CAPLIN

Caplin Platform 6.0

How To Use The Deployment Framework

December 2012

Contents

1 Preface

1.1 What this document contains

This document explains how to install and use the Caplin Platform Deployment Framework. It also describes how Caplin Platform blades are deployed to the Framework to create a working web trading platform.

About Caplin document formats

This document is supplied in Portable document format (*.PDF* file), which you can read on-line using a suitable PDF reader such as Adobe Reader®. The document is formatted as a printable manual; you can print it from the PDF reader.

1.2 Who should read this document

This document is intended for developers who want to create a web trading platform based on Caplin Platform components and Caplin Platform blades.

Before reading this document, you should be familiar with the concepts and terms that are introduced in the following documents:

- **Caplin Platform Overview** (all sections)
- **Caplin Platform: Deployment Framework Overview** (all sections)
- **Caplin Liberator Administration Guide** (Overview section)
- **Caplin DataSource Overview** (all sections)

1.3 Related documents

- **Caplin Platform: Overview**

  A business and technical overview of the Caplin Platform.
- **Caplin Platform: Deployment Framework Overview**

  Gives an overview of the Caplin Platform Deployment Framework, and explains the concept of Caplin Platform blades.
- **How To Create A Platform Java Blade**

  Explains in detail how to create new Java™-based Caplin Platform Blades using the Caplin Integration Suite Toolkit.
- **How To Create C And Lua Platform Blades**

  Explains in detail how to create C and Lua blades for the Caplin Platform.
- **Best Practices For Deploying The Caplin Platform**

  Provides recommendations for deploying the Caplin Platform in a typical live environment, and discusses failover scenarios for achieving high service availability.
- **Caplin DataSource: Overview**

  A technical overview of Caplin DataSource.

- **Caplin Liberator Administration Guide**

  Describes the Caplin Liberator server. Includes configuration reference information and a list of Liberator log and debug messages.
- **KeyMaster: Administration Guide**

  Describes how to set up Caplin KeyMaster and integrate it with the Liberator.
- **Caplin Platform: Monitoring and Management Overview**

  Describes the Caplin Monitoring and Management solution and its place within the Caplin Platform, and introduces the Caplin Management Console (CMC).
- **DataSource For C Configuration Syntax Reference**

  Describes the syntax of the language that is used to configure **DataSource applications** that have been built using Caplin's DataSource for C API. Such applications include Caplin Liberator, Caplin Transformer, and Integration Adapters that use the API.
- **StreamLink Overview**

  A technical overview of Caplin StreamLink.

1.4 Typographical conventions

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

Type	*Uses*
/AFolder/Afile.txt	File names, folders and directories
`Some code;`	Program output and script examples
The `value=10` attribute is...	Script 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
Glossary term	Items that appear in the “Glossary of terms and acronyms”
XYZ Product Overview	Document name
♦	Information bullet point
■	Action bullet point – an action you should perform

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

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

1.5 Feedback

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

Please email your feedback to documentation@caplin.com.

1.6 Acknowledgments

2 Hardware and operating system platforms supported

The **Deployment Framework** can be used on the following hardware and operating system platforms:

- Linux® EL5 64 bit
- Windows XP® 32 bit (requires Cygwin – see http://www.cygwin.com/)
- Apple® Mac 64 bit

3 Installing the Caplin Platform Deployment Framework

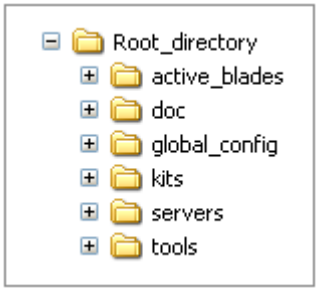
You must install the **Caplin Platform Deployment Framework** kit on all the servers that are to host the **Caplin Platform**. The kit is provided as a zip file. To install it, unzip the file in a directory of your choice, using the following command:

```
unzip –q –o –a <kit-name>
```

Note: On Windows XP, always install the Deployment Framework at the highest directory level possible, preferably at the root (for example, at *C:\<short_folder_name>*).
Windows XP has a 260 character limit on file path names, and if the Deployment Framework is deployed at a deeper level, some files may not be created when other kits are deployed to the Framework.

4 Framework directory structure

The files and directories of the Caplin Platform Deployment Framework are organized in a well defined modular hierarchy, as shown in the following diagram:



The root directory contains the following files and directories.

Files

- *README.txt*

  A short “getting started” guide.

- *start-all.sh*

  A script that starts the system. It stops the **core components** (**Caplin Liberator** and **Caplin Transformer**) and each active **Integration Adapter**, if they are currently running, and then starts them all up.

- *stop-all.sh*

  A script that stops the system by stopping the core components and each active Integration Adapter.

- *clean-all.sh*

  A script that cleans the system logs by stopping the core components and each active Integration Adapter and then removing the log files for each of these components. This script also removes the database files containing information about the Liberator users’ license usage.

***active_blades* directory**

The Deployment Framework uses this directory to manage which blades are active.

***doc* directory**

This directory contains documents about the Caplin Platform Deployment Framework (including this document), and the release note for the **Framework**.

global_config directory

This directory holds files and directories concerning configuration that is global to the Deployment Framework.

Note: To customize the configuration, only change files that are in the *global_config* directory and its subdirectories.
All other configuration files in the Deployment Framework and in Caplin supplied blades are read-only and must not be edited.

- *environment.conf*

  Defines configuration variables that are server specific, such as the port numbers of the core components and the location of the Java Runtime Environment (JRE).

- *hosts.conf*

  Defines the servers that core components and any **Adapter blades** run on. You must edit this file to reflect the deployment.

Tip: An Integration Adapter blade consists of an executable binary file in addition to its configuration and core component configuration.

- *fields.conf*

  A file that defines the names of global fields used in **DataSource** messages.

- *licenses*

  When components are deployed, any licenses required by these components must be placed in this directory.

Note: The Liberator license must be named *license-rttpd.conf*
The Transformer license must be named *license-transformer.conf*.

- *ssl* directory

  Any SSL keys required by the system must be placed in this directory, including the public key file for **Caplin KeyMaster**.

- *Overrides* directory

  This directory contains files that you can update to change configuration; for example, to make the Liberator log events at DEBUG level instead of INFO level. This is described in more detail in section 6.3 “Overriding configuration”.

kits directory

This directory initially contains the Deployment Framework's built-in **Caplin Platform blades** and some scripts.

You use the script called *deploy.sh* to deploy core components and blade kits. The core components and all blades supplied by Caplin Systems are delivered in kit form. *deploy.sh* unpacks each kit and makes the blade components active.

There are also some scripts for managing blades:

activate_blade.sh	Activates a blade or blades
deactivate_blade.sh	Deactivates a blade or blades
check_blades.sh	Checks the state of a blade or blades

servers directory

This directory holds a number of configuration files that contain server specific configuration for the core components, such as port numbers and the location of the Java Runtime Environment (JRE).

tools directory

This directory contains some tools used by the Deployment Framework:

- A python script that ensures all process are stopped when *stop-all.sh* is run.
- Some Cygwin utilities that are used by the Deployment Framework scripts.

5 Deploying Platform components to the Framework

To create a Caplin Platform trading system, you must first deploy the Caplin Platform's core components to the relevant server machines that host the system.

> **Tip:** For information about suitable architectures for deploying Caplin Platform components, please refer to the document **Best Practices For Deploying The Caplin Platform**.

- Obtain kits and licenses from Caplin Systems for the core components and **Caplin Platform Blade**s. See Section 8 for further information about the blades you can deploy.
- Copy any licenses to the *global_config/licenses* directory.

The Liberator requires certain files for HTTPS connections and for Caplin KeyMaster (if used); the following sections explain how to set up these files.

5.1 HTTPs file setup

- If client connections to Liberator are to be via HTTPS, create the *.pem*, *.pwd* and key files as described in the **Caplin Liberator Administration Guide**.

  When these files are used in the Caplin Platform Deployment Framework, they must be named as follows:

  rttpd_https.pem

  rttpd_https.pwd

  rttpd_https.key

- Copy the *.key*, *.pem*, and *.pwd* files to the Deployment Framework's *global_config/ssl* directory on the server machine(s) on which the Liberator runs.

> **Tip:** If you do not generate your own key files, the example HTTPS files in the Liberator kit are automatically copied to *global_config/ssl* when the Liberator kit is deployed. For maximum security, replace these temporary key files with proper user-generated ones as soon as possible.

5.2 Caplin KeyMaster file setup

The Caplin Platform Deployment Framework comes with a default public key file for KeyMaster. However it is recommended that you create your own key pair for KeyMaster as described in the **KeyMaster Administration Guide**.

- Rename the generated public key file to *keymaster_public.der*
- Copy this file to the *global_config/ssl* directory of the Caplin Platform Deployment Framework on the server machine(s) on which the Liberator runs.

5.3 Deploying core component and blade kits

> **Tip:** If the core component kits are deployed and no licenses are found, the 30 minute evaluation licenses that come with these kits are automatically copied to *global_config/licenses*. You should replace these temporary licenses as soon as possible with authorized licenses supplied by Caplin Systems.
>
> Only copy licenses to the server machines on which the licensed components run.

The minimum kits that you must deploy are those for the core components – Caplin Liberator and Caplin Transformer.

- You only need to deploy a core component kit to the server(s) on which the component is to run.
- You must deploy every Caplin Platform blade kit to every server.

To deploy core component and blade kits:

- Copy the kits to the Deployment Framework's *kits* directory.

  For example:

  ```
  cp //software/test_kits/Liberator/Liberator-6.0.0-2/Liberator-6.0.0-2-i686-pc-win32.zip  ./kits
  cp //software/test_kits/Transformer/Transformer-6.0.0-1/Transformer-6.0.0-1-i686-pc-win32.zip ./kits
  ```

- Run the `deploy.sh` script on each server machine:

  ```
  ./kits/deploy.sh
  ```

  Here is an example of the terminal output from this script:

  ```
  Unpacking Transformer kit Transformer-6.0.0-1-i686-pc-win32.zip
  Unpacking Liberator kit Liberator-6.0.0-2-i686-pc-win32.zip
  Setting up active blades
  Activating DirectConnection
  Activating HTTP
  Activating LiberatorWebsite
  Activating OpenPermissioning
  Activating Security
  ```

  If there are no kits to deploy, the `deploy.sh` script exits, otherwise it:

  - stops any Caplin Platform binaries that are running on the server,
  - unpacks (unzips) each kit and then moves the kits' zip files to the *kits/archive* directory.
  - activates each blade that it unpacks (see Section 8.5).

- If an existing kit is being upgraded, the deploy script asks you whether the old kit should be removed. If you know in advance that you want old kits to be removed and replaced with the new ones, run `deploy.sh` with the `-f` option:

  ```
  ./kits/deploy.sh -f
  ```

Note:	The *deploy.sh* script deploys and activates Caplin Platform Blade kits, but *does not* start any of the Caplin Platform components.

Tip:	To ensure that old kit files do not take up too much space on the server, periodically check and clean out the *kits/archive* directory.

- **Now review server specific configuration – see Section 6.**

6 Reviewing server specific configuration

Once you have deployed **core component** kits and blade kits to the server machines that host the Caplin Platform trading system, you must review, and where necessary, change the server specific configuration, as explained in the following sections.

Note: You must make any configuration customizations to the Deployment Framework only by changing the contents of files in the directory *global_configuration* and its subdirectories. Do not directly alter any blade configuration files (see section 6.4 "Reviewing blade configuration").

Tip: The syntax used for the configuration is described in the document **DataSource For C Configuration Syntax Reference**.

6.1 Reviewing server hostnames

Server hostnames are defined in the file *global_config/hosts.conf*; they specify the server machines on which the core components and Integration Adapters run.

The following instructions describe how to edit this file when the Caplin Platform trading system is deployed to a single server. In a multi server deployment, edit and save the file on one server machine and then copy it to the Deployment Framework on each of the other servers.

- For each Adapter blade that is deployed, add the following line to *global_config/hosts.conf*:

```
define <blade_name>${THIS_LEG}_HOST <hostname>
```

In each line that you add, `<blade_name>` is the name of the deployed blade, and `<hostname>` is the hostname of the server machine on which the Integration Adapter runs.

Tip: In a single server deployment, `<hostname>` can be set to `localhost`, but it is good practice to always set it to the actual hostname of the server.

- Edit the lines in *global_config/hosts.conf* that define the hostnames of the servers on which the core components run. The lines look like this:

```
# Core component hosts
define LIBERATOR_HOST_${THIS_LEG}          localhost
define TRANSFORMER_HOST_${THIS_LEG}        localhost
```

To edit these lines, change `localhost` to the hostname of the server on which each core component runs.

- There is also a failover section in *global_config/hosts.conf*, but you can ignore this section unless failover is being deployed. For further information, see section 10 “Failover”.

The following example shows typical host configuration for the core components and for an Adapter blade called *FITradingDataExample*.

```
#
# Liberator and Transformer hosts
#
define LIBERATOR${THIS_LEG}_HOST          liberator1
define TRANSFORMER${THIS_LEG}_HOST       transformer1

#
# Blade hosts
#
define FITradingExample${THIS_LEG}_HOST   prodhost3
```

Note that `THIS_LEG` refers to a variable defined in the file *global_config/environment.conf* (see section 6.2 “Reviewing configuration variables”).

6.2 Reviewing configuration variables

The configuration variables that configure a server component (such as a Liberator or Transformer) are defined in the file *global_config/environment.conf*.

The following instructions describe how to edit this file when the Caplin Platform Deployment Framework is deployed to a single server machine. In a multi server deployment, it is usually sufficient to edit and save the file on one server machine and then copy it to the Deployment Framework on each of the other server machines.

- Check the Java environment settings:
  - `JAVA_HOME`

    The Deployment Framework’s configuration attempts to use the environment variable `JAVA_HOME`. If your server machine has an installed Java Development Kit (JDK), `JAVA_HOME` should be defined. Check that `JAVA_HOME` points to the correct JDK.
  - `JVM_BASE` and `JVM_LOCATION`

    These configuration variables define the location of the Java Virtual Machine (JVM). If certain Java-based features are activated (for example, the **LiberatorJMX** blade), Liberator attempts to load a JVM when it starts up.

If your Java installation is standard, you do not need to change either of these settings. However, if the Java installation is non-standard:

- Assuming that the java binary is located in the directory */<path-up-to-bin>/bin*, set the `JVM_BASE` configuration variable in *global_config/environment.conf* to *<path-up-to-bin>*.
- Set the `JVM_LOCATION` configuration variable in *global_config/environment.conf* to the absolute path of the JVM.

Note: If you copy *global_config/environment.conf* to other server machines in a multi-server deployment, make sure the settings for `JVM_BASE` and `JVM_LOCATION` are suitable for these servers, and change the settings as required.

Note: On Windows, paths can include spaces.
If a Windows path name includes spaces, use double quotes around the name.
For example, define `JVM_BASE` as `"c:/Program Files/Java/jre6"`

- Search *environment.conf* for all configuration variables that have names containing the text `PORT`. These variables specify port numbers that must be available on the server machine on which the component runs. The following is an extract from this file.

```
define THIS_LEG                                          1
...

define LIBERATOR${THIS_LEG}_HTTPPORT                     ${THIS_LEG}8080
define LIBERATOR${THIS_LEG}_HTTPSPORT                    ${THIS_LEG}8081
```

Because `THIS_LEG` is set to **1**, the variable `LIBERATOR`**1**`_HTTPPORT` is set to **1**`8080`, and the variable `LIBERATOR`**1**`_HTTPSPORT` is set to **1**`8081`.

If ports `18080` and `18081` are not available on the server machine on which the Liberator runs, edit the values to port numbers that *are* available on that server, such as `${THIS_LEG}8090` and `${THIS_LEG}8091`.

- You can also review the other configuration variables defined in this file, but the value of these variables do not normally need to be changed.

  Section 11 shows the complete contents of the file *global_config/environment.conf*.
  The configuration variables it defines are used in various configuration files that you do not need to edit. Section 12 lists all these configuration variables and the configuration items they set.

Tip:	Consult your system administrator if you do not know what port numbers are available on a particular server machine.

6.3 Overriding configuration

The directory *global_config/overrides* contains writeable configuration files that override the configuration of core components and blades. The Liberator and Transformer kits come with override files for Liberator and Transformer configuration; these files are in

- *global_config/overrides/servers/**Liberator/etc/***
- *global_config/overrides/servers/**Transformer/etc/***

As an example of using an override, say you want increase the Liberator logging level to DEBUG. Change the `log-level` option in

global_config/overrides/servers/Liberator/etc/rttpd.conf
to

```
log-level DEBUG
```

and then restart the Framework.

Blade overrides

When a Caplin Platform blade is deployed, its override files are copied to directories under *global_config/overrides/<blade_name>*.

You can customize blades supplied by Caplin Systems by editing the appropriate override files. However, you do not normally need to customize built-in Config blades, other than to change port numbers (see section 8.1).

6.4 Reviewing blade configuration

Some Adapter blades have configuration variables that allocate port numbers for JMX monitoring. These variables are defined in the file:

*active_blades/<blade_name>/**DataSource/etc/datasource.conf***

- You must review this file on the server machine on which the Integration Adapter runs to make sure the port numbers it defines are available on that machine.

  The following example shows an extract from such a file.

```
# JMX configuration
#
rmi-registry-port    ${THIS_LEG}1006
rmi-client-port      ${THIS_LEG}2006
#rmi-server-hostname  localhost
```

  The variable `THIS_LEG` is defined in the file *global_config/environment.conf*.
  If `THIS_LEG` is set to **1**, the variable `rmi-registry-port` is set to **1**`1006`, and the variable `rmi-client-port` is set to **1**`2006`.

  If ports `11006` and `12006` are not available on the server machine, add these configuration items, with suitable values, to the Integration Adapter's override file at *global_config/overrides/<blade name>/DataSource/etc/datasource.conf*.
  The configuration item settings in the overrides file will override the equivalent settings in the blade.

7 Starting and stopping the Deployment Framework

The Deployment Framework provides scripts to start and stop Caplin Platform components on the server machines to which they are deployed:

- Navigate to the root directory of the installed Caplin Platform Deployment Framework.
- To start the system, run the script
  `start-all.sh`

Note: On Windows, before starting a multi-server deployment of the Caplin Platform Deployment Framework, first ensure that all the ports used for the components are added to the Windows Firewall exceptions.

- To stop the system, run the script
  `stop-all.sh`
- To stop the system and remove all log files that have been produced by the Platform components, run the script
  `clean-all.sh`
- To just start or stop the core components (Liberator and Transformer), use the `servers` argument:
  `start-all.sh` **`servers`**
  `stop-all.sh` **`servers`**
  `clean-all.sh` **`servers`**

8 Using Caplin Platform Blades

8.1 Built-in blades

The Caplin Platform Deployment Framework is supplied with a number of **Config blades** that implement some of the basic features of a Caplin Platform trading system. These blades are located in the *kits* directory.

The following table lists the built-in Config blades at the time of publication. The blades marked ✓ default to the active state when the Deployment Framework is installed and the relevant Platform components are started. For more about blade states, see section 8.5.

Built-in Config blades

Blade name	Use	Active when deployed?
BlotterExport	Defines blotter export configuration for the Liberator Web Module.	✗
DemoSourceExample	Contains configuration for the C-based demonstration Integration Adapter provided with the Liberator kit.	✗
DirectConnection	Defines direct connection configuration for Liberator.	✓
HTTP	Defines HTTP configuration for Liberator, using the deployment specific configuration defined in *global_config/environment.conf*.	✓
HTTPS	Defines HTTPS configuration for Liberator, using the deployment specific configuration defined in *global_config/environment.conf*.	✗
LiberatorJMX	Defines JMX monitoring configuration for Liberator.	✗
LiberatorWebsite	Provides a full status page for Liberator which is suitable for development.	✓
MinimalLiberatorWebsite	Provides a minimal status page and associated configuration for Liberator that is suitable for secure product deployment.	✗
OpenPermissioning	Defines open authentication configuration for Liberator.	✓

Blade name	Use	Active when deployed?
ServerIdentification	Prevents the Liberator's name, version number, and hostname from being transmitted to client applications in HTTP response headers and in other messages, which is considered to be good security practice. If you deactivate this blade, the above information *is* sent to clients.	✓
TransformerJMX	Defines JMX monitoring configuration for Transformer.	✗

- The built-in blades **HTTP**, **HTTPS**, and **LiberatorJMX** use particular port numbers. You may need to change the port number settings for some or all of these blades to conform to your own port allocation standards. To do this, edit *global_config/environment.conf*
  Do not change any other settings in this file.

8.2 Deploying blades provided by Caplin Systems

Caplin Systems provide a number of blades in kit form that can be deployed to the Deployment Framework.

By convention the filename of each Caplin blade kit is *CPB_<blade_name>-<build_number>.zip*

- Before deploying a Caplin-supplied Adapter blade, make sure there is an entry for it in the *global_config/hosts.conf* file on the server machine(s) where it is to be deployed.
  See section 6.1 “Reviewing server hostnames”.

- To deploy to the Deployment Framework a blade kit provided by Caplin Systems, follow the steps in Section 5.3.

8.3 Creating custom blades

You can create custom blades in Java, using the **Caplin Integration Suite**, and also in C and Lua. For details, see the documents **How To Create A Platform Java Blade** and **How To Create C And Lua Platform Blades**.

8.4 Deploying custom blades

- Before deploying a custom Adapter blade, make sure there is an entry for it in the *global_config/hosts.conf* file on the server machine(s) where it is to be deployed.
  See section 6.1 “Reviewing server hostnames”.

- To deploy a custom blade to the Deployment Framework, follow the steps in Section5.3.

8.5 Changing the state of a blade

Blades can be active or inactive, but only features provided by active blades are available when the Caplin Platform is started. Deploying a new blade kit automatically makes the blade active. If the blade being deployed had previously been deployed and was inactive, the new deployment will make it active.

- To make inactive blades active on the server machines where they are deployed, on each server run the following command from the root directory of the Deployment Framework:

  `./kits/activate.sh <blade-name-1> <blade-name-2> <blade-name-3> ...`

  For example to activate the blades **HTTPS** and **LiberatorWebsite**, run the command:
  `./kits/activate.sh HTTPS LiberatorWebsite`

- To make active blades inactive on the server machines where they are deployed, on each server run the following command from the root directory of the Deployment Framework:

  `./kits/deactivate.sh <blade-name-1> <blade-name-2> <blade-name-3> ...`

  For example to deactivate the blades **HTTPS** and **MinimalLiberatorWebsite**, run the command:
  `./kits/deactivate.sh HTTPS MinimalLiberatorWebsite`

Note: When you run the `activate` and `deactivate` scripts, the Deployment Framework automatically stops the Platform components.

- After changing the state of a blade, restart the Caplin Platform Deployment Framework on every server machine (since a since blade is deployed to every server – see section 5.3 “Deploying core component and blade kits”).

  Use the `start-all.sh script` (see Section 7)

Managing mutually exclusive blades

Some blades are mutually exclusive because they provide different versions of the same feature. *Such blades should not both be activated at the same time.*

Examples of blades supplied by Caplin Systems that are mutually exclusive are:

- **OpenPermissioning** and **Permissioning**
- **LiberatorWebsite** and **MinimalLiberatorWebsite**

If you do activate two mutually exclusive blades, their configurations are loaded in alphabetical order of blade name, so the last loaded will be the active blade. For example, if you activate both the **OpenPermissioning** and **Permissioning** blades, the Permissioning blade will be the one activated.

9 Troubleshooting the Deployment Framework and blades

- To check that the basic Deployment Framework containing Liberator and Transformer has been installed correctly, follow the instructions in the *README.txt* file supplied with the kit.
- If you have problems with the basic installation, contact Caplin Support (see section 13 on page 29).
- Once the basic Deployment Framework is up and running, deploy your required blades.
- The best way to check that a particular blade is correctly deployed and configured is to start the Caplin Platform Deployment Framework and verify that the feature the blade provides is available.
- If you have problems with any of the blades supplied by Caplin Systems, contact Caplin Support (see section 13 on page 29).

Tip: Always ensure that the Java environment settings are correctly defined on each server machine; see section 6.2 “Reviewing configuration variables”.

10 Failover

A Caplin Platform installation is typically deployed with multiple component instances to provide resilience against hardware, software, and network failures. To help achieve this, the hardware and software components can be arranged in processing units called “failover legs”.

In normal operation, all the components in a single failover leg – typically Caplin Liberator, Caplin Transformer, Integration Adapters, and the Bank's internal systems – work together to provide the system's functionality. If a component fails (or a connection to it, or the machine on which it runs), the operations provided by that component are taken up by an alternative copy of the component running in a different failover leg. This transfer of operation is called “failover”.

The Caplin Platform Deployment Framework provides configuration that allows failover components to be deployed to the framework.

Tip: If the **client application** that connects to the Caplin Platform trading system is **Caplin Trader**, client failover is handled at the client by **StreamLink JS**.

For further information, see “Resilience, failover, and load balancing” in the **StreamLink Overview**.

10.1 Configuring failover

Failover components for each leg can be deployed to a single server, but in general the components for each failover leg are deployed to different servers. The first step in configuring failover is to identify the servers on which the failover components will run, and this should be decided in consultation with your system administrator.

Note: Any number of components can be deployed to a single Deployment Framework, but they must all be either primary components or secondary components, and not a mixture of both.

If for some reason the same server must host both primary and secondary components, two Deployment Frameworks must be installed on that server; one containing primary components and the other secondary components.

A failover scheme has two legs, primary and secondary, and each leg is represented in the Caplin Platform Deployment Framework’s failover configuration by the configuration variables `THIS_LEG` and `OTHER_LEG`. By convention, `THIS_LEG` is the primary leg and `OTHER_LEG` is the secondary leg, but either can represent the primary leg as far as client applications are concerned.

Deploying failover components

Liberator and Transformer components are deployed to the primary and secondary server machines on which they will run, but blade components must be deployed to all servers.

- Deploy the component kits as described in section 5.3 ”Deploying core component and blade kits”.

Configuring server hostnames

- Update the file *global_config/hosts.conf* with the hostnames of the server machines that host the primary and secondary failover components.

  The following code example sets the primary host server to `prodhost1` and the secondary host server to `prodhost2`.

```
define LIBERATOR${THIS_LEG}_HOST                    prodhost1
define TRANSFORMER${THIS_LEG}_HOST                  prodhost1
define FXDataExample${THIS_LEG}_HOST                prodhost1

# Failover information
if "${FAILOVER}" == "ENABLED"
    define LIBERATOR${OTHER_LEG}_HOST               prodhost2
    define TRANSFORMER${OTHER_LEG}_HOST             prodhost2
    define FXDataExample${OTHER_LEG}_HOST           prodhost2
endif
```

Enabling failover

- In the *global_config/environment.conf* file of each primary and secondary server machine, change the definition of `FAILOVER` from `DISABLED` to `ENABLED`.
- In the *global_config/environment.conf* file of each secondary server machine, change the definition of `THIS_LEG` from `1` to `2` and the definition of `OTHER_LEG` from `2` to `1`.

The following examples show failover enabled on a primary server and a secondary server.

***global_config/environment.conf* extract for primary server**

```
# Failover configuration
define THIS_LEG                                     1
define OTHER_LEG                                    2
define FAILOVER                                     ENABLED
```

***global_config/environment.conf* extract for secondary server**

```
# Failover configuration
define THIS_LEG                                     2
define OTHER_LEG                                    1
define FAILOVER                                     ENABLED
```


Starting the failover-configured system

- Start each instance of the Caplin Platform Deployment system on each server machine as described in section 7.
- If you have installed the **Caplin Management Console** (CMC), you can use it to check that all primary and secondary components have started and are interconnected as expected.

  For more information about the Caplin Management Console, see the document **Caplin Platform: Monitoring and Management Overview**.

> **Tip:** If you encounter problems starting the system, review the *global_config/hosts.conf* and *global_config/environment.conf* files on each server. If you do not find any obvious configuration errors, contact Caplin Support (see section 13 on page 29).

11 Appendix A: Contents of the file *environment.conf*

This appendix shows the contents of the file *global_config/environment.conf* that is provided with the Caplin Platform Deployment Framework. This file defines configuration variables that have values you can change to customize some of the configuration applied to the Caplin Platform's core components. In particular, it customizes the port numbers assigned to these components.

These configuration variables are used to set the value of configuration items in several configuration files. They provide a level of abstraction that makes it easy to modify core component configuration without needing to directly edit any of the configuration files that use them.

The file also defines configuration variables that specify the location of the Java Runtime Environment (JRE).

Section 6.2 "Reviewing configuration variables" explains how to review and modify the value of these variables, and section 12 lists the configuration items that these variables set.

Contents of *global_config/environment.conf*

```
#
#  Deployment specific configuration.
#
#

#
# Failover configuration. Update these definitions to configure failover
#

define THIS_LEG                                          1
define OTHER_LEG                                        2
define FAILOVER                                         DISABLED

# Liberator peer configuration
define LIBERATOR${THIS_LEG}_DATASRCPORT                 ${THIS_LEG}5001
define LIBERATOR${THIS_LEG}_DATASRCID                   ${THIS_LEG}01
define LIBERATOR${THIS_LEG}_DATASRC_INTERFACE           0.0.0.0

#
# http/https configuration
#
define LIBERATOR${THIS_LEG}_HTTPPORT                    ${THIS_LEG}8080
define LIBERATOR${THIS_LEG}_HTTPSPORT                   ${THIS_LEG}8081
define LIBERATOR${THIS_LEG}_HTTPINTERFACE               0.0.0.0
define SSLCERT_PATH
"${CONFIG_BASE}/ssl"

#
# direct port configuration
#
define LIBERATOR${THIS_LEG}_DIRECTPORT                   ${THIS_LEG}4001
define LIBERATOR${THIS_LEG}_DIRECTINTERFACE             0.0.0.0
```



```
define LIBERATOR${THIS_LEG}_RTTP_HOSTNAME                    "Caplin
Liberator"
define LIBERATOR${THIS_LEG}_HTTP_SERVERNAME                  "Caplin
Liberator"
define LIBERATOR${THIS_LEG}_RTTP_SERVERNAME                  rttpd

define LIBERATOR${THIS_LEG}_BURST_MAX                        0.1
define LIBERATOR${THIS_LEG}_BURST_MIN                        0.01

define LIBERATOR${THIS_LEG}_THREADS_NUM                      1

# Liberator Monitoring
define LIBERATOR${THIS_LEG}_JMX_RMI_REGISTRY_PORT            ${THIS_LEG}1001
define LIBERATOR${THIS_LEG}_JMX_RMI_CLIENT_PORT              ${THIS_LEG}2001
define LIBERATOR${THIS_LEG}_JMX_RMI_SERVER_HOSTNAME          NOT_DEFINED

# Monitor credentials
define LIBERATOR_MONUSER                                     admin
define LIBERATOR_MONPASS                                     admin

# Transformer peer configuration
define TRANSFORMER${THIS_LEG}_DATASRCPORT                    ${THIS_LEG}5002
define TRANSFORMER${THIS_LEG}_DATASRCID                      ${THIS_LEG}02

# Transformer Monitoring configuration
define TRANSFORMER${THIS_LEG}_JMX_RMI_REGISTRY_PORT          ${THIS_LEG}1002
define TRANSFORMER${THIS_LEG}_JMX_RMI_CLIENT_PORT            ${THIS_LEG}2002
define TRANSFORMER${THIS_LEG}_JMX_RMI_SERVER_HOSTNAME        NOT_DEFINED

# Monitor credentials
define TRANSFORMER_MONUSER                                   admin
define TRANSFORMER_MONPASS                                   admin


#
# Failover Liberator and transformer information
#
if "${FAILOVER}" == "ENABLED"
    define FAILOVER                                           ENABLED
    # Other Liberator configuration ports
    define LIBERATOR${OTHER_LEG}_DATASRCPORT
${OTHER_LEG}5001
    define LIBERATOR${OTHER_LEG}_DATASRCID                   ${OTHER_LEG}01
    define LIBERATOR${OTHER_LEG}_DATASRC_INTERFACE           0.0.0.0

    # Other Transformer peer configuration
    define TRANSFORMER${OTHER_LEG}_DATASRCPORT
${OTHER_LEG}5002
    define TRANSFORMER${OTHER_LEG}_DATASRCID                 ${OTHER_LEG}02
endif

#
```

```
# Status page credentials
#
define ADMINUSER                                                admin
define ADMINPASS                                                admin

## User configuration for the location of the JVM
##############################
#
#

## JVM location
if os win32
   define JVM_BASE "${ENV:JAVA_HOME}/jre6"
   define JVM_LOCATION "${JVM_BASE}/${JVM_PATH}/jvm.dll"
   define JVM_BASE32 "${ENV:JAVA_HOME32}/jre6"
   define JVM_LOCATION32 "${JVM_BASE32}/${JVM_PATH}/jvm.dll"
elseif "${OS}" == "darwin"
    define JVM_BASE /usr
    define JVM_LOCATION
/System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Libraries/lib
jawt.dylib
else
   define JVM_BASE ${ENV:JAVA_HOME}/jre
   define JVM_LOCATION ${JVM_BASE}/${JVM_PATH}/libjvm.so
   define JVM_BASE32 ${ENV:JAVA_HOME32}/jre
   define JVM_LOCATION32 ${JVM_BASE32}/${JVM_PATH}/libjvm.so
endif
```

12 Appendix B: Configuration variables and items

This appendix lists the configuration variables that are defined in the file *global_config/environment.conf*, and the configuration items that each of these variables set.

Section 11 shows the contents of this file, and section 6.2 “Reviewing configuration variables” describes how to review and modify the value of these configuration variables.

For further information about the configuration items listed in this appendix, refer to the **Caplin Liberator Administration Guide**.

Configuration variable	Configuration item
JVM_LOCATION	jvm-location
LIBERATOR_BURST_MAX_${THIS_LEG}	burst-max
LIBERATOR_BURST_MIN_${THIS_LEG}	burst-min
LIBERATOR_DATASRCID_${OTHER_LEG}	datasrc-id
LIBERATOR_DATASRCID_${THIS_LEG}	datasrc-id
LIBERATOR_DATASRCPORT_${OTHER_LEG}	datasrc-port
LIBERATOR_DATASRCPORT_${THIS_LEG}	datasrc-port
LIBERATOR_DATASRC_INTERFACE_${OTHER_LEG}	datasrc-interface
LIBERATOR_DATASRC_INTERFACE_${THIS_LEG}	datasrc-interface
LIBERATOR_DIRECTINTERFACE_${THIS_LEG}	direct-interface
LIBERATOR_DIRECTPORT_${THIS_LEG}	direct-port
LIBERATOR_HTTPINTERFACE_${THIS_LEG}	http-interface and https-interface
LIBERATOR_HTTPPORT_${THIS_LEG}	http-port
LIBERATOR_HTTPSPORT_${THIS_LEG}	https-port
LIBERATOR_JMX_RMI_CLIENT_PORT_${THIS_LEG}	jvm-options
LIBERATOR_JMX_RMI_REGISTRY_PORT_${THIS_LEG}	rmi-registry-port
LIBERATOR_JMX_RMI_SERVER_HOSTNAME_${THIS_LEG}	jvm-options
LIBERATOR_RTTP_HOSTNAME_${THIS_LEG}	rttp-hostname
LIBERATOR_RTTP_SERVERNAME_${THIS_LEG}	rttp-server-name
LIBERATOR_THREADS_NUM_${THIS_LEG}	threads-num
SSLCERT_PATH	keyfile
TRANSFORMER_DATASRCID_${OTHER_LEG}	datasrc-id
TRANSFORMER_DATASRCID_${THIS_LEG}	datasrc-id
TRANSFORMER_DATASRCPORT_${OTHER_LEG}	datasrc-port
TRANSFORMER_DATASRCPORT_${THIS_LEG}	datasrc-port
TRANSFORMER_JMX_RMI_CLIENT_PORT_${THIS_LEG}	jvm-options
TRANSFORMER_JMX_RMI_REGISTRY_PORT_${THIS_LEG}	rmi-registry-port
TRANSFORMER_JMX_RMI_SERVER_HOSTNAME_${THIS_LEG}	jvm-options

13 Appendix C: Contacting Caplin Support

If you need to contact Caplin Support to report a problem with the Caplin Platform Deployment Framework, you can do so in the following ways:

Log the problem in the Caplin Issue Management System (Jira) at:	https://jira.caplin.com
or access Jira from the Caplin Client Portal at:	https://support.caplin.com
Email Caplin Support:	support@caplin.com
Telephone Caplin Support:	+44 (0) 20 7826 9601

14 Glossary of terms and acronyms

This section contains a glossary of terms, abbreviations, and acronyms used in this document.

Term	Definition
Adapter blade	A **blade** for the **Caplin Platform** that consists of an **Integration Adapter** and associated configuration.
App	An application that runs in a web browser or on a mobile device.
Blade	A re-usable software module containing the code and resources needed to implement a business feature. In this document the term blade is short for **Caplin Platform blade**.
Blade toolkit	A set of commands to create, build and export **Caplin Platform blades**.
Caplin Integration Suite (CIS)	A set of APIs and tools for creating adapters that integrate the **Caplin Platform** with external systems. It includes the **blade toolkit**.
Caplin KeyMaster	A component that integrates the **Caplin Platform** with any web-based authentication system. It generates a secure encrypted token that enables **Caplin Liberator** to identify authenticated users, and is generally used in conjunction with a single sign-on system.
Caplin Liberator	A financial internet hub that delivers data and messages in real time to and from subscribers over any network.
Caplin Management Console (CMC)	A Java application that communicates with the **Caplin Platform** and **Integration Adapters** via JMX, and provides a GUI for monitoring and controlling these components.
Caplin Platform	An integrated suite of software that supports the services and distribution capabilities needed for web trading. It consists of Caplin Liberator, Caplin Transformer, Caplin KeyMaster, Caplin Director, and Caplin Management Console. For more information, see the **Caplin Platform Overview**.
Caplin Platform blade	A **blade** designed for use with the **Caplin Platform**. A Caplin Platform blade can be an **Adapter blade**, **Config blade**, or **Service blade**.
Caplin Platform Deployment	A configuration and deployment environment for the **Caplin**

Term	Definition
Framework	**Platform** that supports **Caplin Platform blades**.
Caplin Trader	A complete development suite for creating HTML5 trading **apps**.
Caplin Transformer	An event-driven, real-time data transformation engine optimized for web trading services. These services are implemented in **Transformer Modules**.
Client application	In the context of the **Caplin Platform**, a client application is any application that uses the **StreamLink API** to communicate with **Caplin Liberator**.
Config blade	A **blade** for the **Caplin Platform** that enables a feature through configuration.
Container	A data type supported by the **Caplin Platform** that represents a list of items.
Core component	A Caplin Platform component that is supplied with the **Caplin Platform Deployment Framework**, but is not a blade. The core components are **Caplin Liberator** and **Caplin Transformer**.
DataSource	DataSource is the messaging infrastructure used by the **Caplin Platform** and **Integration Adapters**.
DataSource API	An API that allows server applications (including **Integration Adapters**) to communicate with the **Caplin Platform**.
DataSource application	An application that uses the **DataSource API**. Caplin Liberator, Caplin Transformer, and **Integration Adapters** are all DataSource applications.
Deployment Framework	In this document, this term is short for the **Caplin Platform Deployment Framework**.
Framework	In this document, this term is short for the **Caplin Platform Deployment Framework**.
Integration Adapter	A server application that allows an external system to communicate with the **Caplin Platform**. An Integration Adapter is a **DataSource application** and is created using the **Caplin Integration Suite**.
Service blade	A blade for the **Caplin Platform** that includes a **Transformer module** or a **Liberator Auth module**.

Term	Definition
StreamLink API	An API that allows a client application to communicate with a **Caplin Liberator**. There are StreamLink APIs for various technologies; for example, Java, JavaScript, .NET and Silverlight applications, and Objective-C running on iOS.
StreamLink JS	The **StreamLink API** for JavaScript.
Transformer Module	A software module in Caplin Transformer that implements a service. For example, the Refiner module provides a **Container** filtering and sorting service.

CAPLIN

THE WEB TRADING TECHNOLOGY COMPANY

Contact Us

Caplin Systems Ltd

Cutlers Court

115 Houndsditch

London EC3A 7BR

Telephone: +44 20 7826 9600

www.caplin.com

Caplin Platform 6.0: How To Use The Deployment Framework, December 2012, Release 1