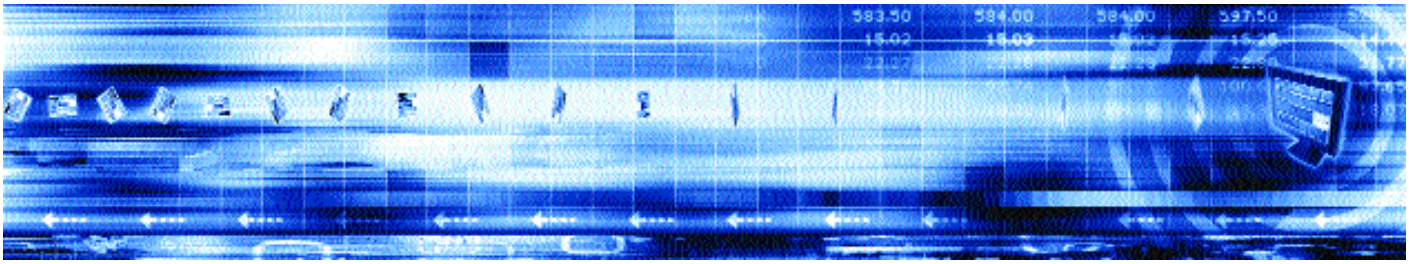

CAPLIN DATASOURCE FOR RMDS 4.0

Administration Guide

March 2005

1 Preface

1.1 What this document contains

This document describes how to configure and operate Caplin's DataSource for RMDS product, which is used to request data from Reuters' RMDS market data system and forward it to applications that interface with Caplin's DataSource protocol.

1.2 Who should read this document

This document is intended for systems administrators who need to configure and manage DataSource for RMDS, and for developers tasked with due diligence and software evaluations.

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

API	Application Program Interface, the method by which a programmer writing an application program can make requests of the operating system or another application.
Caplin Liberator	A suite of software applications and components for publishing real-time information using RTTP protocol over IP networks. Caplin Liberator collects real-time data from one or more sources and redistributes it to suitably permissioned RTTP subscribers.
DataSource	A network protocol that enables most Caplin and RTTP-related products to communicate with each other.
DataSource API	A transmission path which links the architectural elements that use the DataSource protocol.
DataSource Peer	Any remote application that can connect to the DataSource API. All Caplin Liberators are DataSource peers.
DataSource SDK	The DataSource SDK (Software Development Kit) is a library of functions that enables developers to create their own DataSource peer.
Field	A data element of an object, identified by a field name or field number and with a data value of a string.
GMT	Greenwich Mean Time.
HTTP	Hypertext Transfer Protocol, the set of rules for exchanging files on the World Wide Web.
HTTPS	Otherwise known as Secure Sockets Layer or, HTTPS is a version of HTTP used for managing the security of a message transmission on the Web. HTTPS uses the public-and-private key encryption system, which also includes the use of a digital certificate.
Image	A complete set of data currently held on a data source.

Object	There are several types of RTTP object: Directory, Page, Record, News headline, News story and Chat object. Each type is identified by a three digit number.
Page	A fixed format text array used by older financial systems, descended from dumb terminal displays of 80 x 25 characters.
Parameter	A data element of an object, identified by a field name or field number and with a data value of a string. Same as "field".
Protocol	A standard that defines the way in which data is passed between two or more pieces of computer equipment over a telephone line or other communications link. Two pieces of equipment must be using the same protocol in order to communicate.
RFA	Reuters interface for RMDS.
RMDS	The Reuters Market Data System.
RTTP	RTTP (Real Time Text Protocol) is a web protocol developed by Caplin Systems Ltd that implements advanced real-time streaming for almost all types of textual information, including logical records, news and free-format pages.
Sink Distributor	Software responsible for responding to sink requests, maintaining communications with source services and distributing SSL messages to sink clients.
SSL	The Secure Sockets Layer (SSL) is a commonly-used protocol for managing the security of a message transmission on the Internet. SSL uses the public-and-private key encryption system, which also includes the use of a digital certificate.
SSL (2)	Source Sink Library (SSL), Reuters interface for Triarch.

Symbols	The letters used to uniquely identify a financial instrument (e.g. the symbol for Microsoft's common stock on the Nasdaq market is MSFT). Many symbol naming conventions (symbologies) exist, but each is applied consistently in each market, and one symbology is used across all North American equity markets.
TS1	"Time Series" data provided from Reuters, used to create charts. TS1 data is historical price information stored in packed-page format.
UDP	UDP (User Datagram Protocol) is a communications protocol that offers a limited amount of service when messages are exchanged between computers in a network that uses the Internet Protocol.
XML	XML (Extensible Markup Language) contains markup symbols to describe the contents of a page or file. An XML file can be processed purely as data by a program or it can be stored with similar data on another computer or displayed. XML is "extensible" because, unlike HTML, the markup symbols are unlimited and self-defining.
XML tags	The sequence of characters or other symbols that you insert at certain places in a file to indicate how the file should look when it is printed or displayed or to describe the document's logical structure. Tags that have other tags within them are called parent tags; those within are called child tags.

1.5 Feedback

Customer feedback can only improve the quality of Caplin 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.

2 Overview

2.1 Introduction

DataSource for RMDS enables Caplin Liberator, or any Caplin product connected to the DataSource API, to request data from Reuters' RMDS market data system. DataSource for RMDS can then retrieve and forward the data using the DataSource protocol.

2.2 System Requirements

DataSource for RMDS is available for either Solaris or Linux. It requires Solaris 8 or RedHat 9+.

2.3 Basic operation

Figure 2-1 gives a simple indication of how DataSource for RMDS fits into a market data distribution system.

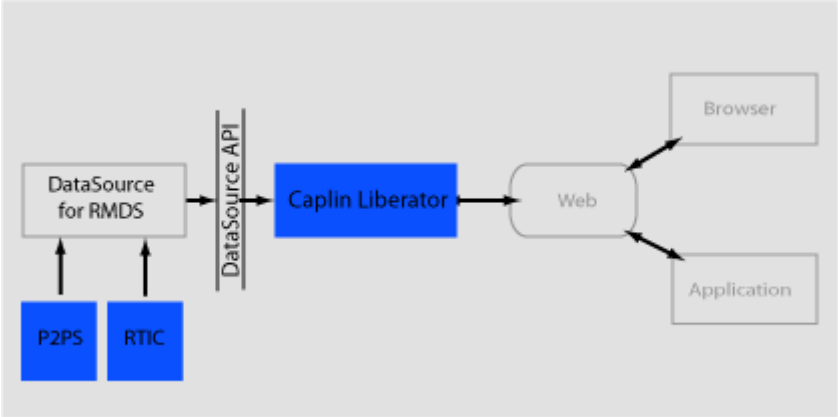

Figure 2-1: Market data distribution system incorporating DataSource for RMDS

- Caplin Liberator sends a request for data to DataSource.
- DataSource extracts the data from RMDS and forwards it to Caplin Liberator via the DataSource API.

❖ Caplin Liberator can then send the data to browsers or applications over the web.

2.4 DataSource for RMDS features

DataSource for RMDS incorporates a number of advanced features that bring greater flexibility and reliability to the way DataSource operates. These are listed below.

Active data sources

An active data source is one that will keep track of which records have been requested and send updates for those records only. Records may also be discarded; this tells DataSource that updates for this record are no longer required.

- For details on how to make DataSource for RMDS an active source, see the *local-type* option of the **add-peer** parameter (see page 28).

Configurable log cycling

Log files containing status messages and information on what data has been sent and to where can be configured to cycle based on time and/or size. Cycled logs are saved using a configurable format (for example, one for each day of the week).

- For details on how to configure log cycling, see **Logging**, starting on page 45.

Message queues

If a DataSource peer loses its connection to DataSource for RMDS, 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 the *queue-size* option of the **add-peer** parameter (see page 28).

Record name mapping

DataSource for RMDS can be configured to map record names that are passed into it into a different format. This can either be used to simply make it a valid RTTP record name, or to create a complex directory structure of records.

- For details on how to implement name mappings, see **Record Name Mapping** on page 58

Charting

DataSource for RMDS can handle Reuters charting data.

- For details on how to configure how charting data is stored, see **TS1 Charting** on page 59.

Outputting news as XML

DataSource for RMDS can be configured to write news stories to an XML file, which permits some degree of formatting on the client side.

- For more information on using XML for news, see **Receiving News from RMDS** on page 50.

UDP messages

DataSource for RMDS includes a UDP command interface that enables you to send UDP messages regarding debugging and the writing of lists of watched objects to files.

- For more information on DataSource for RMDS's UDP interface, see **Using UDP Commands** on page 61.

Active news requests

DataSource for RMDS can output news in response to a client request via RTTP as well as HTTP, and in a variety of formats, including XML files and plain text files.

- For more information on active news requests see **Selecting the output type** on page 59.

Enumerated field values

An enumerated field enables you to give a non-numeric and more meaningful name to a number: for example if an field can take the values 1, 2 or 3, you can define 1 as "London", 2 as "Paris" and 3 as "Berlin". A number of such mappings are used within the DataSource for RMDS configuration.

- For details on how to use enumerated field values, see **enum-enable** and **add-enum** on page 33.

Sending chart data direct to clients

DataSource for RMDS can output TS1 charting data on RTTP for use by client applications.

- For more information on sending chart data to clients see **Sending TS1 charting data to peers** on page 59

Connecting to Caplin Liberator

DataSource for RMDS starts in a disconnected state and will only connect to Caplin Liberator when its connection to RMDS is stable.

- For more information on connecting to Caplin Liberator and other applications see **Communicating with DataSource Peers** starting on page 25.

Automatic disconnection on failover

DataSource for RMDS will now disconnect from all its peers if a specified service goes down.

- For details on how to configure which service's failure should cause disconnection, see **rfa-priority-service on page 24**

Support for non-standard page sizes

Any service supplying pages of non-standard dimensions (80 x 25 characters) can now be identified.

For details on how to configure non-standard page services, see **add-service** on page 36.

2.5 About the data

RMDS data is transmitted as records and pages.

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.

Each field within a record has a data type that specifies the width and character range that the field can hold. Examples of these data types are Bid (the bid price), Ask (the ask price), Time (the time of the last trade in seconds) and Currency (the currency in which the price is quoted).

Records that contain financial market data are identified by a symbol. The symbol that you must use to identify a particular financial instrument depends on the symbology being used by RMDS. For example IBM.N is Reuters symbology for the real-time price of IBM ordinary shares traded on the New York Stock Exchange.

3 Getting started

3.1 Installing DataSource for RMDS—Solaris

Perform the following steps to install DataSource for RMDS:

- Unpack the kit into a suitable directory (for example */opt*) and create a link to this new directory.

Example:

```
$ cd /opt
$ uncompress /tmp/RFA-3.6.0-sparc-sun-solaris2.8.tar.Z
$ tar xf /tmp/RFA-3.6.0-sparc-sun-solaris2.8.tar
$ ln -s RFA-3.6.0 RFA
```

You should now have a directory structure something like this:

```
/opt/RFA
/opt/RFA/bin            (for binary programs)
/opt/RFA/doc            (for documents and examples)
/opt/RFA/etc            (for startup and configuration)
/opt/RFA/extra          (for additional files)
/opt/RFA/lib            (for modules)
/opt/RFA/var            (for log files)
```

- Edit the file *etc/rfasrc* and change the line `RFAsrc_ROOT` to point to the directory in which you installed DataSource for RMDS (for example */opt/RFA*).

- If you want DataSource for RMDS to start automatically on boot-up then create a link from your startup directory.

Example:

```
$ cd /etc/rc3.d
$ ln -s /opt/RFA/etc/rfasrc S99rfasrc
```

Note: *On other systems using SYSV startup scripts this process should be similar.*

The name S99rfasrc tells the system to run the script last. The rfasrc part of the name must match the start script in etc.

3.2 Installing DataSource for RMDS—Linux

Perform the following steps to install DataSource for RMDS:

- Unpack the kit into a suitable directory (for example */usr/local*) and create a link to this new directory.

Example:

```
$ cd /usr/local
$ tar xzf /tmp/RFAsrc-4.0.0-6-i686-pc-linux-gnu.tar.gz
$ ln -s RFA-3.6.0 RFA
```

You should now have a directory structure something like this:

```
/usr/local/RFA
/usr/local/RFA/bin      (for binary programs)
/usr/local/RFA/doc      (for documents and examples)
/usr/local/RFA/etc      (for startup and configuration)
/usr/local/RFA/extra    (for additional files)
/usr/local/RFA/lib      (for modules)
/usr/local/RFA/var      (for log files)
```

- Edit the file *etc/rfasrc* and change the line `RFAsrc_ROOT` to point to the directory in which you installed DataSource for RMDS (for example */usr/local/RFA*).

- If you want DataSource for RMDS to start automatically on boot-up then create a link from your startup directory.

Example:

```
$ cd /etc/rc3.d
$ ln -s /usr/local/RFA/etc/rfasrc S99rfasrc
```

Note: *On other systems using SYSV startup scripts this process should be similar.*

The name S99rfasrc tells the system to run the script last. The rfasrc part of the name must match the application binary.

3.3 Initial configuration

DataSource is configured by editing the entries in two plain text configuration files. The first configuration file is called *rfasrc.conf*, which can be found in the DataSource *etc* directory (as specified at installation—see **Installing DataSource for RMDS** on page 15).

The parameters within *rfasrc.conf* are optional, and are described later in this document. The first parameter gives the location of the second configuration file, *rfasrc_rfa.cfg*. See the **RFA Configuration Guide** for further details.

A sample *rfasrc.conf* is included as Appendix A starting on page 64.

3.4 Specifying the root directory of DataSource

application-root

This configuration option specifies the root directory of the application installation. The standard startup script provided with DataSource will set this explicitly to the installation directory (see **Installing DataSource for RMDS** on page 15).

The application will change directory to **application-root** if it is running as a daemon (background) process.

Default value: [current working directory]

3.5 Sending images instead of updates

image-resend

By default DataSource for RMDS will only send images to Caplin Liberator (i.e. a complete set of data currently held on the source) when requested or when RMDS has reported that data is no longer stale. After this it will only send updates for individual fields when they change.

image-resend overrides this default behaviour and enables DataSource for RMDS to send images at any time during a session.

Default value FALSE

3.6 Debugging

debug-level

Determines the errors and events that are reported to the log files when DataSource for RMDS is operating (for more information on logging, see page 45). Acceptable values are shown in Table 3-1 below.

If the UDP message interface is enabled then the logging level can be changed whilst DataSource for RMDS is running. For details on how to achieve this, see **udpsend on page 62**.

Note: *A list of all error messages and their associated debug level can be found as* ***Appendix B*** *on page 71.*

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 DataSource for RMDS running.

Table 3-1: Debug levels

3.7 Running DataSource for RMDS—Solaris

A startup script to start and stop DataSource for RMDS is provided in the installation.

The startup script is *rfa/etc/rfasrc* and can be used as a standard SYSV startup script, which are often used for starting and stopping system applications on UNIX operating systems.

Using the commandline

- Enter the following:

```
./bin/rfasrc
```

This will read configuration files and write log files relative to the current directory. The standard search path for configuration files includes *etc* so the files in the *etc* directory will be found. The default log directory is *var*.

Using the startup script (recommended)

DataSource for RMDS should be started using the startup script when in production.

- Start the application by entering:

```
./etc/rfasrc start
```

This will run the application in the background.

- Stop the application by entering:

```
./etc/rfasrc stop
```

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

3.8 Naming SASS vs Triarch

DataSource for RMDS can accept either the Triarch style service/item name scheme or the TIB style four-part subject addressing scheme.

Triarch style Market Data Items can be accepted in the format of the following example:

```
/IDN/RTRSY.O
```

TIB style Market Data Items can be accepted in the format of any of the three examples following:

i)

```
/IDN/REC/RTRSY/O
```

ii)

```
/IDN/REC/RTRSY.O
```

iii)

```
/IDN.REC.RTRSY.O
```

3.9 QFORM and Marketfeed data dictionaries

DataSource for RMDS handles QFORM and Marketfeed formats transparently. Sending the UDP command **write-fields** (see page 63) will write out all the fields - which is useful for QFORM data which sends dynamic fields. RFAsrc will internally add fields as required if it comes across ones that are not listed in *fields.conf* - these can then be written out using **write-fields** and added to *fields.conf* later.

3.10 RFA Libconfig

Example RFA Config:

```
\Logger\Applogger\fileLoggerEnabled                    = False
\Logger\ComponentLoggers\Adapter\messageFile           = "lib/MessageFiles/
AdapterMessages.mc"
\Logger\ComponentLoggers\Connections\messageFile       = "lib/MessageFiles/
ConnectionsMessages.mc"
\Logger\ComponentLoggers\SASS3_Adapter\messageFile     = "lib/MessageFiles/
SASS3_AdapterMessages.mc"
\Logger\ComponentLoggers\SessionCore\messageFile       = "lib/MessageFiles/
SessionLayerMessages.mc"
\Logger\ComponentLoggers\RFAED_Adapter\messageFile     = "lib/MessageFiles/
RFAED_AdapterMessages.mc"
#"\Logger\ComponentLoggers\testRFA\messageFile"        = "LogTestMessages.mc"
\Connections\Connection_PTP\connectionType             = "RFAED"
\Connections\Connection_PTP\serverList                 = "server"
\Connections\Connection_PTP\portNumber                 = 8101
\Connections\Connection_PTP\messageThrottleRate        = 500

\Adapters\SASS3_Adapter\requestQueueReadThreshold      = 1
\Adapters\SASS3_Adapter\mainLoopTimerInterval          = 200

\Adapters\RFAED_Adapter\masterFidFile                  = "etc/appendix_a"
\Adapters\RFAED_Adapter\enumTypeFile                   = "etc/enumtype.def"
\Adapters\RFAED_Adapter\downloadDataDict               = false
\Connections\Connection_RV5\connectionType             = "RV"
\Connections\Connection_RV5\service                    = "9020"
\Connections\Connection_RV5\daemon                     = "tcp:server:7500"
\Connections\Connection_RV5\network                    = ";230.1.1.1"
\Connections\Connection_RV5\connectRetryInterval       = 3000
\Connections\Connection_RTIC\connectionType            = "SASS3"
\Connections\Connection_RTIC\subscriberRV_Connection   = "Connection_RV5"
\Connections\Connection_RTIC\contributorRV_Connetion   = "Connection_RV5"
\Connections\Connection_RTIC\marketDataItemSubTimeout = 120000
\Connections\Connection_RTIC\marketDataDictSubTimeout = 30000

\Sessions\rfasrc\connectionList                        = "Connection_PTP"
\DataDictionaries\MF\dataDictType                      = "marketfeed"
```

Please refer to the **RFA Configuration Guide** for further details.

4 Communicating with RMDS

4.1 Checking RMDS services

Use the following parameters in the configuration file *rfasrc.conf* to fine-tune how DataSource for RMDS handles RMDS services.

rfa-priority-service

The most important RMDS service that DataSource uses. If this service goes down DataSource for RMDS disconnects from all its peers, thus enabling failover DataSources to provide the data.

Default value: [no default]

service-hashsize

Size of hashtable for tracking which RMDS services are available.

Default value: 50

5 Communicating with DataSource Peers

5.1 What is a DataSource peer?

DataSource for RMDS receives data from the RMDS system and can then send it to any other application connected to the DataSource API, as illustrated in Figure 5-1 below.

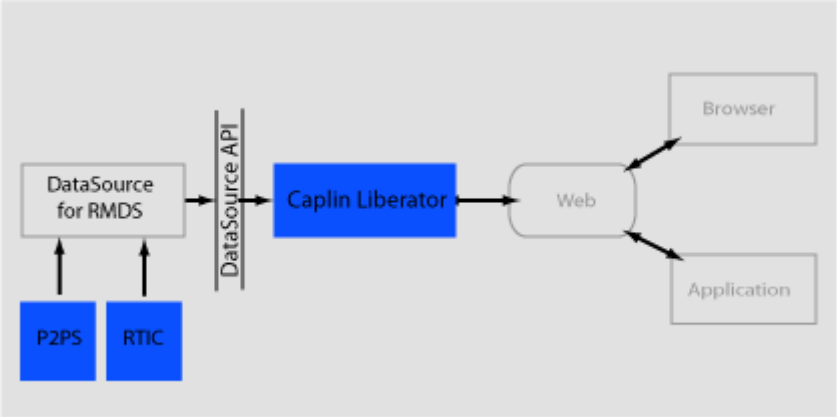

Figure 5-1: DataSource acting as a data source and data sink

These remote applications, which DataSource for RMDS can send data to, are called DataSource peers.

Other potential peers are Caplin Transformer and Caplin Liberator products.

5.2 Setting the network interface

The following configuration parameters in the configuration file *rfasrc.conf* should be used to establish the port and interface for communication with DataSource peers.

datasrc-interface Network interface to listen for connections from DataSource peers.

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 DataSource for RMDS.

Default value: 0

5.3 Identifying your DataSource

In order to communicate with peers, you must give DataSource for RMDS a name and ID number.

datasrc-name

The name of DataSource for RMDS, and how DataSource peers will identify it.

This name can be overridden by putting a value in the *local-name* option of the **add-peer** entry (see page 28).

Default value: %a-%h

Note: *%a is an abbreviation for the application name (in the case of DataSource for RMDS this is rfasrc), and %h is an abbreviation for the name of the host.*

datasrc-id

ID number of DataSource for RMDS.

This ID can be overridden by putting a value in the *local-id* option of the **add-peer** entry (see page 28), in which case it must match the ID given in the **add-peer** entry in the DataSource peer's configuration.

Default value: 0

5.4 Identifying DataSource peers

In order to send data to DataSource peers, the name and ID of each peer must be added to the configuration file *rfasrc.conf*. The following configuration parameter must be used to do this.

add-peer

Adds a DataSource peer.

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]
end-peer
```

The options in this entry are:

Name	Type	Default	Description
remote-id	integer	1	ID number of DataSource peer.
remote-name	string	datasrc1	Name of DataSource peer. Gets overridden by the startup packet when the peer connection is made.
remote-flags	integer	0	DataSource peer flags. Gets overridden by the startup packet when the peer connection is made.
remote-type	integer	0	DataSource peer type. Gets overridden by the startup packet when the peer connection is made.

Name	Type	Default	Description
local-id	integer	**datasrc-id**	ID number of this DataSource. Sent to the DataSource peer.
local-name	string	**datasrc-name**	Name of this DataSource. Sent to the DataSource peer.
local-flags	integer	0	Flags determining restart and reconnection behaviour. Possible values: NONE (0) No special restart or reconnection behaviour; F_SENDFROMSEQ (1) When reconnecting, missed packets should be requested based on sequence number; F_RECVAUTOREPLAY (4) When restarting, this peer should accept replay updates.
local-type	integer	0	Data source type sent to the connecting peer. Possible values: 0 inactive 1 active 2 contrib
addr	string	127.0.0.1 127.0.0.1	Space-separated list of addresses to connect to.
port	integer	22001 22002	Space-separated list of ports to connect to.
queue-size	integer	50	Message queue size.
obj-hash-size	integer	-1	Number of entries in active object hashtable.

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).*

5.5 Identifying which fields can be sent

A set of fields makes up a data object or an update to an object. Standard record objects are made up of fields.

In order for DataSource for RMDS to handle fields, the following must exist:

- The file *rfasrc.conf* must contain the entry

```
include-file  fields.conf
```

- The file *fields.conf* must exist. It contains details of the required fields, including all mandatory entries. Only fields defined in *fields.conf* are propagated.

Both these elements exist in your DataSource for RMDS installation.

add-field

Identifies a field which DataSource for RMDS can output to peers. This parameter should be included in the *fields.conf* file. You can have up to 32,768 entries.

Format: ***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.
FieldFlags	integer	0	The flags passed by the field.
FieldFlagsData			Not used by DataSource for RMDS.
FieldFormat			Not used by DataSource for RMDS.

You must use **add-field** to configure certain fields, as listed in Table 5-1below.

FieldName value	Description
TS1_NUMBER	Field identifying the current segment number of TS1 charting data. For more information see page 59.
TS1_HIGHEST	Field showing the number of segments in a complete TS1 chart.
TS1_DATA	Field containing the base 64 encoded TS1 charting data segment.
TS1_STATUS	Field showing how much of the chart has been sent.
HISTORIC_URL	Field containing the URL of a TS1 chart.

FieldName value	Description
HBTime	Field containing the DataSource for RMDS heartbeat. For more information see **Sending heartbeats** on page 42.
Type	Field identifying the WebSheet template used to display the data.

Table 5-1: Mandatory add-field entries

For example:

```
add-field         HBTime            4200  0
add-field         DELAY_TIME        4300  0
add-field         HISTORIC_URL      4400  0
add-field         TS1_STATUS        4401  4
add-field         TS1_NUMBER        4402  4
add-field         TS1_HIGHEST       4403  0
add-field         TS1_DATA          4404  4
add-field         Type              10002 0
```

enum-enable

A boolean parameter which enables fields to be enumerated to allow configuration files to become more readable.

This parameter should be included in the *rfasrc.conf* file.

An enumerated field enables you to give a non-numeric and more meaningful name to a number. The names can then be used in the configuration file: for example if an field can take the values 1, 2 or 3, you can define 1 as "London", 2 as "Paris" and 3 as "Berlin".

Default value: FALSE

add-enum

Defines the enumerated options for a field. This parameter should be included in the *rfasrc.conf* file.

The client application requires these mappings to be made if it is to display text instead of numbers. Using the above configuration, if an application receives a value of "1" for the specified field it will display "London" instead of "1".

You can have several entries.

Syntax:

```
add-enum
    fieldnum    [value]
    values      [value value]
end-enum
```

The options in this entry are:

Name	Type	Default	Description
fieldnum	integer	[no default]	Number of the field whose values are to be mapped.
values	string	[no default]	Space-separated pair of strings containing the mapping. The first string identifies the incoming value, the second string the name given to that value.

Example:

```
enum-enable

add-enum
      fieldnum    4
      values      0     "   "
      values      1     "ASE"
      values      2     "NYS"
      values      3     "BOS"
      values      4     "CIN"
      values      5     "PSE"
      values      6     "XPH"
      values      7     "THM"
      values      8     "MID"
      values      9     "NYQ"
end-enum
```

5.6 Sending chain records

chain-shuffle

Enables field shuffling. See **shuffle-fids on page 35**.

Default value: FALSE

shuffle-fids

Paired list of FIDs, used to ensure that all links to chain records are sent to the Caplin Liberator on a consistent set of FIDs. Data coming in under the first FID is forwarded to the Caplin Liberator under the second FID.

Example:

shuffle-fids 237 814

This results in the contents of FID 237 being copied to FID 814.

5.7 Sending pages

Page sizes can vary from the standard of 80 x 25 characters. Any page of non-standard dimensions must be specified using the following option.

add-service

Identifies a service that provides pages which are not 80 x 25 characters.

Syntax:

```
add-service
    name      [value]
    pagex     [value]
    pagey     [value]
end-service
```

The options in this entry are:

Name	Type	Default	Description
name	string	[no default]	The service name.
pagex	integer	80	Number of columns in the page.
pagey	integer	25	Number of rows in the page.

5.8 Rippling fields

"Rippling" enables you to keep historical records by identifying a series of fields (for example those which detail the last five previously traded prices) and moving the value of each field on to the next field when a new update arrives.

For example, if the group contains the following fields and values:

Field 1	*Field 2*	*Field 3*	*Field 4*
12.3	12.5	12.1	12.3

when an update arrives, it goes into position one and the rest move on a place:

Field 1	*Field 2*	*Field 3*	*Field 4*
12.6 ⇨	12.3 ⇨	12.5 ⇨	12.1

ripple-enable Switches on the rippling functionality.

Default value: FALSE

add-ripple

Adds a group of FIDs whose order is shuffled when an update is received.

Syntax

add-ripple
ripple
end-ripple

The option in this entry is:

Name	Type	Default	Description
ripple	integer array	[no default]	The FIDs in the ripple chain.

Example:

```
add-ripple
      6 7 8 9 10
end-ripple
```

5.9 Sending data to peers on startup

Use the following parameter in the configuration file *rfasrc.conf* to send records to DataSource peers when DataSource for RMDS starts up (and before the peers have requested them).

add-item

Adds a record to send to the indicated peer or peers when DataSource for RMDS starts up.

Syntax:

```
add-item
    name        [value]
    bcast
    request
    peer        [value]
    internal
end-item
```

The options in this entry are:

Name	Type	Default	Description
name	string		Name is of form /SERVICE/ITEM /SERV/SET/IT/EX or .EX
bcast	boolean	FALSE	Indicates the record should be sent to all connected DataSource peers. This is used when DataSource for RMDS is not being used as an active source.
request	boolean	FALSE	Indicates the record must be requested from RMDS (not necessary if *bcast* is set).
peer	integer array	NULL	Space-separated list of DataSource peers to send the record to. Use a number to represent each peer (where the numbers correspond to the order in which the **add-peer** entries appear in the configuration file *rfasrc.conf*).
internal	boolean	FALSE	Indicates that objects should be marked as being created by DataSource for RMDS and therefore should not be discarded. This is used primarily for news headlines.

5.10 Mapping the / character

Because Reuters use / in their symbol names and RTTP uses / as a directory separator, the receiving application needs to differentiate between the two.

map-character Identifies a character that applications receiving the news story can map a forward slash / to.

Default value: ~

5.11 Adjusting how DataSource tracks data

If DataSource for RMDS has been configured to be an active data source, it will keep track of which records have been requested and send updates for those objects only.

The following parameter in the configuration file *rfasrc.conf* may be adjusted to fine-tune the performance of DataSource for RMDS.

item-hashsize Size of hash table for keeping track of items.

Default value: 4000

5.12 Sending heartbeats

DataSource sends "heartbeat" information to inform recipients of the data that the session is alive. This enables Caplin Liberators to alert the client process if data becomes stale.

The following parameters in the configuration file *rfasrc.conf* may be adjusted to control the emission of a heartbeat packet onto the DataSource API.

heartbeat-symbol The symbol under which the heartbeat information will be sent, and the symbol which applications should subscribe to in order to listen for the heartbeats.

If **heartbeat-symbol** is NULL then the heartbeat will never be issued.

Default value: NULL

heartbeat-symbol-time Frequency of heartbeat signals in seconds.

Default value: 30.0

6 Object Pre-Caching

In some environments when users are requesting a common subset of items and it takes a relatively long time to retrieve these items from the backend infrastructure, it may be desirable to configure DataSource for RMDS elsewhere so that it pre-caches objects.

This feature allows requests for objects within the cache to be responded to quickly, and if used in conjunction with “connect-when-cached” allows an DataSource for RMDS elsewhere to appear "offline" until all configured objects are non-stale.

cache-hash-size

Size of the active cache. This should be configured to be approximately twice the total number of objects that you wish to cache.

Default value: 1024

cache-objects

This indicates that object pre-caching should be enabled.

Default value: False

cache-add-object

Configure an object to be in the active cache.

Default value: None

connect-when-cached

This permits datasource connections only when all objects within the cache are non-stale. (cf **rfa-priority-service on page 24**).

Default values False

Example:

```
# Enable the cache
#cache-objects

# Adjust the size of the hashtable for tracking the objects
#cache-hash-size         16384

# Only connect to our peers when all objects that should be precached
# are cached and are non-stale
#connect-when-cached

# List of items to be precached - one per line
#cache-add-object       /I/VOD.L
#cache-add-object       /I/MSFT.O
```

7 Logging

The log files keep a record of status and debug messages, as well as a list of all data that has passed through DataSource for RMDS and to which destinations the data was sent.

7.1 Setting the default directory

log-dir Default directory in which to store log files.

Default value: %r/var

Note: *%r is an abbreviation for* ***application-root*** *(see page 18).*

7.2 Configuring DataSource status logs

The following parameters within *rfasrc.conf* configure the status message log files.

log-cycle-suffix

Suffix for cycled logs. This is passed through **strftime** (refer to your UNIX manual for further information on **strftime**).

Default value: %u

Note: *%u is an abbreviation for the day of the week. The default value of %u results in a file being created for each day of the week. For a list of* ***strftime*** *abbreviations used within DataSource for RMDS, see Table 8-1 on page 54.*

log-maxsize

Maximum log file size in bytes. The log files will be cycled every **log-cycle-period** if they exceed this size; therefore a value of 0 means log files will cycle every time they are checked.

Default value: 0

log-cycle-time

Time at which logs will cycle, in minutes from midnight.

Default value: 240 (i.e. 0400 hours).

log-cycle-period

Interval between cycling logs, in minutes.

Default value: 1440 (i.e. daily)

log-cycle-offset

Specifies how many minutes to take off the current time when creating the suffix (see **log-cycle-suffix**).

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.

7.3 Configuring the DataSource packet log

The following parameter within *rfasrc.conf* configures the packet log file, which contains a record of all data sent by DataSource for RMDS, and includes all requests for data recieved by the source.

datasrc-pkt-log

Name of DataSource's packet log file. The location of the file must either be relative to **log-dir** (see page 45) or absolute.

Default value: packet-%a.log

Note: *%a is an abbreviation for the application name (for DataSource for RMDS this is rfasrc).*

7.4 The logcat utility

The **logcat** utility is used in much the same way the standard UNIX-based **cat** command (**cat** is short for "concatenate" and strings files together), but only for the binary log files described above.

logcat

The **logcat** utility can be found in the *bin* directory of the DataSource for RMDS installation.

Example output from **logcat**:

```
../bin/logcat packet-rfasrc.log
2002/04/16-11:33:16 192.168.201.208 DATAUPDATE2 0 2652 /I/VOD.L 19
1379=+119.812 1069=1068=A  1067=12:50:02 996=+120 6/8 985=+102380264
975=A 956=+119.654 379=11:50:02 178=+25275 118
.... etc
```

If you are using the default filenames for your log files then the only argument needed is the filename. However, if you have changed the names of your log files you have to tell **logcat** what type of file to expect.

Example:

```
$ ../bin/logcat -t packet my-packet.log
```

The *-t (or --type)* command line option takes a string as an argument. The string can be either "packet" or "p".

Examples:

```
logcat --type packet my-packet.log
```

or

```
logcat -t p my-packet.log
```

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
```

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

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

logcat timestamps

By default **logcat** prints timestamps in the local timezone. To force **logcat** to print in GMT use the *-g* option.

Example:

```
logcat -g packet.log
```

8 Receiving news from RMDS

News data is delivered by the same method as market data—as records consisting of fields. Because news data requires different processing than financial data, DataSource allows you to identify news items by specifying their particular symbols or record names.

A news item consists of two parts: the headline and the story. Stories are generally longer than the headline, and transmitted in segments. You can configure how these segments are stored in DataSource and how they are sent to their destinations, which can help in fine-tuning the performance of DataSource.

DataSource for RMDS can be configured to send news stories to the client via RTTP, write them to an XML file, which permits some degree of formatting on the client side, or output them to a plain text file.

Note: *News will use a lot of disc space over time and the news directory should be cleaned by an external process to ensure that the disc doesn't become full. A short script (clean-news.sh) is supplied in the bin directory as an example of how to clean it up.*

8.1 Identifying news items

The following configuration options in *rfasrc.conf* specify which symbols are used to broadcast news items.

news-headcode

The symbol that news headlines are sent under.

Default value: /NEWS

news-bodycode

The symbol that the body of the news item is sent under.

Note: *This option only applies when data is sent to DataSource peers, and not when output as XML for access via HTTP.*

Default value: /NEWSSTORY

news-wait-time

Number of seconds to wait in between receiving a headline and requesting the associated story (occasionally a headline may appear on RDMS but the associated story may take a few seconds to become available).

Default value: 2.0

8.2 Receiving news stories

The following configuration options in *rfasrc.conf* specify how DataSource receives news stories.

add-item

In order to receive news stories you must include the following **add-item** entry in your configuration file. This requests the symbol from RMDS that news stories are sent under.

Syntax:

```
add-item
    name        /IDN_RDF/N2.UBMS
    request
end-item
```

For more information on **add-item**, see page 52.

news-hashsize

This configuration option in *rfasrc.conf* specifies how DataSource stores news stories. It specifies the size of the hashtable for keeping details of stories whilst they are being assembled. The default value of 1024 is suitable for high numbers of stories being processed every second and in most circumstances will not need to be changed.

Default value: 1024

news-final-time

Number of seconds to wait for an update of the final story segment before considering the story to be complete.

Default value: 300

news-history

If set keeps more than one day's worth of news stories saved on disk.

Default value: FALSE

8.3 Deleting incomplete news stories

The following configuration options in *rfasrc.conf* specify timeout periods used when handling news.

news-expire-time Number of seconds to wait before deleting an article (typically one day).

Note: ***news-expire-time*** *works by deleting internal references to headlines and stories, as these references are repeated each day. However previous days' headlines and stories can still be requested if you have set* ***news-history*** *and the headlines and stories have been saved. If* ***news-history*** *is not enabled, a headline delete message is sent to Caplin Liberator and stories are deleted from disk.*

Default value: 86400

news-garbage-time Time (in seconds) between calls to the news garbage collector. The news garbage collector deletes any incomplete articles that have been in cache beyond the period set by **news-final-time** (see page 52).

Default value: 60.0

8.4 Selecting the output type

The following configuration option in *rfasrc.conf* determines how DataSource outputs a news story.

news-mode

Determines in what form DataSource outputs a news story. Possible values are shown in Table 8-1.

Default value: XML

Code	Meaning
BCAST	Stories are broadcast in plain text to those DataSource peers configured with an interest in the symbol identified by news-bodycode (see page 51).
XML	Stories are saved to a file in XML format which the client requests via HTTP via the Liberator.
XML_REQUEST	Stories are saved to a file in XML format which the client requests over RTTP via the Liberator using **news-bodycode** (see page 51).
PLAIN	Stories are saved to file in plain text format which the client requests via HTTP via the Liberator.
PLAIN_REQUEST	Stories are saved to file in plain text format which the client requests over RTTP via the Liberator using **news-bodycode** (see page 51).

Table 8-1: News output types

8.5 Sending data to DataSource peers

The following configuration option in *rfasrc.conf* only applies for news stories being sent to DataSource peers.

news-maxlen Size of the segments (in bytes) that the body of the news item is transmitted in.

Default value: 1024

8.6 Outputting XML files

The following configuration options in *rfasrc.conf* specify how DataSource outputs a news story as XML.

A note on strftime abbreviations

Some of the following functions are passed through **strftime** to convert the abbreviations into appropriate date and time strings. Please refer to your UNIX manual for further information on **strftime**. Table 8-2 shows some of the abbreviations used in DataSource for RMDS.

Code	Meaning
%b	The abbreviated month name according to the current locale (for example Jan, Feb, Mar).
%d	The day of the month as a decimal number (range 01 to 31).
%u	The day of the week as a decimal (range 1 to 7, Monday being 1).
%D	Equivalent to %m/%d/%y.
%H	The hour as a decimal number using a 24-hour clock (range 00 to 23).
%M	The minute as a decimal number (range 00 to 59).
%Y	The year as a decimal number including the century.

Table 8-2: strftime abbreviations

news-time-format	The output time format for headline display view in the client.	
	Default value:	%b %d %H:%M
news-displaytime-format	The time format as presented in the XML article.	
	Default value:	%d %b %Y %H:%M GMT
news-writedir	The root directory for news articles. The full path must be specified.	
	Default value:	[no default]
news-url	The URL prefix for **news-writedir** via HTTP.	
	Default value:	[no default]
xml-stylesheet	The stylesheet used for news articles. The location must be relative to the *.xml* files via HTTP.	
	Default value:	../../news.xsl
news-rle	If set the xml story text is encoded using Run Length Encoding, a simple form of data compression.	
	Default value:	FALSE

9 Record name mapping

DataSource can be configured to map names of records passed into it into a different format. This can either be used simply to make it a valid RTTP record name (all symbols from Caplin Liberator start with a “/”), or to create a complex directory structure of records.

9.1 Adding a name mapping

The following configuration option of *rfasrc.conf* configures how DataSource renames records in order to pass consistently-named records to its destination peers.

The name pattern can include a single wildcard character (an asterisk **"*"**).

add-pattern

Adds a record name mapping.

Syntax: add-pattern [name to search for] [name to change to]

Examples:

```
add-pattern          *.FX    /FX/*

add-pattern          MSFT*   /Microsoft/*

add-pattern          *       /*
```

10 TS1 charting data

By default DataSource for RMDS sends TS1 charting data to clients in a base 64 encoded form using the RTTP protocol.

DataSource for RMDS can also write incoming Reuters TS1 charting data to a file and then send the URL of the file to the client via RTTP. The client application can then use the URL within a hyperlink to enable users to view the chart using HTTP.

- To allow the client application to access the charts you must define the HISTORIC_URL and TS1_STATUS fields within the Caplin Liberator configuration. See **add-field** on page 31 for more information.

10.1 Sending TS1 charting data to peers

Use the following parameters in the configuration file *rfasrc.conf* to configure how DataSource for RMDS sends TS1 charting data to clients.

ts1-hashsize

Size of TS1 data hashtable.

Default value: 128 items

ts1-active-obj

If set, the chart file URL information and progress indicator are sent to DataSource peers as active records. This means that peers can request and discard this information. If not set, peers only request the information once and then do not request or discard it.

Default value: FALSE

10.2 Writing TS1 charting data to a file

Use the following parameters in the configuration file *rfasrc.conf* to configure how DataSource for RMDS writes TS1 charting data to a file.

ts1-localwrite

Enables incoming Reuters TS1 charting data to be written to a file which can be retrieved using HTTP.

Default value: FALSE

ts1-writedir

Specifies the directory to write the incoming TS1 data files to.

Default value: [no default]

ts1-url

Specifies the url prefix which points to the directory where the TS1 data files are stored.

Default value: [no default]

11 Using UDP commands

DataSource for RMDS includes a UDP command interface that enables you to send UDP messages regarding debugging and the writing of lists of watched objects to files.

11.1 Configuring the UDP interface

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

Default value: [no default]

udp-interface Network interface to listen on for UDP messages. If not specified then DataSource for Triach listens for UDP signals on all interfaces

Default value: [no default]

11.2 UDP commands

The following UDP commands can be sent over DataSource for RMDS's UDP interface.

udpsend

Sends a UDP message.

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

Parameters:

Name	Type	Default	Description
-s	string	127.0.0.1	Host name or IP address of machine to send message to.
-p	integer	10001	Port that DataSource for RMDS listens on for UDP messages.
message	string	[no default]	Message to send. Possible values are listed below.

debug

Dynamically changes the level of error and event reporting. This overrides the level set using the configuration option debug-level (see page 20).

Syntax: ***debug level***

Parameter:

Name	Type	Default	Description
level	string	[no default]	Level of debug messages. For a list of acceptable values see on Table 3-1 on page 20.

write-fields

Writes out all the fields - this is useful for qform data which sends dynamic fields. RFAsrc will add in fields as required if it comes across one it doesn't know about.

new-field-start

This is the field number to start adding new fields from.

Default value: 5000

new-field-add

This is the increment to add to **new-field-start** to get the next new field number.

Default value: 1

insert-enable

This enables contributions.

Default value: FALSE

Note: *You must also set **local-type** on the **add-peer** section on page 28 to include "contrib" for this to work.*

12 Appendix A: Sample rfasrc.conf

```
#
#  Example Config File for DataSource for RMDS v4.0
#

######################################################################
#
#   RFA configuration
#
#   All RFA settings are configured using the standard Reuters
#   configuration file. Please consult your Reuters documentation
#   for details of configuration parameters
#
######################################################################


# The configuration file which defines the RFA session settings
rfa-config-file     %r/etc/%a_rfa.cfg

# The name of the session within the configuration file to use
rfa-session-name    rfasrc


######################################################################
#
#   Some Sources send out images instead of updates, enable this
#   if one of your sources exhibits this property
#
######################################################################
#image-resend
######################################################################
#
# Debug level
#
# One of either:
# CRIT, ERROR, NOTIFY, WARN, INFO, DEBUG
#
# (These are in descending order of importance and increasing order of
#  logging detail)
#
######################################################################
debug-level   INFO
```

```
#####################################################################
#
#   DataSource configuration
#
#####################################################################

# The DataSource heartbeat symbol and frequency
heartbeat-symbol      /HBT/rfasrc
heartbeat-symbol-time 60.0

datasrc-name          rfasrc
datasrc-id            1

# Liberator (peer 0)
add-peer
     addr             127.0.0.1
     port             25000
     local-type       1
end-peer

#####################################################################
#
#          UDP Port configuration
#
#####################################################################

# The port on which we listen for messages (if not specified then no port)
udp-port   10001

# The interface on which we should listen for messages (if not specified
# then all interfaces)
#udp-interface


#####################################################################
#
#     Item Configuration
#
#     We can request items directly from Triarch and send them out
#     automatically to the peers
#
#####################################################################
```

```
# Request the Reusters news symbol if this config requires news
#add-item
#     item       N2_UBMS
#     source     I
#     request
#end-item
# Time that a peer has to be down before objects previously requested by
# this peer are discarded
#item-peerdown-time   60.0

######################################################################
#
#     News Configuration
#
#     We split news into headlines /NEWS and /NEWSSTORY, we want to
#     send these out to peer 0 (Liberator)
#
######################################################################

add-item
     item            NEWS
     peer            0
     internal
end-item

add-item
     item            NEWSSTORY
     peer            0
end-item

# Size of hashtable for maintaining details of stories for the day
news-hashsize         1024

# What's the packet name that we send out the headlines under
news-headcode              /NEWS

# The packetname that we send out the body in BCAST news-mode
news-bodycode              /NEWSSTORY
```

```
# Size of body segments (in bytes)
news-maxlen                1024

# Time to wait for Triarch before requesting the news body
news-wait-time             2.0

# What news mode to operate in
# BCAST, XML, XML_REQUEST, PLAIN, PLAIN_REQUEST
news-mode                  XML_REQUEST

####################################################################
#
#          XML News configuration
#
####################################################################

# Time format for headlines (standard strftime options available)
news-time-format           %b %d %H:%M

# Time format for XML pages
news-displaytime-format    %d %b %Y %H:%M GMT

# Root directory to write news articles to
news-writedir              news/

# Root URL where news-writedir can be accessed from (either fully qualified
# or partical URL valid)
news-url                   /news

# XML Style sheet to reference in .xml files
xml-stylesheet             ../../news.xsl
####################################################################
#
#     TS1 (charting) Configuration
#
####################################################################

# If enabled then TS1 data is written to disc and a URL sent to the client
# otherwise data is base64 encoded and sent over datasrc (thence RTTP to the
client)
#ts1-localwrite
```

```
# Size of hashtable for maintaining chart details during retrieval
ts1-hashsize    128

# The URL that we output (prepended to the chart filename)
ts1-url         /charts/
# Where to write complete chart files to
ts1-writedir    charts/

# Should be enabled if TS1 objects are to be sent out as active objects
ts1-active-obj

#####################################################################
#
#     FID Manipulation
#
#####################################################################

# These lines ensure that all links come through on the same set of FIDs
# Enable the shuffling
chain-shuffle
# And the mappings
# PREV_LR -> LONGPREVLR
shuffle-fids          237 814
# PREF_LINK -> LONGNEXTLR
shuffle-fids          1081 815
# NEXT_LR -> LONGNEXTLR
shuffle-fids          238 815
# LINK_n  -> LONGLINK_n
shuffle-fids          240 800
shuffle-fids          241 801
shuffle-fids          242 802
shuffle-fids          243 803
shuffle-fids          244 804
shuffle-fids          245 805
shuffle-fids          246 806
shuffle-fids          247 807
shuffle-fids          248 808
shuffle-fids          249 809
shuffle-fids          250 810
shuffle-fids          251 811
shuffle-fids          252 812
shuffle-fids          253 813
```

```
#####################################################################
#
#     DataSource mappings (to ensure things are as we expect)
#     (order is important!)
#
#####################################################################

# WebSheet requests charts as /I/CHARTS
add-pattern     /IDN_SELECTFEED/d*    /I/CHARTS/d*
# Reply with an immediate NODATA if we get a request for a /I/d* from the
client
add-pattern     /NOTFOUND/d*          /I/d*
# The default mapping from /I to /IDN_SELECTFEED
add-pattern     /IDN_SELECTFEED/*     /I/*


# Clients map a reuters / into a high ascii character
map-character   \277

#####################################################################
#
#     Rippling FIDs
#
#####################################################################

# Enable rippling and include the configuration file
ripple-enable
include-file    ripple.conf
#####################################################################
#
#     Enumerated FIDs
#
#     RFAsrc can convert enumerated values into the textual version
#     This conversion can be done on the SelectFeed/RDF server, but
#     it may not be desirable to do it at that point
#
#####################################################################

# Enable enum content mapping and include the configuration file
#enum-enable
#include-file   enums.conf
```

```
#####################################################################
#
#     Fields
#
#     RFAsrc can derive certain fields which are consumed by the client
#     these need to be defined in this file and be present in the
#     fields.conf used by the Liberator.
#
#    Note: From RFAsrc 3.6 only fields that are defined in fields.conf will
#    be processed
#
#####################################################################

include-file    fields.conf

#####################################################################
#
#     Precaching
#
#    RFAsrc can precache objects to enable faster recovery from a failover
#    situation
#
#####################################################################

# Enable the cache
#cache-objects

# Adjust the size of the hashtable for tracking the objects
#cache-hash-size         16384

# Only connect to our peers when all objects that should be precached
# are cached and are non-stale
#connect-when-cached

# List of items to be precached - one per line
#cache-add-object       /I/VOD.L
#cache-add-object       /I/MSFT.O
```

13 Appendix B: Debug levels and messages

13.1 CRITICAL debug level messages

```
CRITICAL: Couldn't open RFA library
```

Table 13-1: CRITICAL debug level messages

13.2 ERROR debug level messages

```
ERROR: Couldn't mount Sink channel trying again in %f seconds
ERROR: Could not mount Sink channel, adding re-read event
ERROR: rfaSnkMount failed: %s
ERROR: rfaGetProperty(RFA_OPT_CHANNEL_FD) failed: %s
ERROR: rfaRegisterCallBack Failure: %d %s
ERROR: Strange data %d received
ERROR: Cannot write watchlist to file <%s>
ERROR: Could not open news file %s (%s)
ERROR: Invalid time string supplied: %s
ERROR: Failed to create dir <%s> errno = %s
ERROR: Inappropriate newsstory filename <%s>
ERROR: Could not open the news file <%s>
ERROR: Could not determine size of the file <%s>
ERROR: Could not wholly read the file <%s>
ERROR: Couldn't open file %s for HTTP write
ERROR: Couldn't listen on UDP port %d
```

Table 13-2: ERROR debug level messages

13.3 NOTIFY debug level messages

```
NOTIFY: Received unknown update type, using %d
NOTIFY: Received update with unknown type %d
NOTIFY: Channel %d disconnected: %s
NOTIFY: RFA has reconnected us on channel %d (%d)
NOTIFY: SnkOpen on %s/%s failed
NOTIFY: SnkClose on %s/%s failed
NOTIFY: Dumping repeated update for row %d
NOTIFY: News drop before time not yet supported
NOTIFY: Unknown news message type %d <%s>
NOTIFY: Odd, no story id on body text
NOTIFY: Odd, no story text on body text message
NOTIFY: Story headline has gone..odd!
NOTIFY: Request for <%s> from peer %d is too long
NOTIFY: Discard for <%s> from peer %d is too long
NOTIFY: Maximum news length is long than buffersize, setting to 1024
NOTIFY: No item defined for news headline object %s
NOTIFY: News article has invalid date/time format: <%s> <%s>
NOTIFY: Got long and short links at the same time...
NOTIFY: UDP job (%s)/cmd combo not found so couldn't delete
```

Table 13-3: NOTIFY debug level messages

13.4 WARN debug level messages

```
WARN: Incorrect number (%d) of arguments given for UDP verbose
command
WARN: %s/%s not found in hash table
WARN: Storyid has 0 length <%s>
WARN: Received story <%s> with 0 length/non existent headline
WARN: No delayed prefix set, setting to %s
WARN: Delayed prefix is too long, setting to %s
WARN: Zero length item request, sending nodata
WARN: Unknown service %s, sending nodata
WARN: Zero length item discard
WARN: Incorrect number of arguments for UDP watchlist command
WARN: No root found for bitmap sequence %d/%s (%s)
WARN: RTL for secondary (%d) doesn't match RTL for root (%d) -
aborting
WARN: No memory for job allocation
```

Table 13-4: WARN debug level messages

13.5 INFO debug level messages

```
INFO: <%s> service is %s
INFO: Using cached type %s for unknown update type
INFO: Received RFA_ET_STATUS_CLOSED/%d for %s/%s: %s
INFO: Closing item %s/%s since RFA closed it
INFO: RFA has already closed the item %s/%s..
INFO: %s item %s service %s
INFO: Alert sentout <%s> (%s)
INFO: New story: storyid=<%s> headline=<%s>
INFO: Delayed prefix is  %s
INFO: Adding peer %d for item %s : %s
INFO: Request for %s from peer %d
INFO: Request for item with no service (%s), sending nodata
INFO: Opening %s on service %s
INFO: Opening delayed snapshot %s on service %s for peer %d
INFO: Been told to discard %s/%s by peer %d
INFO: Discard for object with no service (%s)
INFO: Discard for item with no service (%s)
INFO: Item <%s> has DELAY_NODISCARD set for peer %d
INFO: Assuming that storyid <%s> (%s) (%s) has finished updating
INFO: Deleting news article <%s> since over %f seconds oldINFO:
Sending delete for storyid <%s>
INFO: Writing dummy article to file %s
INFO: Writing news article to file %s
INFO: Old news article received
INFO: Received bitmap data for %s on %s
INFO: Received discard, deleting file %s for %s/%s
INFO: Writing to httpfile %s
INFO: Matched %s to delayprefix %s
INFO: Not delaying, sending update out directly as %s
INFO: Not sending empty packet for %s
INFO: Sending live-fids for %s to peer %d
INFO: Sending image for %s to peer %d
INFO: Not sending empty update packet for %s/%s
INFO: Sending status %d for %s
INFO: Received flag for _ONETIME symbol, sending delay instead
INFO: Sending flag %d for %s to peer %d
```

```
INFO: UDP message port not configured
INFO: Listening on UDP port %d for messages
INFO: Added UDP job for cmd %s @%p for "%s"
```

Table 13-5: INFO debug level messages

13.6 DEBUG debug level messages

```
DEBUG: RFA channel is %d
DEBUG: Registering RFA callbacks
DEBUG: Deregistering RFA callbacks
DEBUG: Received page broadcast for %s/%s
DEBUG: Received unrequested image for %s/%s State: %d - ignoring
DEBUG: Received record image for %s/%s State: %d
DEBUG: Received page image for %s/%s
DEBUG: Received unknown type %d for %s/%s
DEBUG: Received unrequested update for %s/%s - ignoring
DEBUG: Received record update for %s/%s
DEBUG: Received page update for %s/%s
DEBUG: Received event %d (%d) for %s/%s (%s)
DEBUG: Received a rename event %s/%s -> %s
DEBUG: Created new item %s/%s
DEBUG: RFA should do recovery for us after disconnect
DEBUG: Insert reply %s srcname: %s item: %s text: %s
DEBUG: Requesting %s/%s from RMDS
DEBUG: Discarding %s/%s from RMDS
DEBUG: Little page template = %d
DEBUG: storyid for text is <%s>
DEBUG: Reached end of story <%s>. Waiting for re-xmit/garbage
collect
DEBUG: RMDS: %d <%s>
DEBUG: Item <%s/%s> is already open for peer %d, asking RFA for image
DEBUG: Item <%s> is already closed for peer %d
DEBUG: Deleting item %s/%s from memory
DEBUG: Cleaning up for disconnect from peer %d
DEBUG: Starting news garbage cleaning
DEBUG: Ending news garbage cleaning
DEBUG: Linking <%s> to <%s>
DEBUG: Unlinking <%s>
```

```
DEBUG: Expiring ts1 data %s/%s
DEBUG: Received bitmap sequence number %d/%d for %s
DEBUG: TS1 Sequence %d not seen
DEBUG: TS1 Output symbol/filename is <%s>
DEBUG: Unlinking TS1 file %s
DEBUG: Couldn't write all the data (%d/%d) to file %s
DEBUG: Name is %s start is %s
DEBUG: Matched %s to %s
DEBUG: Sending delete for %s to peer %d
DEBUG: Broadcasting image for %s/%s
DEBUG: Broadcasting update for %s/%s
DEBUG: Sending update for %s to peer %d
DEBUG:%s: <%s> <%d> = <%s>
DEBUG: Successfully removed UDP job %s
DEBUG: Read %d bytes on UDP port
DEBUG: UDP Command is %s time is %lu
DEBUG: Already received UDP message
DEBUG: Calling functions for command <%s>
```

Table 13-6: DEBUG debug level messages

London

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

www.caplin.com

New York

111 5th Avenue
New York
NY 10003-1005
USA
Telephone: +1 212 994 1770
Fax: +1 212 994 1777

info@caplin.com