

CAPLIN

DS4IDC 4.4

DataSource For IDC Administration Guide

July 2010

CONFIDENTIAL

Contents

1	Preface.....	1
1.1	What this document contains.....	1
	About Caplin document formats	1
1.2	Who should read this document.....	1
1.3	Related documents.....	2
1.4	Typographical conventions	2
1.5	Feedback.....	3
1.6	Acknowledgments.....	3
1.7	Open Source Software.....	3
2	Terminology.....	4
3	Introduction to DataSource for IDC.....	5
3.1	Architecture.....	5
3.2	Features.....	7
	Active DataSource	7
	Configurable log cycling	7
	Name mapping	7
	Connecting to the IDC feed	7
3.3	About the data.....	7
3.4	About configuration files.....	8
4	Installing DS4IDC.....	9
5	Reviewing the supplied configuration.....	10
5.1	General configuration settings	10
	Connecting to a DataSource peer	11
	Setting the log level	12
5.2	Specific configuration settings.....	13
6	Running DS4IDC.....	14
7	Monitoring and management.....	15
7.1	The monitoring and management subsystem.....	15
7.2	The UDP command interface utility	16
	udpsend	16
7.3	Viewing packet logs with the logcat utility.....	18
7.4	Log file cycling.....	20

8	Reference: Types of configuration item.....	21
8.1	Single value configuration items.....	21
8.2	Multi value configuration items.....	21
8.3	Configuration items with nested options	22
8.4	Lists.....	23
9	Reference: DataSource configuration.....	24
9.1	include-file.....	24
9.2	license-file.....	24
9.3	process-usage-period	25
9.4	The network interface.....	26
	datasrc-interface	26
	datasrc-port	26
9.5	DataSource identifiers.....	27
	datasrc-name	27
	datasrc-id	27
9.6	Field mappings.....	28
	add-field	28
9.7	DataSource peers.....	29
	add-peer	29
	Peer connection failover strategy	35
9.8	Name mappings.....	36
	add-pattern	37
9.9	Heartbeats.....	38
	heartbeat-symbol	38
	heartbeat-symbol-time	38
9.10	Logging.....	39
	log-level	39
	log-dir	40
	log-cycle-suffix	40
	log-maxsize	41
	log-cycle-time	41
	log-cycle-period	42
	log-cycle-offset	42
	add-log	42
	datasrc-pkt-log	45
9.11	UDP command interface configuration.....	46
	udp-interface	46
	udp-port	46

9.12	Java configuration.....	47
	jvm-location	47
	jvm-options	47
	jvm-global-classpath	48
	add-javaclass	48
9.13	Monitoring configuration.....	50
	monitor-module	50
	monitor-moddir	50
	add-monuser	51
9.14	Sockmon configuration.....	52
	sockmon-interface	52
	sockmon-port	52
	log-level	52
9.15	JMX configuration.....	53
	jmx-classid	53
	log-level	53
	rmi-registry-port	53
10	Reference: DS4IDC specific configuration.....	54
10.1	Exchange mappings.....	54
	add-exchange	54
10.2	Client side processor connection.....	55
	add-csp-connection	56
10.3	IDC data logging.....	59
	ctf-logfile	59
11	Reference: Log levels and messages.....	60
11.1	CRITICAL level log messages.....	60
11.2	ERROR level log messages.....	61
11.3	NOTIFY level log messages.....	62
11.4	WARN level log messages.....	62
11.5	INFO level log messages.....	63
11.6	DEBUG level log messages.....	64
12	Reference: Monitoring MBeans.....	65
12.1	MBeans summary.....	65
12.2	System information (ctfsrc.server.system).....	66
12.3	DataSource information (ctfsrc.server.datasrc).....	66
12.4	Log file information (ctfsrc.server.logging).....	67
12.5	MessageQueue (ctfsrc.server.msgq).....	68

12.6	Peer information (ctfsrc.server.peers).....	68
12.7	Peer statistics (ctfsrc.server.peerstats).....	70
12.8	IDC server connection (ctfsrc.csp_connection).....	74
13	Glossary of terms and acronyms	75

1 Preface

1.1 What this document contains

This document describes how to install, configure and manage Caplin's DataSource for IDC product, version 4.4.

Note: In this document, DataSource for IDC is also referred to as "DS4IDC".

About Caplin document formats

This document is supplied in three formats:

- ◆ Portable document format (*.PDF* file), which you can read on-line using a suitable PDF reader such as Adobe Reader®. This version of the document is formatted as a printable manual; you can print it from the PDF reader.
- ◆ Web pages (*.HTML* files), which you can read on-line using a web browser. To read the web version of the document navigate to the *HTMLDoc_m_n* folder and open the file *index.html*.
- ◆ Microsoft HTML Help (*.CHM* file), which is an HTML format contained in a single file. To read a *.CHM* file just open it – no web browser is needed.

For the best reading experience

On the machine where your browser or PDF reader runs, install the following Microsoft Windows® fonts: Arial, Courier New, Times New Roman, Tahoma. You must have a suitable Microsoft license to use these fonts.

Restrictions on viewing .CHM files

You can only read *.CHM* files from Microsoft Windows.

Microsoft Windows security restrictions may prevent you from viewing the content of *.CHM* files that are located on network drives. To fix this either copy the file to a local hard drive on your PC (for example the Desktop), or ask your System Administrator to grant access to the file across the network. For more information see the Microsoft knowledge base article at <http://support.microsoft.com/kb/896054/>.

1.2 Who should read this document

This document is intended for System Administrators who need to install, configure, and manage DataSource for IDC.

1.3 Related documents

- ◆ **DataSource Overview**
A technical overview of Caplin DataSource.
- ◆ **Caplin Liberator 4.5 Administration Guide**
A technical overview of Caplin Liberator with instructions on how to configure Liberator to connect to a DataSource.
- ◆ **Caplin Xaqua: Monitoring And Management Overview**
Describes the Caplin Xaqua Management and Monitoring solution and its place in the Caplin Xaqua architecture.
- ◆ **Caplin Xaqua: Getting Started With The XMC**
Describes how to configure the Caplin Xaqua Management Console.

1.4 Typographical conventions

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

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

Note: Important Notes are enclosed within a box like this.
Please pay particular attention to these points to ensure proper configuration and operation of the solution.

Tip: Useful information is enclosed within a box like this.
Use these points to find out where to get more help on a topic.

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

1.5 Feedback

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

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

1.6 Acknowledgments

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

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

Solaris is a trademark of Sun Microsystems, Inc. in the U.S. or other countries.

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

1.7 Open Source Software

Caplin DataSource for IDC incorporates the following Open Source software:

Open Source item	Use in DS4IDC	Further information
OpenSSL	Where high security is required, DataSource connections can be configured to use the Secure Sockets Layer (SSL), providing an encrypted channel over which DataSource applications can exchange data.	www.openssl.org

2 Terminology

For a description of the technical terms used in this document, see the [Glossary of terms and acronyms](#) [75]. For a description of DataSource concepts and features, see the **Caplin DataSource Overview** document.

In this DS4IDC administration guide:

- ◆ A *DataSource application* is sometimes referred to as a *DataSource*.
- ◆ A *peer DataSource application* is sometimes referred to as a *peer*.

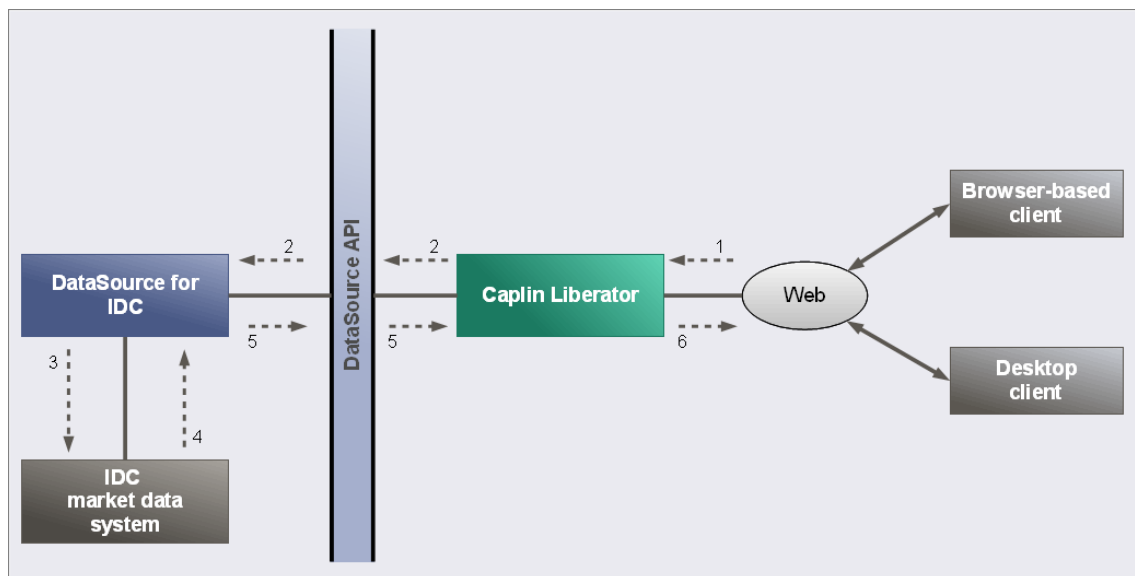
3 Introduction to DataSource for IDC

DS4IDC is a DataSource adapter that enables Caplin Liberator, or any Caplin product that uses the Caplin DataSource API, to request data from an IDC market data system. DS4IDC can then retrieve the requested data and forward it to Liberator (or other product) using the DataSource protocol.

DS4IDC is built on the standard DataSource for C SDK and can communicate with other DataSource applications.

3.1 Architecture

The following diagram shows DataSource for IDC in a market data distribution system.

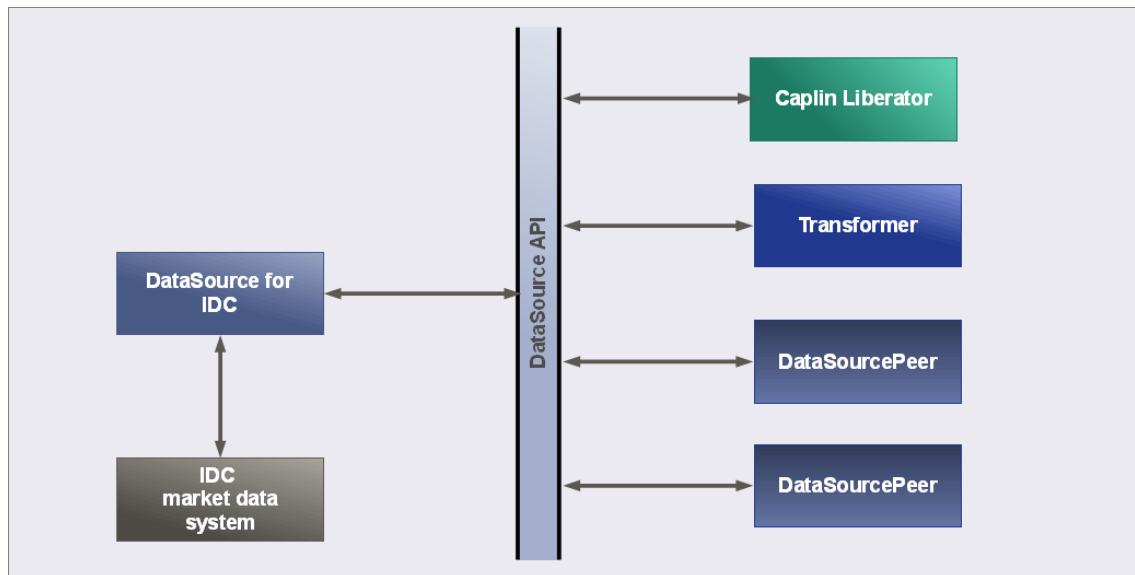


Market data distribution system incorporating a DataSource for IDC adapter

Typical data extraction and distribution:

1. A browser-based or desktop client application subscribes to data from Caplin Liberator.
2. Liberator subscribes to the data from the DS4IDC (using the DataSource API and protocol).
3. DS4IDC requests the data from the IDC market data system.
4. The IDC market data system sends an initial image of the requested data, and subsequent updates, to the DS4IDC.
5. DS4IDC forwards the initial image, and subsequent updates, to Caplin Liberator (using the DataSource API and protocol).
6. Caplin Liberator can then send the data to browser-based and desktop client applications over the Web.

DS4IDC can be configured to communicate with any DataSource application that supports the DataSource protocol. The remote DataSource applications that DS4IDC can communicate with are called DataSource peers.



Communicating with other DataSource applications

Connection requests

DataSource adapters are normally configured to initiate connection requests to Liberator or Transformer, rather than being configured to listen for connection requests. There are two advantages with this kind of configuration:

1. Liberator will not try to connect to DataSource adapters that are not running.
2. Firewalls – Liberator does not need to initiate a connection to a more 'internal' part of the network.

For further information, see [Connecting to a DataSource peer](#) in [Reviewing the supplied configuration](#).

3.2 Features

DS4IDC provides the following features.

Active DataSource

DS4IDC is an active DataSource. An active DataSource is one that keeps track of which records have been requested and sends updates for those objects only, rather than broadcasting all received data. An active DataSource also accepts discard requests, which tells the DataSource that updates for a record are no longer required.

See the [local-type](#)^[32] option of [add-peer](#)^[29] for further information

Configurable log cycling

DS4IDC creates log files that contain status messages and information about the data that has been sent and received. DS4IDC can be configured to cycle these log files based on time and/or size (for example, one log file for each day of the week).

See [Logging](#)^[39] for further information.

Name mapping

DS4IDC can be configured to re-format the names of records passed into it from Liberator or Transformer. This can be used to change RTTP record names to valid IDC record names, and can simplify the configuration required at Liberator or Transformer.

See [Name mappings](#)^[36] for further information.

Connecting to the IDC feed

DS4IDC can be configured to connect to a set of failover IDC feeds, but will only connect to one feed at any time.

3.3 About the data

IDC provides equity data that is transmitted as records, where the body of each record consists of fields. For example, a record containing equity data could have several price fields (for example, the last traded prices) together with time and date fields.

The symbol that identifies these records consists of an exchange identifier followed by the stock (ticker) name.

For example, the symbol `/755/E:VOD` identifies Vodafone shares traded on the London Stock Exchange (755 is the IDC source ID of the London Stock Exchange).

A symbolic name for the exchange, such as LSE for the London Stock Exchange, can be mapped to its IDC source ID (see [Specific configuration settings](#)^[13]). This allows subscriptions to be made to the DS4IDC using either the symbolic name of the exchange or the IDC source ID.

3.4 About configuration files

DS4IDC is configured by settings in a number of configuration files.

<i>ctfsrc.conf</i>	Defines the overall characteristics and functionality of the DS4IDC application. This file can 'include' other named configuration files that configure specific settings (see include-file ^[24]). All configuration items have default values and are therefore optional. Reference information for these configuration items can be found in in Reference: DataSource configuration ^[24] and Reference: DS4IDC configuration ^[54] .
<i>fields-ctfsrc.conf</i>	Defines the mapping between the field names and field numbers that are used by the DataSource communication protocol. This file is supplied with the installation kit, but is used to configure a Liberator or Transformer peer and not the DS4IDC.
<i>jmx.conf</i>	Configures the JMX monitoring module (see JMX configuration ^[53] and Java configuration ^[47]).
<i>sockmon.conf</i>	Configures the Socket monitoring module (see Sockmon configuration ^[52]).

Configuration files are located in the */etc* directory of the DS4IDC installation.

4 Installing DS4IDC

DS4IDC can be installed on a Linux or Solaris platform.

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

Linux example:

```
$ cd /opt
$ tar xzf /tmp/CTFsrc-4.4.0-i686-pc-linux-gnu.tar.gz
$ ln -s CTFsrc-4.4.0 CTFsrc
```

Solaris example:

```
$ cd /opt
$ uncompress /tmp/CTFsrc-4.4.0-sparc-sun-solaris2.8.tar.Z
$ tar xf /tmp/CTFsrc-4.4.0-sparc-sun-solaris2.8.tar
$ ln -s CTFsrc-4.4.0 CTFsrc
```

You should now have a directory structure like this (when installed to `/opt`):

<code>/opt/CTFsrc</code>	(application root)
<code>/opt/CTFsrc/bin</code>	(binary programs)
<code>/opt/CTFsrc/doc</code>	(documents and examples)
<code>/opt/CTFsrc/etc</code>	(startup and configuration)
<code>/opt/CTFsrc/var</code>	(log files)
<code>/opt/CTFsrc/lib</code>	(library files)

- If you want DS4IDC to start automatically when the system boots, create a link to the startup script from your startup directory. Consult your system administrator before you do this.

Example (Linux and Solaris):

```
$ cd /etc/rc3.d
$ ln -s /opt/CTFsrc/etc/ctfsrc S99ctfsrc
```

The name `S99ctfsrc` tells the system to run the startup script last. The `ctfsrc` part of the name must match the name of the application binary.

Tip: On other systems that use SYSV startup scripts this process should be similar.

5 Reviewing the supplied configuration

Before you run the installed DataSource for the first time, open the configuration file *ctfsrc.conf* and review the supplied configuration settings to ensure they are suitable for your installation and network configuration. This file can be found in the *etc* directory of the installed application (see [Installing DS4IDC](#) ^[9]).

The configuration settings in *ctfsrc.conf* specify:

- ◆ The DataSource applications (peers) that this DS4IDC communicates with.
- ◆ The IDC system that this DS4IDC communicates with.
- ◆ Log file and log level settings.

The file contains both [General configuration settings](#) ^[10] and [Specific configuration settings](#) ^[13].

Tip: When a configuration setting is not specified, the default value for that setting is applied.

5.1 General configuration settings

General configuration settings configure characteristics that are common to all DataSource applications. The configuration settings shown below are typical of the settings that you will find in your installation when DS4IDC is first installed.

General settings in *ctfsrc.conf* (typical)

```
#####  
# General datasrc configuration settings  
#####  
  
# The ID number of this DataSource  
datasrc-id      2  
  
# Add a peer that connection requests are sent to  
add-peer  
    addr          127.0.0.1  
    port          25000  
end-peer  
  
# Set the log level  
log-level      INFO
```

The following sections describe the general configuration settings shown above, with advice on the settings that you may need to change.

Connecting to a DataSource peer

These configuration items allow DS4IDC to communicate with a remote DataSource peer.

- ◆ [datasrc-id](#)^[27] Sets the ID number of the DS4IDC.

```
datasrc-id      2
```

The ID number uniquely identifies the DS4IDC when a connection request is sent to a remote DataSource.

If the ID number is not unique and the remote DataSource already has a connection to another DataSource with the same ID, then the original connection will be dropped and the new connection established.

You need to change **datasrc-id** if the supplied value is not unique in your network of connected DataSources.

- ◆ [add-peer](#)^[29] Specifies a peer DataSource application that DS4IDC can communicate with.

```
add-peer
      addr      127.0.0.1
      port      25000
end-peer
```

You need a separate **add-peer** entry for each DataSource peer that DS4IDC can communicate with. Typical DataSource peers are Caplin Liberator and Caplin Transformer.

If **addr** and **port** are specified in the configuration, then DS4IDC initiates all connection requests to the peer. In this case the address of the peer is set to 127.0.0.1 and the port that the peer listens on for connection requests is set to 25000.

Change the **addr** and **port** settings if the DataSource peer is configured to listen on a different address or port number.

To listen for connection requests rather than initiating the request, remove or comment out the **addr** and **port** entries and add a [remote-id](#)^[33] and optional [remote-name](#)^[33] entry to **add-peer**. You will also need to specify the network address that DS4IDC listens on for network connection requests (see [The network interface](#)^[26]).

Liberator and Transformer

To configure Liberator or Transformer to listen for connection requests from the DS4IDC (as configured by the [General configuration settings](#)^[10]), add the following lines to the Liberator or Transformer configuration.

```
datasrc-port 25000
add-peer
      remote-id  2
end-peer
```


When adding these lines:

- ◆ The **datasrc-port** setting in the Liberator or Transformer configuration must match the **add-peer port** setting in the DS4IDC configuration (in this case 25000).
- ◆ The **add-peer remote-id** setting in the Liberator or Transformer configuration must match the **datasrc-id** setting in the DS4IDC configuration (in this case 2).

In addition, add the following line to the Liberator configuration so that the field numbers of records received from the DS4IDC can be translated into field names.

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

You must also copy or link the file *fields-ctfsrc.conf* to the Liberator *etc* directory.

Setting the log level

The log level determines the severity of the errors and events that are logged by the DS4IDC.

- ◆ [log-level](#)³⁹ Sets the log level of the DS4IDC application.

```
log-level      INFO
```

The INFO log level reports serious network connection errors, critical errors that prevent the DS4IDC from running, data corruption errors, minor errors, and information regarding normal operation. Only change this value if you want to set a different log level.

See [log-level](#)³⁹ for a list of the permitted log levels.

5.2 Specific configuration settings

Specific configuration settings configure characteristics that are specific to a DS4IDC. The configuration settings shown below are typical of the settings that you will find in your installation when it is first installed.

Specific settings in ctfsrc.conf

```
add-csp-connection
    addr          10.5.7.110
    port          52200
    username      caplin
    password      caplin
end-csp-connection

add-exchange 755 LSE
add-exchange 558 NYSEC
add-exchange 340 AMEX
add-exchange 594 BATSE1
add-exchange 748 EUREX1
```

◆ [add-csp-connection](#) ⁵⁶

This configures the DS4IDC to attempt to connect to the IDC server at address (**addr**) 10.5.7.110 **port** 52200. The IDC server requires login credentials, and in this configuration the **username** and **password** are supplied as `caplin`, `caplin`. If the connection attempt and login is successful, stocks can be requested from the IDC server.

You need to change these settings to match the values required by your IDC server.

◆ [add-exchange](#) ⁵⁴

Maps an exchange symbol to its IDC source ID. In this case five exchange symbols have been mapped: the London Stock Exchange (`LSE`), the New York Stock Exchange Composite Only (`NYSEC`), the American Stock Exchange (`AMEX`), the BATS Europe Exchange (`BATSE1`), and the EUREX Exchange (`EUREX1`).

This allows DataSource peers (such as Liberator and Transformer) to request stocks by exchange symbol rather than IDC source ID (see [About the data](#) ⁷). Note that the IDC source ID can also be used in a data request, which allows stocks to be requested from exchanges that have not been mapped, without having to stop and reconfigure DS4IDC.

Add to these settings if you want to map other exchanges, or change the supplied mappings if your DataSource peers request stocks using different exchange symbols.

6 Running DS4IDC

DS4IDC is started by running a script provided with the application. The name of the script is *ctfsrc* and it is located in the *etc* directory of the installed application.

Tip: Before you run DS4IDC for the first time, you may want to open the file *ctfsrc.conf* and review the supplied configuration settings to ensure they are suitable for your installation (see [Reviewing the supplied configuration](#) ¹⁰³).

You can run the script from any directory.

- To start the DS4IDC from the installation directory, enter the following command:

```
./etc/ctfsrc start
```

- To stop the DS4IDC from the installation directory, enter the following command:

```
./etc/ctfsrc stop
```

To run the script from another directory, change the path to the script when you enter the *start* and *stop* commands.

7 Monitoring and management

The following sections describe how to monitor and manage DS4IDC.

7.1 The monitoring and management subsystem

DS4IDC supports monitoring and management via a plug-in monitoring subsystem. This is an additional licensable feature. The monitoring subsystem allows the administrator to monitor many different aspects of the DS4IDC, including the objects currently requested, the peers that are configured, and information about the connection to the IDC feed. Two monitoring plug-ins are 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 Xaqua Management Console (XMC) console uses this JMX monitoring plug-in. A number of modifications to the configuration file are needed in order to enable JMX monitoring. These modifications are documented in the Caplin Xaqua document **Getting Started With The XMC**.
- ◆ **Socket Monitoring (sockmon):** 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 **Caplin Xaqua Monitoring And Management Overview** document for more details.

7.2 The UDP command interface utility

The Caplin **udpsend** utility is used to send UDP messages over the UDP command interface to a running DataSource application. The utility is located in the *bin* directory of the DS4IDC installation.

UDP messages can be sent to change the log level of the DS4IDC and to control peer connections.

udpsend

This command sends a UDP message to the specified host and port number. The DataSource application that the message is intended for must be configured to listen for UDP messages (see [UDP command interface configuration](#) ^[46]).

Syntax

```
udpsend [-s <server-ip>] [-p <port>] message
```

Command options

Name	Type	Default	Description
-s	string	127.0.0.1	The IP address that the UDP message is sent to.
-p	integer	10001	The port that the UDP message is sent to.
message	string	<no default>	The message to send.

Permitted messages

Syntax	Description
log-level <new log level>	Dynamically changes the log level of the DataSource to the <new log level>. See the log-level ^[39] configuration item for a list of permitted log levels.
peer-reconnect peer addr-num	Attempt to reconnect with the specified peer after failover. 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. See permitted message arguments ^[17] in the following table.

Syntax	Description
<code>peer-status-up peer</code>	Connect to peer and accept connections. See permitted message arguments ^[17] in the following table.
<code>peer-status-down peer</code>	Disconnect from this peer and don't accept connections from this peer. See permitted message arguments ^[17] in the following table.

Permitted message arguments

Argument	Description
<code>peer</code>	DataSource peer index to connect to or disconnect from. This is not the DataSource ID, but the order of the peer's add-peer ^[29] entry in the configuration file. The first add-peer entry is index 0, the next add-peer entry is index 1, and so on.
<code>addr-num</code>	The index of the address in the failover list to reconnect to (see addr ^[29]). For example, in the configuration: <pre>add-peer addr server1 server2 server3</pre> <code>server1</code> is index 0, <code>server2</code> is index 1, and <code>server3</code> is index 2. Defaults to the first address in the list.

Example

If the DataSource listens on port 1247 for UDP messages, then the following command changes the log level of the DataSource to `WARN`.

```
./bin/udpsend -p 1247 log-level WARN
```

7.3 Viewing packet logs with the logcat utility

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

Packet logs are in binary format and must be viewed using the **logcat** utility, which is used in the same way as the standard **cat** command. **logcat** is located in the *bin* directory of the DS4IDC installation.

The **logcat** utility takes arguments, as listed in the following table.

logcat arguments

Argument (short, long)	Description
-h, --help	Displays detailed help on logcat options.
-F, --print-field-names	Print the field names.
-f, --fields-file	Location and name of the fields file. Default value is <i>fields.conf</i> in the current directory.
-i, --print-info	Displays the version, type and source of the log.
-l, --print-flag-names	Prints the flags.
-t, --type	Forces logcat to process a particular type of file. An additional argument is required to specify the type of file, which must be 'packet' (as in <code>logcat -t packet mypacket.log</code>).
-v, --log-version	Displays the version of the log.
-z, --timezone	Sets all times in the log to the specified time zone. To find the required time zone, look in the system folder <i>zoneinfo</i> , sometimes found at <i>/usr/share/lib/zoneinfo</i> or <i>/usr/share/zoneinfo</i> . The timezone offset is that of the local machine that the logs were written on.

Examples

The following example displays information about the version, type and source of the log.

```
./bin/logcat -i packet-ctfsrc.log
```

The response would look something like this:

```
Logcat: Log Type 'packet' Version 4 created by '<datasourcename>' in timezone  
'Europe/London'
```

The next example displays the content of the log.

```
./bin/logcat packet-ctfsrc.log
```

The response would look something like this:

```
Logcat: Log Type 'packet' Version 4 created by '<datasourcename>'  
2010/06/25-16:46:52.528 +0100: 192.168.201.102 < PEERINFO 1 type2src-devsun1 0  
2010/06/25-16:46:52.528 +0100: 192.168.201.102 > PEERINFO 0 rttdpdevsun2 0  
2010/06/25-16:47:00.000 +0100: 192.168.201.102 > SUBJREQ 1 1 /I/VOD.L  
2010/06/25-16:47:00.000 +0100: 192.168.201.102 > SUBJREQ 1 1 /I/BP.L
```

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

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

To view very large packet logs it is possible to split the log into smaller files using the standard Linux 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.

```
./bin/logcat -t packet packet-xab
```


7.4 Log file cycling

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

- ◆ Define a maximum file size above which the log file is cycled (see [log-maxsize](#)^[41]).
- ◆ Define a fixed time at which the log file is cycled (see [log-cycle-time](#)^[41]).
- ◆ Define a time interval after which the log file is cycled (see [log-cycle-period](#)^[42]).
- ◆ Define a combination of the above – the log file is cycled when any one of the criteria is met.

By default all log files are cycled at 04:00 hours each day, so that a separate log file of each type is created each day.

8 Reference: Types of configuration item

Three different types of configuration item can be used to configure a DataSource application:

- ◆ [Single value configuration items](#) ²¹
- ◆ [Multi value configuration items](#) ²¹
- ◆ [Configuration items with nested options](#) ²²

Some of these configuration items allow you to specify a list of one or more values (see [Lists](#) ²³).

8.1 Single value configuration items

Single value configuration items set something to a single value. An example of this type of configuration item is **datasrc-port**, which specifies the network port that the application listens on for DataSource messages and connection requests.

Example

```
datasrc-port 22001
```

In this case the listening port is set to 22001.

If present in the configuration file, a single value configuration item must define the value to be assigned. If not present in the configuration file, the configuration item is assigned a default value.

In the reference sections of this document, further information about a single value configuration item is shown in a table like this.

Type	Default	Permitted values
The type of value, such as string or integer.	The default value, or <no default> if there is no default value.	Restrictions on the permitted values for this configuration item.

8.2 Multi value configuration items

Multi value configuration items set the value of two or more related items. An example of this type of configuration item is **add-field**, which maps a field name (a string) to a field number (an integer).

Example

```
add-field Bid 22
```

In this case the `Bid` field is mapped to 22.

If a multi value configuration item is present in the configuration file, values must be defined for all related items. Values are assigned according to the position of the value in the list of ordered values. If the configuration item is not present in the configuration file, related items are assigned default values.

In the reference sections of this document, named position indicators in the syntax definition indicate how values of related items must be ordered. In the example **add-field** configuration item above, the syntax would look something like this.

Example Syntax for the add-field configuration item

```
add-field  FieldName  FieldNumber
```

Further information about the items that these named position indicators represent are shown in a table like this.

Options

Name	Type	Default	Description
The name of the position indicator, such as <code>FieldName</code> .	The type of value, such as string or integer.	The default value, or <no default> if there is no default value.	Describes the item represented by the named position indicator, and defines any restrictions on its value.

8.3 Configuration items with nested options

This type of configuration item takes nested options, where each nested option can be any one of the four types of configuration item. An example of this type of configuration item is **add-peer**, which specifies options for connecting to a remote DataSource.

Example

```
add-peer
  addr      liberator.example.com
  port      25000
end-peer
```

In this case the address of the remote DataSource is set to `liberator.example.com`, and the port that it listens on to `25000`.

In the reference sections of this document, the syntax definition specifies the configuration items that can be nested. In the example **add-peer** configuration item above, the syntax would look something like this.

Example Syntax for the add-peer configuration item

```
add-peer
  addr      <value>
  port      <value>
end-peer
```

This syntax indicates that if the **addr** and **port** configuration items are present in the configuration, then a single value must be defined for each of these items. The angle brackets indicate that `<value>` is only a place holder for the single value, and not the value itself.

8.4 Lists

Some configuration items allow you to specify a list of one or more values, where each value is the same type. An example is [addr](#)²⁹, which defines a list of addresses that the DataSource will attempt to connect to.

In the reference sections of this document, list types are identified by the text 'list' in the Type column of the table that describes the configuration item.

List Type	Description	Example
string list	A space separated list of one or more strings.	A space separated list of IP addresses. 127.0.0.0 192.255.129.1
integer list	A space separated list of one or more integers.	A space separated list of port numbers. 22001 22002

9 Reference: DataSource configuration

This section provides reference information for the configuration items that configure characteristics common to all DataSource adapters. Reference information about DS4IDC specific configuration items can be found in [Reference: DS4IDC specific configuration](#)^[54].

9.1 include-file

Specifies that another configuration file has configuration settings for the DS4IDC. A configuration file can have several **include-file** entries, each specifying a different configuration file.

Included files can be nested such that file A includes file B, and file B includes file C, but circular references must be avoided (file C must not include file A or file B).

Type	Default	Permitted values
string	<no default>	<p>The name of the configuration file that contains the configuration settings. The name can be an absolute or relative file path.</p> <p>The substitution characters %a and %h can be used in the string value of the file name, where %a represents the name of the application binary and %h the name of the host machine. Substitution characters can be used to include application or host specific configuration files.</p>

Example

```
include-file myfile-%a-%h.conf
```

9.2 license-file

Specifies the name of the DS4IDC license file. DS4IDC will not run without a valid license file.

Tip: For further information about configuration items related to licensing, refer to the document **Caplin Platform: Guide to User Licensing**.

Type	Default	Permitted values
string	license.conf	This can be an absolute or relative file path.

9.3 process-usage-period

Defines the time interval in seconds at which the DS4IDC CPU time counters **user-cputime-total** and **system-cputime-total** are updated. These counters are available to the JMX monitoring subsystem (see [System information \(ctfsrc.server.system\)](#) for further information).

A typical configuration does not need to explicitly define this configuration item, as the counters are updated using the default value.

Type	Default	Permitted values
float	10.000000	Any non-negative value.

9.4 The network interface

Network interface configuration items specify the network address and port number that DS4IDC listens on for connection requests from other DataSource applications.

datasrc-interface

The network interface to listen on for connection requests from other DataSource applications.

Type	Default	Permitted values
string	<all available interfaces>	Any valid IP address.

If the **datasrc-interface** is not specified, then the DataSource will listen for connections on all available interfaces.

datasrc-port

The network (TCP/IP) port to listen on for connection requests from other DataSource applications.

Type	Default	Permitted values
integer	0	Any valid port number. Min value: 0 Max value: 65335

The default value of 0 means that connections cannot be made to this DataSource, as the DataSource will not listen for connections on any port.

9.5 DataSource identifiers

DataSource identifiers specify how DS4IDC identifies itself when it communicates with other DataSource applications.

datasrc-name

The name that identifies the DS4IDC when it communicates with other DataSource applications. The name should preferably be unique in the network of connected DataSources, as it can appear in log files and is available to monitoring utilities.

Type	Default	Permitted values
string	%a-%h	Any string. The substitution characters %a represent the name of the application binary, and %h the name of the host machine, and can be used in the string value of the name.

If the **local-name** option of [add-peer](#)^[29] is set for this DS4IDC, then it overrides the **datasrc-name** setting.

datasrc-id

An ID number that uniquely identifies the DS4IDC when it communicates with other DataSource applications.

Type	Default	Permitted values
integer	0	<p>Any non-negative integer that uniquely identifies the DS4IDC in the network of connected DataSources.</p> <p>If the local-id^[31] of add-peer^[29] is <i>not</i> set for this DS4IDC, then the value must match the <i>remote-id</i> of add-peer in all peer DataSource configurations.</p>

If the [local-id](#)^[31] of [add-peer](#)^[29] is set for this DataSource, then it overrides the **datasrc-id** setting for that peer connection.

9.6 Field mappings

Field mappings map field names to field numbers. When a DataSource sends a message containing fields to another DataSource application, it is the field numbers and not the field names that are sent in the DataSource message.

add-field

Maps a field name to a field number. There can be a maximum of 32,768 field mappings in a DataSource configuration.

Syntax

```
add-field FieldName FieldNumber
```

Options

Name	Type	Default	Description
FieldName	string	<no default>	The name of the field.
FieldNumber	integer	<no default>	The number of the field. Min value: -65535 Max value: 65535

Example of mandatory field mappings

```
add-field Bid 22
```

In this example the `Bid` field is mapped to 22.

9.7 DataSource peers

Each remote DataSource application that DS4IDC can communicate with must be identified by an [add-peer](#)^[29] configuration item. There can be a maximum of 63 **add-peer** entries in a DS4IDC configuration.

add-peer

Contains nested configuration items identifying a remote DataSource application that DS4IDC can communicate with.

Nested Syntax

```
add-peer
  addr                <value>
  heartbeat-slack-time <value>
  heartbeat-time      <value>
  local-id            <value>
  local-name          <value>
  local-type          <value>
  port                <value>
  remote-id           <value>
  remote-name         <value>
  remote-type         <value>
end-peer
```

addr

(A child of [add-peer](#)^[29])

A space-separated list of addresses to connect to.

This configuration item is used in conjunction with [port](#)^[32] to specify the address and port number that connection requests are sent to when DS4IDC attempts to initiate a connection with a DataSource peer.

The connection request is sent to the first address and port in the list, and if this request fails, a connection request is sent to the next address and port in the list. Addresses and ports are applied in pairs for each connection attempt (for example, *addr1* with *port1* and *addr2* with *port2*). If a **port** is defined but no corresponding **addr** is defined, the default address is used in the request.

This failover strategy is repeated until a connection is established with the remote DataSource (see [Peer connection failover strategy](#)^[35] for an example failover configuration and further details about the failover strategy).

Only specify **addr** and **port** in the configuration of the DS4IDC if the DS4IDC initiates the connection request. The remote DataSource must be configured to accept connection requests (see [datasrc-interface](#)^[26] and [datasrc-port](#)^[26]).

Type	Default	Permitted values
string list	127.0.0.1	Any valid IP address or host name.

heartbeat-slack-time

(A child of [add-peer](#) ^[29])

When DS4IDC does not receive an expected peer DataSource heartbeat, it waits **heartbeat-slack-time** seconds for a heartbeat to arrive before attempting to connect to another DataSource peer. The connection attempt is made using the failover strategy described in [Peer connection failover strategy](#) ^[35].

For example, if **heartbeat-slack-time** is set to 5 seconds and [heartbeat-time](#) ^[30] to 1 second, and a heartbeat is not received for 6 seconds, DS4IDC will attempt to connect to the next peer in the configured list of peers (see [addr](#) ^[29]).

DataSource peers do not compare **heartbeat-slack-time** values.

Type	Default	Permitted values
integer	2	Any non-negative integer.

heartbeat-time

(A child of [add-peer](#) ^[29])

Time in seconds between DataSource heartbeats. The two peers involved in a DataSource connection compare **heartbeat-time** values and use the lowest.

The heartbeat tells connected DataSource peers that the DS4IDC is still running (see [heartbeat-slack-time](#) ^[30]).

Type	Default	Permitted values
integer	<disabled>	Any non-negative integer. The default value disables the heartbeat for this peer.

local-id

(A child of [add-peer](#)^[29])

The ID number of this DS4IDC. The ID number is sent to the remote DataSource when the connection is established.

The **local-id** in this configuration must match the [remote-id](#)^[33] in the remote DataSource configuration.

Set **local-id** to a value other than the default if you want additional connections to the remote DataSource (each configured by a separate [add-peer](#)^[29] entry).

Type	Default	Permitted values
integer	datasrc-id ^[27]	Any non-negative integer that uniquely identifies the DS4IDC in the network of connected DataSources.

local-name

(A child of [add-peer](#)^[29])

The name of this DS4IDC. The name is sent to the remote DataSource when the connection is established.

The name should preferably be unique in the network of connected DataSources as it can appear in log files and is available to monitoring utilities.

Type	Default	Permitted values
string	datasrc-name ^[27]	Any string.

local-type

(A child of [add-peer](#)^[29])

The type of this DS4IDC. The type is sent to the remote DataSource when the connection is established.

Type	Default	Permitted values
string	broadcast	One of the following: broadcast Broadcast DataSource, no contributions. active Active DataSource, no contributions. contrib Broadcast DataSource, with contributions. active contrib Active DataSource, with contributions.

port

(A child of [add-peer](#)^[29])

A space-separated list of ports to connect to.

This configuration item is used in conjunction with [addr](#)^[29] to specify the address and port number that connection requests are sent to when the DS4IDC attempts to initiate a connection with the remote DataSource.

At least one port number must be defined, otherwise DS4IDC does not initiate the connection.

Values

Type	Default	Permitted values
integer list	<no default>	Any valid port number. Min value: 0 Max value: 65335

remote-id

(A child of [add-peer](#)^[29])

The ID number of the remote DataSource that this DS4IDC can communicate with. Only specify **remote-id** if this DS4IDC listens for connection requests from the remote DataSource (see [addr](#)^[29] and [port](#)^[32]).

The **remote-id** in this configuration must match the [local-id](#)^[31] in the remote DataSource configuration.

Type	Default	Permitted values
integer	1	Any non-negative integer.

remote-name

(A child of [add-peer](#)^[29])

The name of the remote DataSource that this DS4IDC can communicate with. This value gets overridden by [local-name](#)^[31] in the remote DataSource configuration when the peer connection is established. **remote-name** can appear in log files and is available to monitoring utilities.

Type	Default	Permitted values
string	datasrc1	Any string.

remote-type

(A child of [add-peer](#)^[29])

The type of the remote DataSource. This value gets overridden by [local-type](#)^[32] in the remote DataSource configuration when the peer connection is established. **remote-type** can appear in log files and is available to monitoring utilities.

Type	Default	Permitted values
string	broadcast	One of the following: broadcast Broadcast DataSource, no contributions. active Active DataSource, no contributions. contrib Broadcast DataSource, with contributions. active contrib Active DataSource, with contributions.

Peer connection failover strategy

DS4IDC can be configured to connect to one of several DataSource peers using a failover strategy.

The failover strategy is:

1. Attempt to connect to each DataSource peer in the configured failover list, with a one second interval between each attempt, for the first ten connection attempts.
2. If a connection is not established, attempt to connect to each peer in the configured failover list, with a two second interval between each attempt, for the next ten connection attempts.
3. If a connection is not established, repeat step (2) first with a four second interval, then an eight second interval, and then a 16 second interval between each connection attempt.
4. If a connection is not established, attempt to connect to each peer in the configured failover list, with a 32 second interval between each attempt, until a connection is established.

This failover strategy also applies if DS4IDC is configured to monitor the connection and the established connection is lost (see [heartbeat-time](#)^[30] and [heartbeat-slack-time](#)^[30]).

The following example configures DS4IDC to connect to one of two DataSource peers.

Example configuration

```
add-peer
    addr          10.5.7.110
    port          25000 7900
end-peer
```

In this case the first failover peer is at address ([addr](#)^[29]) 10.5.7.110 [port](#)^[32] 25000. Because two ports are specified (25000 and 7900) but only one address (10.5.7.110), the second failover peer is at the default address 127.0.0.1 but on configured **port** 7900.

Note that if two addresses had been specified but only one port, the second address would be ignored and DS4IDC would only be configured to connect to one peer. This is because a port that is not configured has no default value.

9.8 Name mappings

DS4IDC can be configured to change the names of records that are passed into it from Liberator or Transformer. This can be used to change RTTP record names to valid IDC record names, and to simplify the configuration required at Liberator or Transformer.

For example, if the DS4IDC provides instrument prices to Liberator from three exchanges: London (LSE), New York Composite (NYSE), and the American (AMEX), the data service for these exchanges could be configured in one of two ways.

Tip: See the **Caplin Liberator Administration Guide** document for further information about Liberator data services.

The first way to configure the data service uses multiple **include-patterns** in the Liberator configuration, one for each exchange.

Example using multiple include-patterns

```
include-pattern    "^/LSE/"
include-pattern    "^/NYSE/"
include-pattern    "^/AMEX/"
```

With this configuration, subscriptions to instruments from these exchanges are sent to the DataSource that supplies the data service for these **include-patterns**, which in this case is the DS4IDC. No name mappings are required at the DS4IDC with this configuration.

The second way to configure the data service uses a single **include-pattern** in the Liberator configuration.

Example using a single include-pattern

```
include-pattern    "^/C/"
```

With this simplified Liberator configuration, subscriptions to instruments that are prefixed by `/C/` are sent to the DataSource that supplies the data service for this **include-pattern**. In this case the client application would prefix `/C/` to the subject of all London, New York Composite, and American instruments. For example, instead of subscribing to `/LSE/VOD/`, the client application would subscribe to `/C/LSE/VOD/`.

DS4IDC must now be configured to change record names to remove the `/C/` prefix (see [add-pattern](#)³⁷) before it requests the data from the IDC market data system.

add-pattern

Maps a search pattern to a replacement pattern. If the name (subject) of a record matches the search pattern, then the replacement pattern is applied to the record name.

Both the search pattern and the replacement pattern can include the * wildcard character to match any string of characters.

Syntax

```
add-pattern <search pattern> <replacement pattern>
```

Options

Name	Type	Default	Description
<search pattern>	string	<no default>	The pattern to search for in a record.
<replacement pattern>	string	<no default>	The new pattern of the record if the search pattern is found.

The following example configures DS4IDC to remove the /C/ prefix from all record names that have that prefix.

Example pattern mapping

```
add-pattern      /C/*      /*
```

This would change the record name /C/LSE/VOD/ to /LSE/VOD/.

9.9 Heartbeats

A DS4IDC can send heartbeat records to connected DataSources to inform them that the session is alive. This allows Caplin Liberator to alert client processes if data becomes stale.

This heartbeat is in addition to the heartbeats sent to DataSource peers (see [heartbeat-time](#) ³⁰)

heartbeat-symbol

The symbol under which the heartbeat is sent, and the symbol that Caplin Liberator and client applications must subscribe to if they want to be sent heartbeats.

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

Type	Default	Permitted values
string	NULL	Any valid string. The convention is to use the string /HBT/<machine-name>-ctfsrc.

heartbeat-symbol-time

Frequency of the heartbeat in seconds.

Type	Default	Permitted values
float	30.0	Any non-negative value.

9.10 Logging

Logging configuration items specify how event and packet logs are recorded.

log-level

Determines the severity of the errors and events that are recorded in event log files.

Type	Default	Permitted values
string	INFO	As defined in the following Permitted log levels table.

Permitted log levels

Value	Description
DEBUG	Reports all errors and events. (Warning – this level can produce very large log file sizes.)
INFO	Reports events and information regarding normal operation, and all errors included in the WARN, NOTIFY, ERROR and CRIT log levels.
WARN	Reports minor errors, and all errors included in the NOTIFY, ERROR, and CRIT log levels.
NOTIFY	Reports errors regarding data corruptions, and all errors included in the ERROR and CRIT log levels.
ERROR	Reports serious errors regarding network connections, and all errors included in the CRIT log level.
CRIT	Reports critical errors that prevent the DataSource running. (This level produces the smallest log file sizes).

If the UDP message interface is enabled, then the logging level can be changed when the DS4IDC is running (see [The UDP command interface utility](#) ¹⁶).

log-dir

The path to the log file directory. By default, all log files will be created in this directory (see [datasrc-pkt-log](#)^[45]).

Type	Default	Permitted values
string	%r/var	Any valid directory path. The substitution characters %r represent the current working directory, and can be used in the string value of the path. When the startup script is used to start the DS4IDC (see Running DS4IDC ^[14]), the current working directory is set to the installation directory .

log-cycle-suffix

A format string that defines the suffix appended to the filename of log files when the log files are cycled. The suffix is created from the format string by the operating system function **strftime** (see <http://linux.die.net/man/3/strftime> for further information about **strftime**).

If a log file with the same name already exists, it will be overwritten when the log file is cycled.

Type	Default	Permitted values
string	%u	See http://linux.die.net/man/3/strftime . Some of the format strings that can be used are shown in the following Common permitted format strings table. The suffix can also be plain text (see the example in log-maxsize ^[41]).

The following table shows some of the permitted **strftime** format strings.

Common permitted format strings

strftime format string	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).

strftime format string	Meaning
%M	The minute as a decimal number (range 00 to 59).
%Y	The year as a decimal number including the century.

log-maxsize

The maximum size of each log file, in bytes. Log files are cycled when they exceed this size; therefore a value of 0 means log files are cycled every time they are checked (see [Log file cycling](#)^[20]).

Type	Default	Permitted values
integer	0	Any non-negative integer.

Example configuration

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

In this example configuration, each log file is checked every 30 minutes and moved to *<logfile>.old* if it is bigger than 1,024,000 bytes.

log-cycle-time

The time that log files are cycled, in minutes from midnight. The default of 240 means that log files cycle at 0400 hours (see [Log file cycling](#)^[20]).

Type	Default	Permitted values
integer	240	Any non-negative integer.

log-cycle-period

Interval between the cycling of log files, in minutes. The default of 1440 means that the interval is 24 hours (that is, daily). See [Log file cycling](#) ^[20].

Type	Default	Permitted values
integer	1440	Any non-negative integer.

log-cycle-offset

Specifies how many minutes to take off the current time when creating the log file suffix (see [log-cycle-suffix](#) ^[40]). The default value of 1440 means that the time to take off is 24 hours (that is, the log file prefix is the previous day).

Type	Default	Permitted values
integer	1440	Any non-negative integer.

add-log

Overrides the global log option settings for a named log file.

You can either override all global settings or only particular settings. For example, you could set `name` and `maxsize` to set the maximum size of the named log file, but leave other values (such as log cycle time) at the global setting.

Nested Syntax

```
add-log
    level           <value>
    maxsize         <value>
    monitor-level   <value>
    name            <value>
    offset          <value>
    period          <value>
    suffix          <value>
    time            <value>
end-log
```

level

(A child of [add-log](#) ⁴²)

Determines the severity of the errors and events that are recorded in the log (only valid for the event log).

Type	Default	Permitted values
string	INFO	See log-level ³⁹ .

maxsize

(A child of [add-log](#) ⁴²)

Maximum size of the log file in bytes. The log file is cycled if it exceeds this size; therefore a value of 0 means the log file will cycle every time it is checked (see [Log file cycling](#) ²⁰).

Type	Default	Permitted values
integer	0	Any non-negative integer.

monitor-level

(A child of [add-log](#) ⁴²)

Determines the severity of the errors and events that are sent to the monitoring client (see [Monitoring and management subsystem](#) ¹⁵).

Type	Default	Permitted values
string	NOTIFY	See log-level ³⁹ .

name

(A child of [add-log](#)^[42])

Name of the log to cycle. If no value is entered, the **add-log** settings are not applied to any log file.

Type	Default	Permitted values
string	<no default>	event_log the event log jmx_log the JMX log packet_log the packet log sockmon_log the sockmon log

offset

(A child of [add-log](#)^[42])

Specifies how many minutes to take off the current time when creating the log file suffix (see [suffix](#)^[45]). The default value of 1440 means that the time to take off is 24 hours (that is, the log file prefix is the previous day).

Type	Default	Permitted values
integer	1440	Any non-negative integer.

period

(A child of [add-log](#)^[42])

Interval between the cycling of the log file, in minutes. The default of 1440 means that the interval is 24 hours (that is, daily). See [Log file cycling](#)^[20].

Type	Default	Permitted values
integer	1440	Any non-negative integer.

suffix

(A child of [add-log](#)^[42])

Defines the suffix that is appended to the filename of the log file when the log file is cycled (see [log-cycle-suffix](#)^[40] for further information).

time

(A child of [add-log](#)^[42])

The time that the log file is cycled, in minutes from midnight. The default of 240 means the log file will cycle at 0400 hours (see [Log file cycling](#)^[20]).

Type	Default	Permitted values
integer	240	Any non-negative integer.

datasrc-pkt-log

The path to the packet log file. The packet log file is a binary file that contains a record of all data sent and received by the DS4IDC.

Type	Default	Permitted values
string	packet-%a.log	<p>The path to the packet log file can either be an absolute path or relative to log-dir^[40].</p> <p>The substitution characters %a represent the name of the application binary, and can be used in the string value of the path.</p>

You must use Caplin's [logcat](#)^[18] utility to read the contents of a packet log file.

9.11 UDP command interface configuration

The following configuration items configure the interface and port that DS4IDC listens on for [UDP messages](#)¹⁶.

udp-interface

The network interface to listen on for UDP messages.

Type	Default	Permitted values
string	<all available interfaces>	Any valid IP address.

If not specified then DS4IDC listens for UDP messages on all interfaces.

udp-port

The network port to listen on for UDP messages.

Type	Default	Permitted values
integer	-1	The network (TCP/IP) port number. Min value: 0 Max value: 65335

The default value of -1 disables UDP messages.

9.12 Java configuration

Java must be configured if JMX Monitoring is enabled (see [Monitoring and management subsystem](#)^[15]).

jvm-location

The location of the JVM file *libjvm.so*.

Type	Default	Permitted values
string	<no default>	The absolute path including the <i>.so</i> suffix.

The following example sets the location of the JVM.

```
jvm-location /usr/local/java/jre/lib/i386/client/libjvm.so
```

jvm-options

Adds a standard startup option for the JVM. More than one startup option can be specified, each by a different **jvm-options** entry.

Type	Default	Permitted values
string list	<no default>	Any valid JVM startup option.

The following example sets some standard startup options.

```
# Configure the rmi client port here
jvm-options -Drmi.client.port=46000

# Set the heap size
jvm-options -Xms256m
```

jvm-global-classpath

Location of the global classpath. These classes are available to all Java modules loaded by the DS4IDC (see [add-javaclass](#) ^[48]).

More than one global classpath can be specified, each by a different **jvm-global-classpath** entry.

Type	Default	Permitted values
string list	%r/lib/java	<p>The substitution characters %r represent the current working directory, and can be used in the string value of the classpath.</p> <p>The current working directory is set to the installation directory of the DS4IDC when the startup script is used to start the DS4IDC (see Running DS4IDC ^[14]).</p>

The following example sets some global class paths.

```
# JARs required in the startup global classpath for the JVM
jvm-global-classpath  %r/lib/java/jmx-default-classloader.jar
jvm-global-classpath  %r/lib/java/javaauth.jar
jvm-global-classpath  %r/lib/java/common-jmx.jar
```

add-javaclass

Identifies the class of a Java module to load.

Nested Syntax

```
add-javaclass
  class-name      <value>
  class-id        <value>
  classpath       <value>
end-javaclass
```

The following example adds the classpath of the JMX monitoring module.

```
# Required by JMX monitoring module
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
```

class-name

(A child of [add-javaclass](#) ⁴⁸)

The fully qualified class name of the Java module.

Type	Default	Permitted values
string	<no default>	A fully qualified class name.

class-id

(A child of [add-javaclass](#) ⁴⁸)

A short identifier for the Java class.

When the module to load is JMX, the value of **class-id** must match the value of **jmx-classid** in the configuration file *jmx.conf* (normally set to `jmx`). For further information, see the Caplin Xaqua document **Getting Started With The XMC**.

Type	Default	Permitted values
string	NULL	Any string.

classpath

(A child of [add-javaclass](#) ⁴⁸)

The Java classpath. More than one classpath can be specified, each by a different **classpath** entry.

Type	Default	Permitted values
string	NULL	<p>Any valid class path.</p> <p>The substitution characters %r represent the current working directory, and can be used in the string value of the classpath.</p> <p>The current working directory is set to the installation directory of the DS4IDC when the startup script is used to start the DS4IDC (see Running DS4IDC ¹⁴).</p>

9.13 Monitoring configuration

Configures DS4IDC to load a particular monitoring module. See [Monitoring and management subsystem](#) [15] for further information about the available monitoring modules and how to configure them.

monitor-module

The name of the monitoring module that DS4IDC loads.

Type	Default	Permitted values
string	NULL	One of the following: sockmon Loads the socket monitoring module. jmx Loads the JMX monitoring module.

monitor-moddir

The path to the monitoring module that DS4IDC loads.

Type	Default	Permitted values
string	%r/lib	Any valid path. The substitution characters %r represent the current working directory, and can be used in the string value of the directory path. The current working directory is set to the installation directory of the DS4IDC when the startup script is used to start the DS4IDC (see Running DS4IDC [14]).

add-monuser

Specifies the credentials that allow a monitoring client application to log into the DS4IDC.

Nested Syntax

```
add-monuser
  user      <value>
  pass      <value>
end-monuser
```

user

(A child of [add-monuser](#) ⁵¹)

The username that the monitoring client application uses to log in to the DS4IDC.

If **user** and [pass](#) ⁵¹ are not specified, then the DS4IDC accepts monitoring login requests from any user.

Type	Default	Permitted values
string	<any>	Any string.

pass

(A child of [add-monuser](#) ⁵¹)

The password that the monitoring client application uses to log in to the DS4IDC.

If [user](#) ⁵¹ and **pass** are not specified, then the DS4IDC accepts monitoring login requests from any user.

Type	Default	Permitted values
string	<any>	Any string.

9.14 Sockmon configuration

Configures the socket monitoring module (see [Monitoring configuration](#) ⁵⁰).

sockmon-interface

The interface to listen on for sockmon connections. When present in the configuration, **sockmon-interface** must be defined in the file *sockmon.conf*.

Type	Default	Permitted values
string	<all available interfaces>	Any valid IP address.

sockmon-port

The port to listen on for sockmon connections. When present in the configuration, **sockmon-port** must be defined in the file *sockmon.conf*.

Type	Default	Permitted values
integer	10000	Any valid port number. Min value: 0 Max value: 65335

log-level

Determines the severity of the errors and events that are recorded in the sockmon log file. When present in the configuration, **log-level** must be defined in the file *sockmon.conf*.

Type	Default	Permitted values
string	INFO	See log-level ³⁹ .

9.15 JMX configuration

Configures the JMX monitoring module (see [Monitoring configuration](#) ^[50]).

jmx-classid

The class ID of the Java module required by the JMX monitoring module. When present in the configuration, **jmx-classid** must be defined in the file *jmx.conf*.

The value of **jmx-classid** must match the value of [class-id](#) ^[49] in the DS4IDC configuration (see [add-javaclass](#) ^[48]).

Type	Default	Permitted values
string	jmx	Any string.

log-level

Determines the severity of the errors and events that are recorded in the JMX log file. When present in the configuration, **log-level** must be defined in the file *jmx.conf*.

Type	Default	Permitted values
string	INFO	See log-level ^[39] .

rmi-registry-port

The port to listen on for JMX connections. DS4IDC listens for JMX connections on this port on all available interfaces.

When present in the configuration, **rmi-registry-port** must be defined in the file *jmx.conf*.

Type	Default	Permitted values
integer	1099	Any valid port number. Min value: 0 Max value: 65335

10 Reference: DS4IDC specific configuration

This section provides reference information about DS4IDC specific configuration items. Reference information about general DataSource adapter configuration items can be found in [Reference: DataSource configuration](#)²⁴.

10.1 Exchange mappings

The IDC source ID that identifies an exchange can be mapped to a more user friendly symbolic name. In this way subjects can be requested from DS4IDC using either the source ID or the mapped symbolic name.

For example, if the source ID for the London Stock Exchange (755) is mapped to the symbolic name `LSE`, subscriptions that start with `/LSE/` or `/755/` will be requested from exchange 755 (LSE UK Equity Market Service Level 1).

add-exchange

Maps the IDC source ID of an exchange to a symbolic name that can be used in subscription requests.

Syntax

```
add-exchange SourceID SymbolicName
```

Options

Name	Type	Default	Description
SourceID	integer	<no default>	The IDC source ID of the exchange.
SymbolicName	string	<no default>	The symbolic name used in subscription requests.

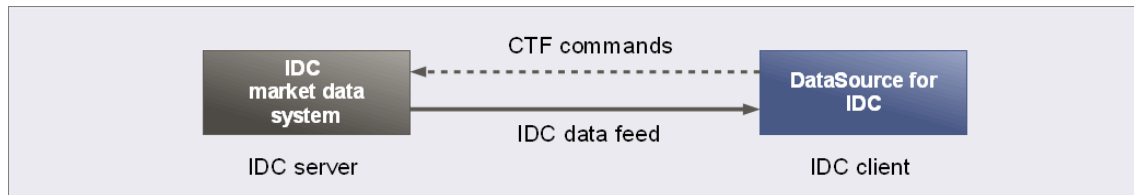
Example exchange mapping

```
add-exchange 755 LSE
```

In this example the source ID 755 is mapped to the symbolic name `LSE`.

10.2 Client side processor connection

In IDC terminology DS4IDC is known as the client, and the system that provides the IDC data feed is known as the server. Internally, IDC clients and servers communicate using the CTF protocol.



IDC client, server, and data feed

DS4IDC can be configured to connect to one of several IDC servers using a failover strategy, and to log in to the server once a connection is established.

The failover strategy is:

1. Attempt to connect to each server in the configured failover list, with a one second interval between each attempt, for the first ten connection attempts.
2. If a connection is not established, attempt to connect to each server in the configured failover list, with a two second interval between each attempt, for the next ten connection attempts.
3. If a connection is not established, repeat step (2) first with a four second interval, then an eight second interval, and then a 16 second interval between each connection attempt.
4. If a connection is not established, attempt to connect to each server in the configured failover list, with a 32 second interval between each attempt, until a connection is established.

This failover strategy also applies if DS4IDC is configured to monitor the connection and the established connection is lost (see [heartbeat-slack](#)^[57]).

The following example configures DS4IDC to connect to one of two failover servers.

Example configuration

```
add-csp-connection
  addr          10.5.7.110
  port          52200 6700
  username      admin
  password      admin
  monitor-interval 5.0
end-csp-connection
```

In this case the first failover server is at address (**addr**) 10.5.7.110 **port** 52200. Because two ports are specified (52200 and 6700) but only one address (10.5.7.110), the second failover server is at the default address 127.0.0.1 but on configured **port** 6700.

Note that if two addresses had been specified but only one port, the second address would be ignored and DS4IDC would only be configured to connect to one IDC server. This is because a port that is not configured has no default value.

The login **username** and **password** for each server is `admin`, `admin`, and monitoring statistics (**monitor-interval**) for the connection are configured to update every 5 seconds.

Tip: The `csp` in `add-csp-connection` is an abbreviation of Client Side Processor, alternative IDC terminology for an IDC client such as DS4IDC (see [Glossary of terms and acronyms](#)^[76]).

add-csp-connection

Configures DS4IDC to connect to an IDC server in a failover sequence.

Nested Syntax

```
add-csp-connection
  addr                <value>
  heartbeat-slack     <value>
  monitor-interval    <value>
  password            <value>
  port                <value>
  response-timeout    <value>
  username            <value>
end-csp-connection
```

addr

(A child of [add-csp-connection](#)^[56])

A space-separated list of addresses to connect to.

This configuration item is used in conjunction with [port](#)^[32] to specify the address and port number that connection requests are sent to when DS4IDC attempts to connect to an IDC server.

The connection request is sent to the first address and port in the list, and if this request fails, a connection request is sent to the next address and port in the list. Addresses and ports are applied in pairs for each connection attempt (for example, *addr1* with *port1* and *addr2* with *port2*).

This failover strategy is repeated until a connection is established with an IDC server (see [Client side processor connection](#)^[55] for an example failover configuration and further details about the failover strategy).

Type	Default	Permitted values
string list	127.0.0.1	Any valid TCP/IP address.

heartbeat-slack

(A child of [add-csp-connection](#)^[56])

When **heartbeat-slack** is set to a value greater than 0 seconds, DS4IDC subscribes to a CTF heartbeat symbol issued by the IDC server. The CTF heartbeat symbol is sent to subscribing IDC clients once every second, allowing these clients to monitor the state of the connection.

If DS4IDC does not receive an expected heartbeat, it waits **heartbeat-slack** seconds for a heartbeat to arrive before attempting to connect to another IDC server. The connection attempt is made using the failover strategy described in [Client side processor connection](#)^[55].

For example, if **heartbeat-slack** is set to 5 seconds and a heartbeat is not received for 6 seconds, DS4IDC will attempt to connect to the next IDC server in the configured list of servers (see [addr](#)^[56]).

The default value disables heartbeat subscription and monitoring.

Type	Default	Permitted values
float	0.000000	Any non-negative value.

monitor-interval

(A child of [add-csp-connection](#)^[56])

Monitoring statistics for the IDC server connection are updated every **monitor-interval** seconds (see [IDC server connection \(ctfsrc.csp connection\)](#)^[74] for further information).

Type	Default	Permitted values
float	5.000000	Any non-negative value.

password

(A child of [add-csp-connection](#)^[56])

The password that DS4IDC uses to log in to the IDC server (see [username](#)^[58]).

Type	Default	Permitted values
string	NULL	Any string.

port

(A child of [add-csp-connection](#)^[56])

A space-separated list of ports to connect to.

This configuration item is used in conjunction with [addr](#)^[56] to specify the address and port number that connection requests are sent to when DS4IDC attempts to connect to an IDC server.

At least one port number must be defined, otherwise a connection cannot be made to an IDC server and DS4IDC will terminate.

Type	Default	Permitted values
integer list	<no default>	Any valid port number. Min value: 0 Max value: 65335

response-timeout

Time to wait (in seconds) for a response to a CTF command, before disconnecting and trying to reconnect to the IDC server. This feature is disabled when **response-timeout** is set to 0.

Type	Default	Permitted values
float	5.000000	Any non-negative value.

username

(A child of [add-csp-connection](#)^[56])

The username that DS4IDC uses to log in to the IDC server (see [password](#)^[57]).

Type	Default	Permitted values
string	NULL	Any string.

10.3 IDC data logging

DS4IDC can be configured to log all data sent to, and received from, the IDC server.

ctf-logfile

The name of the binary log file in which DS4IDC records all CTF data sent to, and received from, the IDC server.

Note: Only set this configuration item if requested to by Caplin Support. The log file can become very large and is only intended to be read by Caplin Support.

Type	Default	Permitted values
string	NULL	The name can include a file path that is either absolute or relative to log-dir ^[40] . The default means CTF data is not logged.

The following example sets the name of the log file to *idc_feed.log* and the location of the log file to [log-dir](#)^[40].

```
ctf-logfile idc_feed.log
```


11 Reference: Log levels and messages

The following sections describe the log levels and messages that are specific to DS4IDC.

In the tables that follow:

- ◆ Text inside braces {like this} indicates place marker text that would be replaced by other text in the actual log message. For example, {symbol name} would be replaced by the name of a symbol in the log message.
- ◆ The **Description/Action** column is left blank if the message is self explanatory and requires no action.

11.1 CRITICAL level log messages

CRITICAL level log messages

Message	Description/Action
CTF Init: No CSP connection configured. ctfsrc exiting	Check the configuration has an add-csp-connection ⁵⁶ section. In addition to being recorded in the log file, this message is also echoed to the terminal.
CTF Init: Incorrect CSP connection configuration, no ports defined. ctfsrc exiting	Check add-csp-connection ⁵⁶ section in configuration for a port number. In addition to being recorded in the log file, this message is also echoed to the terminal.

11.2 ERROR level log messages

ERROR level log messages

Message	Description/Action
CTF response: Command <{CTF command number}> tag <{CTF command tag}> not in command list	Contact Caplin.
CTF response: Command <{command string}> tag <{CTF command number}> with no status	Contact Caplin.
CTF response: Received response for command <{command string}> which is currently not handled	Contact Caplin.
CTF Command: Missing initial / in <{symbol name}>	Requested symbol name must have a leading '/
CTF Command: Missing separator in <{symbol name}>	Requested symbol name must have a '/' separator after the exchange symbol.
CTF Command: Can't service command <{command string}><{symbol name}>, subject badly formed	This message is logged if either of the following format errors occur (as described above): CTF Command: Missing initial / in <{symbol name}>) CTF Command: Missing separator in <{symbol name}>
CTF Send: Can't send command <{command string}>, parsing failed	Contact Caplin.
CTF parse: Received badly formatted CTF message - missing CTF_START	Contact IDC.
CTF parse: Received badly formatted CTF message - missing space character	Contact IDC.
CTF parse: Received badly formatted CTF message <{CTF message text}> length <{CTF message length}> - expected CTF_END, got <{number at end position}><{character at end position}>	Contact IDC.
CTF create command: Unimplemented CTF query command <{CTF command number}>	Contact Caplin.
Item discard: Peer <{peer number}><{peer name}> discarding non-existent item <{symbol name}>	Contact Caplin.

11.3 NOTIFY level log messages

NOTIFY level log messages

Message	Description/Action
CTF accept: Connected to CSP on {address} {port}	Expected on normal startup and connection.
CTF read: Disconnecting bytes read <{number}> errno <{error code of last system error}>	Possible network issue.
CTF Disconnect: Disconnecting from CSP <{reason}>, <{current number of failed attempts}> failed attempts	<p>Disconnection from IDC server. This may be due to:</p> <ul style="list-style-type: none"> ◆ Network issue. ◆ Bad login credentials. ◆ Heartbeat timeout. ◆ Monitor command.
CTF Send: Socket write error <{error code of last system error}> sending command <{command string}>	Possible network issue.
*** Source id {number} {peer label} (peer {number}) is UP ***	Standard peer connection log message.
*** Source id {number} {peer label} (peer {number}) is DOWN ***	Standard peer disconnection log message.
Item send data: Received rogue update for object <{symbol name}>	Contact IDC.
Item send data: Received update for unrequested object <{symbol name}>	Contact Caplin.
Item send nodata: Got rogue request to send nodata for item <{symbol name}>	Contact Caplin.

11.4 WARN level log messages

WARN level log messages

Message	Description/Action
CTF accept: Cannot open socket for {address} {port}	Possible network issue.
CTF Command: Ticker name too long <{stock name}>	IDC have a limit of 30 characters for the name of a stock.
CTF Send: Can't send command <{command string}>, not connected to CSP	Possible network issue.
CTF Send: Can't send command <{command string}>, not logged in to CSP	<p>Possible user credentials issue.</p> <p>Check that the username ^[58] and password ^[57] settings in add-csp-connection ^[56] match the credentials required by the IDC server.</p>

11.5 INFO level log messages

INFO level log messages

Message	Description/Action
CTF response: Command <{command string}> tag <{CTF tag}> subject <{symbol name}> passed	The named CTF command was successful. If the command was a request or discard, the log message includes the command tag and the object symbol name.
CTF response: Command <{command string}> tag <{CTF tag}> subject <{symbol name}> failed, status <{CTF status}>	The named CTF command failed and the log message includes the failure status. If the command was a request or discard, the log message includes the command tag and the object symbol name. The most common failure status codes are: 14 – Data not found -12 – Not authorized to read symbol -5 – Login failure
CTF Command: Exchange code for <{exchange string}> has not been configured	The named exchange string was present in an object request, but there is no add-exchange ⁵⁴ configuration item that maps it to an IDC source ID. The named exchange string could itself be a valid IDC source ID, but if it is not then the requested object was not found.
CTF Send: Sent command <{CTF command string}>	The named CTF command was sent to the IDC server.
Sending UP DOWN alert to peer <{number}>	The named IDC connection state was sent to the named peer.
Item request: Peer <{number}><{label}> requesting <{symbol name}> flags <{flags content}>	The named peer requested the named object.
Item discard: Peer <{number}><{label}> discarding <{symbol name}> flags <{flags content}>	The named peer discarded the named object.
Item send data: Not sending empty packet for object <{symbol name}>	A update received for the named object had no fields. The update was not sent to any peers.
Item send data: Sending image update for <{symbol name}> to peer <{number}><{label}>	The named image or update for the named object was sent to the named peer.
Item send stale: Sending stale for all subscribed symbols	A stale message was sent to all peers for all subscribed objects. This message is logged if DS4IDC disconnects from the IDC server.
Item request: Requesting all subscribed symbols	All objects subscribed to by peers have been requested from the IDC server. This message is logged if DS4IDC reconnects to the IDC server.

11.6 DEBUG level log messages

DEBUG level log messages

Message	Description/Action
Heartbeat: Logged in, subscribing to <{heartbeat symbol}> and scheduling heartbeat <{timed event pointer}>	
CTF response: Heartbeat	
CTF response: Update for <{symbol name}> <{number}> fields	
Heartbeat: Timeout occurred, disconnecting and deleting event <{timed event pointer}>	
CTF parse: Parsing message text <{CTF text}>	
CTF parse: Packet <{symbol name}> number fields	
CTF parse: Packet <{symbol name}> <{index}> = <{field value}>	
Item send nodata: Sending nodata for <{symbol name}> to peer <{number}><{label}>	
Item send stale: For each sending stale <{symbol name}>	
Item request: For each requesting <{symbol name}>	







12 Reference: Monitoring MBeans

The following sections describe the MBeans that expose monitoring information about the DS4IDC.


12.1 MBeans summary

This is a summary of the MBeans used by the JMX server.

Generic MBeans

MBean type	MBean description
ctfsrc.server.system 	Provides information and monitors activity relating to the system that the DS4IDC is running on.
ctfsrc.server.datasrc 	Provides release information regarding the DataSource libraries the DS4IDC is using.
ctfsrc.server.logging 	Log file information.
ctfsrc.server.msgq 	Provides information about, and monitors activity of, the DS4IDC message queue. If a peer loses its connection to a DataSource, messages are queued until the connection can be reestablished. The queue is flushed when a reconnection is successful.
ctfsrc.server.peers 	Provides information about, and monitors activity of, a peer (remote application or feed handler which the DS4IDC can receive data from and send data to). There are two kinds of peer: <ul style="list-style-type: none">◆ Active peers, which keep track of which objects have been requested and send updates for those objects only.◆ Broadcast peers, which send all objects and updates to any connected peers.
ctfsrc.server.peerstats 	Provides statistical information on each of the DataSource peers. Additionally, an instance called "global" is an aggregate of all peers.

DS4IDC specific MBeans

MBean type	MBean description
ctfsrc.csp_connection 	IDC server connection information.

12.2 System information (ctfsrc.server.system)

Provides information and monitors activity relating to the system that the DS4IDC is running on.

Attributes and Notifications

Attribute name	Type	Description
<i>current-directory</i>	java.lang.String	Run directory.
<i>process-uptime</i>	java.lang.String	Process uptime.
<i>process-start-time</i>	java.util.Date	Time the system started, in seconds since January 1970.
<i>process-id</i>	java.lang.Integer	Process ID.
<i>user-cputime-total</i>	java.lang.Float	User CPU time used.
<i>system-cputime-total</i>	java.lang.Float	System CPU time used.
<i>cputime-total</i>	java.lang.Float	Total CPU time used.
<i>cpu-usage</i>	java.lang.Float	Average CPU percentage used by process in the period configured by the configuration item process-usage-period ²⁵ .
<i>memory-usage</i>	java.lang.String	Memory usage. This information is only sent to the monitoring client if the monitoring client requests the information.
<i>hostname</i>	java.lang.String	Hostname.
<i>addresses</i>	java.lang.String	IP address(es).

Operations

Operation name	Type	Arguments	Description
shutdown	java.lang.String	No arguments	Shutdown the process. No return value - monitoring console will indicate an exception.

12.3 DataSource information (ctfsrc.server.datasrc)

Provides release information regarding the DataSource libraries the DS4IDC is using.

Attributes and Notifications

Attribute name	Type	Description
<i>datasrc-version</i>	java.lang.String	DSDK library version.
<i>datasrc-build-number</i>	java.lang.String	DSDK build number.
<i>datasrc-build-time</i>	java.util.Date	DSDK build time.


12.4 Log file information (ctfsrc.server.logging)

Log file information.

Attributes and Notifications

Attribute name	Type	Description
<i>name</i>	java.lang.String	Log name.
<i>filename</i>	java.lang.String	Filename.
<i>maximum-size</i>	java.lang.Long	Maximum file size.
<i>cycle-period</i>	java.lang.Integer	Cycle period (mins).
<i>cycle-time</i>	java.lang.Integer	Cycle time (mins).
<i>cycle-suffix</i>	java.lang.String	Cycle suffix.
<i>debug-level</i>	java.lang.String	Current logging level.
<i>isbinary</i>	java.lang.Boolean	Logfile is binary.
<i>islevel</i>	java.lang.Boolean	Logfile has levels.
<i>loglines</i>	java.lang.String	Last few lines of the log file. This information is only sent to the monitoring client if the monitoring client requests the information.
<i>message</i>	java.lang.String	Last monitorable message.

Operations

Operation name	Type	Arguments	Description
cycle	java.lang.String	No arguments	Cycle the logfile
set-debug-level	java.lang.String	level 	Set logging level

set-debug-level arguments

Argument name	Type	Optional	Description
<i>level</i>	java.lang.String	No	Logging level to set the log file to.

12.5 MessageQueue (ctfsrc.server.msgq)

Provides information about, and monitors activity of, the DataSource message queue.

If a peer loses its connection to a DataSource, messages are queued until the connection can be reestablished. The queue is flushed when a reconnection is successful.

Attributes and Notifications

Attribute name	Type	Description
<i>msgq-id</i>	java.lang.Long	Message queue ID.
<i>bytes-sent</i>	java.lang.Long	Total bytes sent.
<i>bytes-read</i>	java.lang.Long	Total bytes received.
<i>write-queue-size</i>	java.lang.Long	Write queue size.

12.6 Peer information (ctfsrc.server.peers)

Provides information about, and monitors activity of, a peer (remote application or feed handler that the DS4IDC can receive data from and send data to).

There are two kinds of peer:

- ◆ Active peers, which keep track of which objects have been requested and send updates for those objects only.
- ◆ Broadcast peers, which send all objects and updates to any connected peers.

Relationships:

- ◆ **msgq (ctfsrc.server.msgq)** [One to one]

There is one message queue for each peer.

Attributes and Notifications

Attribute name	Type	Description
<i>peer-number</i>	java.lang.Integer	Number.
<i>label</i>	java.lang.String	Label.
<i>local-type</i>	java.lang.Integer	Local type.
<i>local-id</i>	java.lang.Integer	Local ID.
<i>local-name</i>	java.lang.String	Local name.
<i>remote-type</i>	java.lang.Integer	Remote type.
<i>remote-id</i>	java.lang.Integer	Remote ID.
<i>remote-name</i>	java.lang.String	Remote name.
<i>state</i>	java.lang.String	Connection state.
<i>configured-addresses</i>	java.lang.String	Connection addresses (configured).

Attribute name	Type	Description
<i>configured-ipaddresses</i>	java.lang.String	Connection addresses (resolved).
<i>configured-ports</i>	java.lang.Integer	Configured ports.
<i>connected-port</i>	java.lang.Integer	Connected remote port.
<i>connected-address</i>	java.lang.String	Connected remote address.
<i>local-port</i>	java.lang.Integer	Connected local port.
<i>local-address</i>	java.lang.String	Connected local address.
<i>recent-events</i>	java.lang.String	<p>Description of recent peer events. One of:</p> <ul style="list-style-type: none"> ◆ Lost connection to peer ◆ Connection closed to peer ◆ Connecting to peer ◆ Connected to peer ◆ Failed connecting to peer ◆ Accepting peer <p>This information is only sent to the monitoring client if the monitoring client requests the information.</p>
<i>last-update-time</i>	java.util.Date	Last update time.
<i>failed-connection-count</i>	java.lang.Long	Failed connection count.

Operations

Operation name	Type	Arguments	Description
peer-reconnect	java.lang.String	dest ⁶⁹	Reconnect peer.
set-down	java.lang.String	No arguments	Set status to DOWN.
set-up	java.lang.String	No arguments	Set status to UP.

peer-reconnect arguments

Argument name	Type	Optional	Description
<i>dest</i>	java.lang.Integer	Yes	<p>The index of the peer to connect to.</p> <p>This is the order of the add-peer²⁹ entry in the configuration file. The first add-peer entry is index 0, the next add-peer entry is index 1, and so on.</p>

12.7 Peer statistics (ctfsrc.server.peerstats)

Provides statistical information on each of the DataSource peers.
Additionally, an instance called "global" is an aggregate of all peers.

Attributes and Notifications

Name	Type	Description
<i>identifier</i>	java.lang.String	Identifier.
<i>messages-read-count</i>	java.lang.Long	All messages in.
<i>messages-read-count-since-reset</i>	java.lang.Long	All messages in since reset.
<i>messages-read-rate</i>	java.lang.Float	All messages in rate.
<i>messages-written-count</i>	java.lang.Long	All messages out.
<i>messages-written-count-since-reset</i>	java.lang.Long	All messages out since reset.
<i>messages-written-rate</i>	java.lang.Float	All messages out rate.
<i>bytes-read-count</i>	java.lang.Long	Bytes in.
<i>bytes-read-count-since-reset</i>	java.lang.Long	Bytes in since reset.
<i>bytes-read-rate</i>	java.lang.Float	Bytes in rate.
<i>bytes-written-count</i>	java.lang.Long	Bytes out.
<i>bytes-written-count-since-reset</i>	java.lang.Long	Bytes out since reset.
<i>bytes-written-rate</i>	java.lang.Float	Bytes out rate.
<i>updates-read-count</i>	java.lang.Long	Update messages in.
<i>updates-read-count-since-reset</i>	java.lang.Long	Update messages in since reset.
<i>updates-read-rate</i>	java.lang.Float	Update messages in rate.
<i>updates-written-count</i>	java.lang.Long	Update messages out.
<i>updates-written-count-since-reset</i>	java.lang.Long	Update messages out since reset.
<i>updates-written-rate</i>	java.lang.Float	Update messages out rate.
<i>requests-read-count</i>	java.lang.Long	Request messages in.
<i>requests-read-count-since-reset</i>	java.lang.Long	Request messages in since reset.
<i>requests-read-rate</i>	java.lang.Float	Request messages in rate.
<i>requests-written-count</i>	java.lang.Long	Request messages out.
<i>requests-written-count-since-reset</i>	java.lang.Long	Request messages out since reset.
<i>requests-written-rate</i>	java.lang.Float	Request messages out rate.
<i>discards-read-count</i>	java.lang.Long	Discard messages in.
<i>discards-read-count-since-reset</i>	java.lang.Long	Discard messages in since reset.
<i>discards-read-rate</i>	java.lang.Float	Discard messages in rate.
<i>discards-written-count</i>	java.lang.Long	Discard messages out.
<i>discards-written-count-since-reset</i>	java.lang.Long	Discard messages out since reset.

Name	Type	Description
<i>discards-written-rate</i>	java.lang.Float	Discard messages out rate.
<i>nodatas-read-count</i>	java.lang.Long	NoData messages in.
<i>nodatas-read-count-since-reset</i>	java.lang.Long	NoData messages in since reset.
<i>nodatas-read-rate</i>	java.lang.Float	NoData messages in rate.
<i>nodatas-written-count</i>	java.lang.Long	NoData messages out.
<i>nodatas-written-count-since-reset</i>	java.lang.Long	NoData messages out since reset.
<i>nodatas-written-rate</i>	java.lang.Float	NoData messages out rate.
<i>notfound-read-count</i>	java.lang.Long	NotFound messages in.
<i>notfound-read-count-since-reset</i>	java.lang.Long	NotFound messages in since reset.
<i>notfound-read-rate</i>	java.lang.Float	NotFound messages in rate.
<i>notfound-written-count</i>	java.lang.Long	NotFound messages out.
<i>notfound-written-count-since-reset</i>	java.lang.Long	NotFound messages out since reset.
<i>notfound-written-rate</i>	java.lang.Float	NotFound messages out rate.
<i>readdeny-read-count</i>	java.lang.Long	ReadDeny messages in.
<i>readdeny-read-count-since-reset</i>	java.lang.Long	ReadDeny messages in since reset.
<i>readdeny-read-rate</i>	java.lang.Float	ReadDeny messages in rate.
<i>readdeny-written-count</i>	java.lang.Long	ReadDeny messages out.
<i>readdeny-written-count-since-reset</i>	java.lang.Long	ReadDeny messages out since reset.
<i>readdeny-written-rate</i>	java.lang.Float	ReadDeny messages out rate.
<i>writedeny-read-count</i>	java.lang.Long	WriteDeny messages in.
<i>writedeny-read-count-since-reset</i>	java.lang.Long	WriteDeny messages in since reset.
<i>writedeny-read-rate</i>	java.lang.Float	WriteDeny messages in rate.
<i>writedeny-written-count</i>	java.lang.Long	WriteDeny messages out.
<i>writedeny-written-count-since-reset</i>	java.lang.Long	WriteDeny messages out since reset.
<i>writedeny-written-rate</i>	java.lang.Float	WriteDeny messages out rate.
<i>deleteobject-read-count</i>	java.lang.Long	DeleteObject messages in.
<i>deleteobject-read-count-since-reset</i>	java.lang.Long	DeleteObject messages in since reset.
<i>deleteobject-read-rate</i>	java.lang.Float	DeleteObject messages rate.
<i>deleteobject-written-count</i>	java.lang.Long	DeleteObject messages out.
<i>deleteobject-written-count-since-reset</i>	java.lang.Long	DeleteObject messages out since reset.
<i>deleteobject-written-rate</i>	java.lang.Float	DeleteObject messages out rate.
<i>unavailable-read-count</i>	java.lang.Long	Unavailable messages in.
<i>unavailable-read-count-since-reset</i>	java.lang.Long	Unavailable messages in since reset.
<i>unavailable-read-rate</i>	java.lang.Float	Unavailable messages in rate.
<i>unavailable-written-count</i>	java.lang.Long	Unavailable messages out.
<i>unavailable-written-count-since-reset</i>	java.lang.Long	Unavailable messages out since reset.

Name	Type	Description
<i>unavailable-written-rate</i>	java.lang.Float	Unavailable messages out rate.
<i>info-read-count</i>	java.lang.Long	Info messages in.
<i>info-read-count-since-reset</i>	java.lang.Long	Info messages in since reset.
<i>info-read-rate</i>	java.lang.Float	Info messages in rate.
<i>info-written-count</i>	java.lang.Long	Info messages out.
<i>info-written-count-since-reset</i>	java.lang.Long	Info messages out since reset.
<i>info-written-rate</i>	java.lang.Float	Info messages out rate.
<i>status-read-count</i>	java.lang.Long	Status messages in.
<i>status-read-count-since-reset</i>	java.lang.Long	Status messages in since reset.
<i>status-read-rate</i>	java.lang.Float	Status messages in rate.
<i>status-written-count</i>	java.lang.Long	Status messages out.
<i>status-written-count-since-reset</i>	java.lang.Long	Status messages out since reset.
<i>status-written-rate</i>	java.lang.Float	Status messages out rate.
<i>heartbeats-read-count</i>	java.lang.Long	HeartBeat messages in.
<i>heartbeats-read-count-since-reset</i>	java.lang.Long	HeartBeat messages in since reset.
<i>heartbeats-read-rate</i>	java.lang.Float	HeartBeat messages in rate.
<i>heartbeats-written-count</i>	java.lang.Long	HeartBeat messages out.
<i>heartbeats-written-count-since-reset</i>	java.lang.Long	HeartBeat messages out since reset.
<i>heartbeats-written-rate</i>	java.lang.Float	HeartBeat messages out rate.
<i>rogue-update-count</i>	java.lang.Long	Rogue updates.
<i>rogue-update-count-since-reset</i>	java.lang.Long	Rogue updates since reset.
<i>rogue-update-rate</i>	java.lang.Float	Rogue updates rate.
<i>active-subscriptions-count</i>	java.lang.Long	Number of active subscriptions made to this peer.
<i>active-subscriptions-count-since-reset</i>	java.lang.Long	Number of active subscriptions made to this peer since reset.
<i>active-subscriptions-rate</i>	java.lang.Float	Active subscriptions rate.
<i>monitoring-interval</i>	java.lang.Float	Current monitoring interval.

Operations

Name	Type	Arguments	Description
reset-counters	java.lang.String	No arguments	Reset counters.
set-monitoring-interval	java.lang.String	interval ⁷³	Set monitoring interval.

set-monitoring-interval arguments

Argument name	Type	Optional	Description
<i>interval</i>	java.lang.Integer	No	Interval in seconds.

12.8 IDC server connection (ctfsrc.csp_connection)

IDC server connection information.

Attributes and Notifications

Name	Type	Description
<i>connected</i>	java.lang.Boolean	CSP is connected and logged in.
<i>connected-address</i>	java.lang.String	Connected address.
<i>connected-port</i>	java.lang.Integer	Connected port.
<i>configured-addresses</i>	java.lang.String	Configured addresses.
<i>configured-ports</i>	java.lang.Integer	Configured ports.
<i>commands-sent-count</i>	java.lang.Long	All commands sent.
<i>commands-sent-count-since-connection</i>	java.lang.Long	All commands sent since last connection.
<i>commands-sent-rate</i>	java.lang.Float	All commands sent rate.
<i>bytes-written-count</i>	java.lang.Long	Bytes out.
<i>bytes-written-count-since-connection</i>	java.lang.Long	Bytes out since last connection.
<i>bytes-written-rate</i>	java.lang.Float	Bytes out rate.
<i>command-response-count</i>	java.lang.Long	All command responses.
<i>command-response-count-since-connection</i>	java.lang.Long	All command responses since last connection.
<i>command-response-rate</i>	java.lang.Float	All command responses rate.
<i>updates-received-count</i>	java.lang.Long	All received updates.
<i>updates-received-count-since-last-connection</i>	java.lang.Long	All Received Updates Since Last Connection.
<i>updates-received-rate</i>	java.lang.Float	All received updates rate.
<i>bytes-read-count</i>	java.lang.Long	Bytes in.
<i>bytes-read-count-since-connection</i>	java.lang.Long	Bytes in since last connection.
<i>bytes-read-rate</i>	java.lang.Float	Bytes in rate.
<i>outstanding-subscriptions</i>	java.lang.Long	Current number of subscribed objects.
<i>failed-connection-attempts</i>	java.lang.Long	Number of failed connection attempts.
<i>failed-login-attempts</i>	java.lang.Long	Number of failed login attempts.
<i>disconnect-events</i>	java.lang.Long	Number of disconnect events.

Operations

Name	Type	Arguments	Description
<i>next-connection</i>	java.lang.String	No arguments	Connect to next configured connection

13 Glossary of terms and acronyms

This section contains a glossary of terms, abbreviations, and acronyms relating to the DataSource for IDC product.

Term	Definition
Active source	A DataSource application that only sends data to DataSource peers that subscribe to the data.
Broadcast source	A DataSource application that sends data to all connected DataSource peers.
Caplin DataSource+	Caplin DataSource+ is the interface between Caplin Xaqua and the bank's systems. It comprises a set of domain specific APIs representing the key Caplin Xaqua subsystems.
Caplin Liberator	Caplin Liberator is a real-time financial internet hub that delivers trade messages and market data to and from subscribers over any network.
Caplin Transformer	Caplin Transformer is an event-driven real-time business rules engine.
Caplin Xaqua	A framework for building single-dealer platforms that enables banks to deliver multi-product trading direct to client desktops.
Contributions	In addition to requesting data from an external data feed, some DataSource adapters (but not DS4IDC) can be configured to contribute data to the external data feed (see local-type ^[32] and remote-type ^[34]).
CSP	<u>C</u> lient <u>S</u> ide <u>P</u> rocessor IDC terminology for an IDC client such as DS4IDC .
CTF	<u>C</u> omStock <u>T</u> oken <u>F</u> ormat The protocol that an IDC server and IDC client use to communicate with each other.
DataSource	DataSource is the internal communications infrastructure used by Caplin Xaqua's server components such as Caplin Liberator , Caplin Transformer , and DataSource adapters . In this and other documents, a DataSource application is sometimes referred to as a DataSource.
DataSource adapter	A DataSource application that integrates with an external (non-Caplin) system, exchanging data and/or messages with that system.
DataSource application	A Caplin Xaqua application that uses the Caplin DataSource+ APIs to communicate with other Caplin Xaqua applications via the DataSource protocol.
DataSource for IDC	A DataSource adapter that retrieves data from an IDC server for onward transmission to other DataSource applications , such as Caplin Liberator and Caplin Transformer .
DataSource peer	A DataSource application that another DataSource application is configured to communicate with.
DataSource protocol	A bidirectional protocol that DataSource applications use to communicate with each other.
DS4IDC	An abbreviation of <u>D</u> ata <u>S</u> ource for (4) <u>I</u> DC.

Term	Definition
DSDK	<u>D</u> ata <u>S</u> ource <u>S</u> oftware <u>D</u> evelopment <u>K</u> it
Failover	A technique to support software resilience whereby, when an application loses the service provided by a server, it reconnects (fails over) to an alternative server in order to minimize the interruption to the service.
IDC client	The client side of an IDC client/server connection. DS4IDC is an IDC client.
IDC data feed	The data feed from an IDC server to an IDC client .
IDC server	The server side of an IDC client/server connection.
Peer	Another name for a DataSource peer .

Contact Us

Caplin Systems Ltd
Triton Court
14 Finsbury Square
London EC2A 1BR
Telephone: +44 20 7826 9600
Fax: +44 20 7826 9610
www.caplin.com

The information contained in this publication is subject to UK, US and international copyright laws and treaties and all rights are reserved. No part of this publication may be reproduced or transmitted in any form or by any means without the written authorization of an Officer of Caplin Systems Limited.

Various Caplin technologies described in this document are the subject of patent applications. All trademarks, company names, logos and service marks/names ("Marks") displayed in this publication are the property of Caplin or other third parties and may be registered trademarks. You are not permitted to use any Mark without the prior written consent of Caplin or the owner of that Mark.

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

This publication could include technical inaccuracies or typographical errors and is subject to change without notice. Changes are periodically added to the information herein; these changes will be incorporated in new editions of this publication.

Caplin Systems Limited may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time.

This publication may contain links to third-party web sites; Caplin Systems Limited is not responsible for the content of such sites.