Blueprint

IT Administration Guide

blueprint

Contents

Introduction

This guide is intended for IT administrators. This guide contains a variety of tasks that administrators may want to perform for configuration or troubleshooting purposes. Reference material is also available in this guide, such as an architecture overview and ALM integration information.

Information about additional administrator tasks can be found in other Blueprint guides, including:

- *Blueprint Installation Guide*
- *Blueprint Upgrade Guide*
- *Blueprint Migration Guide*
- *Blueprint Instance Administration Guide*
- *Blueprint Project Administration Guide*
- *Blueprint ALM Integration Configuration Guide*

Architecture overview

> **Note:** All of the diagrams in this section are based off of the standard Blueprint installation and configuration (same-server). Scenarios where Blueprint is installed and configured on a separate server differ slightly and are not illustrated in this guide.

There are two types of setups that impact system architecture:

- Standard setups
- Setups enabling support for HP ALM legacy versions 12 and earlier

In both setups, job services run on a separate thread of execution from the main Blueprint application. Job services operate like any other Windows service, running in the background. Within Blueprint, the *Job Management* dialog allows users to initiate jobs. Jobs are placed into a job queue that runs off of the database. You can view the statuses of jobs from the *Job Services* tab in the Instance Administration Console. Jobs services can be stopped and restarted from the *Services* window that is accessible from the Control Panel.

Standard setup

The standard setup of Blueprint includes two 64-bit job services. The two 64-bit services process all jobs, including exports to Microsoft Team Foundation Server.

The following diagram displays a standard setup of Blueprint:
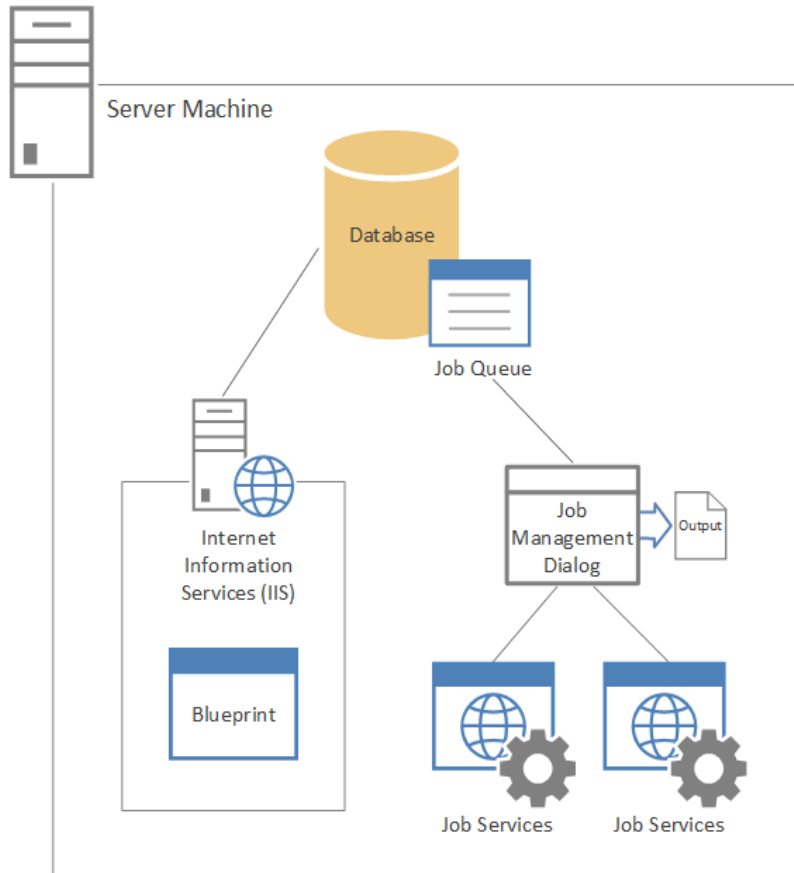
> **Note:** A setup containing 32-bit support for legacy HP ALM can be installed and configured alternatively. For more information about 32-bit architecture, see Setup with HP ALM legacy support.

Setup with HP ALM legacy support

The setup supporting HP ALM legacy versions (12 and earlier) is architecturally different from the standard setup.

Support for HP ALM legacy versions 12 and earlier can be enabled during Blueprint configuration. As a part of configuration, one 32-bit job service is deployed in addition to the standard two 64-bit job services. A legacy connector is also set up. Both the 32-bit job service and the legacy connector communicate with the external HP ALM legacy web site, as the diagram below illustrates:
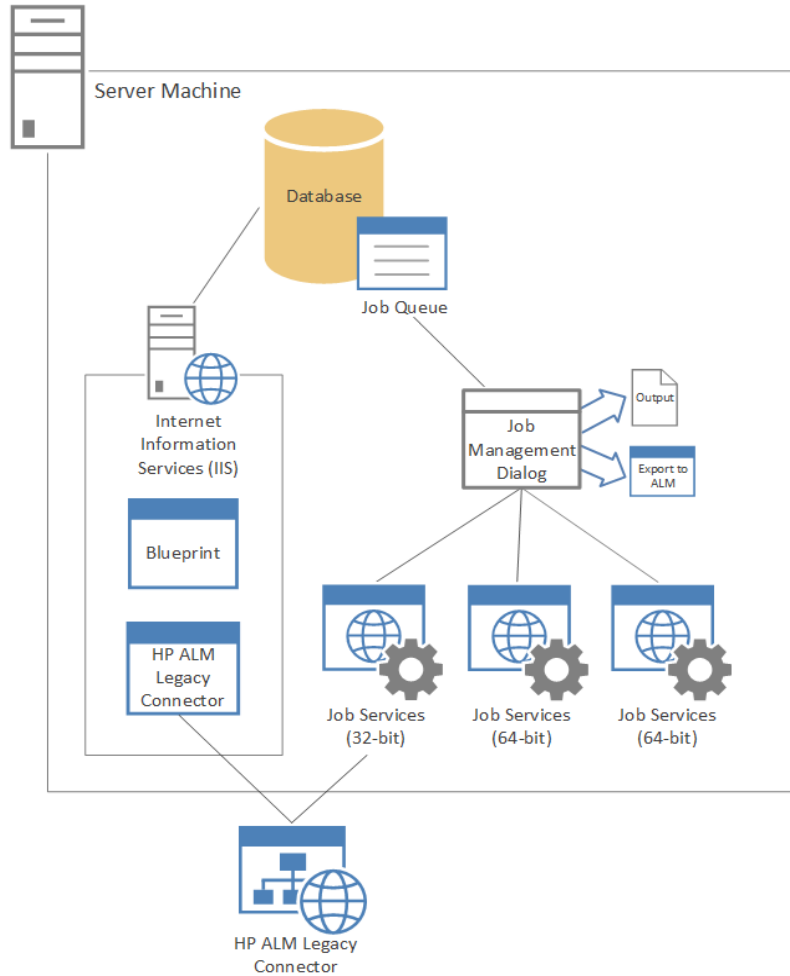
blueprint

Integrations with ALM systems

Overview

Blueprint offers two ALM integration options:

- The built-in ability to export artifacts to Microsoft Team Foundation Server. Export jobs are managed using a queuing system.

  > **Note:** Alternatively, legacy support for HP ALM versions 12 and earlier can be enabled during the installation and configuration of Blueprint.

- The ability to sync artifacts with various other application life-cycle management systems using OpsHub (licensed).

OpsHub integration

Blueprint, in partnership with OpsHub, offers bi-directional integrations with various application life-cycle management systems to optimize the development of requirements.

Supported ALM systems include:

- IBM Rational Team Concert
- Rally
- JIRA
- VersionOne
- Microsoft Team Foundation Server
- HP ALM

OpsHub communicates with Blueprint by using the Blueprint REST API and must be deployed within a corporate network at a point that has network connectivity to both the Blueprint Cloud REST API and the integration API of the ALM system:

ALM Server   OpsHub   Blueprint REST Web Service
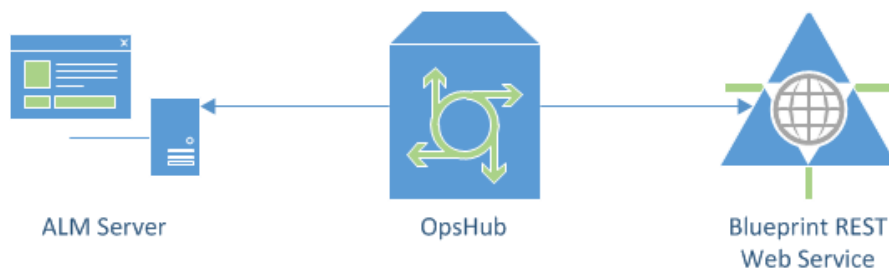
About administrator accounts and passwords

There are multiple accounts and passwords that system administrators use in order to configure Blueprint and integration with other systems. Most of the accounts and passwords are used for communication with external systems and can be managed externally from Blueprint.

The table below lists all possible accounts and passwords system administrators may need depending on how Blueprint is configured and licensed:

Account	Purpose	Managing the password
Customer Portal	Provides access to Blueprint downloads and resources.	If you have forgotten your Customer Portal password, email support@blueprintsys.com to request a new password.
Application Pool identity	Provides access to the Blueprint database in order to perform a distributed installation.	For information on changing an Application Pool identity password, go here: http://technet.microsoft.com/en-us/library/cc263454 (v=office.12).aspx
SQL Server administrator	Allows you to create a database during Blueprint installation.	For information on changing a SQL password, go here: http://msdn.microsoft.com/en-us/library/ms189828.aspx
ALM system account(s) used for integration	Allows you to connect with any supported ALM system.	The account password is managed in the ALM system that is used for integration.
Active Directory bind user	Allows you to enable LDAP integration in Blueprint.	For information on changing an Active Directory password, go here: http://technet.microsoft.com/en-us/library/cc782255 (v=ws.10).aspx
Blueprint Instance Administrator account	Allows you to configure email settings and to enable Blueprint notifications (SMTP), among other administrative tasks.	If you forgot your password, another Blueprint instance administrator can change your password or you can contact Blueprint support.

Account	Purpose	Managing the password
CloudConnect (gateway administrator and policy manager administrator)	The gateway administrator account (ssgconfig) allows you to configure the gateway appliance. The policy manager administrator account (ssm admin) allows you to apply licenses and certificates in addition to creating or modifying assertions for the web services.	For information on changing a CloudConnect password, see the *Layer 7 Installation and Maintenance Manual*.
OpsHub	Allows you to configure and manage integration with supported ALM system(s).	For information about changing an OpsHub password, see the *OpsHub User Manual*.

Changing a Windows service password

To change a Windows service password:

1. In the *Control Panel*, open **Administrative Tools**.

   The *Services* window appears.

2. Open *Services*.

3. Select a service in the list.

4. Right-click the service and then click **Properties**.

   The *Properties* dialog appears.

5. Click the *Log on* tab.

6. Change your password.

7. Click **OK**.

Your password has successfully been changed.

About security

Blueprint provides a couple of security enhancements that administrators can configure. For example, federated authentication can be configured from the Instance Administration Console to provide a secure and convenient single sign-on for users. You can also enable HTTPS to allow Blueprint users to access a secure connection.

Enabling HTTPS

To enhance security, you can enable HTTPS so end users access Blueprint using a secure connection by default.

To enable HTTPS:

1. Open the `web.config` file in a text editor.

2. Add the following attribute:

```
<httpCookies httpOnlyCookies="true" requireSSL="true"/>
```

3. Add the following line:

```
<authentication mode="None">
<forms requireSSL="true"/>
</authentication>
```

4. Uncomment the following line:

```
<!--<behaviorExtensions>
<add name="errorHandlingBehavior"
type="BluePrintSys.RC.Services.ErrorHandlerBehaviorExtensionElement,
BluePrintSys.RC.Service.RIAServices"/>
</behaviorExtensions>-->
```

5. Uncomment the following lines and add the bolded line:

```
<!--<behavior name="HideStackTrace">
<behavior name="HideStackTrace">
<serviceMetadata httpGetEnabled="true"/>
<serviceDebug includeExceptionDetailInFaults="false"/>
<serviceDebug includeExceptionDetailInFaults="false"/>
<errorHandlingBehavior/>
<errorHandlingBehavior/>
</behavior>-->
```

6. Uncomment the following line:

```
<service
name="BluePrintSys.RC.Service.RIAServices.ContentDomainService">
<endpoint address="" binding="webHttpBinding"
bindingConfiguration="streamBinding"
contract="BluePrintSys.RC.Service.RIAServices.ContentDomainService"/>
```

```
<!-- This should only be enabled for IIS hosted web site that contains
https site binding -->
<!--<endpoint address="" binding="webHttpBinding"
bindingConfiguration="securedStreamBinding"
contract="BluePrintSys.RC.Service.RIAServices.ContentDomainService"/>-
->
</service>
```

7. Uncomment the following line:

```
<!--<service
name="BluePrintSys.RC.Client.RIA.Models.Services.AuthenticationDomainS
ervice" behaviorConfiguration="HideStackTrace"/>
<service behaviorConfiguration="HideStackTrace"
name="BluePrintSys.RC.Service.RIAServices.UserManagementDomainServic
e"/>
<service
name="BluePrintSys.RC.Service.RIAServices.UserManagementDomainService"
behaviorConfiguration="HideStackTrace"/>-->
```

8. Add the following attribute:

```
<endpoint address="" binding="basicHttpBinding"
bindingConfiguration="securedBasicHttpBinding"
contract="DanielVaughan.Logging.IClogService"/>
```

9. Save the file.

Improving performance

Overview

Blueprint should run reliably and without issue but there are a few factors that can slow Blueprint performance, such as:

- **Running an older version of Blueprint**

  > **Note:** Running many document generations and export operations to ALM systems on older versions may slow application performance.

  Upgrading Blueprint is a necessary part of performance optimization and additionally provides access to new features.

- **Any additional job services that have been installed and are running**

  We recommend running the default job services that are deployed automatically. For more information on the default job services, see Managing job services.

- **Irregular index maintenance**

  We recommend performing routine database and server maintenance to optimize Blueprint performance and reduce index issues. For more information on performing routine maintenance on the Blueprint database, see Maintaining the Blueprint database.

  As a best practice, we recommend setting up a reoccurring job in SQL server to automatically reindex and rebuild statistics.

Setting up a maintenance plan in SQL Server

To set up a maintenance plan in SQL Server:

1. Open SQL Management Studio.
2. Log on to Management Studio with your credentials.
3. In *Object Explorer*, expand **Databases**, right-click the Blueprint database, point to **Tasks** and then click **Maintenance Plan**.
4. Enter the name of the new plan in the *New Maintenance Plan* dialog and then click **OK**.
5. In the *Toolbox* pane, double-click **Rebuild Index Task**.

   The *Rebuild Index Task* box appears.
6. Double-click the *Rebuild Index Task* box and configure the task as needed.
7. In the *Toolbox* pane, double-click **Update Statistics Task**.

   The *Update Statistics Task* box appears.
8. Double-click the *Update Statistics Task* box and configure the task as needed.
9. Connect the two tasks you have added.

10. Click the **schedule** button on your subplan.

    The *Job Schedule* dialog appears.

11. Configure the job to run every week during off-time.

    For example: Sunday at 4 am.

12. Click **OK**.

13. Click the **Save** button.

You have successfully created a weekly SQL maintenance plan to run automatically.

Accessing and understanding logs

Overview

Blueprint provides two files that log Blueprint user activity: the Blueprint log zip file and the job services log. The Blueprint log zip file contains the server log, the audit log, the API log and the client log. The job services log contains information about job services activity on the system and job activity within the Blueprint application.

About the log configuration file (logging.config)

A variety of log settings can be configured within the **logging.config** file, including log file paths and log levels. In the **logging.config** file, the log file paths can be found in a different section from log level values. In the following sections, you can find information on changing a log file path and changing a log level.

> **Note:** We recommend backing up the **logging.config** file prior to upgrading because this file is overwritten during the Blueprint upgrade process. If you have changed log file paths or log levels in this file, you can use the backup file as configuration reference after the upgrade. After the upgrade, you can re-configure the values in **logging.config**.

Re-configuring the log configuration file path

The path to the log configuration file is set within the `<appSettings>` section of the **web.config** file. By default, the file path is **Configuration\Logging.config**.

The log configuration file path value can be configured in the following tag:

```
<add key="LoggingConfigurationPath" value="Configuration\Logging.config" />
```

Changing a log file path

To change a log file path:

1. Open the log configuration file in a text editor.

   The file can typically be found in this folder: **Configuration\Logging.config**

2. Locate the value you want to edit and then change the value.

   > **Note:** See the tables below for log file path values that can be changed.

3. Save the file.

The file has been successfully edited.

Log file paths

Server logging and client logging file paths can be found in separate sections within the log configuration file. The file path value of any log can be changed (that is, server, audit, API or client logs).

Server log file paths

The following table lists where each type of server log can be found in the server section:

Log	Appender name(s)	Default file value	Description
Server	**ServerFileAppender, ServerPerformanceFileAppender, ServerPerformanceTraceFileAppender**	C:\ProgramData\Blueprint Software Systems\Logs\Blueprint_ c\Blueprint.log	All of the **appender name** values listed in this row contain server log information and write to the same server log file path. For more information about the server log, see Viewing the server log. If you want to change the location of the server log, all of the file values in the appender name sections listed must be changed to the same location.
Audit	**ServerAdminAuditFileAppender**	C:\ProgramData\Blueprint Software Systems\Logs\Blueprint_ c\BlueprintAuditLog.csv	This file contains administrator and audit information on the server side. The file is available as a CSV file, which can be opened in Microsoft Excel. For more information on the audit log, see Viewing the audit log.
API	**WebApiAuditFileAppender**	C:\ProgramData\Blueprint Software Systems\Logs\Blueprint_ c\BlueprintApiAudit.log	This file contains audit information on the Blueprint API. For more information on the API log, see Viewing the API log.

Client file log path

The default client log path is as follows:

```
C:\ProgramData\Blueprint Software Systems\Logs\Blueprint_
c\Blueprint.Client.log
```

The above client log path file can be found in two locations: **ClientFileAppender** and **ClientPerformanceFileAppender** (*Client* section). If you change the location of the client log, the file values in the **ClientFileAppender** and **ClientPerformanceFileAppender** sections must be changed to the same value. Both appender names contain client log information and write to the same path.

For information on the client log, see Viewing the client log.

Changing the server log level

> **Note:** We only recommend changing the server log level from `Warn` (default) to `Info` or `Debug`.

Blueprint provides you with the ability to change the amount of log detail that appears in the server log. By default, the log level is set to `Warn`.

If needed, the log level can be changed to one of the following:

- **All**

  All levels of log information are included.

  > **Note:** We do not recommend enabling this log level as it can cause issues with server performance.

- **Debug**

  A high level of information is logged, including client-to-server web traffic information.

  > **Note:** This level should not be used for prolonged periods of time as this level will cause issues with server performance.

- **Info** (recommended)

  All informational statements including warnings and errors are logged.

  For example: When a user exports artifacts to an ALM system, all entries that are being exported to the ALM system are logged.

- **Warn**

  Only warning and higher priority messages, such as error and fatal messages, are logged. This is the current default Blueprint log level setting.

- **Error**

  Only operations that cause errors appear in the server log.

- **Fatal**

  Only messages that relate to fatal or critical operations appear in the server log.

- **None**

  No information pertaining to server operations are logged.

To change the server log level:

1. Open the **`logging.config`** file in a text editor.
2. Change the level value in the following entry:

```
<!-- Server logger -->
<logger name="BlueprintServerLogger">
<!-- Modify level value in order to increase/decrease logging level
for server logging. -->
```

```
<!-- Valid values are None < Debug < Info < Warn < Error < Fatal < All
-->
<level value="Warn" />
<appender-ref ref="ServerFileAppender" />
</logger>
```

3. Save the file.

The file has been successfully edited and the log level has been changed.

About the Blueprint log zip file

The Blueprint log zip file provides information about system usage and can be helpful for troubleshooting. The Blueprint log zip file contains four different log files (the server log, audit log, API log and the client log).

Within the Instance Settings, Blueprint provides a log zip file that contains the following log files:

- Server log

  Provides information about debugging and errors that have occurred in order to help you with troubleshooting.

- Audit log (CSV file)

  Provides a record of changes administrators have made within the Instance Administration Console and the Project Administration Console.

- API log

  Provides activity, error and debugging information for API developers.

- Client log

  Provides information about Blueprint user activity in order to help you with troubleshooting.

Viewing the server log

The Blueprint server log file provides information about system usage. This information is useful for troubleshooting purposes.

The log file is provided as a comma-separated file that lists each log entry on a new line. Here's an example of a single log entry:

```
04/07/2012 11:27:12 AM,GMT-
05:00,dmnptkx5w5dvrseyxtr2pwyh,acme\wecoyote,Info,ChangeSummary - Finish -
for (int i =(path.Count -1);i>=0;i--)
```

Here's an explanation of the data contained in each log entry, outlined from left to right:

Log Entry Data	Description	Example
Date/Time	Indicates the date and time that the item was logged.	`04/07/2012 11:27:12 AM`
Time Zone	Indicates the time zone of the Date/Time value that was logged.	`GMT-05:00`

Log Entry Data	Description	Example
Session ID	Indicates the session ID of the user that triggered the log entry.	`dmnptkx5w5dvrseyxtr2pwyh`
User	Indicates the user name of the user that triggered the log entry.	`acme\wecoyote`
Type	Indicates the type of log message. This value can be set to: ■ Info ■ Debug ■ Warn ■ Error ■ Fatal	`Info`
Action	Indicates whether or not the log entry occurred due to a login or logout action. This value can be set to: ■ [blank] ■ Login ■ Logout	`Login`
Change Summary	Provides a detailed summary of the log entry. This information can be useful for the Blueprint Support team if you require assistance troubleshooting a problem.	`ChangeSummary - Start - for (int i =(path.Count - 1);i>=0;i--)`

Viewing the audit log

> **Note:** Because it is geared towards maintaining security within an enterprise structure, the audit log is only accessible at the Instance Administration level.

The audit log provides a detailed record of administrative activities, helping you keep of track important operations that have taken place within the system. Whereas artifact versioning and history allows you to view the changes that have occurred in an individual artifact or project, the audit log provides an account of administrative actions that have taken place within the Instance Administration Console and the Project Administration Console. For example, audit logging can facilitate insight into a variety of administrative activities, such as granted privileges and modified instance settings.

Audit logging provides the additional benefit of helping you troubleshoot high-level issues effectively.

Provided in the main log zip file, the audit log is a CSV file that lists each log entry on a new line. Here's an example of an audit log:

> **Important:** Depending on whether the log entry category is applicable to the action that occurred, the log entry field either contains data or is blank.

Here's an explanation of the data contained in each log entry, outlined in the order the columns appear:

Log Entry Data	Description	Example
DateTime	Indicates the date that the action was logged.	`05/23/2013 14:40:45`
UserID	Indicates the ID of the user that performed the logged action.	`348`
UserName	Indicates the user name of the user that performed the logged action.	`jsmith`
Scope	Indicates if the action was instance-wide or specific to a project. The scope can be either: ■ Project --Or-- ■ Instance	`Project`
ProjectID	If the logged action was specific to a project, indicates the ID of the project.	`74840`
ProjectName	If the logged action was specific to a project, indicates the project name.	`OnlineBankingProject`
Area	Indicates the functional area that the action applies to. Possible areas include: ■ Users ■ Groups ■ Roles ■ Group Assignment ■ Projects ■ Project Role Assignments - User ■ Project Role Assignments - Group ■ Project Settings ■ Properties ■ Property Assignments	`Groups`
Action	Indicates the type of action that the user performed. Possible actions include: ■ Add ■ Edit ■ Delete	`Edit`
ObjectId	Indicates the ID of the object that the user acted upon.	`571`
ObjectName	Indicates the name of the object that the action was applied to. An object can be a wide variety of things, including (but not limited to): projects, artifact types, custom properties, users, roles, groups, ALM integrations, document generation templates.	`Collaborator`
Attribute	Indicates the attribute that the action applies to.	`Email`

Log Entry Data	Description	Example
NewValue	Indicates the new value that the attribute has been set to.	`newemail@address.com`
OldValue	Indicates the previous value of the attribute.	`oldemail@address.com`

Details	Depending on the object type that has been added, removed or edited, provides any additional details.	`TYPE=Database EMAIL=`
		`  SCOPE=/MainProject`
		`ISLICENSED=False`

Viewing the API log

The API log provides a detailed record of API requests and responses, helping you keep of track important actions that have been performed on Blueprint data.

The log is provided as a comma-separate file that lists each new entry on a new line. Here's an example of a single log entry:

```
Error, 02/10/2014 10:46:46, 191.161.21.31, admin,
http://localhost:80/projects/83907/artifacts, POST, 401.1722, 500,
Processing of the HTTP request resulted in an exception.,
System.Web.Http.HttpResponseException
```

> **Important:** Depending on whether the log entry category is applicable to the action that occurred, the log entry field either contains data or is blank.

Here's an explanation of the data contained in each log entry, outlined in the order the columns appear:

Log Entry Data	Description	Example
TypeOfEntry	Indicates the type of action that occurred. Possible actions include: ■ Error ■ Audit	`Error`
Date	Indicates the date that the action was logged.	`05/23/2013 14:40:45`
SourceIp	Indicates the IP address of the user performing the request.	`191.161.21.31`
Username	Indicates the user name of the user that performed the request.	`jsmith`
UriRequest	Indicates the URI that was used in the request.	`http://localhost:80/projects/83907/artifacts`
HttpMethod	Identifies the HTTP method that was performed.	`POST`

Log Entry Data	Description	Example
ElapsedTime	Indicates the amount of time (in milliseconds) that elapsed between the request and the response.	`401.1722`
Status Code	Indicates the status code that appeared in response to the request.	`500`
Message	If a message appeared, this data indicates the response message that appeared.	`Processing of the HTTP request resulted in an exception.`
Exception	If an exception occurred, this data indicates the type of exception that occurred.	`System.Web.Http.HttpResponseException`

Viewing the client log

The client log provides information about Blueprint user activity in order to help you with client troubleshooting. The log is available within the Blueprint log zip file (*Instance Administration Console*).

The log file is provided as a comma-separated file that lists each log entry on a new line. Here's an example of a single log entry:

```
Client: 21/03/2014 12:24:43, GMT-05:00, acme\wecoyote, Info,Information
message: Shared View settings are applied.
```

Here's an explanation of the data contained in each log entry, outlined from left to right:

Log Entry Data	Description	Example
Date/Time	Indicates the date and time that the item was logged.	`21/03/2014 12:24:43`
Time Zone	Indicates the time zone of the Date/Time value that was logged.	`GMT-05:00`
User	Indicates the user name of the user that triggered the log entry.	`acme\wecoyote`
Type	Indicates the type of log message. This value can be set to: ■ Info ■ Debug ■ Warn ■ Error ■ Fatal	`Info`
Action	Provides a detailed summary of the log entry. This information can be useful for the Blueprint Support team if you require assistance troubleshooting a problem.	`Information message: Shared View settings are applied.`

Tip: To download a log of an individual user's activity: using the user's account, click the *application menu* `Menu` and then click **Profile Options**. When the dialog appears, click the **Download** button under the *Client log* section.

To view the client logs:

Note: If you are using Internet Explorer 8, you must enable the *automatic prompting for file downloads* security setting before you can download the file from Blueprint. To enable this setting, click **Tools** > **Internet Options** > **Security** > **Custom level...** > **Downloads** and then enable the **Automatic prompting for file downloads** option.

1. Open the *Instance Administration Console*.

2. Click **Instance Settings**.

3. Click **Logging**.

4. Click the **Download Log** button.

   The download dialog appears.

5. Click **Open**.

   The file is unzipped.

6. Open `Blueprint.Client.Log` in a text editor.

The client log file appears.

Viewing the job services log

The job services log contains information about the functionality of job services as well as job activity within Blueprint. The service log file is available in the job services log folder. The log file has a similar path to the following: `C:\Program Files (x86)\Blueprint Software Systems\Blueprint\JobExecutorService\Log`.

The job services log contains the following information to help you with troubleshooting:

Log Entry Data	Description	Example
Date/Time	Indicates the date and time that the item was logged.	`04/07/2012 11:27:12 AM`
Time Zone	Indicates the time zone of the Date/Time value that was logged.	`GMT-05:00`
Session ID	Indicates the session ID of the user that triggered the log entry.	`dmnptkx5w5dvrseyxtr2pwyh`
User	Indicates the user name of the user that triggered the log entry.	`acme\wecoyote`

Log Entry Data	Description	Example
Type	Indicates the type of log message. This value can be set to: - Info - Debug - Warn - Error - Fatal	`Info`
Action	Indicates what job service action occurred. Provides a detailed summary of the log entry. This information can be useful for the Blueprint Support team if you require assistance troubleshooting a problem.	`Server logging initialized`

> **Tip:** The *Job Management* dialog in Blueprint contains a high-level log of job activity within the *Details* pane.

To view the job service log:

1. In Windows explorer, navigate to the folder containing the job services log.

   The job services log can typically be found in this folder: `C:\Program Files (x86)\Blueprint Software Systems\Blueprint\JobExecutorService\Log`

2. Open the `BlueprintJobService.log` file in a text editor.

The job service log appears.

Managing the Blueprint database

This section covers a variety of database-related tasks that can be performed, such as:

- Database server configuration parameters
- Maintaining the Blueprint database
- Setting up a new database
- Moving the application to a new web application server
- Pointing the web application server to a different database
- Clearing database contents and re-initializing the database
- Changing database connection settings
- Changing a SQL port number in the connection string
- Changing the Blueprint Server User of the Blueprint application

Database server configuration parameters

> **Tip**
>
> You can type the following commands to view more information about the command parameters:
>
> `blueprintwebcmd.exe /help`
>
> `blueprintdbcmd.exe /help`
>
> `blueprintqcwebcmd.exe /help`

Parameter	Description	Default	Example
/object	Defines the object type of the command. This parameter can be set to one of the following values: - DB - administration of database - FILESTOREDB - administration of FileStore database - ADMINSTOREDB - administration of AdminStore database - USER - administration of database user		
/command	Defines the command to perform. This parameter can be set to one of the following values: - ADD - add new object to the server - INIT - initialize the object. Currently supported only for DB object. - UPGRADE - upgrade the object. Currently only supported for DB object.		
/datasource	Defines your database and instance names.		DBSERVER\INSTANCE01
/catalog	Defines the name of the service database.	Blueprint	BlueprintDB

Parameter	Description	Default	Example
/catalogprefix	Defines the initial catalog prefix for the service database. (Note the _AdminStorage suffix is automatically added to this value, and should be omitted from the command-line value.)		Blueprint (which becomes Blueprint_ AdminStorage)
/integratedsec	Defines whether or not Windows security is used. This parameter can be set to one of the following values: ■ TRUE ■ FALSE If /integratedsec is set to FALSE, you must specify a /userid and /password.		
/userid	Defines the username of the *Database System Administrator* user. This parameter is only required if /integratedsec is set to FALSE.		
/password	Defines the password of the *Database System Administrator* user. This parameter is only required if /integratedsec is set to FALSE.		
/nuseridentity	Defines the username of the *Blueprint Server User*.		acme\rrunner

Maintaining the Blueprint database

Basic Maintenance

Performing routine maintenance on the Blueprint SQL database is important for ensuring optimum application performance. Over time, changes to the content of a database can cause index fragmentation and the index statistics to be out of date. Indexes play an important role in database performance and proper index maintenance can have a dramatic impact on the performance of Blueprint. Index maintenance is especially important if a large amount of data is being loaded or many changes are being made on the Blueprint database server.

We recommend performing index maintenance once a week when using Blueprint in a typical business environment. Additionally, we recommend updating index statistics after every project migration and after every upgrade of Blueprint.

As a part of maintenance, perform the following steps:

1. Verify the date of the latest index statistics update by running this query:

```
EXEC sp_autostats 'ItemVersions'
```

2. If index statistics are not up to date or query execution times are slow, recalculate index statistics with the following query:

```
EXEC sp_updatestats
```

For more information on statistics recalculation, visit the following links:

- http://msdn.microsoft.com/en-us/library/ms190397.aspx
- http://msdn.microsoft.com/en-us/library/ms187348.aspx
- http://msdn.microsoft.com/en-us/library/ms173804.aspx

3. Run the following query to find out the fragmentation level of your indexes, replacing `BPDatabase` with your Blueprint instance database's name:

```
USE [BPDatabase]

SELECT db_name() AS DatabaseName
,OBJECT_NAME(a.object_id) AS ObjectName
,a.index_id
,b.NAME AS IndexName
,avg_fragmentation_in_percent
,index_type_desc
,record_count
,avg_page_space_used_in_percent --(null in limited)
FROM sys.dm_db_index_physical_stats(db_id(), NULL, NULL, NULL,
'SAMPLED') AS a
INNER JOIN sys.indexes AS b ON a.object_id = b.object_id
AND a.index_id = b.index_id
WHERE b.index_id <> 0
AND avg_fragmentation_in_percent <> 0
```

For more information on detecting fragmentation, go here: http://msdn.microsoft.com/en-us/library/ms189858.aspx#Fragmentation.

4. If your Blueprint database is fragmented, we recommend taking the database offline outside of peak hours and rebuilding the database using the following command:

```
EXEC sp_MSforeachtable @command1 = "print 'Rebuilding indexes for
?...' ALTER INDEX ALL ON ? REBUILD WITH (FILLFACTOR = 80)"
```

For more information on reorganizing and rebuilding indexes, go here: http://msdn.microsoft.com/en-us/library/ms189858.aspx

Advanced Maintenance

For more in-depth and comprehensive maintenance, see the Advanced Database Maintenance article.

Setting up a new database

Warning: Be careful completing the steps below. If you run the re-initialize command on an existing database, you will lose all data!

1. Create the new database:

```
blueprintdbcmd.exe /object ADMINSTOREDB /command ADD /datasource
[DBSERVER\INSTANCE01] /catalogprefix [BlueprintDB] /integratedsec
FALSE /userid [dbadmin] /password [pAssw0rd]
```

> **Note**: The `ADD` command automatically appends `_AdminStorage` to the database name.

2. Add security:

```
blueprintdbcmd.exe /object USER /command ADD /datasource
[DBSERVER\INSTANCE01] /catalog [BlueprintDB] /integratedsec FALSE
/nuseridentity [acme\rrunner] /userid [dbadmin] /password [pAssw0rd]
```

3. Initialize the database:

```
blueprintdbcmd.exe /object ADMINSTOREDB /command INIT /datasource
[DBSERVER\INSTANCE01] /catalogprefix [BlueprintDB] /integratedsec
FALSE /userid [dbadmin] /password [pAssw0rd]
```

> **Note**: The catalog that will be updated is `[BlueprintDB]_AdminStorage`.

Moving the application to a new web application server

The steps below may be required if a server is being decommissioned and you want to move the application to a new web application server.

1. Run the MSI installation on the new web application server.
2. Proceed with the configuration of the application (using the Blueprint Configuration Wizard or the Advanced/Manual installation steps).

> **Warning**: Do NOT proceed with the installation of the database.

3. Run the connection string command.
   For example:

```
blueprintwebcmd.exe /object DBCONFIG /command SET /datasource
[DBSERVER\INSTANCE01] /wsname [Blueprint] /integratedsec TRUE /catalog
[BlueprintDB]
```

Pointing the web application server to a different database

Complete the following steps if you want the web application server to point to a different database:

1. Make sure the new database is setup. Learn more about setting up a new database.
2. Run the connection string command, using the new database name and location as parameters. For

example:

```
blueprintwebcmd.exe /object DBCONFIG /command SET /datasource
[DBSERVER\INSTANCE01] /wsname [Blueprint] /integratedsec TRUE /catalog
[BlueprintDB]
```

Clearing database contents and re-initializing the database

Warning: The steps below will result in data loss!

1. Back up your data!

2. Run the database initialization command. For example:

```
blueprintdbcmd.exe /object ADMINSTOREDB /command INIT /datasource
[DBSERVER\INSTANCE01] /catalogprefix [BlueprintDB] /integratedsec
FALSE /userid [dbadmin] /password [pAssw0rd]
```

Note: The catalog that will be updated is `[BlueprintDB]_AdminStorage`.

Changing database connection settings

The database connection details, specifically the server and database name, can be configured within the `web.config` file.

To change the database connection settings:

1. Open the **`web.config`** file in a text editor.

2. Enter the new database server and database name found within the `connectionStrings` tag:

```
<connectionStrings>
<add
connectionString="metadata=res://*/Models.Instance.csdl|res://*/Models
.Instance.ssdl|
res://*/Models.Instance.msl;provider=System.Data.SqlClient;provider
connection string=&quot;Data Source=192.168.10.10;Initial
Catalog=blueprint;Integrated
Security=True;MultipleActiveResultSets=True&quot;"
name="InstanceContainer" providerName="System.Data.EntityClient" />
</connectionStrings>
```

3. Save the file.

Changing the Blueprint Server User of the Blueprint application

You can use the steps below to change the Blueprint Server User of the Blueprint application. For example, complete the steps below if you installed Blueprint using a temporary account as the Blueprint Server User (example: tempuser) and now you want to change the Blueprint Server User to a different user (example: acme\rrunner).

To change the Blueprint Server User:

1. Add the new user to both the web application server and the database server. The user must exist on both servers with the same user name.

2. Change the `Identity` of your Blueprint application pool by performing the following steps:

   1. Open *Internet Information Services (IIS) Manager* on the web application server that is hosting Blueprint.

   2. Right-click your Blueprint application pool and select **Advanced Settings**:

      > **Note**: The Application Pool is named **Blueprint** in the example image below. The name, however, depends on the name that was chosen during the installation of Blueprint.
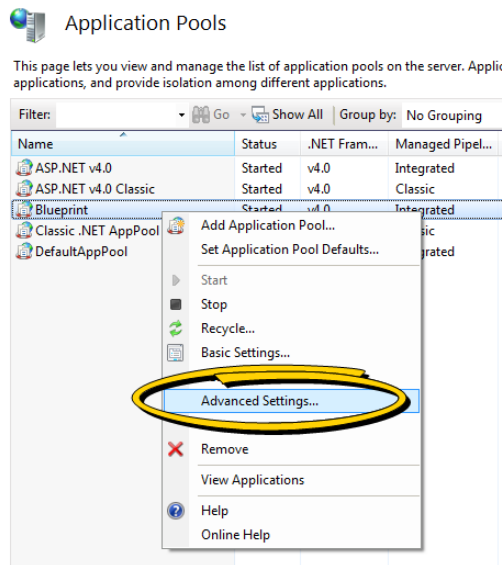
   3. On the *Advanced Settings* dialog, select **Identity** and then click the ellipsis button.
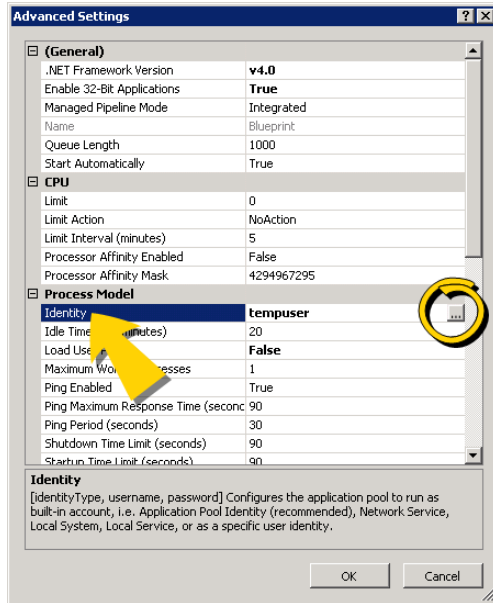
The *Application Pool Identity* dialog appears:
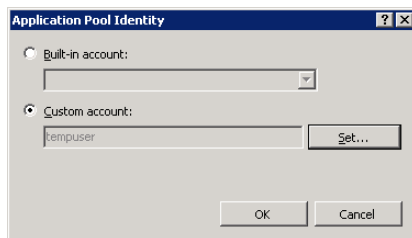
4. Select one of the following options:

   ■ **Built-in account**: Allows you to use the Built-in account for the Blueprint Server User. The ApplicationPoolIdentity built-in account in IIS is created as **IIS APPPOOL\<AppPoolName>** in SQL.

   > **Note**: The Built-in account should only be used for single-server installs, where the datasource is always localhost.

   ■ **Custom account**: Allows you to set a specific user account for the Blueprint Server User. Click the **Set** button to open the *Set Credentials* dialog. Enter the user name and password for the new user, and then click **OK** to save the changes.
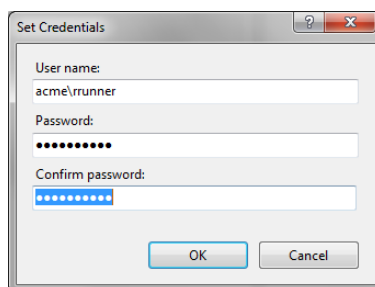
3. Run the Blueprint database utility command to set the user. You must specify the new user using the

`/nuseridentity` parameter:

- If you chose the **built-in account** option:

```
blueprintdbcmd.exe /object USER /command ADD /datasource
LOCALHOST /catalog [database] /integratedsec TRUE /nuseridentity
"IIS APPPOOL\[AppPoolName]"
```

Example:

```
blueprintdbcmd.exe /object USER /command ADD /datasource
LOCALHOST /catalog BlueprintDB /integratedsec TRUE /nuseridentity
"IIS APPPOOL\BlueprintAP"
```

- If you chose the **Custom account** option:

```
blueprintdbcmd.exe /object USER /command add /datasource [db_
server]\[instancename] /catalog [database] /integratedsec TRUE
/nuseridentity [new_user_id]
```

Example:

```
blueprintdbcmd.exe /object USER /command ADD /datasource
DBSERVER\INSTANCE01 /catalog BlueprintDB /integratedsec TRUE
/nuseridentity "acme\rrunner"
```

Managing the Blueprint application

This section contains information on managing the Blueprint web application, including:

- Web application server configuration parameters
- About the web.config file

Web application server configuration parameters

> **Tip**
>
> You can type the following commands to view more information about the command parameters:
>
> `blueprintwebcmd.exe /help`
>
> `blueprintdbcmd.exe /help`
>
> `blueprintqcwebcmd.exe /help`

Parameter	Description	Default	Example
/object	Defines the object type of the command. This parameter can be set to one of the following values: - SITE - administration of web sites - APPPOOL - administration of application pools - DBCONFIG - administration of database connection configuration sections - SERVICEKEYCONFIG - administration of key configuration sections - SERVICEDBCONFIG - administration of database connection configuration in services		

Parameter	Description	Default	Example
/command	Defines the command to perform. This parameter can be set to one of the following values: ■ LIST - list the existing objects in the server ■ ADD - add new objects to the server ■ ADDAUX - add new objects to the auxiliary server ■ DELETE - delete the object from the server ■ START - start the object instance on the server ■ STOP - stop the object instance on the server ■ SET - sets the specific configuration values (currently supported only by DBCONFIG object)		
/wsname	Defines the name of the site. This should be the same as the application pool name.	Blueprint	Blueprint
/wsid	Defines the ID of the site.		25
/port	Defines the port number used for the site.		8080
/auxSiteUrl	Defines the ID of the site.	Blueprint_ Auxiliary	Blueprint_Auxiliary
/auxSitePort	Defines the port number used by the auxiliary site.		9101
/httpHostname	Hostname for the site		
/https	Flag specifying SSL install of the site		
/sslcert	Name of IIS server certificate		
/sslcertstore	Name of IIS certificate store		
/dir	Defines the location of the Blueprint installation.		C:\Program Files\Blueprint\Web
/apppoolname	Defines the name of the application pool.	Blueprint	Blueprint
/serviceapppoolname	Defines the name of the primary services application pool.	Blueprint_ PrimaryServices	Blueprint_ PrimaryServices
/datasource	Defines your database and instance names.		DBSERVER\INSTANCE01
/catalog	Defines the name of the database.	Blueprint	

Parameter	Description	Default	Example
/adminstoragedatasource	Defines the AdminStorage database.		
/adminstoragecatalogprefix	Defines the initial catalog prefix for AdminStorage. (Note the _ AdminStorage suffix is automatically added to this value, and should be omitted from the command-line value.)		Blueprint (which becomes Blueprint_ AdminStorage)
/filestoragedatasource	Defines the FileStorage database.		
/filestoragecatalogprefix	Defines the initial catalog prefix for FileStorage. (Note the _FileStorage suffix is automatically added to this value, and should be omitted from the command-line value.)		Blueprint (which becomes Blueprint_FileStorage)
/integratedsec	Defines whether or not Windows security is used. This parameter can be set to one of the following values: ■ TRUE ■ FALSE If /integratedsec is set to FALSE, you must specify a /userid and /password.		
/userid	Defines the username of the Service Account/Application Pool user.		
/password	Defines the password of the Service Account/Application Pool user.		
/multisite	Flag used for multi-site deploy		

About the web.config file

Overview

> **Note:** Before upgrading Blueprint, we recommend backing up the **web.config** file. None of your existing customizations are preserved during the upgrade process. The backup file can be used as a reference to make changes to the new version of **web.config** after upgrading.

A variety of Blueprint application settings can be configured within the **web.config** file, such as LDAP timeout. Typically the **web.config** file is installed here: **C:\Program Files\Blueprint Software Systems\Blueprint\Web**.

Instructions on editing the web.config file

> **Tip:** See the table of configuration settings below for a list of commonly edited parameters.

To edit the web.config file:

1. Open the **web.config** file in a text editor.
2. Locate the value you want to edit and then change the value.
3. Save the file.

The file has been successfully edited.

Web.config parameters

Blueprint web.config

The Blueprint **web.config** file contains many Blueprint web application settings, most of which should not be edited in common configuration scenarios. The following table identifies Blueprint settings that are commonly configured as well as their possible values:

Section	Parameter name	Default value	Description
`<appSettings>` `<add key="UseLegacyDomainName">`	value	false	The `UseLegacyDomainName` key indicates the domain name section. You can use the original parsing algorithm to determine the domain name. There is no need to change the default value in most circumstances. However, if you have problems integrating Blueprint with LDAP, you may need to change the value to `TRUE`. Please contact Blueprint Support before changing this value.

Section	Parameter name	Default value	Description

`<appSettings>`
 	
`  <add key="ExcelImportArtifactLimit">`
	value	500	This section defines the amount of artifacts that a user can import from Microsoft Excel at a time.
			
			Note: We do not recommend changing this value unless it is necessary.

`<appSettings>`
 	
`  <add key="LdapGetTimeout">`
	value	300	Indicates how many seconds it takes for LDAP retrieval to time out. By default, the LDAP retrieval timeout is `300` seconds (recommended).
			
			The value can be changed. The minimum value it can be changed to is `30` and the maximum is `600`.

`<appSettings>`
 	
`  <add key="QCMaxRetries">`
	value	30	This section defines the maximum number of connection re-attempts that will be made when connection to HP Quality Center initially fails. By default, the value is `30` (recommended).
			
			To change the value, the following tag must be manually added to the `<connectionStrings>` section first:
			
			`<add key="QCMaxRetries" value="">`
			
			The minimum value it can be changed to is `3` and the maximum is `60`.

Section	Parameter name	Default value	Description
`<appSettings>` `  <add key="QCDelayInSeconds">`	value	30	This section defines the delay between connection re-attempts that will be made when connection to HP Quality Center initially fails. The delay is measured in seconds. To change the value, the following tag must be manually added to the `<connectionStrings>` section first: `<add key="QCDelayInSeconds" value="">` The recommended value is 5. The minimum value it can be changed to is 5 and the maximum is 60.
`<appSettings>` `  <add key="AccessControl">`	value	None	Indicates the location of the auxiliary AccessControl service.
`<appSettings>` `  <add key="ConfigControl">`	value	None	Indicates the location of the auxiliary ConfigControl service.

`<connectionStrings>`
`  <add connectionString>`
	connectionString	`metadata=res://*/Models.Instance.csdl|`
		
		`res://*/Models.Instance.ssdl|`
		`  res://*/Models.Instance.msl;`
		
		`provider=System.Data.SqlClient;`
		`  provider connection string=`
		`    "data source=BlueprintDevDB;`
		`  Initial Catalog=Blueprint;`
		`  Integrated Security=True;`
		
		`MultipleActiveResultSets=True;`
			The connection string that points to the Blueprint database.

Blueprint AdminStore web.config

The AdminStore service is the interface to administration functionality and logins. Its **web.config** file's settings likely do not require editing in typical deployment scenarios, with the exception of the following connection parameters:

Section	Parameter name	Default value	Description
`<appSettings>` `<add key="AccessControl">`	value	http://localhost:9101/svc/accesscontrol/	The URL for the auxiliary AccessControl service.
`<appSettings>` `<add key="ConfigControl">`	value	http://localhost:9101/svc/configcontrol/	The URL for the auxiliary ConfigControl service.

`<connectionStrings>`
  `<add name="AdminStorage">`
	connectionString	```data source=BlueprintDevDB;
		    initial catalog=Blueprint_AdminStorage;
		    integrated security=True;
		    min pool size=4;
		    max pool size=128;
		    load balance timeout=2;
		    packet size=16384;
		    connectretrycount=3;
		    connectretryinterval=3;
		    MultipleActiveResultSets=True;```
			The connection string that points the AdminStore service to the AdminStorage database.

`<connectionStrings>`
  `<add name="RaptorMain">`
	connectionString	```data source=BlueprintDevDB;
		    initial catalog=Raptor;
		    integrated security=True;
		    min pool size=4;
		    max pool size=128;
		    load balance timeout=2;
		    packet size=16384;
		    connectretrycount=3;
		    connectretryinterval=3;
		    MultipleActiveResultSets=True;```
			The connection string that points the AdminStore service to the Blueprint database.

Blueprint FileStore web.config

The FileStore service is the interface to files accessed from Blueprint. Its **web.config** file's settings likely do not require editing in typical deployment scenarios, with the exception of the following storage and connection parameters:

Section	Parameter name	Default value	Description
`<appSettings>` `<add key="FileChunkSize">`	value	1	The chunk size for files stored in the FileStorage database, in megabytes. The recommended value is the default of 1.

Section	Parameter name	Default value	Description

`<appSettings>`
 	
`  <add key="ConfigControl">`
	value	http://localhost:9101/svc/configcontrol/	The URL for the auxiliary ConfigControl service.

`<connectionStrings>`
 	
`  <add name="FileStoreDatabase">`
	connectionString	`data source=BlueprintDevDB;`
		`  initial catalog=Blueprint_`
		`FileStorage;`
		`  integrated security=True;`
		`  min pool size=4;`
		`  max pool size=128;`
		`  load balance timeout=2;`
		`  packet size=16384;`
		`  connectretrycount=3;`
		`  connectretryinterval=3;`
		`  MultipleActiveResultSets=True;`
			The connection string that points the FileStore service to the FileStorage database.

`<connectionStrings>`
 	
`  <add name="FileStreamDatabase">`
	connectionString	`data source=BlueprintDevDB;`
		`  initial catalog=Raptor;`
		`  integrated security=True;`
		`  min pool size=4;`
		`  max pool size=128;`
		`  load balance timeout=2;`
		`  packet size=16384;`
		`  connectretrycount=3;`
		`  connectretryinterval=3;`
		`  MultipleActiveResultSets=True;`
			The connection string that points the FileStore service to the Blueprint database.
			
			The use of the Blueprint database is a fallback, for files not found in the FileStorage database. This scenario applies in cases where Blueprint is upgraded to v7.0, but the FileStore Migration Utility has not been run.

Auxiliary AccessControl web.config

The AccessControl service handles sessions and authentication. Its **web.config** file's settings likely do not require editing in typical deployment scenarios, with the exception of the following timeout and connection parameters:

Section	Parameter name	Default value	Description
`<appSettings>` `<add key="SessionTimeoutInterval">`	value	1200	Interval to remove terminated sessions from the database (in seconds). This value overrides the `<add key="SessionLifetimeInterval" />` section in the Blueprint `web.config` file. (Note that the latter is measured in minutes.)
`<appSettings>` `<add key="ConfigControl">`	value	http://localhost:9101/svc/configcontrol/	The URL for the auxiliary ConfigControl service.

`<connectionStrings>`
`<add name="AdminStorage">`
	connectionString	`data source=BlueprintDevDB;`
		`  initial catalog=Blueprint_`
		`AdminStorage;`
		`  integrated security=True;`
		`  min pool size=4;`
		`  max pool size=128;`
		`  load balance timeout=2;`
		`  packet size=16384;`
		`  connectretrycount=3;`
		`  connectretryinterval=3;`
		
		`MultipleActiveResultSets=True;`
			The connection string that points the AccessControl service to the AdminStorage database.

Auxiliary ConfigControl web.config

The ConfigControl service handles configuration settings and logging. Its **web.config** file's settings likely do not require editing in typical deployment scenarios, with the exception of the following:

Section	Parameter name	Default value	Description

`<connectionStrings>`
`<add name="AdminStorage">`
	connectionString	`data source=BlueprintDevDB;`
		`  initial catalog=Blueprint_AdminStorage;`
		`  integrated security=True;`
		`  min pool size=4;`
		`  max pool size=128;`
		`  load balance timeout=2;`
		`  packet size=16384;`
		`  connectretrycount=3;`
		`  connectretryinterval=3;`
		`  MultipleActiveResultSets=True;`
			The connection string for the AdminStorage database.

Changing a SQL port number in the connection string

Some database administrators prefer to use a different port number than the standard SQL port 1433. You can change the SQL port number to access the Blueprint database by appending the port number to the connection string in the **web.config** file.

To change the SQL port number:

1. Open the **web.config** file in a text editor.
2. Locate the `<connectionStrings>` section.

3. In `Data Source`, add the database server name and the port number separated by a comma, as follows:

```
<connectionStrings>
<add
connectionString="metadata=res://*/Models.Instance.csdl|res://*/Models
.Instance.ssdl|
res://*/Models.Instance.msl;provider=System.Data.SqlClient;provider
connection string=&quot;Data Source=192.168.10.10,1433;Initial
Catalog=blueprint;Integrated
Security=True;MultipleActiveResultSets=True&quot;"
name="InstanceContainer" providerName="System.Data.EntityClient" />
</connectionStrings>
```

4. Save the file.

Clearing the Silverlight cache

Every Blueprint user has a Silverlight cache on his or her computer that stores their Blueprint settings.

Clearing the Silverlight cache is a last resort troubleshooting step to the following issue:

- The login dialog box does not appear when opening Blueprint in a browser.

> **Warning:** When you clear a Blueprint user's cache, all of the user's settings are reset to the default.

We do not recommend clearing the Silverlight cache as a solution to most other issues.

Impact

The Silverlight cache contains the user settings for the following commands:

- *Options* dialog box (on the menu)
    - Default Font Size
    - Auto Save
      If Auto Save is enabled, how often work is auto-saved in minutes
    - Disable Spell Check
    - Messages in information bar section
- *View* tab (on the ribbon)
    - **Views** button (*Use Cases* group)
    - Wrap Text (*Artifact List* group)
    - Full Text
    - Alternative Layout
    - Curved Lines
- *Show Indicators* group (*View* tab)

- **Relationships** indicator
- **Discussions** indicator
- **Follow** indicator
- **Files** indicator
- **UI Mockup** indicator

- Explorer
  - *Explorer* panel (expanded or minimized)
  - *Explorer* panel width
  - Activities (*Explorer*, Activity Center)

- Utility Panel
  - Utility Panel (expanded or minimized)
  - Utility Panel width
  - *Properties* section, expanded or collapsed (*Properties* tab)
  - *Author History* section, expanded or collapsed (*Properties* tab)
  - **Description** field (*Properties* tab)
  - *Details* section, expanded or collapsed (*Properties* tab)
  - **Search** button filters (*Browse* tab)
  - **Relationships** menu (*Relationships* tab)

- *Confirmation* dialog boxes
  - Changes to traces that cannot be undone (confirmation dialog box)
  - Changes to attachments and document references that cannot be undone (confirmation dialog box)
  - *Traceability matrix confirmation* dialog box

- Miscellaneous
  - System view settings for all folders (column visibility, width, order, sorting and filtering)
  - The IDs of projects that the user has open
  - Login username

To clear the Silverlight cache, complete the following steps:

1. Close any browsers with Blueprint you have open.

2. Delete the contents of the local is folder, replacing `USER ID` with the ID of the user: `C:\Users\USER ID\AppData\LocalLow\Microsoft\Silverlight\is`

> **Note:** If you are unable to access the `AppData` folder above, you need to show hidden files. For more information on showing hidden files, go here: http://windows.microsoft.com/en-au/windows-vista/show-hidden-files.

The Blueprint user settings are restored to the default.

Changing the Blueprint Server User of the Blueprint application

Blueprint gives administrators the ability to change the Blueprint server user of the Blueprint application. For more information on this topic, see Changing the Blueprint Server User of the Blueprint application.

Using a later version of Team Foundation Server

By default, support for Team Foundation Server 2010 is configured within the **web.config** file. The default support settings can be overridden to optimize support for a newer version of Team Foundation Server.

To change the default Team Foundation Server settings:

1. Open the **web.config** file in a text editor.
2. Uncomment the following block of code to override with support for TFS 2012 and later:

```
<!--
<dependentAssembly>
<assemblyIdentity name="Microsoft.TeamFoundation.Client"
publicKeyToken="b03f5f7f11d50a3a" culture="neutral" />
<bindingRedirect oldVersion="10.0.0.0-10.65535.65535.65535"
newVersion="11.0.0.0" />
</dependentAssembly>
<dependentAssembly>
<assemblyIdentity name="Microsoft.TeamFoundation.Common"
publicKeyToken="b03f5f7f11d50a3a" culture="neutral" />
<bindingRedirect oldVersion="10.0.0.0-10.65535.65535.65535"
newVersion="11.0.0.0" />
</dependentAssembly>
<dependentAssembly>
<assemblyIdentity
name="Microsoft.TeamFoundation.VersionControl.Client"
publicKeyToken="b03f5f7f11d50a3a"
culture="neutral" />
<bindingRedirect oldVersion="10.0.0.0-10.65535.65535.65535"
newVersion="11.0.0.0" />
</dependentAssembly>
<dependentAssembly>
<assemblyIdentity
name="Microsoft.TeamFoundation.TestManagement.Common"
publicKeyToken="b03f5f7f11d50a3a"
culture="neutral" />
<bindingRedirect oldVersion="10.0.0.0-10.65535.65535.65535"
newVersion="11.0.0.0" />
</dependentAssembly>
<dependentAssembly>
<assemblyIdentity
name="Microsoft.TeamFoundation.TestManagement.Client"
publicKeyToken="b03f5f7f11d50a3a"
culture="neutral" />
```

```
<bindingRedirect oldVersion="10.0.0.0-10.65535.65535.65535"
newVersion="11.0.0.0" />
</dependentAssembly>
<dependentAssembly>
<assemblyIdentity
name="Microsoft.TeamFoundation.WorkItemTracking.Client"
publicKeyToken="b03f5f7f11d50a3a"
culture="neutral" />
<bindingRedirect oldVersion="10.0.0.0-10.65535.65535.65535"
newVersion="11.0.0.0" />
</dependentAssembly>
-->
```

3. Save the file.

Managing spell check and the dictionary

Blueprint gives users the ability to add terms to the dictionary. Once a user adds a word to the dictionary, spell check identifies the word as valid and no longer underlines the word in red. After being added to the dictionary, the word cannot be removed except by clearing the entire dictionary database.

The default language of the spell check and the dictionary is **en-US (English - United States)**. However, the spell check and dictionary features can be configured to use any of the supported languages. All of the languages Blueprint supports are visible in the following folder: **C:\Program Files\Blueprint Software Systems\Blueprint\Web\Dictionary\Languages** You can enable as many spell check and dictionary languages as you wish.

Clearing the dictionary database

Blueprint provides administrators with the ability to clear the dictionary database. In some circumstances the dictionary database may need to be cleared in order to clean up the dictionary.

> **Note:** Once a user adds a term to the dictionary, the term cannot be removed except by clearing the entire dictionary database.

Clearing the dictionary database consists of a copy and paste operation that overrides the current dictionary database. The default dictionary language is **en-US (English - United States)** but, if needed, any supported language can be used to override the current dictionary database.

To clear the dictionary database:

1. Navigate to the folder containing the language you want to use.

   For example, the default language is English (United States) and thus the folder is **C:\Program Files\Blueprint Software Systems\Blueprint\Web\Dictionary\Languages\en-US (English - United States)**

2. Copy the **dictionary.dct** file.

> **Warning:** Once the dictionary database is cleared, all terms that have been added are cleared and cannot be retrieved.

3. Overwrite the active (default) dictionary file with the copied dictionary file.

    The active (default) dictionary file is stored here:

    ```
    C:\Program Files\Blueprint Software
    Systems\Blueprint\Web\Dictionary\default\dictionary.dct
    ```

The dictionary has been immediately restored to the language default. If a client was already using Blueprint prior to the dictionary update, the dictionary cleanup becomes effective after the client browser cache is cleared.

Changing the spell check and dictionary language

Blueprint allows you to change the default spell check and dictionary language. Alternatively, you can add more languages to the default spell check and dictionary.

After a dictionary has been configured in the **web.config** file, all Blueprint users on that particular instance will automatically utilize that particular dictionary for their spell check needs. Users can disable spell checking but they cannot configure the dictionary.

To change the spell check dictionary language:

1. Locate the dictionary file that you want to use.

    The dictionary files (**dictionary.dct**) for each supported language are available in the following folder:

    ```
    C:\Program Files\Blueprint Software
    Systems\Blueprint\Web\Dictionary\Languages\
    ```

2. Overwrite the active (default) dictionary file with the desired dictionary file.

    The active (default) dictionary file is stored here:

    ```
    C:\Program Files\Blueprint Software
    Systems\Blueprint\Web\Dictionary\default\dictionary.dct
    ```

> **Example**
>
> If you want your users to utilize a French spell check dictionary, copy this file:
>
> ```
> C:\Program Files\Blueprint Software
> Systems\Blueprint\Web\Dictionary\Languages\fr-FR (French -
> France)\dictionary.dct
> ```
>
> and use it to overwrite this file:
>
> ```
> C:\Program Files\Blueprint Software
> Systems\Blueprint\Web\Dictionary\default\dictionary.dct
> ```

The new spell check dictionary becomes effective immediately. If a client was already using Blueprint prior to the dictionary update, the new spell check dictionary becomes effective after the client browser cache is cleared.

Enabling a language in the spell check and the dictionary

Any of the languages Blueprint supports can be enabled in the spell check and the dictionary. There is no limit to the supported languages you can enable. All of the languages Blueprint supports can be found in the following folder: `C:\Program Files\Blueprint Software Systems\Blueprint\Web\Dictionary\Languages\`

To enable a language in the spell check and the dictionary, complete the following:

1. Create a new folder named **Backup** in the following directory: `C:\Program Files\Blueprint Software Systems\Blueprint\Web\Dictionary\`
2. Copy the `dictionary.dct` file (the default) that you can find in the following folder: `C:\Program Files\Blueprint Software Systems\Blueprint\Web\Dictionary\default`
3. Paste the copied `dictionary.dct` file into the **Backup** folder.
4. Rename the original `dictionary.dct` file (found in `C:\Program Files\Blueprint Software Systems\Blueprint\Web\Dictionary\default`) to `dictionary.zip`.
5. Navigate to the `C:\Program Files\Blueprint Software Systems\Blueprint\Web\Dictionary\Languages\` folder and then open the folder containing the language you want to enable.
6. Rename the `dictionary.dct` file to `dictionary.zip`.
7. Unzip the `dictionary.zip` file.
8. Extract the `.words` file from the `dictionary.zip` file.
9. Rename the `dictionary.zip` file to `dictionary.dct`.
10. Copy the extracted `.words` file to the `C:\Program Files\Blueprint Software Systems\Blueprint\Web\Dictionary\default` folder.
11. Paste the newly copied `.words` file into the `dictionary.zip` file.
12. In the `C:\Program Files\Blueprint Software Systems\Blueprint\Web\Dictionary\default` folder, rename the `dictionary.zip` file back to `dictionary.dct`.

The language has successfully been enabled.

Managing job services

This section provides the following information on managing job services:

- Installing services
- Adding a database user (command line)
- Configuring a job service

Installing services

Typically, most administrators deploy job services as a part of the Blueprint installation or upgrade process (that is, using the install wizard). For more information on installing job services alongside Blueprint, see the *Blueprint Installation Guide* or *Blueprint Upgrade Guide*.

This section addresses how to manually install services. You may choose to manually install services if you did not select the option to install services when you installed or upgraded Blueprint.

> **Note:** Only Blueprint database users can manually install the Windows services. For more information about how to add a database user, see Adding a database user (command line).

The following Windows services are available to install if needed:

- **Job services (recommended)**

  This functionality is necessary to perform the following jobs in Blueprint: document generation, exporting artifacts to ALM systems and test generation.

- **Legacy support for HP ALM versions 12 and earlier (optional)**

  If you require support for HP ALM version 12 or earlier, setup of this component is necessary.

  > **Note:** HP ALM support is only available for COM library. It is not available for the REST API.

To install the Windows-based services:

- Continue to the single-server section to host job services and the Blueprint database on the same server.
- Continue to the distributed-server section to host job services and the Blueprint database on separate servers.

Adding a database user (command line)

You can add any of the following database user types using the `/nuseridentity` parameter:

- Local Service: **LocalService**
- Local System: **LocalSystem**
- Network Service: **NetworkService**
- A Windows services account. Example: **blueprint\jsmith**.

Add a database user:

```
blueprintdbcmd.exe /object USER /command ADD /datasource
[DBSERVER\INSTANCE01] /catalog [BlueprintDB] /integratedsec FALSE /userid
[dbadmin] /password [pAssw0rd] /nuseridentity LocalService
```

Setting up services (single-server setup)

You can set up one or both of the following:

- 64-bit job services
- HP ALM legacy support

Setting up 64-bit job services (single-server)

Installing 64-bit job services involves completing the following steps:

- Step One: Configuring 64-bit job services
- Step Two: Finalizing the 64-bit job services setup

Step One: Configuring 64-bit job services

1. Open the following configuration file in a text editor: **BluePrintSys.RC.JobExecutor.exe.config**

2. Make sure that the value in the `<add key="Service.Jobs">` tag is as follows:

    ```
    <add key="Service.Jobs" value="
    DocGen,TfsExport,HpAlmRestExport,TfsChangeSummary,HpAlmRestChangeSumma
    ry,TfsExportTests,HpAlmRestExportTests" />
    ```

3. Make sure the `<add key="Service.Name">` tag specifies the 64-bit job service as follows:

    ```
    <add key="Service.Name" value="Blueprint Job Service (64 bit)" />
    ```

    > **Note:** If you are setting up the second 64-bit job service in this step, specify the `Service.Name` value as `Blueprint Job Service 2 (64 bit)` instead of the value listed above.

4. Replace the following `connectionString` value with the Blueprint database connection string:

    ```
    <connectionStrings>
    <add name="InstanceContainer" connectionString="
    metadata=res://*/Models.Instance.csdl|res://*/Models.Instance.ssdl|res
    ://*/Models.Instance.msl;provider=System.Data.SqlClient;provider
    connection string=&quot;Data Source=.\MSSQLSERVER;Initial
    Catalog=Blueprint;Integrated
    Security=True;Pooling=True;MultipleActiveResultSets=True&quot;"
    providerName="System.Data.EntityClient" />
    </connectionStrings>
    ```

> **Note:** `Data Source` must specify the SQL instance name and the Blueprint instance name. If your SQL instance has a name that is different from **`MSSQLSERVER`** and/or your Blueprint instance is not named **`Blueprint`**, you need to change the value(s).

Step Two: Finalizing the 64-bit job services setup

1. Install both 64-bit job services by running the following command with your user name and password:

```
BluePrintSys.RC.JobExecutor.exe -c Install -a User -u [USERNAME] -p
[PASSWORD]
```

> **Note:** To install the service using the default Windows account, run the following command instead:
>
> ```
> BluePrintSys.RC.JobExecutor.exe -c Install -a LocalService
> ```

2. Start the 64-bit job services with the following command:

```
BluePrintSys.RC.JobExecutor.exe -c Start
```

You have successfully deployed the 64-bit job services.

> **Important:** Repeat steps one and two to configure the second 64-bit job service.

Setting up HP ALM legacy support (single-server)

Installing HP ALM legacy support involves completing the following steps:

- Step One: Setting up the HP ALM legacy support connector
- Step Two: Configuring 32-bit job services for HP ALM legacy support
- Step Three: Finalizing the HP ALM legacy support setup

Step One: Setting up the HP ALM legacy support connector

1. Set up the HP ALM application pool by running the following command:

```
blueprintqcwebcmd.exe  /object AppPool /command ADD /apppoolname
HPQCLegacyConnector
```

2. Set up the HP ALM web site by running the following command (where the number after `/port` is `QcLiteWeb`'s port number):

```
blueprintqcwebcmd.exe  /object Site /command ADD /wsname
HPQCLegacyConnector /dir "C:\Program Files\Blueprint Software
Systems\Blueprint\QcLiteWeb" /port [8081] /apppoolname
HPQCLegacyConnector
```

3. Start the HP ALM application pool by running the following command:

```
blueprintqcwebcmd.exe  /object AppPool /command START /apppoolname
HPQCLegacyConnector
```

4. Start the HP ALM web site by running the following command:

```
blueprintqcwebcmd.exe  /object SITE /command START /wsname
HPQCLegacyConnector
```

5. Set the HP ALM key for Blueprint with the following command (where the number after `/port` is `QcLiteWeb`'s port number):

```
blueprintqcwebcmd.exe /object Config /command SET /dir "C:\Program
Files\Blueprint Software Systems\Blueprint\Web" /port [8081]
```

You have successfully set up the HP ALM legacy support connector.

Step Two: Configuring 32-bit job services for HP ALM legacy support

1. Open the following configuration file in a text editor: **BluePrintSys.RC.JobExecutor32.exe.config**

2. Make sure that the value within `<add key="Service.Jobs">` is as follows:

```
<add key="Service.Jobs" value="QcExport,QcChangeSummary,QcExportTests"
/>
```

3. Make sure the `<add key="Service.Name">` tag includes the Blueprint HP ALM legacy job service as follows:

```
<add key="Service.Name" value="Blueprint HP QC Legacy Job Service (32
bit)" />
```

4. Replace the following `connectionString` value with the Blueprint database connection string:

```
<connectionStrings>
<add name="InstanceContainer" connectionString="
metadata=res://*/Models.Instance.csdl|res://*/Models.Instance.ssdl|res
://*/Models.Instance.msl;provider=System.Data.SqlClient;provider
connection string=&quot;Data Source=.\MSSQLSERVER;Initial
Catalog=Blueprint;Integrated
Security=True;Pooling=True;MultipleActiveResultSets=True&quot;"
providerName="System.Data.EntityClient" />
</connectionStrings>
```

> **Note:** `Data Source` must specify the SQL instance name and the Blueprint instance name. If your SQL instance has a name that is different from **MSSQLSERVER** and/or your Blueprint instance is not named **Blueprint**, you need to change the value(s).

Step Three: Finalizing the HP ALM legacy support setup

1. Install the 32-bit service by running the following command with your user name and password:

   ```
   BluePrintSys.RC.JobExecutor32.exe -c Install -a User -u [USERNAME] -p
   [PASSWORD]
   ```

   > **Note:** To install the service using the default Windows account, run the following command instead:
   >
   > ```
   > BluePrintSys.RC.JobExecutor32.exe -c Install -a LocalService
   > ```

2. Next, start the 32-bit job executor Windows service with the following command:

   ```
   BluePrintSys.RC.JobExecutor32.exe -c Start
   ```

You have successfully set up legacy support for HP ALM.

Setting up Windows-based services (distributed-server setup)

Setting up services on a separate machine (distributed-server) involves these steps:

- Step One: Configuring services
- Step Two: Testing the connection to the database
- Step Three: Finalizing the job services setup

Step One: Configuring services

You have the option of setting up the following services:

- 64-bit job services

- HP Quality Center legacy support

Configuring the 64-bit services

1. Open the following configuration file in a text editor: **BluePrintSys.RC.JobExecutor.exe.config**

2. Make sure that the value in the `<add key="Service.Jobs">` tag is as follows:

   ```
   <add key="Service.Jobs" value="
   DocGen,TfsExport,HpAlmRestExport,TfsChangeSummary,HpAlmRestChangeSumma
   ry,TfsExportTests,HpAlmRestExportTests" />
   ```

3. Make sure the `<add key="Service.Name">` tag specifies the 64-bit job service as follows:

   ```
   <add key="Service.Name" value="Blueprint Job Service (64 bit)" />
   ```

> **Note:** If you are setting up the second 64-bit job service in this step, specify the `Service.Name` value as `Blueprint Job Service 2 (64 bit)` instead of the value listed above.

4. Replace the following `connectionString` value with the Blueprint database connection string:

```
<connectionStrings>
<add name="InstanceContainer" connectionString="
metadata=res://*/Models.Instance.csdl|res://*/Models.Instance.ssdl|res
://*/Models.Instance.msl;provider=System.Data.SqlClient;provider
connection string=&quot;Data Source=.\MSSQLSERVER;Initial
Catalog=Blueprint;Integrated
Security=True;Pooling=True;MultipleActiveResultSets=True&quot;"
providerName="System.Data.EntityClient" />
</connectionStrings>
```

> **Note:** `Data Source` must specify the SQL instance name and the Blueprint instance name. If your SQL instance has a name that is different from **MSSQLSERVER** and/or your Blueprint instance is not named **Blueprint**, you need to change the value(s).

> **Important:** Repeat all of the steps above to configure the second 64-bit job service.

Setting up HP ALM legacy support

Setting up legacy support for HP ALM involves the following steps:

- Step One: Setting up the HP ALM legacy support connector
- Step Two: Configuring the 32-bit job services

STEP ONE: SETTING UP THE HP ALM LEGACY SUPPORT CONNECTOR

1. Set up the HP ALM application pool by running the following command:

```
blueprintqcwebcmd.exe  /object AppPool /command ADD /apppoolname
HPQCLegacyConnector
```

2. Set up the HP ALM web site by running the following command (where the number after `/port` is `QcLiteWeb`'s port number):

```
blueprintqcwebcmd.exe  /object Site /command ADD /wsname
HPQCLegacyConnector /dir "C:\Program Files\Blueprint Software
Systems\Blueprint\QcLiteWeb" /port [8081] /apppoolname
HPQCLegacyConnector
```

3. Start the HP ALM application pool by running the following command:

```
blueprintqcwebcmd.exe  /object AppPool /command START /apppoolname
HPQCLegacyConnector
```

4. Start the HP ALM web site by running the following command:

```
blueprintqcwebcmd.exe  /object SITE /command START /wsname
HPQCLegacyConnector
```

5. Set the HP ALM key for Blueprint with the following command (where the number after `/port` is `QcLiteWeb`'s port number):

```
blueprintqcwebcmd.exe /object Config /command SET /dir "C:\Program
Files\Blueprint Software Systems\Blueprint\Web" /port [8081]
```

You have successfully set up the HP ALM legacy support connector.

STEP TWO: CONFIGURING THE 32-BIT JOB SERVICES

Complete the following instructions, pasting the copied files onto your target machine:

1. Open the following configuration file in a text editor: **BluePrintSys.RC.JobExecutor32.exe.config**

2. Make sure that the value within `<add key="Service.Jobs">` is as follows:

```
<add key="Service.Jobs" value="QcExport,QcChangeSummary,QcExportTests"
/>
```

3. Make sure the `<add key="Service.Name">` tag includes the Blueprint HP ALM legacy job service as follows:

```
<add key="Service.Name" value="Blueprint HP QC Legacy Job Service (32
bit)" />
```

4. Replace the following `connectionString` value with the Blueprint database connection string:

```
<connectionStrings>
<add name="InstanceContainer" connectionString="
metadata=res://*/Models.Instance.csdl|res://*/Models.Instance.ssdl|res
://*/Models.Instance.msl;provider=System.Data.SqlClient;provider
connection string=&quot;Data Source=.\MSSQLSERVER;Initial
Catalog=Blueprint;Integrated
Security=True;Pooling=True;MultipleActiveResultSets=True&quot;"
providerName="System.Data.EntityClient" />
</connectionStrings>
```

> **Note:** `Data Source` must specify the SQL instance name and the Blueprint instance name. If your SQL instance has a name that is different from **MSSQLSERVER** and/or your Blueprint instance is not named **Blueprint**, you need to change the value(s).

Step Two: Testing the connection to the database

> **Note:** This step must be performed for each job service you intend to set up.

This testing procedure must validate the configuration of the following values:

- The `Service.Name` value is unique and no other job service has this name on the current machine
- The `connectionString` value is valid and the job service can connect to Blueprint database.

If the test is not successful, you must specify the correct value(s) in the job service configuration file and re-attempt the test.

To test the job services connection to the Blueprint database:

- Run the following command (where `[JobServiceExecutableFile]` is the name of the job service executable file):

```
[JobServiceExecutableFile] -c test
```

> **Note:** The 32-bit job services file name is typically **BluePrintSys.RC.JobExecutor32.exe** and the 64-bit job services file name is typically **BluePrintSys.RC.JobExecutor.exe**.

Step Three: Finalizing the job services setup

> **Note:** This step must be performed for each job service you intend to set up.

This is the final step in setting up job services on a separate machine from the Blueprint database.

To finalize the setup:

1. Install the job service by running the following command (where `[JobServiceExecutableFile]` is the name of the job service executable file):

```
[JobServiceExecutableFile] -c install
```

2. Start the job service by running the following command (where `[JobServiceExecutableFile]` is the name of the job service executable file):

```
[JobServiceExecutableFile] -c start
```

You have deployed job services.

To verify whether job services have been successfully installed and configured:

1. Log on to Blueprint.

2. Open the *Instance Administration Console* from the **Menu** Menu .

   The *Instance Administration Console* appears.

3. Click **Job Services**.

   The *Job Services* screen appears.

Any job services that have been successfully installed and configured appear in the *Job Services* list. Information about the configured service name and supported jobs also is available in the list.

Adding a new job service

> **Caution:** We do not recommend creating additional services because they place a greater load on the server, affecting Blueprint performance.

A new job service can be created by copying an existing job service folder with the desired 32-bit or 64-bit capabilities, pasting the file into the topmost job services folder and then re-configuring the copied folder and files.

To add a new job service:

1. Navigate to the location of the main job services folder.

   For example, the job services folder is typically found here: `C:\Program Files (x86)\Blueprint Software Systems\Blueprint\JobExecutorService`

2. Copy one of the individual job service folders that contains the 32-bit or 64-bit capability you want to duplicate.

3. Paste the folder into the main job services folder.

4. Rename the folder.

You have successfully completed added a new job service.

Next, configure the job service.

Configuring a job service

If necessary, you can edit the settings of any job service. Each job service has a configuration file containing various settings, including start-up, application and connection strings.

The main settings are as follows:

Section	Key	Description
`<appSettings>`	`Service.Jobs`	The list of job types that users can execute. **Note:** We do not recommend editing or removing any services. If a job service is removed, the unsupported job can still be initiated on the server and any initiated jobs will still appear in the queue.
`<appSettings>`	`Service.Name`	The name of the individual service, which is visible on the *Job Services* tab in Blueprint and in the Windows *Services* window. **Important:** All job services must have unique names. For example, if you have installed two 64-bit job services, you have to edit both configuration files within their respective job services folders.
`<connectionStrings>`	`InstanceContainer`	The connection string to the Blueprint database.

To edit a job service:

1. Navigate to the folder where the job service is installed.

   For example, the first 64-bit service is typically found here: **C:\Program Files (x86)\Blueprint Software Systems\Blueprint\JobExecutorService\x64**

2. Right-click the configuration file and open it with a text editor.

   For example, the configuration file is typically named **BluePrintSys.RC.JobExecutor.exe.config**.

3. Specify new values for the settings you want to change.

   - If this job service has been copied from one of the other job services, specify a unique name in the **Service.Name** tag (where the value equals the unique name).

   > **Note:** We do not recommend editing or removing any services. If a job service is removed, the unsupported job can still be initiated on the server and any initiated jobs will still appear in the queue.

4. Save the file.

After you have configured all job services, verify whether they have been successfully installed and configured.