25-13:52:17.455 +0100: INFO: Created object /(220) [0x8c37578/0]
2005/08/25-13:52:17.455 +0100: NOTIFY: Field CONTRIB_USER not known, setting unique user
fieldnumber to 20000
2005/08/25-13:52:17.459 +0100: INFO: 2 CPUs CONFIGURED
2005/08/25-13:52:17.459 +0100: INFO: 2 CPUs ONLINE
2005/08/25-13:52:17.493 +0100: INFO: Created object /SYSTEM(200) [0x9f41478/1]
2005/08/25-13:52:17.493 +0100: INFO: Created object /SYSTEM/NODE-0(200) [0x9f41628/2]
2005/08/25-13:52:17.493 +0100: INFO: Created object /SYSTEM/NODE-0/INFO(200) [0x9f417d0/3]
2005/08/25-13:52:17.493 +0100: INFO: Changing type of /SYSTEM/NODE-0/INFO from 200 to 201
[0x9f417d0/3]
2005/08/25-13:52:17.493 +0100: INFO: Created object /SYSTEM/INFO(200) [0x9f41a58/4]
2005/08/25-13:52:17.493 +0100: INFO: Changing type of /SYSTEM/INFO from 200 to 201
[0x9f41a58/4]
2005/08/25-13:52:17.493 +0100: INFO: Created object /SYSTEM/LICENSE(200) [0x9f41d10/5]
2005/08/25-13:52:17.493 +0100: INFO: Changing type of /SYSTEM/LICENSE from 200 to 201
[0x9f41d10/5]
2005/08/25-13:52:17.493 +0100: INFO: Created object /SYSTEM/NODE-0/SRC-0(200) [0x9f41fd8/6]
2005/08/25-13:52:17.493 +0100: INFO: Changing type of /SYSTEM/NODE-0/SRC-0 from 200 to 201
[0x9f41fd8/6]
2005/08/25-13:52:17.493 +0100: INFO: Created object /SYSTEM/NODE-0/SRC-1(200) [0x9f42248/7]
2005/08/25-13:52:17.493 +0100: INFO: Changing type of /SYSTEM/NODE-0/SRC-1 from 200 to 201
[0x9f42248/7]
2005/08/25-13:52:17.493 +0100: INFO: Created object /SYSTEM/NODE-0/SRC-2(200) [0x9f425a8/8]
2005/08/25-13:52:17.493 +0100: INFO: Changing type of /SYSTEM/NODE-0/SRC-2 from 200 to 201
[0x9f425a8/8]
```

```
2005/08/25-13:52:17.493 +0100: INFO: Created object /SYSTEM/NODE-0/SRC-3(200) [0x9f42890/9]
2005/08/25-13:52:17.493 +0100: INFO: Changing type of /SYSTEM/NODE-0/SRC-3 from 200 to 201
[0x9f42890/9]
2005/08/25-13:52:17.494 +0100: INFO: Created object /SYSTEM/NODE-0/SRC-4(200) [0x9f42ba0/
10]
2005/08/25-13:52:17.494 +0100: INFO: Changing type of /SYSTEM/NODE-0/SRC-4 from 200 to 201
[0x9f42ba0/10]
2005/08/25-13:52:17.494 +0100: INFO: Created object /SYSTEM/NODE-0/SERVICE(200) [0x9f42eb0/
11]
2005/08/25-13:52:17.494 +0100: INFO: Created object /SYSTEM/NODE-0/SERVICE/svc1(200)
[0x9f43048/12]
2005/08/25-13:52:17.494 +0100: INFO: Changing type of /SYSTEM/NODE-0/SERVICE/svc1 from 200
to 201 [0x9f43048/12]
2005/08/25-13:52:17.494 +0100: INFO: Created object /MT1(200) [0x9f43428/13]
2005/08/25-13:52:17.494 +0100: INFO: Changing type of /MT1 from 200 to 222 [0x9f43428/13]
2005/08/25-13:52:23.675 +0100: INFO: Accepted connection from 127.0.0.1 42371
2005/08/25-13:52:23.676 +0100: NOTIFY: Accepting peer id 1 on 127.0.0.1 42371
2005/08/25-13:52:33.642 +0100: INFO: Created object /SYSTEM/USERS(200) [0x9f45b80/14]
2005/08/25-13:52:33.642 +0100: INFO: Created object /SYSTEM/USERS/demouser-0(200)
[0x9f45ce8/15]
2005/08/25-13:52:33.642 +0100: INFO: Changing type of /SYSTEM/USERS/demouser-0 from 200 to
202 [0x9f45ce8/15]
2005/08/25-13:52:37.114 +0100: INFO: Removed object /SYSTEM/USERS/demouser-0(202)
[0x9f45ce8/15]
2005/08/25-13:52:37.114 +0100: INFO: Adding to batch-delete timer for /SYSTEM/USERS/
demouser-0(202) [0x9f45ce8/15]
2005/08/25-13:52:39.619 +0100: NOTIFY: Lost connection to peer 1 on 127.0.0.1 42371
2005/08/25-13:52:42.616 +0100: INFO: Deleted object /SYSTEM/USERS/demouser-0(202)
[0x9f45ce8/15]
2005/08/25-13:52:42.863 +0100: NOTIFY: Received signal SIGINT (2)
2005/08/25-13:52:42.864 +0100: NOTIFY: Shutting down - SIGNAL (6)
```

14 Appendix C: Debug Levels and Messages

Please refer to the log message reference on Caplin's Client Portal available at demo.caplin.com/clientportal for a list of Liberator and DataSource log messages and their explanations.

These log message references include the internal message label, the message severity (in decreasing order - Error, Critical, Notify, Warn, Info, Debug, Config), the message that is written to the log file, and an explanation of that message.

15 Appendix D: 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   XXXXXXXXXXXXXXXXXXXX
      company     Caplin Systems
      hostname    hostname1
      max-users   500
      expire      2005030330
      https       1
      module      cfgauth auth
      module      openauth auth
      module      xmlauth auth
      module      javaauth auth
end-license
```

- Ensure there is a Sun 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-id          authenticator
    class-name        examples/OpenAuthenticator
    classpath         %r/lib/java/javaauth-examples.jar
end-javaclass
```

Please see Java.conf configuration on page 224 for details on the above parameters.

- Start the Liberator.

16 Appendix E: Performance benchmark

16.1 Benchmark test methodology

Liberator is started on the target test machine with a pre-configured list of 100 symbols that will be subscribed to by the test clients. These symbols will initially have no field values but act as containers which can be added to the users' watchlists.

The RTTP clients are then started up on the client machine(s). The number of client sessions started is the only metric that changes from test to test. Each client subscribes randomly to 20 of the 100 symbols.

Once all the clients have been connected, logged in and subscriptions acknowledged, a lightweight DataSource program is started on the target server machine that provides continuous updates to the 100 symbols. The update rate is increased every 100 seconds throughout the test with the CPU utilisation logged as the average over each of these 100 second intervals.

A typical message consists of 5 fields or 5 updates.

Hardware

Server

Model:	SUN 420R
Processors:	4 x 450MHz, 4Mb Cache
Random Access Memory:	2Gb
Operating System:	Solaris 8

Client 1

Model:	SUN 220R
Processors:	2 x 450MHz, 4Mb Cache
Random Access Memory:	2Gb
Operating System:	Solaris 8

Client 2

Model	DELL/Intel
Processors:	Xeon 4 x 550MHz, 2Mb Cache
Random Access Memory:	2Gb
Operating System:	RedHat Linux 2.4.2

16.2 Message content

The update message used for all these tests was a standard record object. An example of a message actually used in the tests is shown below

```
6c0R0001 4=/B/OBJ_00 5=1732 6=1000 7=1000 8=0032
```

16.3 Results

The following headline results represent achievable performance with the system running at no more than 50% utilisation in each case.

Note: *These results were obtained using Liberator running on a Solaris platform.*

At 50% CPU utilisation

- 5,000 concurrent users can each receive 250 updates per second giving 1.25 million user updates per second.
- 7,500 concurrent users can each receive 116 updates per second giving 870,000 user updates per second.
- 10,000 concurrent users can each receive 110 updates per second giving 1.1 million user updates per second.

At 36% CPU utilisation

- 2,500 concurrent users can each receive 500 updates per second giving 1.25 million user updates per second.

A

active request 44
active-discard-timeout 122, 175
add-authdir 24, 105, 150, 164
add-cluster-node 41, 186
add-field 95, 187
add-javaclass 225
add-log-cycle 130, 159
add-newscodes 98, 214
add-object 91, 92, 93, 175
add-peer 112, 113, 116, 123, 152, 190
add-sigkey 110, 217, 222
add-source-mapping 210
add-thread 84, 90, 212
add-type-mapping 91, 178
add-user 109, 218
add-virtual-host 86, 172
application-name 154, 155
application-root 154
architecture 18
    internal 18
    system 21
Auth Module 24, 104, 107, 181, 222
auth_new_user 106
authentication 104
auth-login-timeout 106, 182
auth-map-timeout 107, 182
auth-moddir 104, 181
auth-module 105, 108, 181
authorisation 104

B

broadcast 29, 43, 112
buf-cache-size 102, 211
buf-elem-len 102, 142, 211
buffering 101, 142
burst-max 101, 142, 211
burst-min 101, 142
bursts 101, 142

C

cache 28, 82, 91
catch-crash 131, 154
cfgauth 108, 218, 221
cfgpass 221
chat 79
cipher 195, 196
cluster-addr 185
cluster-cache-request-objects 185
cluster-cache-source-routing 185, 186
cluster-index 42, 185
Clustering 23
clustering 185
cluster-port 185

D

data 25
    sources 25
    Type 1 81, 115, 179
    Type 2 81, 96, 115, 143, 179
    Type 3 82, 97, 115, 179
Data health checking 78
DataSource peer 111
    configuration parameters 189
    connecting to 113
    failover 113
    Liberator as 112
    reconnecting 115
    replaying data from 122, 124
    requesting updates from 116
DataSource, protocol 80, 235
datasrc_id 116
datasrc-auto-replay 123, 199
datasrc-auto-replay-days 123, 199
datasrc-auto-replay-files 123, 199
datasrc-default-obj-hash-size 118, 190, 193
datasrc-id 112, 189, 192
datasrc-interface 113, 189
datasrc-name 112, 189, 192

datasrc-pkt-log 128, 189
datasrc-port 113, 190, 194
datasrc-reject-new-peers 115, 189
datasrc-sslport 113, 151, 190
default-type 91, 178
definition of Liberator 17
direct-interface 90, 174
direct-max-line-length 145, 166
directory 79
direct-port 90, 174
direct-tcp-nodelay-off 144, 212
discarding 233, 235

E

encrypted-passwords 109, 221
event log 239
event-log 128, 154

F

failover 236
fields 80
fields.conf 235
fields-file 95, 187
file descriptors 146, 148

H

hashtables 143
header 145
heartbeat 136, 184
heartbeat-slack-time 115, 195
heartbeat-time 115, 195
http-access-log 128, 164, 238
http-connection-cookie-enable 85
http-connection-cookie-expires 85
http-def-content-type 163
http-err-content-type 164
http-error-log 128, 164
http-idx-content-type 164
http-indexfile 163
http-interface 84, 162
http-keepalive-max 84, 162
http-keepalive-timeout 84, 163
http-max-body-length 145, 166
http-max-header-line-length 145, 166
http-max-header-lines 145, 166
http-max-request-length 145, 166
http-port 84, 162
http-rttp-content-type 163
https-certificate 87, 169, 172
https-cipher-list 170
https-enable 85, 169
https-interface 86, 169
https-passwordfile 86, 88, 170, 173
https-port 87, 169
https-privatekey 87, 169, 173
http-tcp-nodelay-off 144, 213
http-wwwroot 162, 172

I

improving performance 142
include-file 155
installation
    licence 38
    secure 38
IP address 25

J

Java Virtual Machine 224
java-file 224
JVM See Java Virtual Machine
jvm-global-classpath 225
jvm-options 226

K

key 24, 82
KeyMaster Integration 24

L

Liberator 17
- as DataSource peer 112
- features 22

licence 38, 231
licence.conf 38
license-file 155
Linux 23, 30
logcat 127
log-cycle-offset 130, 157
log-cycle-period 129, 156, 157, 161
log-cycle-suffix 130, 157
log-cycle-time 129, 156
log-dir 128, 156
logging 127, 156
log-maxsize 130, 156

M

max-user-limit 106, 181
max-user-ok 106, 181
max-user-warn 106, 181

N

news 79, 102
- configuring 214
- replaying 124

newscode-exceptions 98, 214
newscode-hash-size 98, 215
newscode-max-length 98, 214
newscodes-valid-chars 98, 215
news-datetime-format 102, 215
newsitems-max 102, 214, 215
newsitems-saved 102, 214
news-log 124, 216
news-purge-days 215
news-purge-time 215
news-replay 124, 216
news-replay-days 124, 216
news-replay-files 124, 216
noauth-reconnect 107, 183

O

object-hash-size 144, 210
object-log 129, 183
object-map 91, 179
object-precache-enable 179
objects 78
object-throttle-default-level 101, 175
object-throttle-off 101, 175
object-throttle-times 94, 100, 175, 177
openauth 108, 220
OpenSSL 16, 87, 88, 89, 170, 171
output-queue-size 102, 211

P

packet log 234, 235, 236
page 79
parameter 80
peer-reconnect 116
permissioning 24
Persistent virtual connection 78
pid-filename 155
port 25
priority 186
purge-age 92, 93
purge-period 92, 93
purge-time 92
purging 28, 83, 91

Q

queue 27

R

read-access 108, 220
record 79, 80

record-max-cache 97, 175
record-type1-clear-on-failover 115, 179
record-type2-clear-on-failover 115, 179
record-type2-hash-size 143
record-type2-hashsize 97, 180
record-type3-clear-on-failover 115
replaying data 122, 124, 199
requested-fields-only 144, 188
requesting 26, 79, 232, 233, 235
request-log 128, 183
request-timeout 193
RTML 135
RTSL 135
RTTP 77
    definition 77
    features 77
    fields 80
    objects 78
runtime-user 154

S

SDK 224
security 145
service 121, 201
service-request-timeout 121
session log 230, 231, 232
session-hash-size 143, 210
session-heartbeat 137, 184
session-log 128, 183
session-reconnect-timeout 107, 184
session-timeout 107, 183
signature 24, 222
signature-hashsize 110, 217
signature-validtime 109, 217, 218
SL4B 25, 149
Solaris 26
source 122
source-request-timeout 201
sources, data 25
ssl-engine-flags 89, 172
ssl-engine-id 88, 171
ssl-random-seed 87, 170
start-ssl 151
startup 27, 28
symbol 80, 82
system-max-files 146, 154

T

TCP nodelay 144
threads 142
threads-num 143, 212
throttling 175
timestamp 231, 232, 233, 236
tunnelling 77
type1-host 185
type1-port 186
type2-url 186

U

UDP messages 115, 219
udp-interface 116, 219
udp-port 116, 219
user-hash-size 106, 143, 210
users, concurrent 17

W

web site, Liberator 25
Windows 26, 40
write-access 108, 220
XML 24
XMLauth 107, 181

X-Y-Z

175

This publication is provided "as is" without warranty of any kind, either express or implied, including, but not limited to, warranties of merchantability, fitness for a particular purpose, or non-infringement.

This publication could include technical inaccuracies or typographical errors and is subject to change without notice. Changes are periodically added to the information herein; these changes will be incorporated in new editions of this publication. Caplin Systems Limited may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time.

Contact Us

Triton Court
14 Finsbury Square
London EC2A 1BR
UK
Telephone: *+44 20 7826 9600*
Fax: *+44 20 7826 9610*

www.caplin.com

info@caplin.com