IBM Rational Build Forge Installation Guide Version 7.1.1.2

Document # GC23-7682-09

Table of Contents

List of Figures

Build Forge Installation Guide

Note

Before using this information and the product it represents, read the information in "Notices for IBM Rational Build Forge Installation Guide" on page 262.

First Edition October 2009.

This edition applies to version 7.1.1.2 of IBM® Rational® Build Forge® and all subsequent releases and modifications until otherwise indicated in new editions.

Document version and build: 7.1.1.2.0.0012.

About IBM Rational Build Forge

This section describes product editions, terminology, and product components.

Build Forge product editions

The following editions of the product are as available:

- IBM Rational Build Forge Standard Edition
- IBM Rational Build Forge Enterprise Edition
- IBM Rational Build Forge Enterprise Plus Edition

The table lists the distinct components or features for the editions.

Component or Function	Standard Edition	Enterprise Edition	Enterprise Plus Edition
Management Console	Windows, UNIX/Linux	Windows, UNIX/Linux	Windows, UNIX/Linux
Database	Supported Windows databases (including DB2 Express) Supported UNIX/Linux databases	Supported Windows databases (including DB2 Express) Supported UNIX/Linux databases	Supported Windows databases (including DB2 Express) Supported UNIX/Linux databases
Agents	All supported operating systems	All supported operating systems	All supported operating systems
License Server	required (25 concurrent users)	required (150 concurrent users)	not required (250 concurrent logins)
Adaptor Toolkit	Supported	Supported	Supported
Quick Report	Supported	Supported	Supported
APIs (Perl, Java)	Supported	Supported	Supported
Dynamic server management	Not supported	Supported	Supported

Components

The Build Forge® system is made up of the following components:

- **Web Client**: Users and administrators use Web browsers to access the the system. Browser clients access the **Web Interface** component.
- **API Client**: Any program using the Java API or Perl API to access Build Forge. API clients access the **Services Layer** directly.

- **Build Forge**: a collective term for the system. During installation the system is shown as made up of **Core Product Features** :
  - **Web Interface**: also referred to as the Management Console or console. This component is made up of a set of PHP modules.
  - **Process Engine**: also referred to as the engine. The engine manages job scheduling and execution.
  - **Services Layer**: a database abstraction layer through which API Clients, Web Interface, and Process Engine make requests.

  In simple installations all three components are installed on the same host. They can be installed on separate hosts. In that case the Build Forge configuration file `buildforge.conf` must be modified so that the components can communicate.
- **Database**: Information storage for the system. The database stores project definitions, system configurations, and user configurations.
- **Agent**: A program installed on a host. An agent must be installed on every host that you want the Management Console to use as a server resource. The agent receives requests to perform work (steps) and runs them on the host where it is installed.

The components can be deployed in a variety of ways, ranging from all components on a single host to a system that uses clustered consoles and a large number of distributed server resources.

Terminology

The primary purpose of the Build Forge system is to enable you to define and run projects. A project accomplishes a task, such as building and packaging software. The power of this system comes from its ability to track, control, and report on the execution of complex tasks distributed over multiple hosts.

Project:	A set of steps to be performed.
Step:	A single executable operation that can be individually scrutinized for success or failure. A project is made up of one or more steps. A step is made up of one or more commands. Each command can start an executable file, a batch file, or script that issues many other commands. A command is anything that can be invoked from the command line of a given server. When you run a project, each step runs in the specified order on the specified server resources. A step can be run in an environment that you specify.
Job	An executing project.
Environment:	Consists of one or more environment variables. Environments can be assigned to servers, projects, or steps.
Server:	An object defined in the console that represents a resource to run a project or step. The Server definition includes a property that points to an agent.

Requirements

This topic lists hardware and software requirements for installing the Build Forge® product components.

Each of the following has specific requirements:

- Management Console
- Database
- Agent
- Web client
- Networking requirements for IPv6 support
- Integration
- National language support

Management Console requirements

This section identifies licensing and operating system requirements.

Operating system requirements

Resource Requirements:

- Disk space: 1 GB of disk space for all product components except the database, which is typically installed on a separate host.

  If the provided DB2 Express database is installed on Windows and on the same host as the core product components, an extra 1 GB of disk space is recommended.

- Memory: 1 GB of RAM, if you are using Quick Report, an extra 1 GB of RAM is recommended.

The Management Console runs on the following operating systems and hardware platforms. The Management Console is compiled as a 32-bit application that can be run on 64-bit operating systems; for example, Solaris 64-bit SPARC.

Windows operating systems:

- **Windows Server 2003**: Service Pack 1 or 2 (32-bit Intel)
- **Windows Server 2003**: Service Pack 2 (64-bit Intel)
- **Windows XP Professional**: Service Pack 2 (32-bit Intel)

UNIX and Linux Systems:

- **AIX 5.3**: all Service Packs (System p, Power5). Also known as AIX 5300-05.
- **Red Hat Enterprise Linux 4 Server**: includes all updates and Service Packs, (32-bit Intel)
- **Red Hat Enterprise Linux 5 Server**: includes all updates and Service Packs (32-bit Intel)
- **Red Hat Enterprise Linux 5.1 Server**: includes all updates and Service Packs (32-bit Intel and 64-bit Intel)
- **SUSE Linux Enterprise Server 10**: all Service Packs (32-bit Intel).
- **Solaris 9**: all Service Packs, (32-bit SPARC and 64-bit SPARC)
- **Solaris 10**: all Service Packs, (32-bit SPARC and 64-bit SPARC)

z/Linux:

- **SUSE Linux Enterprise Service 10**: Service Pack 2 (IBM System z: S390x). *This platform is supported only with the Enterprise Plus Edition.*

Licensing requirements

Rational Build Forge licensing is either file-based or server-based, depending on your product edition.

For server licenses, Rational License Server V7.0, is included in the product distribution. The 7.0 version supports IPv4 addresses and must be installed on a host machine that uses IPv4 addresses. If you need to support IPv6 addresses, review the following information:

- Install Rational License Server V7.1; it supports IPv4 and IPv6 addresses. Obtain version 7.1 from IBM Rational Support or the IBM Rational Download and Licensing Center.
- For information about IPv6 support in the product, see “Networking requirements for IPv6 support” on page 12.

For server licenses, a FLEXlm client is automatically installed and configured. The provided FLEXlm client supports Rational License Server versions 7.0 and 7.1. Only configure a different version of the FLEXlm client licensing software if you must use IPv6 addresses. For details, see Setting up the FLEXlm client.

For more information about licensing, see “Licensing requirements for product editions” on page 8.

Database requirements

A copy of DB2 Express is packaged with the product. Your Rational Build Forge product license entitles you to use the DB2 Express that is bundled with the product on Windows only; you must install DB2 Express on the Management Console host.

For any database you choose to install, you must click **Test connection** to test the settings. Invalid settings, or fields that you need to provide a valid entry for, will be highlighted. You must enter valid settings before you can proceed with any installation. You may need to scroll down to see the **Test Connection** button.

The following databases and versions are supported:

- DB2 Express 9.1.1
- DB2 9.1
- MySQL 5.0.45 and 5.0.15 a-community-nt
- Microsoft® SQL Server 2005 Service Pack 3**on Windows platforms only**. *Support for SQL Server is for English-only, because it does not support UTF-8.* The database must be created using a case-insensitive collation, such as the default: `SQL_Latin1_General_CP1_CI_AS`
- Oracle 10.2.0 (a separate install of Oracle Instant Client is required for PHP oci8 drivers)

Note

DB2/390 on System z is not a supported database.

Agent requirements

The agent runs on the following operating systems and hardware platforms.

All agents require less than 5 MB of disk space when installed. An agent running idle requires less than 2 MB of memory.

An agent running steps requires more memory resources. For example, running a step to perform a code checkout is a common agent task. Set up a test environment to determine how much RAM memory-intensive tasks will require.

Important

The agent is provided as a 32-bit application that can be run on a 64-bit operating systems; for example, Solaris 64-bit SPARC. Do not compile the agent source pack as a 64-bit application.

Windows

All Windows operating systems use this Windows program binary file: win-bfagent-<*version*>.exe.

- **Microsoft Windows Server 2003**: all Service Packs, 32-bit and 64-bit Intel
- **Microsoft Windows Server 2008**: Service Pack 2, 32-bit and 64-bit Intel
- **Microsoft Windows XP Professional**: Service Pack 2, 32-bit and 64-bit Intel
- **Microsoft Windows Vista Business, Windows Vista Enterprise, Windows Vista Ultimate**: all Service Packs, 32-bit and 64-bit Intel

UNIX and Linux

Use the program binary file that is unique to your UNIX/Linux operating system. For example, rhel5-bfagent-<*version*>.rpm.

- **AIX 5.3**: 32-bit and 64-bit, System p, Power 5. The agent can run in a System WPAR on AIX.
- **HP-UX 11i (v1, v2, v3)**: 32-bit or 64-bit, HP 700, 800: PA-RISC
- **Solaris 9, 10** : 64-bit SPARC, 32-bit SPARC
- **Red Hat Enterprise Linux 4**: 32-bit Intel
- **Red Hat Enterprise Linux 5**: 32-bit Intel
- **Red Hat Enterprise Linux 5.1**: 64-bit Intel
- **Macintosh OS X 10.x** :
- **Macintosh OS X Server 10.x**:

System i

Use this System i binary file: iseries-bfagent-<*version*>-tar.gz.

- **IBM System i i5/OS V5R4**: AS/400

System z

Use the source pack to build the agent on z/OS: src-bfagent-<*version*>.tar.gz. The agent source code for z/OS is provided as uncompiled source only; a binary distribution is not available.

- **IBM System z 1.6, 1.7, 1.8, and 1.9**: S390x

Agent version interoperabilility

In Version 7.1 there was a change in the protocol used by the agent and console to communicate.

- Agents from prior versions still work with version 7.1+ consoles.
- Agents from version 7.1 onward do not work with consoles older than version 7.1.

It is preferable to maintain the console and agent at the same version.

Web client requirements

Minimum resolution levels and certain Web browser versions are required when you access the Management Console from a client computer.

Use a web browser to access Management Console as a client.

- These browsers are supported:

  - **Internet Explorer version 6 and 7**: Enable ActiveX controls; otherwise, the console server host must be added to the Trusted Sites list. (In the browser, click **Tools > Internet Options > Security**). Note that Windows 2003 disables ActiveX controls by default.
  - **Mozilla Firefox:** versions 2.0 and 3.0
- Minimum display resolution:
  - 1024 x 768 minimum in order to display the Management Console correctly. A resolution of 1280 x 1024 or higher displays the Management Console more clearly.

Note

Do not shrink the browser window smaller than the size that the Build Forge application uses (1024 x 768). In some cases shrinking the window hides necessary controls and causes unpredictable behavior.

Note

All Web clients must clear their cache after you update Management Console (upgrade installations) or apply an iFix.

Licensing requirements for product editions

The licensing mechanism that you use depends on the product edition that you have. See “Build Forge product editions” on page 2.

Edition	License mechanism
Rational Build Forge Standard Edition	License server
Rational Build Forge Enterprise Edition	License server
Rational Build Forge Enterprise Plus Edition	License file

This topic contains details about setting up a Rational license server for Build Forge. You can specify a new license server through the Build Forge UI. Managing licenses through IBM Installation Manager is not supported.

Specifying a license file during installation

This topic applies only to product editions that require file-based licensing.

- Build Forge Enterprise Plus Edition

The license file is included with the product distribution and installed with the Build Forge product files.

1. In the Installation Manager installer, on the License Server Configuration page, select **License File**.

2. Click **Browse** to locate the license key file, irbf_license.properties, in the root installation directory, for example:

Windows	C:\Program Files\IBM\Build Forge
UNIX/Linux	/usr/local/bin

3. Double-click the **irbf_license.properties** file to select it and click **Next** to continue with the installation.

Configuring a Rational license server for Build Forge

Your license administrator sets up a Rational license server and provides you the license server host name that you specify during installation.

This section applies only to product editions that require server-based licensing.

- Build Forge Standard Edition
- Build Forge Enterprise Edition

Build Forge configuration requirements for the license server

Before configuring the license server for Build Forge, review the following requirements.

- All of the Management Console host machines in your environment must be able to connect to the Rational license server host machine.
- The UNIX/Linux or Windows Rational license server must be set up to start automatically and run as a service.
- To install and configure the product to use the license server, the license administrator must provide you the host name and port number of the Rational license server.
- The Rational Build Forge 7.1 product distribution includes Rational License Server 7.0. The FLEXlm client software that is required to communicate with the license server is installed and configured as part of the Management Console installation.

Specifying a license server during installation

During installation, you provide the host name and TCP/IP port of the Rational license server. Obtain this information from your license administrator.

To configure a server-based license:

1. In the Installation Manager installer, on the License Server Configuration page, select **License Server** .

2. At **License Server** , provide a valid host name for the Rational license server.

   If you provide a host name, this information is automatically added to Build Forge system settings.

   If you do not know the host name, enter any character or value in this field and update Build Forge system settings in the UI after installation is complete.

   Important

   Do not leave this field blank. Leaving this field blank might result in an incomplete installation and an unusable product.

3. At **Port** , provide the TCP/IP port for the license server. The default port is 27000.
4. Click **Next** to continue with the installation.

Changing the license server for the Management Console

To change the Rational license server for the Management Console, make the following modifications to the product license server configuration. You must perform this procedure if:

- You entered an incorrect host name or other value during installation.
- The license server host name was greyed out during installation, indicating that the FLEXlm client already registered a license server for the host.

1. For your operating system, change the value of the RATIONAL_LICENSE_FILE variable.

Windows	RATIONAL_LICENSE_FILE is in the registry at `HKEY_LOCAL_MACHINE\SOFTWARE\FLEXlm License Manager`
UNIX/Linux	RATIONAL_LICENSE_FILE is set in the `.flexlmrc` file, located in the home directory of the user who is running Build Forge

   Set this variable to the correct host name of the license server:

   ```
   port@hostname | @hostname
   ```

   If the license server port is 27000 (the default), then a port is not required.

2. In the Management Console UI, select **Administration** → **System** .
3. Locate the License Server setting and set its value to the host name of the new Rational license server.

   Use one of the following formats.

   ```
   <host_name>:<port> | <host_name>  | <port>:<host_name>
   ```

   If the license server port is 27000 (the default), then a port is not required.

4. Click **Refresh** on your Web browser to verify that the Management Console can connect to the new license server.

Obtaining license keys and setting up a Rational licensing server

If an existing license server is not available, the following table identifies the general tasks that the license administrator completes to obtain license keys and set up the Rational license server.

To install and configure a Rational license server, review the documentation for your version of the Rational License Server software. Go to http://www-306.ibm.com/, select **Support and downloads**, and search for the Rational License Management Guide.

License administrator task	Resource
Obtains license keys from the Rational License Key Center.	Rational License Management Guide Quick Start Guide for Rational License Key Center at http://www-306.ibm.com/software/rational/support/licensing
Verifies network connectivity for Management Console hosts and the Rational license server.	Rational License Management Guide
Obtains the required version of the Rational License Server software by either: • Accessing the software included with the product distribution. • Downloading the software from the IBM Rational Download and Licensing Center at: https://www14.software.ibm.com/webapp/iwm/web/reg/signup.do?source=rational	Rational License Management Guide
Installs a Rational license server	Rational License Management Guide
Installs or imports license keys to the Rational license server, as follows: • On Windows, use the IBM Rational License Key Administrator (LKAD), installed with many IBM Rational products and with the Rational License Server software. • On UNIX/Linux, use the license_setup script and licensing executables from the IBM Rational Download and Licensing Center at: https://www14.software.ibm.com/webapp/iwm/web/reg/signup.do?source=rational	Rational License Management Guide
Starts the Rational license server and sets up the Rational license server to start up automatically and run as a service, as follows:	Rational License Management Guide

License administrator task	Resource
• On Windows, the Rational License Server software is automatically set to start up as a service when the computer starts. If it does not start automatically, see the instructions for automatically starting the license server on Windows. • On UNIX/Linux, create a startup script using the template startup script provided and modify it for your installation. Obtain the template from the IBM Rational Download and Licensing Center at: https://www14.software.ibm.com/webapp/iwm/web/reg/signup.do?source=rational	

Networking requirements for IPv6 support

The Build Forge system can run on Internet Protocol version 6 (IPv6) and mixed IPv6-IPv4 networks with some restrictions.

IPv6 support requires that your computers and network are configured correctly to support IPv6. Network configuration problems will prevent host names and addresses that are specified from within the Build Forge system from resolving correctly.

You must manually configure Build Forge for IPv6. To do this, see Modifying httpd.conf.

Use the correct address format when you enter IP addresses in Build Forge. In Build Forge, administrators or users enter host names or IP addresses in only a few places:

- During installation, administrators specify a host name or IP address that the Management Console uses to communicate with the license server (Standard and Enterprise Editions) and the database.
- In the agent configuration (bfagent.conf file), an optional setting restricts to a particular address or range of addresses connecting to the agent.
- Users enter a URL in a browser in order to view the Management Console user interface. The URL consists of the host name or IP address of the server where the Management Console is running. For example, to access the Management Console installed on a server named BFmachine that has IPv4 and IPv6 addresses configured, a user can enter any of the following addresses in the Web browser:
  - `http://BFmachine/`
  - `http://localhost/`
  - `http://127.0.0.1/` (IPv4 loopback address)

    - `http://::127.0.0.1/`, `http://0:0:0:0:0:0:127.0.0.1/`(IPv6 abbreviations of an IPv4 loopback address), or simply `http://::1/` (IPv6 compressed notation for the loopback address)

The IPv4 and IPv6 addresses differ in format and length.

- **IPv4 format:** The length is 32 bits. The address is specified as four decimal-separated decimal values, for example, 255.255.255.255
- **IPv6 format:** The length is 128 bits. The address is specified as eight colon-separated hexadecimal values, for example FE80:0000:0000:0000:0202:B3FF:FE1E:8329. There are a number of conventions for using the higher order fields. There are also rules for abbreviation. Build Forge does not perform any interpretation of IP addresses, they are passed directly to the network. Therefore any legal and valid abbreviation should work. Please see other references for more information on IPv6 address conventions.

Components that do not support IPv6

Components that do not support IPv6 must be installed on a host machine with an IPv4 address. Install these components on a computer that has an IPv4 address:

- Rational License Server 7.0, the license server for Build Forge Standard and Enterprise Editions that is included in product distribution.

  To support IPv6 addresses, you must install Rational License Server 7.1. Obtain it from IBM Rational Support or the IBM Rational Download and Licensing Center. You must also configure the version of the FLEXlm client software that supports IPv6. See Setting up the FLEXlm client.
- DB2 database: the PHP database drivers do not yet support IPv6.
- MySQL database: the PHP database drivers do not yet support IPv6.
- Oracle database other than 11g–the PHP database drivers do not yet support IPv6.

Integration requirements for other products

This section identifies product and version requirements for source configuration management (SCM) adaptors, integrated development environment (IDE) plug-ins, and other products that you can integrate with Build Forge core components

Alternate configuration of required components

Build Forge packages the following prerequisite products: Apache Tomcat server, Apache HTTP Server, and required PHP modules. If you already have these components or IBM equivalent products installed in your environment, you can configure them for Build Forge use.

The following products and technologies are supported:

- Apache Tomcat server 5.5.9
- Apache HTTP Server 2.2.4

- PHP 5.2.4
- Websphere Application Server 6.1
- IBM HTTP Server 6.1

Prerequisite: due to restrictions in the license server, the Build Forge console and IHS or WAS must be running on the same operating system and hardware platform.

Java API requirements

Use the Java client to write Java programs that access the Management Console.

JDK 1.5 is required for use with the Java API. To use the Java API, you must:

- Get the client API package from your Management Console machine.
- Add the .jar pathname to your CLASSPATH.

Note

A Build Forge user must be defined on the Management Console for programs to use to authenticate.

You can use the Java API to create Java programs that run on a client machine and access data on the Management Console. The Java API consists of a .jar file containing classes that define Management Console objects methods that provide operations on those objects.

Documentation is provided in JavaDocs.

ClearQuest integration

A Build Forge/Clear Quest integration can be a powerful tool to help an organization keep accurate records of builds in a ClearQuest format.

The following is a link to a white paper on the IBM Rational web site detailing this integration:

http://www-01.ibm.com/support/docview.wss?uid=swg27015041

The document includes information about required packages ClearQuest will apply to the database being used for the integration (specifically BuildTracking and DeploymentTracking).

SCM adaptor requirements

With adaptors, Build Forge can be integrated into source configuration management systems. Adaptors are provided for the following systems at the designated version(s):

- ClearQuest: version 7.0 or later
- ClearCase: version 6.0 or later

Note

If you integrate with *both* products, consult their documentation to determine what versions can interoperate. As a rule of thumb, both must be at the same version level.

IDE plug-in requirements

IDE users can operate the Management Console from their IDE user interface. Integrations are provided for the following applications and versions:

- Eclipse: version 3.02 or later running on Java 5. A plug-in is provided that users can install in their client.
- Rational Application Developer: version 7.0 or later. A plug-in is provided that users can install in their client.
- Rational Team Concert: version 1.0.1 or later. Integration requires installation of a server extension. After that installation, users can install a provided client plug-in.

Tivoli IT Asset Manager requirements

Build Forge is shipped ready for use with Tivoli IT Asset Manager products. The following file is installed in the installation root directory and must not be deleted, renamed, or edited:
`IRBFSVR0701.SYS2`

National language support requirements

The Build Forge product provides localized support for French, German, Italian, Brazilian Portuguese, Spanish, Japanese, Korean, Simplified Chinese, and Traditional Chinese.

This section provides information about language support in Build Forge.

Language settings for the Management Console and Agent

This topic describes how Build Forge configures language settings for the Management Console, agent, and engine components.

Management Console language settings

The language setting for the current user determines the language used to display interface controls in the Management Console.

Set the Management Console language setting for a user account as follows:

Root user	The installation program creates a root user account that the administrator uses to log in to the console the first time. On initial log in, the language setting for the root user is based on the operating system language for the Build Forge engine host.

Language default for other users	For administrator-created user accounts, the default console language is initially set using the language preference of the Web browser that the user logs in on. (This is a configuration setting of the Web browser, not Build Forge.) If the administrator does not change the language preference for the user, it is inherited from the Web browser. If the administrator wants to change the language for the user , complete this procedure. In the UI, click **Administration > Users > Language,** and then select a language. LDAP-created user accounts always use the language preference that is configured for the Web browser.
Changing the default user language	After logging in to the console, the administrator can select a different language setting for individual Build Forge users by selecting **Administration > Users > Language** from the left navigation panel of Build Forge window. If language preferences for users are configured in this way, the Management Console displays interface controls in the language selected for the user regardless of the language configured for the Web browser.

Agent and Build Forge engine language settings

The operating system language of the Build Forge engine host dictates:

- the language that the Build Forge engine uses
- the default language of the Build Forge agent

  The agent language setting controls the language of the system messages and job output.

Regardless of whether the language is set for the Management Console language is set in a Web browser or as a Build Forge preference, the agent logs data for system messages and job output in the operating system language of the Build Forge engine host.

To avoid mixed languages in the Management Console interface, ensure that the language selected for the Management Console matches the language used by the Build Forge engine host.

Language setup

To ensure that the language displayed in the Management Console matches the language that the agent uses to log data for system messages and job output, use the same language for the Web browser, the operating system on the Management console host, and the Build Forge user.

International data support for the database host

To display and manipulate international data, configure the host machine for the Management Console database as follows:

- Use the Unicode UTF-8 character set.

- Install the fonts that you intend to use to display data.

Changing user language preference in the Management Console

The default language for all Management Console users is initially set to the language of the Web browser.

To change the language setting for a Build Forge user, select **Administration > Users > Language**.

The Management Console displays user interface controls in the language selected for the user, but the agent continues to log data for system messages and job output in the operating system language of the Build Forge engine host.

Determining the language/charset for UNIX/Linux hosts

If the Management Console or agent is installed on a UNIX/Linux host, use the `locale` command as follows.

- To determine the language/charset currently being used by the operating system:

  ```
  $ locale
  ```

- To determine the language/charset combinations available to the operating system:

  ```
  $ locale -a
  ```

- To set locale on login, use an rc or profile script.

Determining the language code page for Windows hosts

If the Management Console or agent is installed on a Windows host, use the `chcp` command as follows:

- To determine the active code page number, type:

  ```
  > chcp
  ```

- To set the code page, enter the number for the language:

  > chcp *code_page*

The following table lists the Windows character encodings for the NLV1 languages that Build Forge supports:

Language	Code Page
English	1252
French	1252

Language	**Code Page**
Spanish Italian German Portuguese	
Japanese Shift-JIS	932
Korean	949
Simplified Chinese GBK	936
Traditional Chinese Big5	950

Planning the installation

This topic describes the planning needed for installing the Rational® Build Forge® product components. It describes the following installation configurations:

- Standalone installation
- Normal production installation
- Distributed installation
- Redundant installation
- IDE Integrated deployments

The system is highly configurable. Within a deployed system, you can configure project definitions to take advantage of resources to meet different needs.

Standalone installation

A standalone deployment includes the database, Management Console, and one agent deployed on a single host. This deployment is typically used for evaluation or development purposes.

Normal production installation

A normal installation puts the database, Management Console, and agents on different hosts. This deployment is the most typically used production environment.

Distributed installation

In a distributed installation, the database, Management Console, and multiple agents are deployed on multiple hosts.

Redundant installation

Build Forge can be installed on multiple hosts to increase availability. The first installation is performed normally. Additional installations on other hosts are made to point to the same database. If one of the running systems fails, only the jobs running at the point of failure are lost. Incoming job requests continue to be serviced by the other installations. See

"Configuring redundancy" on page 87.

Integration with IDE applications

The product can be integrated with several IDE applications. Product plug-ins allow users to perform key tasks, such as viewing projects and running jobs from their IDE.

Pre-installation setup

This section describes the pre-installation setup needed before running the installer.

Note

If you already have Build Forge version 7.1 or earlier installed, please see “Upgrade overview” on page 189.

- International data support (required). You must configure the web browser and your database to support the UTF-8 character set before installing Build Forge.
- Set up a database (required): set up a database for use with Build Forge or plan to install DB2 Express along with Build Forge.
    - If you intend to use an existing database, you typically need to create database objects (including a database user and password), install database clients on the host where Build Forge runs, and assemble information that you are prompted for during Build Forge installation.
    - If you intend to install Build Forge and a new installation of the included DB2 Express database, you need to assemble information that you are prompted for during Build Forge installation. This installation is available for Windows platforms only.
- Security (optional). If you intend to use SSL/HTTPS, you need to provide a certificate or plan to have Build Forge install a self-signed certificate.

International data setup

You must set up Build Forge components to support international data.

1. Configure web browsers.
    a. Set the language.
    b. Be sure that the correct fonts are installed.
2. Configure agent hosts to use the UTF-8 character set.

   On Windows, use the `chcp` command to check the code page:

   ```
   > chcp
   ```

   On UNIX or Linux, use the following command to check the locale and character set:

   ```
   locale
   ```

   You should see values that designate your language and character set. The following example is from a Solaris system where US English is the language and UTF-8 is the character set:

   ```
   LANG=en_US.UTF-8
   LC_CTYPE="en_US.UTF-8"
   ```

3. Configure databases to use the UTF-8 character set and fonts that support international data.

   - **DB2**:

     a. Set the codeset and territory. Example: CREATE DATABASE USING CODESET UTF-8 TERRITORY US (or select the appropriate codeset and territory in Control Center).

     b. Set DB2CODEPAGE=1208.

     If an existing database has data in it that you need to migrate to UTF-8, the following document can help: http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp?topic=/com.ibm.db2.udb.admin.doc/doc/t0024033.htm

   - **Microsoft SQL Server** (for use with Management Console on Windows only): *No support for international data.* Microsoft SQL Server uses UCS-2 for storing Unicode data and does not support UTF-8.

   - **MySQL**: Set the server character set and collation. If your installation of MySQL does not currently support international data, you can recompile it from source and use `./configure --with-charset=utf8 --with-collation=utf8_bin`. The Build Forge engine will not start if this support is not configured.

   - **Oracle**: Set the character set to **UTF8 - Unicode 3.0** on the instance when you install it. In the Database Configuration Assistant, the setting is made on the Initialization Parameters step on the Character Sets tab. If you use the command line, set the character set to **AL32UTF8** .

Database setup

This section contains setup instructions for each supported database.

The database can reside on the same host as the Management Console or on a different host.

Note

The Rational Build Forge 7.1 release requires that you install using an empty database. The installation will fail if the database is not empty.

Setup requires that you complete some or all of the following tasks, depending on your database:

- Create database objects for Build Forge to use (database, database user).
- Install the necessary client software for Build Forge to use.
- Determine what additional information you will need during installation. Typically this is the location of JDBC drivers that will be used by the Apache Tomcat application server.
- Configure support for the UTF-8 character set and an appropriate collation. Typically, support for international data is specified when you create your database; international data support

cannot be configured after database creation. You must install the fonts you intend to use to display data. **Build Forge requires the use of international data (UTF-8 character sets)**.

DB2 Express setup

Two types of installation are supported for DB2 Express:

- Install Build Forge along with the provided DB2 Express installation. Both are installed on the same host.
- Install Build Forge to access an existing installation of DB2 Express. The existing installation may be on the same host or a different host.

DB2 Express information needed when installing the provided database

During installation, if you select Yes for **Install DB2 Express** on the **Database Configuration** panel, the following fields are filled automatically:

- Database type: DB2
- Do you wish to populate this database at install time: Yes
- Database Host: 127.0.0.1 (localhost)
- Database Name: BUILD
- Database Schema Name: BUILD

You are asked for additional information in the **Database Configuration** panel:

Database Configuration

- **Database Port**: The default is 50000. Enter the port number you use if it is different.
- **Database Username**: User name for Build Forge to use when accessing the database. You can enter a local Windows account that already exists or one that you want Installation Manager to create.
- **Password**: Password for the database user name. Enter the password for the local Windows account that already exists or one that you want Installation Manager to create.

Important

If you choose to create the account, use a strong password that meets your local requirements for complexity. Build Forge does not validate password complexity. If you enter a password that does not meet requirements for complexity, the installation fails. The password policy is maintained in Windows.

- **Confirm Password**: Enter the password again to confirm it.

DB2 Express Installation Specifics

- **DB2 Express Installation Directory** - Where to install the files. The default is `C:\Program Files\IBM\SQLLIB\`.
- **Drive to store DB2 Data on:** - The drive on which to create the `DB2` directory. Choose a driver from the list of all drivers on your system. The default is `C:`.
- **Create User Account** : If you select this check box, Installation Manager creates the user and password in Windows. Clear the check box if you specify an existing account.

DB2 Express requirements when connecting to an existing database

If you are installing Build Forge to connect to an existing instance of DB2 Express, you need to do the following:

- Create the required system and database objects
- Collect information that is needed during Build Forge installation

DB2 Express system and database object requirements

The following objects must be defined before running the installer:

- User: a user account must be defined in Windows on the database host. The account should be used exclusively by Build Forge to access the database.
- Database objects: Create the following objects in your DB2 Express instance.
  - database: create a database and name it BUILD.
  - schema: create a database schema and name it BUILD. The user you created should have authorization for this schema.

DB2 Express requirements when connecting to an existing database

During installation, you select No for **Install DB2 Express** on the **Database Configuration** panel, because you are using an existing DB2 Express instance. The following fields are presented.

You are asked for additional information in the **Database Configuration** panel:

Database Configuration

- **Database Type**: Choose **DB2** .
- **Do you wish to populate this database at install time?**: Choose Yes if you want Build Forge to automatically populate at installation, and No to manually populate it after install.
- **Database Host**: Enter the host name where the DB2 Express installation is located.
- **Database Name**: Enter the name of the database you created for Build Forge to use.
- **Database Schema Name**: Enter the name of the schema you created for Build Forge to use.

- **Database Port**: The default is 50000. Enter the port number you use if it is different.
- **Database Username**: User name for Build Forge to use when accessing the database. Enter the Windows account you created for Build Forge to use.
- **Password**: Password for the Database Username. Enter the password for the user account.
- **Confirm Password**: Enter the password for the user account.

Test the Database Configuration

- Path to the DB2 Client Libraries: by default these libraries are in `C:/Program Files/IBM/SQLLIB/java` on the drive where you installed DB2 Express. Build Forge needs to use these files: `db2jcc.jar`, `db2jcc_license_cu.jar`.
- JDBC Driver location: by default these libraries are in `C:/Program Files/IBM/SQLLIB/java` on the drive where you installed DB2 Express. The JDBC drivers are used by Apache Tomcat to access the database.

Note

When No is selected for *Do you wish to populate this database at install time?*, the test only checks for the correct JDBC driver location.

DB2 setup

Use this procedure to set up support for DB2.

Database objects for DB2

1. *In your operating system*, create a user. The Management Console will use this name to log in to the database. Example: user name **build**, password **build**.

   The database name is case-sensitive.

   #### Note

   DB2 does not support creating a user from a SQL script.

   Perform the remaining steps in DB2:

2. Create a user table space with a 16K page size.
3. Create an empty database named **build** with the necessary table spaces.

   Database names are case-sensitive.

4. Grant user access to BFUSE_TEMP table space.

Note

Build Forge accesses the database using the schema for the user.

Example 1. Sample DB2 SQL command script

Use the following commands at a DB2 Command Line Processor to create the table space and database.

```
// Create database
db2 "CREATE DATABASE BFT ALIAS BUILD USING CODESET UTF-8 TERRITORY US AUTOCONFIGURE USING
 MEM_PERCENT 40 APPLY DB ONLY"
db2 "CONNECT TO BUILD"

db2 "CREATE BUFFERPOOL 'BUFFP1' IMMEDIATE SIZE 1000 PAGESIZE 16384 NOT EXTENDED STORAGE"
db2 "CONNECT RESET"
db2 "CONNECT TO BUILD"
db2 "CREATE SCHEMA schema name"

// Create table spaces
db2 "CREATE SYSTEM TEMPORARY TABLESPACE TEMPSPACE2 PAGESIZE 16384 MANAGED BY SYSTEM
  USING ('path to database/SQL003.0')
  EXTENTSIZE 64
  PREFETCHSIZE 64
  BUFFERPOOL BUFFP1"

db2 "CREATE USER TEMPORARY TABLESPACE BFUSE_TEMP PAGESIZE 16384 MANAGED BY SYSTEM
  USING ('path to database/SQL004.0')
  EXTENTSIZE 64
  PREFETCHSIZE 64
  BUFFERPOOL BUFFP1"

db2 "CREATE REGULAR TABLESPACE USERSPACE2 PAGESIZE 16384 MANAGED BY SYSTEM
  USING ('path to database/SQL005.0')
  EXTENTSIZE 64
  PREFETCHSIZE 64
  BUFFERPOOL BUFFP1"

// User must be granted use of table spaces
db2 "GRANT USE OF TABLESPACE BFUSE_TEMP TO USER BUILD WITH GRANT OPTION"
db2 "GRANT USE OF TABLESPACE USERSPACE2 TO USER BUILD WITH GRANT OPTION"
db2 "commit work"
db2 "CONNECT RESET"
db2 "terminate"
```

Tuning parameters recommended for DB2

Setting DB2 tuning parameters can improve the performance and scalability of Build Forge systems using an existing DB2 database.

Note

If you change these parameters *after* Build Forge is installed and running, stop Build Forge before making the changes. Restart Build Forge after you restart DB2.

1. Set the tuning parameters.

   Run the following DB2 commands:

   ```
   db2set DB2_EVALUNCOMMITTED=ON
   db2set DB2_SKIPDELETED=ON
   db2set DB2_SKIPINSERTED=ON
   ```

See DB2 documentation for more information on the effects of these settings.

2. Restart DB2.

   This step is required in order to put the parameters into effect. Be sure there are no sessions running on the database first.

   ```
   db2stop force
   db2start
   ```

DB2 client drivers

DB2 client drivers must be installed on the host before you install Build Forge. On UNIX or Linux, use the 32–bit drivers.

Install DB2 Connect Client to provide the drivers. .

Important

Reboot after installing the DB2 client. The Build Forge installation fails if you do not.

DB2 information needed during installation

During installation you are asked for the following information in the **Database Configuration** panel:

Database Configuration

- **Database Host**: the host where DB2 is installed.
- **Database Port**: Build Forge puts the default port of 50000 in this field for DB2. Be prepared to enter the port number if you use a different port.
- **Database Name**: name of the database for Build Forge to use. You created this database in a previous setup step.
- **Database Schema Name**: name of the schema for Build Forge to use.
- **Database Username**: user name for Build Forge to use when accessing the database. You created this user in a previous setup step.
- **Password**: password for the database user name.

Test the Database Configuration

- **Path to the DB2 client libraries** - the directory where the DB2 client libraries are located.

Important

When installing Build Forge on UNIX or Linux, this directory must be the one containing the 32–bit client driver libraries.

- **JDBC driver location** - the directory where the JDBC driver is located. This driver is used by Apache Tomcat to access the database. Typical location:
  - Windows: `<db2install>/IBM/SQLLIB/java`
  - UNIX or Linux: consult the documentation for your system.

The following information is displayed:

- **Required driver JAR files** - Displays the required driver JAR files. For DB2 these are db2jcc.jar and db2jcc_license_cu.jar.
- **Required JDBC driver class** - Displays the required JDBC driver class. For DB2 this is com.ibm.db2.jcc.DB2Driver.

Microsoft SQL Server setup

Use these procedures to install and configure access to a Microsoft SQL Server database from a Windows®-based Management Console.

Note

MS SQL Server 2005 is supported only on Windows. Microsoft SQL Server is not supported in Build Forge Version 7.1. Microsoft SQL Server 2005 is supported in version 7.1.1.

Install and configure the following items using the instructions in the following sections.

Database objects for Microsoft SQL Server

In these steps you create a user to be the database owner and the database for Build Forge to use.

1. **Create a user to serve as the database owner.**

   Build Forge uses this username to log on to the database. Use **build** unless you must use a different name. The user must have full permissions.

   a. Open SQL Server Management Studio.

   b. Open the database server in the Object Explorer (left pane).

   c. Right-click the **Security** folder and choose **New → Login** .

   d. In the Login - New dialog, specify the login name and choose options as follows. Important: uncheck User must change password at next login.

   - Choose **SQL Server authentication** and provide a password.
   - Uncheck **Enforce password expiration**
   - Uncheck **User must change password at next login**

2. **Create the database.**

   You must use mixed-mode authentication.

   a. Open SQL Server Management Studio.

   b. Open the database server in the Object Explorer (left pane).

   c. Right-click the **Databases** folder and chose **New Database** .

   d. In the New Database dialog, specify parameters for the database:

      - Specify a Database name. Use **build** unless you must use another name. The name is case-sensitive. The name of the data and log files are updated automatically in the Database files box.
      - Specify the Database owner.
        - Click the **[..]** control to the right of the field.
        - In the Select Database Owner dialog, click **Browse** .
        - Check the name of the user you created, then click **OK** .
        - Click **OK** in the Select Database Owner dialog.
      - Specify the Database files parameters. In the Database files table, do the following:
        - For both files: set the Initial Size to 500 (in MB)
        - For both files: set Autogrowth. In the Autogrowth column, click the **[..]** control to open the dialog. Check the **Enable Autogrowth** box, set growth to 500 MB, and select **Unrestricted Growth** , then click **OK** .

   Alternatively, you can use the following script to create the database.

```
CREATE DATABASE [build] ON  PRIMARY
( NAME = N'build', FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL.2\MSSQL\DATA\build.mdf' , SIZE = 2048KB , FILEGROWTH = 1024KB )
 LOG ON
( NAME = N'build_log', FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL.2\MSSQL\DATA\build_log.ldf' , SIZE = 1024KB , FILEGROWTH = 10%)
GO
EXEC dbo.sp_dbcmptlevel @dbname=N'build', @new_cmptlevel=90
GO
IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [build].[dbo].[sp_fulltext_database] @action = 'disable'
end
GO
ALTER DATABASE [build] SET ANSI_NULL_DEFAULT OFF
GO
ALTER DATABASE [build] SET ANSI_NULLS OFF
GO
ALTER DATABASE [build] SET ANSI_PADDING OFF
GO
```

```
ALTER DATABASE [build] SET ANSI_WARNINGS OFF
GO
ALTER DATABASE [build] SET ARITHABORT OFF
GO
ALTER DATABASE [build] SET AUTO_CLOSE OFF
GO
ALTER DATABASE [build] SET AUTO_CREATE_STATISTICS ON
GO
ALTER DATABASE [build] SET AUTO_SHRINK OFF
GO
ALTER DATABASE [build] SET AUTO_UPDATE_STATISTICS ON
GO
ALTER DATABASE [build] SET CURSOR_CLOSE_ON_COMMIT ON
GO
ALTER DATABASE [build] SET CURSOR_DEFAULT  GLOBAL
GO
ALTER DATABASE [build] SET CONCAT_NULL_YIELDS_NULL OFF
GO
ALTER DATABASE [build] SET NUMERIC_ROUNDABORT OFF
GO
ALTER DATABASE [build] SET QUOTED_IDENTIFIER OFF
GO
ALTER DATABASE [build] SET READ_COMMITTED_SNAPSHOT ON
GO
ALTER DATABASE [build] SET RECURSIVE_TRIGGERS OFF
GO
ALTER DATABASE [build] SET AUTO_UPDATE_STATISTICS_ASYNC OFF
GO
ALTER DATABASE [build] SET DATE_CORRELATION_OPTIMIZATION OFF
GO
ALTER DATABASE [build] SET PARAMETERIZATION SIMPLE
GO
ALTER DATABASE [build] SET  READ_WRITE
GO
ALTER DATABASE [build] SET RECOVERY FULL
GO
ALTER DATABASE [build] SET  MULTI_USER
GO
ALTER DATABASE [build] SET PAGE_VERIFY CHECKSUM
GO
USE [build]
GO
IF NOT EXISTS (SELECT name FROM sys.filegroups WHERE is_default=1 AND name = N'PRIMARY')
 ALTER DATABASE [build] MODIFY FILEGROUP [PRIMARY] DEFAULT
GO
```

Note

The READ_COMMITTED_SNAPSHOT attribute must be set to ON. It is set to ON in the script. To test for it, run the following SQL statement.

```
SELECT is_read_committed_snapshot_on FROM sys.databases WHERE name=<build>
```

Substitute your database name if you did not use build. The statement must return "1".

3. **Set the default database for the user.**

   a. Open SQL Server Management Studio.

b. Open the database server in the Object Explorer (left pane).

c. In Object Explorer, open **Security** → **Logins** .

d. Right-click the user you created and choose **Properties**

e. On the General page, select a Default database. Select the database you created.

f. Click **OK**

TCP/IP setup on Microsoft SQL Server

You must enable TCP/IP on Microsoft SQL Server in order to use it with Management Console. It is disabled by default on MS SQL Server 2005.

To enable TCP/IP on MS SQL Server 2005, do the following:

1. Open MSSQLServer Configuration Manager.
2. Under **SQL Server 2005 Network Configuration** , click **Protocols for MSSQLSERVER** .
3. Right-click **TCP/IP** , then choose **Enable** from the menu.

Microsoft SQL Server client and JDBC driver installation

You need to install the Microsoft SQL Server client and JDBC driver on the Management Console host. The Management Console uses them to access the database.

1. Install SQL Native Client.

   Version 2005.90.4035.00 is required. It is included in Service Pack 3 for Microsoft SQL Server.

2. Install JDBC drivers.

   JDBC Library version 1.2 is required. Download it from Microsoft at the following location:

   `http://www.microsoft.com/downloads/details.aspx?FamilyId=C47053EB-3B64-4794-950D-81E1EC91C1BA&displaylang=en`

   After completing the Microsoft install process, the location of the JAR is as follows:

   `/sqljdbc_1.2/enu/sqljdbc.jar`

ODBC Data Source Setup for Microsoft SQL Server

In these steps you create an ODBC data source that is used by a Windows-based Management Console to access the database you created in Microsoft SQL Server.

1. From the Windows start menu, select **Settings** → **Control Panel** → **Administrative Tools** → **Data Sources** .

   The **ODBC Data Source Administrator** dialog appears.

2. On the **System DSN** tab, click **Add** .

A list of drivers appears.

3. Select **MS SQL Server** from the list of drivers, then click **Finish** .

   In the **ODBC Setup** dialog box that appears, enter the following information:

   - **Data Source Name**: the name for this data source, *must be the same as Database Name and must not be the same as the schema name associated with the database.*
   - **Description**: description for this data source.
   - **Server Name**: host name of the host where the MS SQL Server database is installed.
   - **Database Name**: database name you created above.

   Click **OK** to close **ODBC Setup** , then **OK** to close **ODBC Data Source Administrator** . The data source is created.

Make a note of the following information. It is requested by the installation program when you install the Management Console.

- **Data source name**: as you assigned when creating the ODBC data source
- **Data source type**: SQL Server
- **User name**: user name you created for the database (for example, **build**)
- **Password**: password for the user name (for example, **build**)

Important

Use the same value for the Database Name and Data Source Name. A limitation in JDBC drivers requires this constraint. If they are not the same, the Quick Report reporting feature and the services-layer APIs for Java and Perl do not work.

Microsoft SQL Server 2005 information needed during installation

During installation you are asked for the following information in the **Database Configuration** panel:

Database Configuration

- **Database Host**: the host where SQL Server is installed.
- **Database Port**: Build Forge puts the default port of 1521 in this field for SQL Server. Be prepared to enter the port number if you use a different port.
- **Database Name**: name of the database for Build Forge to use. You created this database in a previous setup step.
- **ODBC Data Source Name**: the name of the ODBC data source.

- **Database Username**: user name for Build Forge to use when accessing the database. You created this user in a previous setup step.
- **Password**: password for the database user name.

Test the Database Configuration

- **Path to the SQL Server Client libraries** - the directory where the SQL Server client libraries are located.

  #### Important

  Microsoft SQL Server is supported on Windows platforms only.

- **JDBC driver location** - the directory where the JDBC driver is located. This driver is used by Apache Tomcat to access the database.

The following information is displayed:

- **Required driver JAR files** - Displays the required driver JAR files. For SQL Server this is /sqljdbc_1.2/enu/sqljdbc.jar.
- **Required JDBC driver class** - Displays the required JDBC driver class. For SQL Server this is com.mysql.jdbc.Driver.

MySQL setup

Use this procedure to install and configure support for MySQL.

Install and configure the following items. Use the instructions in the following sections.

Red Hat Linux 4 Requirements for MySQL

MySQL on Red Hat Linux 4 requires additional setup before configuration for Rational Build Forge.

- Install compatibility shared libraries.

  You need to install the `MySQL-shared-compat` package or RPM (but not both).

- Rename startup script.

  You need to rename the startup script to cause it to cause MySQL to be loaded earlier in the startup process than it does by default. The following example should move it up enough.

  ```
  mv /etc/rc3.d/S99mysql /etc/rc3.d/S50mysql
  ```

Database Objects for MySQL

1. Create an empty database named **build**.
2. Create a user associated with it (user name **build**, password **build**).

You could use the following commands to create the database **build** and create a user **build@localhost** with the password (“identified by”) **build**:

```
mysql -u root
mysql> create database build;
mysql> grant all on build.* to build@localhost
       ->     identified by "build";
```

MySQL client drivers

The MySQL native client drivers must be installed on the host before you install Build Forge. On UNIX or Linux, use the 32–bit drivers.

MySQL Configuration

- Increase maximum database connections to 200.

  Edit the `[mysqld]` section in `<`*`mysql-installdir`*`>/my.ini` (Windows) or `/etc/my.cnf` (UNIX/Linux) as follows:

  ```
  max_connections=200
  ```

  The value should at least equal the total of your **Max Console Procs** and **Run Queue Size** system settings (in the Management Console's **Administration** → **System** page).

MySQL information needed during installation

During installation you are asked for the following information in the **Database Configuration** panel:

Database Configuration

- **Database Host**: the host where MySQL is installed.
- **Database Port**: Build Forge puts the default port of 3306 in this field for MySQL. Be prepared to enter the port number if you use a different port.
- **Database Name**: name of the database for Build Forge to use. You created this database in a previous setup step.
- **Database Username**: user name for Build Forge to use when accessing the database. You created this user in a previous setup step.
- **Password**: password for the database user name.

Test the Database Configuration

- **Path to the MySQL Client libraries** - the directory where the MySQL client libraries are located.

Important

When installing Build Forge on UNIX or Linux, this directory must be the one containing the 32–bit client driver libraries.

- **JDBC driver location** - the directory where the JDBC driver is located. This driver is used by Apache Tomcat to access the database.

The following information is displayed:

- **Required driver JAR files** - Displays the required driver JAR files. For MySQL this is mysql-connector-java-5.*-bin.jar.
- **Required JDBC driver class** - Displays the required JDBC driver class. For MySQL this is com.mysql.jdbc.Driver.

Oracle setup

Use this procedure to set up support for an Oracle database.

Install or configure the following items. Use instructions in the following sections.

Database Objects for Oracle

Create a local user on the Oracle host: user name **build** and password **build**.

- Add appropriate grants, including CREATE SESSION and CREATE TABLE.
- Add an appropriate QUOTA size in the DEFAULT TABLESPACE, to provide enough space for the system to store data.

```
create user build
    identified by password
    default tablespace users
    quota unlimited on users;

grant create session, create table
    to build;
```

Important

During installation, the same value is used for the database name and the Oracle SID. A limitation in JDBC drivers requires this constraint. If they are not the same, the Quick Report reporting feature and the services-layer APIs for Java and Perl do not work.

Tuning parameters recommended for Oracle

Some Oracle parameters must be changed from their default in order for Build Forge to run correctly.

Note

If you change these parameters *after* Build Forge is installed and running, stop Build Forge before making the changes. Restart Build Forge after you restart the database server.

1. Set the tuning parameters.

   Run the following DB2 commands:

   ```
   ALTER SYSTEM SET open_cursors=1000 SCOPE=BOTH
   ALTER SYSTEM SET processes=500 SCOPE=BOTH
   ```

   See Oracle documentation for more information on the effects of these settings.

2. Restart the database server.

   This step is required in order to put the parameters into effect. Be sure there are no sessions running on the database first.

Oracle Client Configuration

To install and configure the client for Oracle:

1. Install Oracle Instant Client on the Management Console host.

   You must install the 32–bit version, regardless of whether you are running on a 32–bit or 64–bit platform. Use only version 10.2 of **Instant Client Package - Basic** for your operating system. Download it from Oracle at http://www.oracle.com/technology/software/tech/oci/instantclient/index.html.

2. Set up the environment on the Management Console host.

   Several environment variables must be set.

   - LD_LIBRARY_PATH: set to include the client installation directory.

     ### Note

     You can specify this value on the Database Configuration page from Installation Manager.

   - NLS_LANG: set to an appropriate value for international language support. It must include AL32UTF8. Example: AMERICAN_AMERICA.AL32UTF8.

     NLS_LANG must be set explicitly as described. The default charset that is set during the client installation is not correct for use with Build Forge.

   - ORACLE_HOME: set to the path of your Oracle client installation directory.

     ### Note

     You can specify this value on the Database Configuration page from Installation Manager.

- ORA_NLS10: set to the path where character-set data is located *on the server*.
- PATH: set to include the client installation directly.
- TNS_ADMIN: set to the path where the tnsnames.ora file is located *on the server* Ensure that full access permissions have been set for the tnsnames.ora file.

Note

You can specify this value on the Database Configuration page from Installation Manager.

To check the current language setting on the Oracle server, log in to Oracle and execute the following command:

```
SQL> host echo $NLS_LANG
```

Important

During installation, tnsnames.ora is set up to use the same value for the database name and the Oracle SID. A limitation in JDBC drivers requires this constraint. If they are not the same, the Quick Report reporting feature and the services-layer APIs for Java and Perl do not work.

Oracle Client Example Configuration on Windows

Example environment:

- Instant Client - Basic at `C:\instantclient_10_2`, to use American English.
- Oracle 10.2 on a Windows system at `C:\oracle\product\10.2.0\db_1`, installed to support international data

Variable settings on the system where the client and Build Forge are installed:

- LD_LIBRARY_PATH includes `C:\instantclient_10_2\`
- NLS_LANG=AMERICAN_AMERICA.AL32UTF8
- ORACLE_HOME=`C:\instantclient_10_2\`
- ORA_NLS10=`C:\oracle\ocommon\nls\admin\data`
- PATH includes `C:\instantclient_10_2\`
- TNS_ADMIN=`C:\oracle\product\10.2\db_1\network\admin`

Oracle Client Example Configuration on UNIX or Linux

Example environment:

- Instant Client - Basic at `/usr/local/instantclient_10_2`, to use American English.

- Oracle 10.2 on a UNIX system at `/usr/local/oracle/product/10.2.0/db_1`, installed to support international data

Variable settings on the system where the client and Build Forge are installed:

- LD_LIBRARY_PATH includes `/usr/local/instantclient_10_2`
- NLS_LANG=AMERICAN_AMERICA.AL32UTF8
- ORACLE_HOME=`/usr/local/instantclient_10_2`
- ORA_NLS10=`/usr/local/oracle/ocommon/nls/admin/data`
- PATH includes `/usr/local/instantclient_10_2`
- TNS_ADMIN=`/usr/local/oracle/product/10.2/db_1/network/admin`

Oracle information needed during installation

During installation you are asked for the following information in the **Database Configuration** panel:

Database Configuration

- **Database Host**: the host where Oracle is installed.
- **Database Port**: Build Forge puts the default port of 1521 in this field for Oracle. Be prepared to enter the port number if you use a different port.
- **Database Name**: name of the database for Build Forge to use. You created this database in a previous setup step.
- **Database Username**: user name for Build Forge to use when accessing the database. You created this user in a previous setup step.
- **Password**: password for the database user name.

Test the Database Configuration

- **Path to the Oracle Client libraries** - the directory where the Oracle client libraries are located.

Important

When installing Build Forge on UNIX or Linux, this directory must be the one containing the 32–bit client driver libraries.

- **ORACLE_HOME Environment Variable** - the directory where Oracle is installed.
- **Path to tnsnames.ora file (TNS_ADMIN)** - the directory containing the tnsnames.ora file. Ensure that full access permissions have been set for the tnsnames.ora file.

- **JDBC driver location** - the directory where the JDBC driver is located. This driver is used by Apache Tomcat to access the database.

The following information is displayed:

- **Required driver JAR files** - Displays the required driver JAR files. For Oracle this is ojdbc14.jar.
- **Required JDBC driver class** - Displays the required JDBC driver class. For Oracle this is oracle.jdbc.driver.OracleDriver.

Security setup

During installation you are asked questions about how you want to set up security.

- Certificates: you are given the option of installing a personal certificate or importing a certificate that you already have.
- Secure HTTP: you are asked whether you want to install the Apache server enabled for HTTPS/SSL. The certificate you choose is used. If you need to use a port number other than the default of 443, you need to enter the port number at that time.

Using the provided personal certificate

The provided certificate has the following attributes set:

- Subject DN: "CN=*hostname*", where *hostname* is the fully-qualified name of the host where you are performing the installation.
- Expiration period: 10 years (expressed as 3650 days). You can change this value. Expiration periods of one to two years are typical. Expiration periods longer than that increase vulnerability to security attacks that attempt to guess the key.

You are given the opportunity to modify the provided certificate. If you do modify the certificate, the following fields can be specified.

- Common Name (required)
- Locality
- State/Province
- Organization Name
- Country/Region Name (required)
- Street Address

The Common Name and Country/Region Name are concatenated into an X500Principal type Subject DN to be specified during certificate creation.

You are prompted for a password to use for the keystore that the installer creates. Record this password. It is required to complete setup of HTTPS/SSL.

Important

Changing the password later is possible but a fairly long process. Use a strong password that meets your local requirements for complexity.

Using your own certificate

If you have a certificate, you can import it for use by all components and connections in the system that use SSL:

- The certificate must be available on the host where you are installing Build Forge. Copy the certificate to a temporary directory. You are prompted for the fully qualified path during installation.
- You must specify the keystore password during installation.
- The certificate must be in PKCS12 keystore type. If your certificate is another type, you can use the OpenSSL `openssl` utility or the JDK `keytool` utility to convert the copy into PKCS12.
- You are prompted for a password to use for the keystore that the installer creates. Record this password. It is required to complete setup of HTTPS/SSL.

  ### Important

  Changing the password later is possible but a fairly long process. Use a strong password that meets your local requirements for complexity.

Installing the Management Console

This section describes how to install the Management Console on Windows, UNIX, and Linux platforms. Use the following procedure for any installation scenario supported by Installation Manager.

1. Perform pre-installation setup, as described in “Pre-installation setup” on page 20. Tasks include the following.
    - International data support setup (required)
    - Database setup (required). This step typically involves creating database objects, installing a database client, and collecting information that is needed during the installation steps in Installation Manager.
    - Security setup (optional, depending on your needs)
2. Start launchpad to run Installation Manager.

    Note

    Launchpad searches for Installation Manager on the host where you run it. If it does not exist, launchpad runs a packaged Installation Manager to install Installation Manager on your host. It then uses the installed Installation Manager to install Build Forge. If you prefer, you can install Installation Manager manually rather than from Launchpad.
3. Perform the installation steps in Installation Manager.
4. Perform required post-installation checks.

See “Alternative installation methods” on page 50 for installation procedures for the following scenarios:

- Using your own installations of the following required applications, instead of those that are provided and installed by Build Forge:
    - Apache HTTP server
    - PHP
    - Apache Tomcat
- Silent installation of product components using IBM Installation Manager
- Installation on VMWare

Starting Installation Manager with launchpad

Use launchpad to start Installation Manager and install Build Forge.

You can start launchpad in the following ways:

- Start launchpad from the product DVDs.
- Start launchpad from a downloaded file package.

Launchpad detects whether Installation Manager is installed on your host:

- If Installation Manager exists, launchpad starts it. In Installation Manager, you select the **Build Forge** package to install.
- If Installation Manager does not exist, launchpad uses a packaged Installation Manager. Within it, you select both the **Installation Manager** and **Build Forge** packages to install. The packaged Installation Manager installs Installation Manager, then starts it to install Build Forge.

Launchpad can also install an agent on the local host, if it is a Window shost. You cannot use the launchpad to install the agent on a non-Windows operating system. For agent installation instructions, see "Installing agents" on page 151.

Starting launchpad from the product DVDs

Use these instructions to start launchpad from the product DVDs.

Choose the instructions for your operating system.

- UNIX or Linux
    1. Insert the first DVD on the host where you are installing Build Forge.
    2. Mount the drive.
    3. In the root directory of the drive, run launchpad.sh.
    4. Select the package to install in Installation Manager.
        - If IBM Installation Manager is found on your host, launchpad starts it.

          On the first **Install Packages** page, select the **Build Forge** package, then click **Next** .
        - If IBM Installation Manager is not found, a packaged Installation Manager starts in order to install Installation Manager, then use it to install Build Forge.

          On the first **Install Packages** page, select the **Installation Manager** and **Build Forge** packages , then and click **Next** .
- Windows
    1. Insert the first DVD on the host where you are installing Build Forge.
    2. If autorun is enabled, launchpad starts automatically. If it is not enabled: in the root directory of the drive, run launchpad.exe

3. Select the package to install in Installation Manager.

   - If IBM Installation Manager is found on your host, launchpad starts it.

     On the first **Install Packages** page, select the **Build Forge** package, then click **Next** .

   - If IBM Installation Manager is not found, a packaged Installation Manager starts in order to install Installation Manager, then use it to install Build Forge.

     On the first **Install Packages** page, select the **Installation Manager** and **Build Forge** packages , then and click **Next** .

Starting launchpad from a downloaded package

Use these instructions to download an installation package and start launchpad.

1. From IBM Passport Advantage, download the installation package for your operating system to a temporary directory on the host where you are installing Build Forge.
2. Extract the installation image from the downloaded file to a local directory. The contents of the file are extracted to the local directory.
3. Start the launchpad program from the directory where you extracted the files, as follows:
   - Windows: run launchpad.exe.
   - UNIX/Linux: run launchpad.sh,
4. Select the package to install in Installation Manager.
   - If IBM Installation Manager is found on your host, launchpad starts it.

     On the first **Install Packages** page, select the **Build Forge** package, then click **Next** .

   - If IBM Installation Manager is not found, a packaged Installation Manager starts in order to install Installation Manager, then use it to install Build Forge.

     On the first **Install Packages** page, select the **Installation Manager** and **Build Forge** packages , then and click **Next** .

Installation steps in Installation Manager

Use IBM Installation Manager to install product components on most platforms.

You must have started Installation Manager and selected the **Build Forge** package to install in order to follow these instructions.

Follow the prompts to install the **Build Forge** package:

1. *Install Packages*

– Select the **Build Forge** and **Version** check boxes. If Installation Manager is not already installed, select the **Installation Manager** check box. Click **Next** after you make the selections.

2. *Install Packages: Location - Shared Resources*

   – Enter or choose the directory where you want shared resources installed, then click **Next** .

3. *Install Packages: Location - Package Group* – Choose the directory where you want the installation packages installed, then click **Next** .

   The default location is: C:\Program Files\IBM\Build Forge.

4. *Install Packages: Features*

   – By default all three core product modules are installed: Web Interface, Process Engine, and Services Layer. Click **Next** .

5. *Install Packages: License Server Configuration*

   – Select the type of license the console will use. Fill in the following information, then click **Next** .

   - **Rational License Server Based** : Enter the host name of the Rational License Server.
     - Enter a valid host name for the license server. If you plan to provide the host name later, do not leave this field blank. Enter a character or value in this field. Leaving this field blank might result in an incomplete and unusable product.

       After installation is complete, provided the correct host name. For instructions, see "Configuring a Rational license server for Build Forge" on page 9.
     - If a license server is displayed but greyed out, your FLEXlm license client has already registered a license server for the host.

       After installation is complete, provide the correct host name. For instructions, see "Changing the license server for the Management Console" on page 10.
   - **File Based** : browse to the location where you downloaded the license file.

6. *Database Configuration*

   Depending on the OS platform you are installing Build Forge on, and the database you want to install, you must specify certain information. Refer to database setup instructions in "Database setup" on page 21 for the following:

   - For DB2 Express, see"DB2 Express setup" on page 22.
   - For DB2, see "DB2 setup" on page 24.
   - For Microsoft SQL Server, see "Microsoft SQL Server setup" on page 27.
   - For MySQL, see "MySQL setup" on page 32.

- For Oracle, see "Oracle setup" on page 34.

  Note

  On UNIX and Linux platforms, you must install and use 32-bit database client drivers if you are using an Oracle, DB2, or MySQL database. On the Database Configuration page in Installation Manager, for your specific database type, make sure that you specify the 32-bit version of driver libraries in the **Path to the [DB2|Oracle|MySQL] client libraries** field on the Database Configuration page.

7. *Install Packages: Application and Web Server Configuration* – Fill in the requested information, then click **Next**.

   a. *Web Server/PHP Configuration*

      - **Supply your own webserver?** Select **Yes** if you want to supply your own web server. **No** is the default.
      - **Do you wish to use Secure HTTP?** Select **Yes** if you want to use Secure HTTP. **No** is the default.
      - **Which port should the web server use?** If you do not want the web server to use port 80, enter a different port number. 80 is the default.
      - **Memory Limit for PHP:** Enter a memory limit for PHP if you do not want to use the default. 256 MB is the default.

   b. *Application Server Configuration*

      - **Supply your own application server?** Select **Yes** if you want to configure Build Forge to use an application server that you have already installed. **No** is the default.

        Note

        You do this only if you are configuring Build Forge to use one or more components that you have already installed. Normally Build Forge installs these components during installation. See "Installing using your own components" on page 50.

   c. *Security Configuration*

      Enter and verify a password for the keystore that the installer creates for Build Forge in the following fields.

      Note

      If these fields are not visible, scroll down to find them.

      - **Keystore Password**
      - **Verify Password**

The password is required if you intend to configure Build Forge to use HTTPS/SSL.

Important

Changing the password later is possible but a fairly long process. Use a strong password that meets your local requirements for complexity.

You have the following options:

- Use the self-signed certificate that the installer creates as is. Do the following:
    - i. **Do you wish to modify default or upload a custom certificate?** Select No.
    - ii. **Do you have an existing secure certificate?** Select No.
- Use the self-signed certificate that the installer creates, but modify its fields. Do the following:
    - i. **Do you wish to modify default or upload a custom certificate?** Select Yes. Additional fields for the certificate are displayed. Fill them in.
    - ii. **Do you have an existing secure certificate?** Select No.
- Provide the location of your own certificate. The certificate must be on the host, it must be in pkcs12 format, and you must provide the existing password for the keystore it is in. Do the following:
    - i. **Do you wish to modify default or upload a custom certificate?** Select Yes.
    - ii. **Do you have an existing secure certificate?** Select Yes. Additional fields are displayed.
    - iii. **Please specify a signed certificate in a keystore of type pkcs12.** Enter the filename or use **Browse** to locate it.
    - iv. **Keystore Password** Enter the password for the keystore containing your certificate.
    - v. **Verify Password** Re-enter the password for the keystore containing your certificate.

Note

Browsers typically warn about accessing a secure site that has a self-signed certificate. Users are typically given the option to proceed, but may be required to confirm the exception.

8. *Services Configuration*

   Fill in the requested information, then click **Next** .

- **Will services layer run on this machine?** Click **No** if you do not want the services layer to run on this machine. **Yes** is the default.

  If you click No, you are prompted for the host name of the application server that runs the services layer application.

- **Listen on port**: 3966 is the default. One or both ports must be selected as the services port.

- **Listen on secure port**: 49150 is the default. One or both ports must be selected as the services port.

9. *Install Packages: Console Start Options* – fill in the requested information, then click **Next**.

   a. • **Create a shortcut on the desktop?** The default is create a shortcut on the desktop (for Windows).

      Note

      On Linux, a desktop shortcut is not created.

      - **Directory for file storage:** The Windows default is C:\Program Files\IBM\Build Forge\temp. The Linux default is /opt/buildforge/temp.

      - **Specify your own storage path:** Click **Browse** to locate and use a different file storage path than the default.

   b. *Startup Options*

      - **Do not start the Console:** The Console starts by default. Click the radio button if you do not want to the Console to start automatically.

      - **Start the Console in Service Mode:** The Console starts in service mode by default.

      - **Start the Console in Foreground Mode:** The Console does not start in foreground mode, by default. If you want the Console to start in foreground mode, click the radio button.

10. *Install Packages: Summary review* – Review the summary information on this page to confirm where the Build Forge components will be installed, then click **Install**.

    a. *Target Location*

       - **Package Group Name:** `buildforge.console` is the default package name

       - **Installation Directory:** The default installation directory is C:\Program Files\IBM\Build Forge.

       - **Shared Resources Directory:** The default shared resources directory is C:\Program Files\IBM\SDP70Shared.

    b. *Features*

- **Build Forge Features** You can review what features or modules will be installed. For example, the core product modules are: Web Interface, Process Engine, and Services Layer.

c. *Environment*

- English is the default environment.

d. *Repository Information*

- **Files will be retrieved from the following locations:** – Use this section to review and confirm the repository location.

11. If you elected to install DB2 Express:

a. *Install DB2 Express*.

The installer starts automatically and displays a progress bar.

b. *Reboot Now*

When you are asked when to reboot, reboot now. DB2 Express requires a reboot before you can access it through Build Forge.

12. *Access the Management Console*.

Start a browser. Go to the URL for the Management Console:

- General form: `http://<hostname>[:<portnumber>]`. The port number is optional if you used the HTTP default, port 80.
- Local: if you are running the browser on the same host where the Management Console is running, you can use `http://localhost`.

Note

If you have installed a database other than DB2 Express and you cannot log in, wait a minute or so and try again. On first startup the engine (bfengine) has to load the database schema.

Important

Do not stop bfengine immediately after an install. Doing so can corrupt the database schema. In that case you would need to drop all Build Forge tables from the database and reinstall Build Forge.

13. *Log in.*

Use user name **root**, password **root**. Change the root password immediately.

Post-Installation checklist

This section describes what you need to do after installing the Build Forge system.

- Check the PATH variable.
- Identify the proxy server for PHP to use, if Management Console must go through a proxy server to access the database.
- Set JVM memory

Check PATH variable

The PATH environment variable needs to include the path to database client or driver DLLs. Check the PATH manually for these databases:

- DB2 - directory containing `db2cli.dll` and `sqlar.dll`
- MySQL - directory containing `libmysql.dll`
- Oracle - directory containing `oci.dll`

For other databases (Microsoft SQL Server, Oracle, or Sybase), installing the client or setting up the ODBC connection takes care of this requirement.

Identify proxy server

Optional: this step is needed only if the Management Console needs to use a proxy server to access its database. You must configure PHP to use the proxy server.

- Edit the php.ini file. It is located in `<`*`bf-install`*`>/Apache/php`, for example `C:\Program Files\IBM\Build Forge\Apache\php`.

  Add the following entries:

  ```
  bf_proxyHost=<your_proxy_server_hostname>
  bf_proxyPath=<your_proxy_path>
  bf_symlinkPath=<symlink_to_proxy_path>
  ```

Set JVM memory (required for Quick Report)

Optional: this step is needed only if your edition includes Quick Report (Standard Edition, Enterprise Edition, Enterprise Plus Edition).

- Set the maximum memory for the JVM to 1 MB or more. Running reports requires a minimum 1 GB (1024 MB) heap size. If you get Out of Memory errors while running reports (possible on large reports), increase this setting. That may in turn require you to add memory to the host.

  ```
  JAVA_OPTS -Xmx1024M
  ```

Increasing the number of file handles for Linux

Important: For best results, before you use your Rational product, increase the number of file handles available for Rational Build Forge. A system administrator might need to make this change.

Exercise caution when you follow these steps to increase the file descriptors on Linux. Failure to follow the instructions might result in a computer that does not start correctly. For the best results, have your system administrator perform this procedure.

To increase the file descriptors:

1. Log in as root. If you do not have root access, you must obtain it before you continue.
2. Change to the etc directory.
3. Locate the initscript shell script. Open the file or create it with a Linux text editor.

   Important: Do not leave an empty initscript file on your computer. If you do so, your computer will not start the next time that you turn it on or restart it.
4. On the first line, set ulimit to a number significantly larger than 1024, the default on most Linux computers.

   ```
   ulimit -n 4096
   ```

   Caution: Setting ulimit too high can impact system-wide performance.
5. On the second line, type `eval exec "$4"`.
6. Save and close the shell script.

For more information on the ulimit command, refer to the man page for ulimit.

Alternative installation methods

This section describes alternative methods of installing the Management Console. Instructions are provided for the following scenarios:

- Using your own installations of the following applications, instead of those that are provided and installed by Build Forge:
    - Apache HTTP server and PHP
    - Apache Tomcat
- Silent installation of product components using IBM Installation Manager
- Installation on VMWare

Installing using your own components

Use this section to set up required technologies if you already have these components installed and want to use the components that you already have and not those provided by Rational Build Forge.

Build Forge automates the installation and configuration of the following required components and technologies:

- Apache HTTP server and PHP
- Apache Tomcat

You must configure your installation of one or more technologies to meet Build Forge requirements, then run Installation Manager to install Build Forge components. During installation you are given the option of using technology that you already have set up.

The following sections describe how to set up each technology for use with Build Forge, and then how to install Build Forge to use them.

Prerequisites

You need the following in order to perform an installation on UNIX or Linux:

- Internet access. If you do not have Internet access from the machine where you are installing Build Forge, you need to download files from a machine that does have access and transfer them to the Build Forge machine to complete the steps.
- C compiler that is valid and working on your platform (for example, the gcc compiler on Linux).
- make facility that is suggested by your compiler's manufacturer (for example, gnu-make for use with gcc).
- Privileges as `root`

- To use SSL, you must compile Open SSL.

International data support

Build Forge must be set up to support international data in the Management Console.

- **Web browsers**:
  - must have the language set
  - must have the fonts you use to display data installed
- **Agents**

  Build Forge recommends using the UTF-8 character set on agent servers.

  On UNIX/Linux, use the following command to check the locale and character set:

  ```
  locale
  ```

  You should see values that designate your language and character set. The following example is from a Solaris system where US English is the language and UTF-8 is the character set:

  ```
  LANG=en_US.UTF-8
  LC_CTYPE="en_US.UTF-8"
  ```

- **All databases**:

  Typically support for international data is specified when you create the database; international data support cannot be configured after database creation.

  The fonts you intend to use to display data must be installed on the database host machine.

  Build Forge requires the use of international data (UTF-8 character sets).

- **DB2**:

  1. Set the codeset and territory. Example: CREATE DATABASE USING CODESET UTF-8 TERRITORY US (or select the appropriate codeset and territory in Control Center).
  2. Set DB2CODEPAGE=1208.

  If an existing database has data in it that you need to migrate to UTF-8, the following document can help: http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp?topic=/com.ibm.db2.udb.admin.doc/doc/t0024033.htm

- **MySQL**: Set the server character set and collation. If your installation of MySQL does not currently support international data, you can recompile it from source and use `./configure --with-charset=utf8 -–with-collation=utf8_bin`. The Build Forge engine will not start if this support is not configured.

- **Oracle**: Set the character set to **UTF8 - Unicode 3.0** on the instance when you install it. In the Database Configuration Assistant, the setting is made on the Initialization Parameters step on the Character Sets tab.

Database installation and configuration

Use this section to install the database you intend to use with a Build Forge Management Console installation using your own components.

A database must be installed and configured with database objects before you install and configure other technologies and Build Forge. You need to do the following:

In general you must complete the following tasks:

- Identify the database system you intend to use. Verify that it is one that Build Forge supports and that the necessary network connectivity exists between the database host and the Build Forge host. If a proxy server is required to access the database, get the proxy server name and path.
- If you intend to use international data, verify that the database is configured to use a UTF-8 character set.
- Create database objects and permissions, in general as follows:
  - Database: It is named **build** in examples, but you can use a different name.
  - Database User: The Management Console uses this user name to access the database. The user is named **build** in examples, but you can use a different user name.
  - Database User Password
  - Permissions for the **build** user to create tablespaces in the **build** database. Owner permissions are required in order to create, modify, and delete data.

  Specific instructions are provided for each database type.

Apache HTTP Server installation and configuration

Installation Manager installs and configures Apache HTTP Server as the web server for Build Forge. Using the provided Apache HTTP Server is the quickest way to configure a web server for Build Forge.

As an alternative to the standard configuration, you can configure an existing Apache HTTP Server inplace of the one installed and configured by Build Forge. The instructions provided assume that you have experience setting up and configuring Apache HTTP Server on your operating system.

To use your existing Apache HTTP Server, modify your installation as follows:

1. Modify your Apache HTTP Server configuration file (httpd-vhosts.conf) to point to the Build Forge application.

2. Install PHP and configure the PHP modules required for the Apache HTTP Server; your Build Forge database; and password encryption, if you want to use this security feature.

3. Configure Apache for your database

Install Build Forge using Installation Manager

In Installation Manger, at the Application and Web Server Configuration page, select **Yes** at the **Supply your own web server** prompt.

Prerequisite software

- Apache HTTP Server 2.2.4
- PHP 5.2.4

Edit the Apache server configuration file

1. Locate the Apache http-vhosts.conf file in the extras directory of your server installation.

```
cd <apache-dir>/conf/extras/
vi httpd-vhosts.conf
```

2. Edit the Apache http-vhosts.conf file. To add information about Build Forge to httpd-vhosts.conf, add the following lines:

```
<VirtualHost *:80>
  ServerAdmin build@yourdomain.com
  DocumentRoot /usr/local/buildforge/webroot/public
  ServerName ausbuild01.yourdomain.com
  ServerAlias build.yourdomain.com mc.yourdomain.com
  ErrorLog logs/ausbuild.error_log
  CustomLog logs/ausbuild.access_log common
</VirtualHost>
```

3. Modify the DocumentRoot setting to point to the Build Forge web application. In the example, the Build Forge installation directory is /usr/local/buildforge.

4. Leave the port as 80 or change it to the port you run the Apache HTTP Server on locally.

```
<VirtualHost *:80>
```

Important

Do not use port 8080; it is the default port for Apache Tomcat.

5. Modify any other settings in http-vhosts.conf as required for your Apache HTTP server:

   - `ServerAdmin`: e-mail address of the Build Forge administrator
   - `DocumentRoot`: location of the entry page for the Build Forge application

- `ServerName`: server where the Build Forge application is installed
- `ServerAlias`: optional aliases for the Build Forge `ServerName` URL
- `ErrorLog`: Apache error log for the Build Forge application
- `CustomLog`: Apache error log for logging access to the Build Forge application

Install and configure PHP for the Apache HTTP Server

PHP is not installed with the Apache HTTP Server. You must install PHP 5.2.4 and configure it to point to the httpd-vhosts.conf for the Apache HTTP Server.

Install and configure PHP for your Build Forge database

During PHP installation, select and install the PHP extensions for the database type that you are using as the Build Forge database.

(Optional) Configure the PHP OpenSSL module to support password encryption

To support SSL, Build Forge uses the PHP OpenSSL module. This support is provided with PHP 5.2.4; no additional configuration is required.

To support password encryption, some additional configuration is required. PHP 5.2.4 is required to support this configuration. You must locate the patch files for the OpenSSL extension, install them in the OpenSSL directory and recompile PHP, as follows:

1. Locate the php_openssl.h and openssl.c patch files in the misc directory, located in the Build Forge installation directory, for example:

Windows	C:\Program Files\IBM\Build Forge\misc
UNIX/Linux	/usr/local/buildforge/misc

2. Copy the patch files to the openssl directory, located in the Build Forge installation directory.
3. Compile PHP using the --with-openssl=<*path_to_openssl*> configure option, where <*path_to_openssl*> is the Build Forge openssl directory.

Configure Apache for your database

You need to add specific information to httpd.conf, depending on your database.

Apache configuration for DB2

1. Add the following line to the beginning of the Apache startup script (normally /etc/init.d/httpd or /etc/init.d/apache2, depending on your distribution).

```
source /home/db2bf/sqllib/db2profile
```

2. Add the following lines to `httpd.conf`:

   ```
   PassEnv LD_LIBRARY_PATH
   PassEnv CLASSPATH
   PassEnv LIBPATH
   PassEnv VWSPATH
   ```

Apache configuration for MySQL

No additional configuration is required.

Apache configuration for Oracle

1. Add the following lines to httpd.conf:

   ```
   PassEnv LD_LIBRARY_PATH
   PassEnv NLS_LANG
   PassEnv ORACLE_HOME
   PassEnv ORA_NLS
   PassEnv ORA_NLS32
   PassEnv TNS_ADMIN
   ```

2. Add the following lines to the script that starts Apache at boot time (commonly /etc/init.d/httpd or /etc/init.d/apache2) and provides values for the following settings.

   ```
   export LD_LIBRARY_PATH=<value>
   export NLS_LANG=<value>
   export ORACLE_HOME=<value>
   export ORA_NLS=<value>
   export ORA_NLS32=<value>
   export TNS_ADMIN=<value>
   ```

Start IBM HTTP Server

Before you start the Build Forge engine and start the Management Console, start your Apache HTTP Server.

PHP Installation and Configuration

Use this procedure to set up PHP for use with the Management Console.

Requirements:

- Version: PHP must be 5.2.4 or later
- Database drivers: PHP modules for the Build Forge database installed

Follow the instructions in this section to configure PHP for the Apache HTTP Server or other web server. The instructions assume that you have already downloaded the required version of the PHP.

- Download PHP
- Install PHP

- Configure PHP
- Edit the Apache configuration file
- (optional) Identify proxy server to use to access the database (needed only if the Management Console host accesses the database through a proxy server)

Install PHP

This section describes how to compile and install PHP from source. If you have an existing installation of PHP and do not wish to recompile, you need only determine if the appropriate database drivers are installed. If you need to install a database driver, consult PHP documentation for the installation method to use. Install the database driver for the database to use with Build Forge, as follows:

- DB2: `ibm_db2` driver
- MySQL: `mysqli` driver
- Oracle Instant Client: `oci8` driver

Note

Currently, the full Oracle client is not compatible with PHP oci8. Use Oracle Instant Client only.

1. Configure PHP for installation in the working directory you just created.

```
$ ./configure --prefix=/usr/local/php-5.2.4 --with-<database>=shared \
--with-apxs2 --with-ldap=shared --enable-mbstring --enable-shmop \
--with-xml --with-zlib=shared
```

   Replace `--with-<database>` as follows:

   - DB2: `--with-ibm_db2[=dir]`. If `=dir` is not specified, the default value is used:`/home/db2inst1/sqllib`
   - MySQL: `--with-mysqli[=file]. The optional file parameter is a pathname to mysql_config.`
   - Microsoft SQL Server:
   - Oracle: **You must install a separate installation of Oracle Instant Client to use PHP oci8**. When using Oracle Instant Client to connect to the database, use `--with-oci8=instantclient,lib` where `lib` is the path to the Instant Client lib directory.

   Note the line-continuation character `\` in the code block. This step specifies where PHP will be installed and what options it will be installed with. It is installed in `/usr/local` by default. The example shows how to put it in `/usr/local/php-5.2.4`. This location is used in later examples.

2. Compile PHP.

```
$ make
```

This step compiles executables in your local directory.

3. Install PHP (do as `root`).

```
# make install
```

This step must be performed as a user who has write privileges for the directory where Apache is installed (`/usr/local/apache-2.2.4` in this example). It is normally done as `root`. Your local administrative setup may vary.

Configure PHP

1. Copy extension files to the extension directory.

The extension files for the database need to be copied from the repository up to the active extensions directory. The following example assumes that PHP is installed in `/usr/local/php-5.2.4`. Note that *`<datestamp>`* is a string of numbers.

```
$ cd /usr/local/php-5.2.4/lib/php/extensions/no-debug-non-zts-<datestamp>/
$ cp <db-extensions> ..
```

The *<db-extensions>* files correspond to your database for Build Forge, as follows:

- DB2: `ibm_db2.so`
- MySQL: `mysql.so` and `mysqli.so`
- Oracle: `oci8.so`

2. Edit the PHP configuration file `php.ini`.

The following example assumes that PHP is installed in `/usr/local/php-5.2.4`:

```
$ cd /usr/local/php-5.2.4/lib/
$ vi php.ini
```

Add the following entries:

```
extension_dir=/usr/local/php-5.2.4/lib/php/extensions
upload_tmp_dir=
                        <directory>

extension=<db-extension-so>
```

Use the *<db-extensions-so>* file name (or file names) for your database, as follows:

- DB2: `extension=ibm_db2.so`
- MySQL: two entries -

```
extension=mysql.so
extension=mysqli.so
```

- Oracle: `extension=oci8.so`

  Note

  the directory used for `upload_tmp_dir` must be writable by the user that the Apache web server runs as. Commonly this user is `nobody`, but your local administrative practice may vary.

Edit Apache Configuration File

1. Edit Apache configuration file. Add information about PHP in `httpd.conf`.

   ```
   cd <apache-dir>
   vi httpd.conf
   ```

   Add the following lines:

   ```
   LoadModule php5_module modules/libphp5.so
   AddHandler php5-script .php
   AddType text/html .php
   DirectoryIndex index.php
   ```

Identify Proxy Server

Optional: this step is needed only if the Management Console needs to use a proxy server to access its database.

- Edit the PHP configuration file `php.ini`.

  It is located in `<php-install>/lib`, for example `/usr/local/php-5.2.4`.

  Add the following entries:

  ```
  bf_proxyHost=<your_proxy_server_hostname>
  bf_proxyPath=<your_proxy_path>
  bf_symlinkPath=<symlink_to_proxy_path>
  ```

Apache Tomcat installation and configuration

Installation Manager installs and configures Apache Tomcat as the application server for Build Forge. Using the provided Apache Tomcat application server is the quickest way to configure an application server for Build Forge.

As an alternative to the standard configuration, you have the option to use an existing Apache Tomcat instead of the one provided by Build Forge. This section describes the prerequisite software, pre-installation setup, installation, and post-install requirements for this alternative. The instructions provided assume that you have experience setting up and configuring Apache Tomcat.

Software prerequisites

- Apache Tomcat Server:
  - 5.5.26 for Solaris platforms
  - 5.5.9 for all other platforms
- J2SE 5: IBM or Sun
- JDBC database drivers for the Build Forge database: Java Database Connectivity (JDBC) drivers are required for use with Apache Tomcat. Sun provides a JDBC vendors list at: http://developers.sun.com/product/jdbc/drivers.

Install the jar file for the JDBC driver

Download and unzip the JDBC driver for your database.

Important

The JDBC driver download may contain many files and subdirectories. Locate the jar file for the JDBC driver and copy the jar file only to `$CATALINA_HOME/common/lib`.

`$CATALINA_HOME` is the Tomcat installation root and must be set as an environment variable. See the installation documentation for your JDBC driver.

- DB2 - `http://www-306.ibm.com/software/data/db2/express/download.html` Click the download link next to **DB2 Driver for JDBC and SQLJ** ; IBM account registration is required. You also need to locate and install a licensing .jar, `db2jcc_license_cu.jar`.
- MySQL - `http://www.mysql.com/products/connector/j/` Click the link for **MySQL Connector/J 5.0 or 5.1** . Select the JDBC driver version that corresponds to your MySQL version.
- Oracle - `http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/index.html` Click the download link next to your version of Oracle; account registration is required.
- Microsoft SQL Server- `http://msdn.microsoft.com/en-us/data/aa937724.aspx` Click the **Download SQL Server JDBC Driver** link.

Configuring your Apache Tomcat server in Installation Manager

These instructions identify the information that you need to configure Apache Tomcat through Installation Manager.

1. Shut down Apache Tomcat.

   ### Important

   Before you start Installation Manager, Apache Tomcat must be stopped.

2. Start Installation Manager.

3. On the Start page, click **Install** .

4. Follow the instructions in the Installation Manager wizard to install the product.

5. On the Application Server Configuration page, click **Yes** to configure your own application server.

✓	**Field**	**Description**
	Redirection URL	Enter the host name and port number of your application server. You must specify rbf-services as the context path. For example: `http: | https://<app_server_host>:<app_server_port>/rbf-services.`
	Specify the directory where you want to install the BF Services Plug-ins.	Specify a directory local to the application server host. Installation Manager installs the Build Forge services layer application plug-in extensions in this directory. The user who is running the application server must have read, write, and execute permission to this directory.
	Specify the WAR deployment directory: $CATALINA_HOME/webapps.	Specify a directory local to the application server host. To avoid additional configuration, use $CATALINA_HOME/webapps. Installation Manager installs the Build Forge services layer application WAR (rbf-services.war) file in this directory. The user who is running the application server must have read, write, and execute permission to this directory. If you do not specify $CATALINA_HOME/webapps, see Updating the buildforge.conf file and complete the relevant steps to manually add buildforge.conf to rbf-services.war.
	Specify an installed jar executable	Specify the directory on the application server host where the Java archive tool used by the application server resides. This directory must contain the jar executable: jar on UNIX/Linux and jar.exe on Windows.

6. Complete installation through Installation Manager.

Post-installation configuration of Apache Tomcat

After you have completed installation through Installation Manager, complete the following post-installation steps.

1. If you did not specify the default location for the rbf-services.war file, you must manually add the buildforge.conf file to the rbf-services.war file.

   The default location for the rbf-services.war is: $CATALINA_HOME/webapps. See Updating the buildforge.conf file and complete the relevant steps.

2. Increase JVM heap size for the Apache Tomcat server.

   Set the JVM maximum heap size option -Xmx to 1024 M.

Use the CATALINA_OPTS or the JAVA_OPTS environment variable in catalina.bat or catalina.sh to set this JVM option.

3. Before you start Build Forge, start Apache Tomcat: `$CATALINA_HOME/bin/catalina.sh start`.

Manually installing Installation Manager

IBM Installation Manager is installed automatically or updated if you use launchpad program to start the product installation. See “Starting Installation Manager with launchpad” on page 40.

Users who are experienced with Installation Manager or who want to set up a silent installation can install Installation Manager manually. Perform these steps:

1. Obtain the product installation package by downloading it from Passport Advantage or by using the product DVDs.
2. Locate the Installation Manager files for your platform:
   - InstallerImage_linux
   - InstallerImage_solaris
   - InstallerImage_win32
3. Enter one of the following commands to start the installation program.
   - To run the installation as an Admin user, run the following command:

     ```
     install
     ```

   - To run the installation as a non-Admin user, run the following command:

     ```
     userinst
     ```

4. Follow the installation instructions to install Installation Manager.

After installation, you can use Installation Manager or the Installation Manager installer to silently install packages.

Starting Installation Manager

Start Installation Manager on Windows or on UNIX/Linux.

If you use the launchpad program to start the product install, Installation Manager starts automatically. If you have already installed Installation Manager, you can start it in one of these ways:

- Windows: Click **Start > All Programs** → **IBM Installation Manager** → **IBM Installation Manager** .
- Change to the <*IM-installdir*> and run ./IBMIM.

Specifying the repository URL

IBM Installation Manager uses an embedded URL in each product package to connect to a repository server through the Internet and search for the latest product installation packages.

In Installation Manager, you can set repository locations on the Repositories page in the Preference window. Your organization might require you to redirect the repository to use intranet sites.

Note

Before starting the installation process, be sure to obtain the installation package repository URL from your administrator or IBM.

To specify a repository, complete the following steps:

1. Start IBM Installation Manager.
2. On the Start page, click **File > Preferences** .
3. In the Preferences window, click **Repositories** . The Repositories page opens, showing available repositories, their locations, and whether they are connected.
4. On the Repositories page, click **Add Repository** .
5. In the Add repository dialog box, type the URL of the repository location or use **Browse** to find a .zip or JAR file that contains a repository, a diskTag.inf file, or the repository.config file of an expanded repository; then click **OK** .

   The new repository location is listed. If the repository is not connected, a red x is displayed in the Connection column.

   ### Note

   To search for updated packages, make sure **Search service repositories during installation and updates** is selected. This option is selected by default.
6. Click **OK** to close the Preferences window.

Performing a silent installation of product components

You can install Rational Build Forge product components silently by running Installation Manager in silent installation mode. In silent mode, the user interface is not available; instead, a response file inputs the commands that are required to install the product package.

The following tasks are required for silent installation:

1. Install Installation Manager.
2. Create the response file.

3. Run Installation Manager in silent installation mode.

For more information about Installation Manager and installing silently, see the Installation Manager Information Center: http://www.ibm.com/software/awdtools/installmanager/support/index.html.

Creating a response file with Installation Manager

You can create a response file by recording your actions as you install a product package using Installation Manager. When you record a response file, all the selections that you make in the Installation Manager UI are stored in an XML file. When you run Installation Manager in silent mode, Installation Manager uses the XML response file to complete the installation.

You can create the response file and install the product or just skip the product installation and create the response file only by using -skipInstall <agentDataLocation> argument. The following instructions provide example syntax for both options.

To create a response file for installation:

1. At a command line, change to the eclipse subdirectory in the directory where you installed Installation Manager. For example:

Windows	C:\Program Files\IBM\Installation Manager\eclipse
UNIX/Linux	/opt/IBM/InstallationManager/eclipse

2. At a command line, use one of the following commands to start Installation Manager, substituting your own file name and location for the response file and (optionally) the log file.

   Ensure that the file paths that you enter exist; Installation Manager does not create directories for the response file and the log file. If you use the -skipInstall option, the <agentDataLocation> must be a writable directory.

   - **Record a response file and install the product**:

   ```
   IBMIM -record <response file path and name> -log <log file path and name>
   ```

   - **Record a response file without installing the product**:

   ```
   IBMIM -record <response file path and name> -log <log file path and name>
   -skipInstall <agentDataLocation>
   ```

3. Follow the instructions in the Install Packages wizard to make your installation choices.

4. Click **Finish** , and then close Installation Manager.

An XML response file is created and resides in the location specified in the command.

Installing and running Installation Manager in silent mode

Use Installation Manager to silently install product packages from a command line.

The following tasks are required for silent installation:

To run Installation Manager in silent mode, run the command for your platform from the eclipse subdirectory:

Windows	**`IBMIMc.exe --launcher.ini silent-install.ini -input <response file path and name> -log <log file path and name>`** For example, `IBMIMc.exe --launcher.ini silent-install.ini -input c:\mylog\responsefile.xml -log c:\mylog\silent_install_log.xml`
UNIX/Linux	**`IBMIM --launcher.ini silent-install.ini -input <response file path and name> -log <log file path and name>`** For example, IBMIM --launcher.ini silent-install.ini -input /root/mylog/responsefile.xml –log /root/mylog/silent_install_log.xml

When the Installation Manager runs in silent installation mode, it reads the response file and writes a log file to the directory that you specified. A response file is required; log files are optional. The result of this execution should be a status of 0 for success and a non-zero number for failure.

The following table describes the arguments to use with the silent installation command:

Argument	**Description**
-vm	Specifies the Java launcher. In silent mode, always use java.exe on Windows, and java on other platforms.
`-nosplash`	Suppresses the splash screen.
`--launcher.suppressErrors`	Suppresses the JVM error dialog box.
-silent	Runs the Installation Manager installer in silent mode.
`-input`	Specifies an response file to be used as the input to Installation Manager. The response file contains commands that installer or Installation Manager runs.
`-log`	(Optional) Creates a log file that records the result of the silent installation. The log file is an XML file.

Performing a silent upgrade of product components

You can upgrade Rational Build Forge product components silently by running Installation Manager in silent installation mode.

The following prerequisites are required for silent upgrade:

- The existing Build Forge console installation must have been installed using the Installation-Manager-based silent installation.
- Installation Manager must be installed on the same host as the Build Forge console host.

The following tasks are required for silent installation:

1. Create the upgrade response file.

2. Run Installation Manager in silent mode, specifying the response file as input.

For more information about Installation Manager and installing silently, see the Installation Manager Information Center: http://www.ibm.com/software/awdtools/installmanager/support/index.html.

Creating an Update response file with Installation Manager

Create a response file by recording your actions as you install a product package using Installation Manager.

When you record a response file, all the selections that you make in the Installation Manager UI are stored in an XML file.

To create a response file for an Update installation:

1. Run Installation Manager. In Preferences, add the URL of the product repository for the update installation to the list of IM repositories and be sure it is selected.

2. Exit Installation Manager.

3. At a command line, change to the eclipse subdirectory in the directory where you installed Installation Manager. For example:

Windows	C:\Program Files\IBM\Installation Manager\eclipse
UNIX/Linux	/opt/IBM/InstallationManager/eclipse

4. Start recording the installation without actually installing the product.

   Enter a full path, including file name, for response_file and log_file. Ensure that the file paths that you enter exist. Installation Manager does not create directories for the response file and the log file. The agentDataLocation must be a writable directory.

   ```
   IBMIM -record response_file -log log_file -skipInstall agentDataLocation
   ```

5. Installation Manager starts. In Installation Manager, click **Update** and then respond to the prompts.

6. When Installation Manager finishes, click **Finish** . then close

7. Exit Installation Manager.

An XML response file is created and resides in the location specified in the command.

Running the Update installation in silent mode

Use Installation Manager to silently install product packages from a command line.

To run Installation Manager in silent mode, the general form of the command is as follows:

```
IBMIMc.exe --launcher.ini silent-install.ini -input response_file -log log_file
```

Use a full path and file name for *response_file* and *log_file*.

- Windows example

  ```
  IBMIMc.exe --launcher.ini silent-install.ini -input c:\mylog\responsefile.xml -log
  c:\mylog\silent_install_log.xml
  ```

- UNIX or Linux example

  ```
  IBMIM --launcher.ini silent-install.ini -input /root/mylog/responsefile.xml –log
  /root/mylog/silent_install_log.xml
  ```

When the Installation Manager runs in silent installation mode, it reads the response file and writes a log file to the directory that you specified. A response file is required; log files are optional. The result of this execution should be a status of 0 for success and a non-zero number for failure.

The following table describes the arguments to use with the silent installation command:

Argument	**Description**
-vm	Specifies the Java launcher. In silent mode, always use java.exe on Windows, and java on other platforms.
`-nosplash`	Suppresses the splash screen.
`--launcher.suppressErrors`	Suppresses the JVM error dialog box.
`-silent`	Runs the Installation Manager installer in silent mode.
`-input`	Specifies an response file to be used as the input to Installation Manager. The response file contains commands that installer or Installation Manager runs.
`-log`	(Optional) Creates a log file that records the result of the silent installation. The log file is an XML file.

Installing the Build Forge system on VMware

Build Forge can be installed and run on VMWare.

Use these guidelines:

- Install the database that Build Forge uses on a separate host, preferably a physical host rather than a VMWare image.
- Set the memory that VMware Workstation uses to run your virtual machine uses to at least 1GB. In VMWare Workstation, click **Edit** → **Preferences** → **Memory** to adjust this value.

- You may need to address other system resource parameters to maximize performance on VMWare.

Note

During installation, if you choose to install a new copy of DB2 Express along with Build Forge, it is installed on the same host. Avoid using this configuration on a VMWare image in performance-critical environments.

Installing the Management Console on SUSE Linux on System z

Use the mc-<*version*>-<*build*>.tar.gz tar file provided with the installation media to install and configure the Management Console on z/Linux. IBM Installation Manager is not used for this installation.

After you install the Management Console, install the agent source pack (src-bfagent-<*version*>.tar.gz) on z/Linux to set up a z/Linux server for Build Forge. For installation instructions, see "Installing the agent on UNIX and Linux systems" on page 153.

Prerequisite software

Required operating system:

- SUSE Linux Enterprise Server (SLES) version 10 SP2 (64-bit mode)

  Supplies required versions of the following technologies: Apache HTTP Server, Perl, and PHP.

  The required modules for PHP and Perl that you must install with SUSE or update your existing SUSE installation to include are listed in the following tables.

Other software:

- IBM J9 VM (build 2.3, J2RE 1.5.0 IBM J9 2.3 Linux s390-31 j9vmxz3123ifx-20080811 (JIT enabled), provided and installed by the bfinstall program.
- Perl security modules that we provide and install: IO-Socket-SSL-1.16.tar.gz and Net-SSLeay-1.35.tar.gz, provided and installed by the bfinstall program.

Supported databases:

- DB2 9.1
- Oracle 10.2
- MySQL 5.0.45 and 5.0.15 a-community-nt

Openssl packages required and provided by SUSE:

compat-openssl097g-0.9.7g-13.9	openssl-doc-0.9.8a-18.26	php5-openssl-5.2.5-9.5

openssl-32bit-0.9.8a-18.26	compat-openssl097g-32bit-0.9.7g-13.9	openssl-0.9.8a-18.26
openssl-devel-32bit-0.9.8a-18.26	openssl-devel-0.9.8a-18.26	

Apache modules required and provided by SUSE:

apache2-mod_macro-1.1.6-19.2	apache2-mod_php5-5.2.5-9.5	apache2-worker-2.2.3-16.18
apache2-2.2.3-16.18	apache2-devel-2.2.3-16.18	apache2-prefork-2.2.3-16.18

PHP modules required and provided by SUSE:

apache2-mod_php5-5.2.5-9.5 # also provided by Apache	php5-openssl-5.2.5-9.5	php5-zlib-5.2.5-9.5
php5-mhash-5.2.5-9.5	php5-xsl-5.2.5-9.5	php5-mbstring-5.2.5-9.5
php5-xmlrpc-5.2.5-9.5	php5-mcrypt-5.2.5-9.5	php5-ldap-5.2.5-9.5
php5-ctype-5.2.5-9.5	php5-bcmath-5.2.5-9.5	php5-gettext-5.2.5-9.5
php5-sysvshm-5.2.5-9.5	php5-imap-5.2.5-9.5	php5-devel-5.2.5-9.5
php5-gd-5.2.5-9.5	php5-posix-5.2.5-9.5	php5-sysvsem-5.2.5-9.5
php5-ftp-5.2.5-9.5	php5-dom-5.2.5-9.5	php5-pcntl-5.2.5-9.5
php5-sysvmsg-5.2.5-9.5	php5-exif-5.2.5-9.5	php5-iconv-5.2.5-9.5
php5-xmlreader-5.2.5-9.5	php5-dbase-5.2.5-9.5	php5-suhosin-5.2.5-9.5
php5-json-5.2.5-9.5	php5-dba-5.2.5-9.5	php5-sockets-5.2.5-9.5
php5-curl-5.2.5-9.5	php5-5.2.5-9.5	php5-tokenizer-5.2.5-9.5
php5-calendar-5.2.5-9.5	php5-snmp-5.2.5-9.5	php5-bz2-5.2.5-9.5
php5-shmop-5.2.5-9.5		

Perl modules required and provided by SUSE:

perl-IO-Socket-SSL-0.97-14.2	perl-Net-Daemon-0.38-61.2	perl-Bit-Vector-6.4-13.5
perl-XML-LibXML-Common-0.13-17.2	perl-DBI-1.50-13.2	perl-Archive-Tar-1.24-15.2
perl-File-Tail-0.99.3-12.2	perl-Bootloader-0.4.19.12-0.3	perl-Archive-Zip-1.16-13.2
perl-Net-Server-0.90-12.2	perl-XML-NamespaceSupport-1.09-13.2	perl-Convert-ASN1-0.19-13.2
perl-Net-IP-1.23-14.2	perl-Config-IniFiles-2.39-13.4	perl-MailTools-1.67-13.2
perl-Time-Period-1.20-317.2	perl-TimeDate-1.16-136.2	perl-Convert-TNEF-0.17-274.2
perl-IO-stringy-2.110-13.2	perl-32bit-5.8.8-14.7	perl-Compress-Zlib-1.35-14.2
perl-TermReadKey-2.30-13.2	perl-PlRPC-0.2018-13.2	perl-Convert-BER-1.3101-206.
perl-DateManip-5.44-12.2	perl-IO-Zlib-1.04-14.6	perl-libwww-perl-5.805-12.2
perl-Net-SNMP-5.2.0-12.2	perl-5.8.8-14.7	perl-Date-Calc-5.4-14.5
perl-AppConfig-1.56-43.2	perl-HTML-Tagset-3.10-12.2	perl-Unix-Syslog-0.100-41.2
perl-Convert-UUlib-1.051-13.2	perl-Tie-IxHash-1.21-600.2	perl-gettext-1.05-13.2
perl-XML-LibXML-1.58-17.2		

Pre-installation setup

Install and configure prerequisite software on z/Linux:

1. Install SUSE Linux Enterprise Server (SLES) version 10, SP2 (64-bit mode) on z/Linux.

   The program binaries for the required versions of the following technologies are included: Apache HTTP Server, Perl, and PHP.

   ### Important

   You may select and install these technologies when you install SUSE, or update your existing SUSE installation to include them.

2. Set up a supported database to use as the Build Forge database. Review these topics:

   - Enable international data, using "International data setup " on page 20.
   - Set up your database, using "Database setup" on page 21.

3. Record the following values for the Build Forge database.

   During installation, the bfinstall.pl installation program prompts you for these parameter values.

Parameters	**Description**
Database host name	Fully qualified host name of the host where you installed the Build Forge database. For example, my.host.com. You can also use an IP address. If it is installed on the local host, use 127.0.0.1. Do not use the localhost name.
Services host name	Fully qualified host name of the host where the Build Forge services component is installed. For example, my.host.com. You can also use an IP address. If it is installed on the local host, use 127.0.0.1. Do not use the localhost name. Normally the services component is installed on the same host as the Build Forge console.
Database name	The name of the Build Forge database that you created. For example, build.
Database user	The name of the Build Forge database user that you created.
Database user password	The password for the Build Forge database user that you created.
Database TCP connection port	The TCP connection port for the Build Forge database.

Installing Build Forge files on z/Linux

Run the bfinstall program to install Build Forge files and configure the Build Forge database.

Before you begin, complete the following prerequisite tasks:

- Locate the tar file: mc-<*version*>-<*build*>.tar.gz. Obtain this file from the download site or the installation media.
- Verify that the Build Forge database is running and accessible from the z/Linux host.
- Create an operating system user that you want to run Build Forge as; this user is referred to as the build user.
- Ensure that installation program, bfinstall.pl, can access the required environment variables for your Build Forge database:

DB	**Environment variable**	**Value**
DB2	DB2_HOME	The installation directory of the DB2 server. For example, /opt/ibm/db2/V9.1.
DB2	DB2DIR	The location of the DB2 client libraries. For example, /opt/ibm/db2/V9.1.
Oracle	ORACLE_HOME	Oracle client software installation directory
Oracle	LD_LIBRARY_PATH	$ORACLE_HOME/lib, where ORACLE_HOME is the Oracle client software installation directory.
Oracle	TNS_ADMIN	$ORACLE HOME/network/admin, where ORACLE_HOME is the Oracle client software installation directory.

1. As the build user or the user that you want to run Build Forge as, create an installation directory for Build Forge, for example: /usr/local/buildforge.
2. As the build user or the user that you want to run Build Forge as, copy the tar file to the installation directory.
3. As the build user or the user that you want to run Build Forge as, extract the tar file contents to the installation directory; for example: tar -xzf mc-<*version*>-<*build*>.tar.gz.
4. As the root user, change to the installation directory for Build Forge or the directory where you extracted the tar file and run the bfinstall.pl program.

   The bfinstall.pl program takes the following option:

Parameter	**Description**
- -with-dbtype= db2 | oracle | mysql [, DBlibdir]	Required. The database type is required. Optionally, for DB2, specify value of the DB2DIR environment variable. This option ensures that the bfinstall program can access the DB2 client libraries.

5. The bfinstall program prompts you for the following database values. For most parameters, default values are not provided; example user responses are shown.

```
What host is housing the DB used for buildforge?
   localhost
What is the name of the database for buildforge?
   build
What is the name of the database user for buildforge?
   build
What is the password of the database user for buildforge?
   build
If you database is running on the non-default port enter it here:
   60000
```

6. At the following prompt, type the fully qualified host name of the z/Linux host. The Apache Tomcat server is installed on the z/Linux host.

```
What is the hostname of the services (Tomcat) machine for Build Forge?
   linux142.rtp.raleigh.ibm.com
```

Note

If you are using Web Application Server instead of the provided Apache Tomcat, specify the fully-qualified host name of the Web Application Server.

7. At the following prompt, type **y** to create the Build Forge schema tables in the Build Forge database.

```
Are you sure you want to delete all the tables in your database? y|n> y
```

A command prompt displays after the schema tables have been created. Creating the schema tables takes a few minutes.

Note

If there are errors in the data you entered, the script fails. If the script fails, do the following:

a. Delete `<`*`bfinstall`*`/Platform/buildforge.conf`

b. Re-run the script.

Post-installation tasks for Apache Tomcat

To update the rbf-services.war file for Apache Tomcat, the bfinstall.pl program changes the owner of rbf-services.war to root.

After installation, manually change the owner of the rbf-services.war file to the user that you are running Build Forge as. In the following example, the user is build:

```
chown build tomcat/webapps/rbf-services.war
```

Post-installation tasks for Apache HTTP Server

Complete post-installation tasks for the Apache HTTP Server. Restart the Apache HTTP Server to pick up the changes to httpd.conf.

Set permissions for the Apache user

1. Log on as root or the user that Apache HTTP Server runs as.
2. Change to the Build Forge installation directory and set the following permissions for the Apache user. In the example, substitute wwwrun with the user name who is running the Apache HTTP Server.

   ```
   chown wwwrun ui/templates_c
   ```

Edit Apache HTTP Server configuration

Edit the <*ApacheDir*>/etc/apache2/httpd.conf file:

1. Change the DocumentRoot directive to point to the entry page for the Management Console UI:

   ```
   DocumentRoot <bf-install>/ui/public
   ```

2. Change the Directory directive to point to the entry page for the Build Forge UI:

   ```
   <Directory /<bf-install>/ui/public>
   </Directory
   ```

3. Ensure that the Options directive includes Indexes.

   ```
   <Directory /<bf-install>/ui/public>
   Options Indexes
   </Directory
   ```

Edit the PHP initialization file

Do the following:

1. Go `/etc/php5/apache2/php.ini`
2. Set the database extension for PHP to use. Place the appropriate `extension=` line after the `extensiondir=` line in the file.
   - DB2:

     ```
     extension=ibm_db2.so
     ```

   - Oracle:

     ```
     extension=oci8.so
     ```

- MySQL (two lines):

```
extension=mysql.so
extension=mysqli.so
```

Configure Apache HTTP Server for DB2

To pass environment variables for DB2 to the Build Forge UI, add the following lines to the <*ApacheDir*>/etc/apache2/httpd.conf file.

```
PassEnv LD_LIBRARY_PATH
PassEnv CLASSPATH
PassEnv LIBPATH
PassEnv VWSPATH
```

Add the following lines to the Apache startup script (/etc/init.d/apache2) after the `END INIT INFO` comment line.

```
### END INIT INFO
. /opt/share/db2inst1/sqllib/db2profile
DB2_HOME=/opt/share/db2inst1/sqllib
LIBPATH=$LIBPATH:$DB2_HOME/lib:$DB2_HOME/lib64
VWSPATH=$DB2_HOME

export DB2_HOME LIBPATH VWSPATH
```

Configure Apache HTTP Server for Oracle

Edit the <*ApacheDir*>/etc/apache2/httpd.conf file to pass environment variables for Oracle to the Build Forge UI.

```
PassEnv LD_LIBRARY_PATH
PassEnv ORACLE_HOME
PassEnv TNS_ADMIN
PassEnv NLS_LANG
PassEnv ORA_NLS
PassEnv ORA_NLS32
```

Add the following lines to the Apache startup script (/etc/init.d/apache2) after the `END INIT INFO` comment line.

```
### END INIT INFO
export LD_LIBRARY_PATH=<value>
export ORACLE_HOME=<value>
export TNS_ADMIN=<value>
export NLS_LANG=<value>
export ORA_NLS=<value>
export ORA_NLS32=<value>
```

Restart Apache HTTP Server

To pick up changes to the httpd.conf file, you must restart the Apache HTTP Server after you are done editing the configuration file.

Starting the Management Console

1. Start the Apache HTTP Server:

```
service apache2 start
```

2. Start the Management Console:

   ```
   <bf-install>/rc/buildforge start
   ```

3. Verify that services layer (the Apache Tomcat server) started; open catalina.out and verify that start up messages are logged.

   ```
   <bf-install>/tomat/logs/catalina.out
   ```

4. Start a Web browser and enter the fully-qualified z/Linux host name. For example: linux142.rtp.raleigh.ibm.com.

   The Management Console starts and displays the login prompt. Login as **root/root** .

Installing the license file

The license file for z/Linux is located in <*bf-install*> directory. The license file name is IRBF_license.

To install the license file:

1. Start the Management Console.
2. Log in as **root/root**.
3. Select **Administration > System** .
4. Locate the License Server setting and set its value to fully-qualified path to the license file.

   For example: <*bf-install*>/IRBF_license.

Enabling SSL for the Management Console

You can enable SSL to encrypt the data that is transferred between Build Forge components:

- Web browser client and the Apache HTTP Server
- Apache Tomcat Server and the Apache HTTP Server

Complete the tasks described in this section to enable SSL:

1. Create new certificates: personal certificates and keystores
2. Configure Apache Tomcat Server for SSL
3. Configure Apache HTTP Server for SSL
4. Update keystore passwords in the Build Forge database
5. Enabling SSL in the Management Console UI
6. Enabling debugging for SSL

Note

Other security features, such as password encryption and Single Sign-On (SSO), are not supported for Build Forge on z/Linux in this release.

Prerequisite software:

- IBM JDK for z/Linux, installed by the bfinstall program
- OpenSSL is provided with the required version of SUSE.
- Perl modules for SSL: Net::SSLeay and IO::Socket::SSL, are installed by the bfinstall program

Create new certificates: personal certificates and keystores

Create a self-signed certificate and a set of keystores (PEM and PKCS12). The following keystores are created:

Keystore	Description
buildForgeKeyStore.p12	Contains a password protected keyEntry (personal certificate with public/private key pair).
buildForgeTrustStore.p12	Contains a password protected trustedCertEntry (certificate with public key only).
buildForgeKey.pem	Contains a password protected private key.
buildForgeCert.pem	Contains a non-password protected certificate with public key corresponding to the private key in buildForgeKey.pem.
buildForgeCA.pem	Initially, contains the same information as buildForgeCert.pem; other peer certificate are added to establish trust.
buildForgeKeyForApache.pem	This keystore is needed to enable SSL for Apache HTTP Server. Unlike buildForgeKey.pem, it is not password protected allowing the Apache HTTP Server to start up without a password prompt.

The keytool utility is part of the IBM Java Development Kit (JDK) and installed by the bfinstall program. The openssl utility is part of OpenSSL that is provided by SUSE Linux Enterprise Server (SLES) version 10, SP2.

1. Change to the Build Forge installation directory: <*bf-install*>/Platform
2. Create a keystore directory:

   ```
   mkdir keystore
   ```

3. Change to the keystore directory:

   ```
   cd keystore
   ```

4. Export the path to the IBM JDK:

   ```
   export PATH=<bf-install>/ibmjdk/bin:$PATH
   ```

5. Export the JAVA_HOME environment variable:

   ```
   export JAVA_HOME=<bf-install>/ibmjdk
   ```

6. Create the personal certificate in the JSSE keystore:

   ```
   keytool -genkey -alias buildforge -keyalg RSA -keysize 1024
    -validity <days> -dname <distinguished_name>
    -keystore buildForgeKeyStore.12 -storepass <password>
    -keypass <password> -storetype pkcs12
   ```

   Provide values for the following variables:

-validity <*days*>	Required. At least 365 days or 1 year. You must replace the certificate before it expires to avoid outages. There is no maximum value.
-dname <*distinguished_name*>	Required. The fully-qualified host name where the Apache HTTP Server is installed. Use the following format: `CN=linux142.rtp.raleigh.ibm.com`
-storepass <*password*>	Required. The storepass is used to change the default password for the keystore. The password must be at least 6 characters in length and comply with your password policies. **Important** The following setup commands assume that the -storepass and the -keypass are the same.
-keypass <*password*>	Required. The keystore password is used to assign individual passwords to the private keys. The password must be at least 6 characters in length and comply with your password policies. **Important** The following setup commands assume the -storepass and the -keypass are the same.

7. Export the public key to a file:

   ```
   keytool -export -alias buildforge -file cert.der
    -keystore buildForgeKeyStore.p12
    -storepass <
                           password
                        > -storetype pkcs12
   ```

   Provide a value for the following variable:

   - A password is required. Use the keystore password created by the keytool -genkey command.

8. Import the public key into the JSSE trust store:

```
keytool -import -noprompt -trustcacerts -alias buildforge
 -file cert.der -keystore buildForgeTrustStore.p12
 -storepass <
                        password
                     > -storetype pkcs12
```

Provide a value for the following variable:

- A password is required. Use the keystore password created by the keytool -genkey command.

9. Create the OpenSSL cert store containing the public key:

```
openssl pkcs12 -clcerts -nokeys -in buildForgeKeyStore.p12
 -passin pass:<
                        password
                     > -out buildForgeCert.pem
```

Provide a value for the following variable:

- A password is required. Use the keystore password created by the keytool -genkey command.

10. Create the OpenSSL key store containing the public and private key:

```
openssl pkcs12 -in buildForgeKeyStore.p12 -passin pass:<
                        password
                     >
 -passout pass:<
                        password
                     > -out buildForgeKey.pem
```

Provide a value for the following variable:

- A password is required. Use the keystore password created by the keytool -genkey command.

11. Create the OpenSSL CA store: `cat buildForgeCert.pem > buildForgeCA.pem`

12. Optional. Create the OpenSSL key store for Apache HTTP Server:

```
openssl rsa -in buildForgeKey.pem -passin pass:<
                        password
                     >
 -out buildForgeKeyForApache.pem
```

Provide a value for the following variable:

- A password is required. Use the keystore password created by the keytool -genkey command.

13. Optional. Verify that you have created all of the required keystores. Run a directory listing on the keystore directory:

```
ls <bf-install>/Platform/keystore
```

Configuring Apache Tomcat Server for SSL

Complete the following steps to configure Apache Tomcat for SSL on port 8443. The keystore password is entered and stored in the server configuration file (server.xml) file in plain text.

Important

Restrict access to this file to the user ID who runs Build Forge and users who need to modify the server configuration file.

1. Change to the Apache Tomcat conf directory:

   ```
   cd <bf-install>/tomcat/conf
   ```

2. Edit the server.xml file as follows:

   a. In the existing file, replace the commented section starting with `<!--<Connector port="8443"` with the following section:

   ```
   <Connector port="8443"
   acceptCount="100"
    bufferSize="8192"
    algorithm="IbmX509"
    connectionTimeout="60000"
    clientAuth="false"
    disableUploadTimeout="true"
    enableLookups="false"
    maxHttpHeaderSize="8192"
   maxSpareThreads="25"
    maxThreads="200"
    minSpareThreads="4"
    scheme="https"
    secure="true"
    sslProtocol="SSL_TLS"
   timeout="300000"
   keystoreFile="<
                                 bf-install
                              >/Platform/keystore/buildForgeKeyStore.p12"
   keystorePass="<password>"
   keystoreType="PKCS12"
   truststoreFile="<
                                 bf-install
                              >/Platform/keystore/buildForgeTrustStore.p12"
   truststorePass="<
                                 password
                              >"
   truststoreType="PKCS12"/>
   ```

   b. Provide values for the following variables:

   - <*bf-install*>. The Build Forge installation directory.
   - truststorePass=<*password*>. The keystore password used to access the keystore.

3. Start the Apache Tomcat server from the <*bf-install*>/Platform directory.

```
../tomcat/bin/catalina.sh start
```

Configuring Apache HTTP Server for SSL

The following updates the SSL configuration for the Apache HTTP Server to use the keystores created for Build Forge.

Note

As a prerequisite, your Apache HTTP Server has been configured for SSL and is using port 443.

1. Change to your <*ApacheDir*>/etc/apache2 directory (or wherever the ssl-global.conf is located).
2. Edit the ssl-global.conf file, as follows:
   a. Add the following lines to the <IfModule mod_ssl.c> section of /etc/apache2/ssl-global.conf.

   ```
   SSLCertificateKeyFile <

   bf-install>/Platform/keystore/buildForgeKeyForApache.pem
   SSLCertificateFile <
                                 bf-install>/Platform/keystore/buildForgeCert.pem
   SSLCACertificateFile <
                                 bf-install>/Platform/keystore/buildForgeCA.pem
   ```

   b. Substitute the Build Forge installation directory as the value of the <*bf-install*> variable.
3. Restart Apache with SSL enabled:

   ```
   rcapache2 restart -DSSL
   ```

Updating the Build Forge Database with the Keystore Passwords

When the Management Console UI starts, it accesses the keystore password information from the in the bf_security_keystore table in the Build Forge database.

Update the keystore password in the database, as follows:

1. Connect to your database from the command line.
2. Update the passwords in the database with the following SQL statement:

   ```
   update BF_SECURITY_KEYSTORES set bf_password = ‘<
                            password
                        >'
    where bf_password is not null
   ```

   Provide a value for the following variable:

   - <*bf_password*>. The keystore password used to access the keystore.

Enabling SSL in the Management Console UI

Use the Management Console UI (**Administration > Security**) settings to enable SSL in the Management Console and update the Build Forge database. Then, check that the required property values are updated in the bfclient.conf configuration file.

1. Start Build Forge.
2. Login to the UI.
3. Go to **Administration > Security** .
4. Change SSL Enabled: to **Yes** .
5. Click **Save** .
6. Click **Update Master BFClient.conf** .
7. Stop Build Forge.

Verify that bfclient.conf has been updated

1. Change to the `<bf-install>/Platform` directory.
2. Open the file `bfclient.conf`.
3. Verify that the property values listed in bold have the correct values:

```
# Connection configuration
# These properties are only used when services layer API does not supply them.
 bf_services_hostname=localhost

                    bf_services_tcp_port=3966

                    bf_services_ssl_port=49150
 bf_services_preferred_protocol=ssl

# Login configuration
# These properties are only used when services layer API does not supply them.
bf_login_user=root
bf_login_password=fe5d71bccdf6a11500f78a381cccf998ce6258467105cafbbb42
#bf_login_realm=ADDomain

# OpenSSL configuration
# This SSL configuration is only used by Perl and PHP code.
bf_ssl_usage=openssl
bf_ssl_key_ref=openssl_key
bf_ssl_cert_ref=openssl_cert
bf_ssl_ca_ref=openssl_ca
bf_ssl_cipher_group=ALL
#bf_ssl_cipher_override=SSL_RSA_WITH_RC4_128_MD5
bf_ssl_protocol=TLSv1

# JSSE configuration
# This SSL configuration is only used by Java code.
bf_ssl_usage=jsse
bf_ssl_keystore_ref=jsse_keystore
#bf_ssl_cert_alias=buildforge
bf_ssl_truststore_ref=jsse_truststore
```

```
bf_ssl_cipher_group=ALL
#bf_ssl_cipher_override=SSL_RSA_WITH_RC4_128_MD5
bf_ssl_protocol=SSL_TLS

# PKCS12 keystores are for IBMJSSE2 used by Java
bf_keystore_alias=jsse_keystore
bf_keystore_location=./keystore/buildForgeKeyStore.p12
bf_keystore_type=PKCS12
bf_keystore_password=ab956bca9598991a00f7dff006baa1f6f66d2592976648aff393
bf_keystore_alias=jsse_truststore
bf_keystore_location=./keystore/buildForgeTrustStore.p12
bf_keystore_type=PKCS12
bf_keystore_password=c12f5c37143e6e0200f7b54a3147205001759b7ee8ca25f0497f
# PEM keystores are for OpenSSL used by Perl and PHPs
bf_keystore_alias=openssl_key
bf_keystore_location=./keystore/buildForgeKey.pem
bf_keystore_type=PEM
bf_keystore_password=2b9d0aaf763b2b9b00f75ff867df425544ec24ac480c34033dcd
bf_keystore_alias=openssl_cert
bf_keystore_location=./keystore/buildForgeCert.pem
bf_keystore_type=PEM

# This keystore is critical for OpenSSL clients (Perl/PHP) to validate certificates.
# To create a PEM with multiple certificates, use "cat file1.pem file2.pem >
buildForgeCA.pem".
bf_keystore_alias=openssl_ca
bf_keystore_location=./keystore/buildForgeCA.pem
bf_keystore_type=PEM

# Password encryption configuration
# Enabling password encryption will cause this file to be modified when BuildForge
starts up.
bf_pw_crypt_enabled=false
```

4. **Important**: Start the engine from the <*bf-install*>/Platform directory. Use the following command to start the engine: `./buildforge.pl`

Enabling Debugging for SSL

To debug issues with SSL in the Management Console, use the following instructions to log additional information needed for SSL.

1. Enable debugging in the engine. Set the following environment variable before you start the Build Forge engine:

   ```
   export BFDEBUG_SECURITY=1
   ```

   a. Restart the Build Forge engine.

   b. Restart Apache. Restarting Apache enables PHP to use this debug parameter.

2. Enable debugging in Tomcat. Make the following changes in `<bf-install>/tomcat/common/classes/logging.properties`:

   a. Add the following line:

```
com.buildforge.level = ALL
```

b. In the handlers section, change all other levels from `FINE` to `ALL`.

Restart Tomcat to make the changes take effect.

Configuring additional features in Management Console

This section describes ways to configure Build Forge® to enable additional features or provide alternatives to the default configuration.

Build Forge configuration file (buildforge.conf)

The buildforge.conf file is the Build Forge product configuration file. It contains configuration settings that are used by different Build Forge components to start up and communicate with the Build Forge database.

The buildforge.conf file is stored in two locations and must be updated in both locations, if you need to modify it after installation. See Updating the buildforge.conf file.

- In the rbf-services.war file used by the application server, also called the services layer.
- In the installation root directory. The following table lists the default or standard installation directories for the product:

Windows	C:\Program Files\IBM\Build Forge
UNIX/Linux	/usr/local/buildforge/Platform

Updating the buildforge.conf file

The buildforge.conf file is located in two locations and must be updated in both locations, if you need to update it after installation.

You might need to edit the buildforge.conf file to update the database host if your Build Forge database is moved to a different host machine. Another common reason to edit this file is to update your database password, which must be changed regularly to comply with network security policies.

Use the following procedure to update the buildforge.conf file and then recreate the rbf-services.war file with the updated copy of buildforge.conf.

Note

If you only need to add buildforge.conf to rbf-services.war, skip the steps for editing buildforge.conf.

1. Stop the Build Forge engine.
2. Locate the buildforge.conf file in your Build Forge installation root directory.
3. Use a text editor to open the file, modify configuration settings, and save the file.

Note

You need root or Administrator privileges to edit this file.

4. Navigate to the directory that contains the rbf-services.war file, for example:

Apache Tomcat server	`<`*`bf-install`*`>/Apache/tomcat/webapps`
	`$CATALINA_HOME/webapps`

5. Create a backup copy of rbf-services.war, for example:

Windows	`copy rbf-services.war rbf-services.war.bak`
UNIX/Linux	`cp rbf-services.war rbf-services.war.bak`

6. Delete the rbf-services directory, for example:

Windows	`rmdir rbf-services /s /q`
UNIX/Linux	`rm -rf rbf-services`

7. Recreate the rbf-services directory, for example:

Windows	`mkdir WEB-INF\classes`
UNIX/Linux	`mkdir -p WEB-INF/classes`

8. Copy the updated buildforge.conf file into the new rbf-services directory, for example:

Windows	`copy <`*`path_to_bf_conf`*`>\buildforge.conf WEB-INF\classes`
UNIX/Linux	`cp <`*`path_to_bf_conf`*`>/buildforge.conf WEB-INF/classes`

9. Update the rbf-services.war file, as follows:

Windows	`jar -uvf rbf-services.war WEB-INF`
UNIX/Linux	`jar -uvf rbf-services.war WEB-INF`

10. Restart the Build Forge engine.

    The rbf-services file is redeployed and automatically creates the rbf-services directory again with the updated buildforge.conf.

Buildforge.conf reference

The buildforge.conf file stores settings for how the Build Forge Management Console runs.

This file is in the installation directory. It is built automatically by the installer. If you need to make edits, the file can be saved as an ASCII.txt file or an XML file. The syntax is as follows:

- Type the keywords and their values on one line.
- Separate keywords and values with an equal sign (no spaces). For example, type the keyword: `bf_file_storage=C:\Program Files\IBM\BuildForge\files`

Keyword	**Value**
bf_file_storage	Directory where temporary Build Forge files reside. Example: `C:\Program Files\IBM\BuildForge\temp`
bf_plugin_dir	Directory where IDE plug-ins connecting to a Management Console reside.
birt_home	File location of Eclipse reporting tool (BIRT).
db_database	The name of the database that you created for console use.
db_hostname	The host name/IP address of the machine running the database. When entering a value for db_hostname, use the actual name or an IP address. Do not use the default of `localhost`.
db_password	The password that you created for the database user name.
db_provider	The database you elected to install Build Forge to. Do not edit this value.
db_schema	The name of the schema of your database (usually the same as db_username, but you may have chosen another schema name).
db_tcp_port	The database connection port you are using.
db_type	The type of database Build Forge was installed to. Default is `odbc`. Do not edit this value.
db_username	The user name of the database. This is set up before running the installer. For DB2 and DB2 Express, create a user in Windows, not in DB2. For all other database types, you create a database user name. See "Database setup" on page 21. This value is required for all console types.
services_hostname	The host name/IP address of the machine running the Build Forge services layer. It is the

Keyword	Value
	fully qualified domain name (FQDN) in your services configuration.
services_ssl_port	The SSL port for connecting to Build Forge services securely.
services_tcp_port	The TCP port for connecting to Build Forge services when SSL is not specified.
services_url	URL that specifies the port used for the services layer. Example: `services_url  http://mybfhost.com:8080`

Configuring Management Console to use an alternate port

You can run Management Console on a port other than default port 80.

You can configure Management Console to run on an alternate port in two ways:

- During installation, set the port to the value that you want.
- If Management Console is already installed, complete these steps:

  1. Change two settings in *bf-install*/Apache/conf/httpd.conf. For example, if myHost is your local computer and you want to use port 81, specify these settings:

     ```
     Listen 81
     ServerName myHost:81
     ```

  2. Start the console and log in as root or a user name with administrative privileges.
  3. Click **Administration** → **System** and change the following setting:

     Set **Console URL** to the URL and port that Management Console runs on. In this example it is `http://myHost:81`. Do not use a trailing slash.

  4. Stop and then restart the engine.

     - Windows: Click **Start** → **Programs** → **IBM Rational Build Forge Management Console** → **Stop Engine Service** and then click **Start Engine Service** .

       If Build Forge is running in the foreground, go to the Windows console where it is running, then enter Ctrl-C.

     - UNIX or Linux: Use the script provided for the rc file. Typically the rc file is in /etc/init.d.

       ```
       $ /<path_to_rc_file>/buildforge start
       ```

       ```
       $ /<path_to_rc_file>/buildforge stop
       ```

You may also use manual commands.

a. To stop, find the process ID and kill the process.

```
$ ps aux | grep buildforge
$ kill ${<PID>}
```

b. To start, use the following command, where *<bfinstall>* is the path to the installation directory:

```
<bfinstall>/Platform/buildforge.pl
```

Configuring redundancy

You can set up multiple machines to run Build Forge, all communicating with the same Build Forge database. This setup is called redundancy.

About redundancy

Redundancy helps balance job processing and increases availability should one installation fail.

Important

Redundancy does not provide failover capability or other high-availability capabilities. It simply increases job-processing capacity. If one of the redundant installations fails, all executing jobs managed by that installation are lost, but the remaining installations continue to process their executing jobs and accept new jobs.

When users start jobs, entries for the job are made in the database. The process engine polls the database to see if there are new jobs. When there are multiple process engines, load balancing occurs naturally because each engine polls independently during its non-busy cycles.

When you set up redundancy, you perform a normal installation of management console, then install management console on additional hosts. All of the installations are configured to access the same Build Forge database.

Important

Every installation must be installed on its own host. You cannot install multiple management consoles on the same host.

Installing redundant systems

These instructions assume that you have already set up a database and installed the first installation of management console to use it. To create additional installations on other hosts, do the following:

1. Perform pre-installation setup on the host according to instructions for your database.

   Depending on your database, the setup may require installing a database client on the host and performing other configuration. See “Pre-installation setup” on page 20.

2. Make sure the database server is configured to accept external connections (TCP).
3. Follow the installation instructions.

   Starting at “Installing the Management Console” on page 40, do the following:

   - Install Installation Manager (if necessary)
   - Start Installation Manager
   - Run the installation.

   Only panels that require specific entries are described in the following steps.
4. At the **Install Packages: Features** panel, make sure all features are selected.
5. At the **Database Configuration** panel, do the following:
   - Provide the database name and schema name. These must be the same as specified on the first console.
   - Provide the database user name and password. Use the same as specified on the first console.
   - At **Do you wish to populate this database at install time?** , select **No** .

     ### Caution

     Risk of data loss: if you select Yes, the Build Forge database schema is overwritten in the database. All data from any Build Forge operations on the first installation is lost.
   - Click **Perform Test** . When the test passes, you can click **Next** to proceed.

     ### Note

     when **No** is selected, the only test performed is a check for the correct path to the JDBC driver.
6. At the **Application and Web Server Configuration** panel, use the same values that you provided for the first console.
   - Port: this engine should listen to the same port as the first console.
7. At the **Console Start Options** panel, the **Do Not Start Build Forge** option is selected.

   It is also greyed out so it cannot be changed. You need to start the console manually after installation.
8. Perform any necessary post-installation configuration.

   See “Post-installation tasks” on page 180. If necessary, catalog the database so that the database client can connect to it.

Working with redundancy

After setting up redundancy, you work with it as follows:

- Point users to the URL of the first installation.
- You can stop the Apache server on the additional installations if you do not want users to access them.
- If you want increase capacity for handling HTTP requests, run Apache on all installations. Install a load balancer to distribute requests among the installations.

Enabling IPv6 network support

You can configure Management Console with IPv6 and mixed IPv6/IPv4 networks.

Configuring Management Console for IPv6 requires the following::

1. Change the server entry in httpd.conf.
2. Set up a FlexLM license client required for use with IPv6 networks.

Review the requirements and configuration steps for using Build Forge with IPv6. See "Networking requirements for IPv6 support" on page 12.

Modifying httpd.conf

IPv6 support requires that your computers and network are configured correctly to support IPv6. Network configuration problems will prevent host names and addresses that are specified from within the Build Forge system from resolving correctly.

You must manually configure Build Forge for IPv6. To do this, modify an entry in the main Apache configuration file, httpd.conf. :

1. Navigate to your httpd.conf file (`buildforge/server/apache/conf/httpd.conf`).
2. Add the ServerName that references the fully qualified domain name (FQDN). For example, `ServerName qlnx500-v6.ipv6.lexma.ibm.com`
3. Modify the Listen directive from `0.0.0.0:80` as follows.
   - Windows: `[::]:80`
   - UNIX or Linux:`80`

Setting up the FLEXlm client

Running Build Forge on IPv6 networks requires a different version of the FLEXlm license client.

The FLEXlm client that is installed and configured with the product supports versions 7.0 and 7.1 of the Rational License Server.

Important

Do not complete this task unless you are explicitly required to use IPv6 addresses with Rational License Server 7.1.

To configure the IPv6 version of the FLEXlm client:

1. Stop the Build Forge engine.
2. Navigate to the directory where FLEXlm client software is installed:

Windows	C:\<*bf-installdir*>\Apache\tomcat\webapps\rbf-services\bin, where bf-installdir is the Build Forge installation directory. The default installation directory is C:\Program Files\IBM\Build Forge.
UNIX/Linux	<*bf-installdir*>/server/tomcat/webapps/rbf-services/bin, where bf-installdir is the Build Forge installation directory. On UNIX/Linux, the recommended installation directory is usr/local/buildforge.

3. Back up the current FLEXlm client license file for your operating system by renaming it, for example:

Windows	Rename flexhelper.exe to flexhelp.exe.bak.
UNIX/Linux	Rename flexhelper-Linux-i386 to flexhelper-Linux-i386.bak.

4. Rename the FLEXlm client license file for IPv6 so that the product can use it as the current FLEXlm client license file:

Windows	Rename flexhelper-IPv6.exe to flexhelper.exe.
Linux	Rename flexhelper-Linux-i386-IPV6 to flexhelper-Linux-i386.

5. Start the Build Forge engine.

Integration with other products

This section describes ways to integrate Build Forge® with other products:

- Rational Team Concert
- Websphere

Rational Team Concert integration

Build Forge can be integrated with Rational Team Concert.

When integration is set up, users of Rational Team Concert can do the following:

- Set up Build Forge as an RTC build server
- Set up Build Forge projects as RTC build definitions
- View projects, run jobs, and view job results from the RTC client

The integration includes three components:

- Server: the JazzSCM.xml adaptor template allows Build Forge to access RTC source repositories.The adaptor is installed with Build Forge and is set up and added to projects in the same way as other source adaptors. In the Build Forge online help, see **Adaptor integrations → Adaptor task overview → Adding adaptors to projects** for details.
- Server Extension for Rational Team Concert Server enables communication between Rational Team Concert and Build Forge. The server extension is installed on the RTC server. Installation instructions are in Installing Rational Team Concert Server Extension.
- JazzSCM adaptor: a source type adaptor to allow projects to access files in the Rational Team Concert repository
- Client: a plug-in feature for the Rational Team Concert client: allows users to access Build Forge to view projects, run jobs, and view job results. Installation instructions are in Installing the client plug-in for Rational Team Concert.

Note

The client plug-in must be installed on each RTC client that will access Build Forge.

Important

Build Forge limits a user to one login session. The same user name cannot be logged in through the Management Console and the RTC client at the same time. When a new session is started, any existing session is terminated. RTC client users should have two IDs:

- UserRTC - used when they configure a build definition to run a build. This user name is used to access the Build Forge console to run the build.
- UserBF - used when they check build results. A direct login to the Build Forge Management console is required within a window in the RTC client.

Note

Additional Build Forge users are needed for this integration:

- One for the Rational Team Concert Server to connect to Build Forge.

Installing Rational Team Concert Server Extension

Prerequisites:

- Rational Team Concert must be version 1.x or 2.x. Note where the instructions and files differ by version in the procedure.
- The Build Forge system must be running

Perform these steps from the Rational Team Concert Server host:

1. Log on to the Rational Team Concert server host.
2. Navigate to the Build Forge server using the following URL.

   `http://<bf_console_hostname>/clients`

   - If you are running Rational Team Concert on the same system that Build Forge is running on, you can use `localhost` as the host name.
   - Include the port number if the console is running on a port other than 80. Example: `http://myhostname:11812/clients`
3. Download the server extension to a temporary location.

   Choose the extension that corresponds to your version of Rational Team Concert, one of the following:

   - **Rational Team Concert 1.x Server Extension**
   - **Rational Team Concert 2.x Server Extension**

   The file downloaded in either case is `BuildForgeConnectorServer.zip`.
4. Unzip the file.

   Unzip the file to

   `<RTC_install>/jazz/server`

   The following files are added if you downloaded the extension for Rational Team Concert version 1.x:

   ```
   <RTC_install>/jazz/server/buildforgeconnector-update-site (directory)
   <RTC_install>/jazz/server/provision_profiles/buildforgeconnector-profile.ini
   ```

   The following files are added if you downloaded the extension for Rational Team Concert version 2.x:

   ```
   <RTC_install>/jazz/server/buildforgeconnector-update-site (directory)
   <RTC_install>/jazz/server/conf/provision_profiles/buildforgeconnector-profile.ini
   ```
5. Restart the Rational Team Concert Server.

Installing the client plug-in for Rational Team Concert

Install the client plug-in for Rational Team Concert from the Build Forge server.

Prerequisites:

- Rational Team Concert version 1.0.1 or later
- The Build Forge Connector server extension must be installed on the Rational Team Concert server
- The Build Forge system must be running

The client plug-in must be installed on every Rational Team Concert client that will access Build Forge.

To install the plug-ins, perform these steps from a Rational Team Concert client.

1. Select **Help** → **Software Updates** → **Find and Install** .
2. Select **Search for new features to install** , then click **Next** .

   The system displays the **Update Sites to Visit** dialog.
3. Click **New Remote Site** .

   The system displays the **New Remote Site** dialog.

   a. Enter **`Build Forge Connector Update Site`** in the name field.

   b. Enter the update site URL in the **URL** field.

   The URL varies according to the version of Rational Team Concert that you are using.

   ```
   Rational Team Concert version 1.x:
   http://<console_host_name>/rtc/site.xml

   Rational Team Concert version 2.x:
   http://<console_host_name>/rtc2/site.xml
   ```

   - If you are running Rational Team Concert on the same system that Build Forge is running on, you can use `localhost` as the host name.
   - Include the port number if the console is running on a port other than 80. The following example works if you are using Rational Team Concert version 1.x and Build Forge is running on port 11812. `http://myhostname:11812/rtc/site.xml`

   c. Click **OK** .
4. In the **Update sites to visit** dialog, select the **Build Forge Connector Update Site** check box, then click **Finish** .
5. The system displays a list of available features from the update site in the **Search Results** dialog. Select all the offered features, then click **Next** .
6. Read the license agreements, then select **I accept the terms in the license agreements** , then click **Next** .
7. Select the location where you want to install the features.

To add a new location, click **Change Location** , then browse to the location you want.

8. Click **Finish** .

9. If a **Feature Verification** dialog is shown, click **Install** .

   The dialog appears because the plug-ins are unsigned features. The dialog is shown once for each feature installed unless you selected **Install All** .

10. You are asked to restart Rational Team Concert to make the changes take effect. Click **Yes** .

11. Update the console.

    If you changed the port number used by the console, then in **Administration** → **System** , change this settings:

    - **Console URL** set to the full URL for the console, including port number. Example if the console is running on mymachine.mydomain.com on port 81:

      ```
      http://mymachine.mydomain.com:81
      ```

When the integration is complete, you can use the plug-in to run jobs and examine job results. You must specify RationalBuildForgeConnector as the Build Engine for your builds.

Alternate Installation when SSL is enabled

Use an alternate installation method when SSL is enabled on the Build Forge system.

The current version of Rational Team Concert is not enabled for SSL. Therefore the plug-in cannot be installed from the Build Forge system when the Build Forge system has SSL enabled. As a workaround, make the plug-in installation files available on a non-secure web server or distribute them to users manually.

The following files are required from either `<`*`bfinstall`*`>/webroot/public/rtc` (for RTC version 1.x) or `<`*`bfinstall`*`>/webroot/public/rtc2` (for RTC version 2.x):

- `features` directory
- `plugins` directory
- `site.xml` file

Once you have made the `rtc` or `rtc2` directory available, users perform these steps from within their Rational Team Concert client:

1. Select **Help** → **Software Updates** → **Find and Install** .

2. Click **Search for new features to install** , then click **Next** .

   The system displays the **Update Sites to Visit** dialog.

3. Create a new site.

   Choose one of the following procedures.

   - Getting the files from a remote server:

     a. Click **New Remote Site** . The system displays the **New Remote Site** dialog.

     b. Enter “Build Forge Connector Update Site” in the name field.

     c. Enter the location of the files:

        ```
        http://host/prism/eclipse/updateSite/site.xml
        ```

        The *host* is the host name or IP address of the web server.

        The *path* is the path from the server root to where you placed the files.

     d. Click **OK** .

   - Getting the files from the local host:

     a. Click **New Local Site** . The system displays the **New Local Site** dialog.

     b. Enter “Build Forge Connector Update Site” in the name field.

     c. Enter the location of the files:

        ```
        file://path/prism/eclipse/updateSite/site.xml
        ```

        The *path* specifies the location of the files.

     d. Click **OK** .

4. In the **Update sites to visit** dialog, select the **Build Forge Connector Update Site** check box, then click **Finish** .

5. The system displays a list of available features in the **Search Results** dialog. Select all the offered features, then click **Next** .

   ### Note

   The Reflector plug-in requires the Frequency plug-in. It will not run if installed alone.

6. Read the license agreements, then select **I accept the terms in the license agreements** , then click **Next** .

7. Select the location where you want to install the features.

   To add a new location, click **Change Location** , then browse to the location you want.

8. Click **Finish** .

9. If a **Feature Verification** dialog is shown, click **Install** .

The dialog appears because the plug-ins are unsigned features. The dialog is shown once for each feature installed unless you selected **Install All** .

10. You are asked to restart Rational Team Concert to make the changes take effect. Click **Yes** .

11. Update the console.

    In **Administration** → **System** , change this setting:

    - **Console URL** set to the full URL for the console, including https:// protocol and port number. Example if the console is running on mymachine.mydomain.com on the default HTTPS port of 443:

      ```
      https://mymachine.mydomain.com:443
      ```

Configuring the Rational Team Concert adaptor

An adaptor to connect Build Forge to Rational Team Concert source repository is installed automatically with Build Forge. You can configure individual Build Forge projects to access Rational Team Concert source as follows.

1. Define a Server resource.

   - Enter the **Name** for this server resource.
   - Enter the **Host** : this is the fully qualified domain name of the host where Rational Team Server is running.
   - Enter values for other properties as appropriate.

   ### Note

   The RTC tool `scm` must be on the PATH for the user profile or the startup profile for the RTC server. The adaptor uses `scm` to access source files.

2. Set up adaptor links for each project that uses the Rational Team Concert repository.

   The following instructions describe how to set up an adaptor link for a project.

   a. In the console, go to **Projects** → **Adaptor Links**

   b. Click **New Adaptor Link** . Set its properties in the **Details** tab, then click **Save** .

      - State - select **Active**
      - Name - enter the name for the adaptor link.
      - Adaptor - choose JazzSCM.
      - Project - choose the project to which to apply the adaptor link.

- Environment - choose the environment to use for this adaptor link. It must be an existing environment.

c. Click the adaptor link you just created.

d. Click **Populate Env** . This step populates the specified environment with the variables defined in the adaptor (JazzSCM.xml).

e. Click **Save** .

3. Update the variables provided by the adaptor.

   In the environment you set up and populated, edit the variables provided by the adaptor.

   Four environment variables are set in the adaptor:

   - Current_Date - sets the current date. It is used to apply timestamps. Do not change this definition.
   - Last_Run - automatically updated by the system. Do not change this definition.
   - Directory_Path - sets the location for source files retrieved from the repository. It is set to C:\temp by default. Change this directory to the temporary directory you want to use.

     The directory is not cleaned up by default after a job runs. Delete old directories from jobs that have already run.
   - Jazz_Server - sets the location of the Rational Team Concert server. It is set to $BFServer by default. You must change this setting to the **Name** property of the Server resource that points to the Rational Team Concert Server.

   a. In the console, go to **Environments** .

   b. Select the environment that is used by the adaptor link.

   c. Select and edit the variables you want to change.

   d. Click **Save** .

You can now run the project. Each time it runs, it connects to the Rational Team Concert repository. It updates source files that have changed at the RTC server since the last time the project ran.

Additional resources:

- The JazzSCM adaptor is located in `<bfinstall>/interface/JazzSCM.xml`. You can open it with a text editor or XML reader.
- The SCM commands used by the adaptor are documented in the Rational Team Concert documentation. Use them and others when testing the connection to the server and testing commands to be used by the adaptor.

Websphere integrations

This section describes ways to integrate Build Forge® with Websphere products:

- Using Websphere Application Server rather than Apache Tomcat to run the Build Forge services and Quick Reports
- Using IBM HTTP Server (IHS) rather than Apache as the web application server

Using Websphere Application Server instead of Apache Tomcat

You can manually configure Websphere Application Server 6.1 to be the application server for the Build Forge service layer applications instead of using the provided Apache Tomcat application server.

Prerequisite: due to restrictions in the license server, the Build Forge console and WAS must be running on the same operating system and hardware platform.

Important

After configuring Websphere Application Server for Build Forge, the Build Forge services layer must be running on the Websphere Application Server before you start the Build Forge engine and open the Build Forge Management Console.

Perform the steps in the following procedure from the Websphere Admin Console.

1. Open the Websphere Admin Console.

   The URLs for the console are:

   - http://<*was_host*>:<*was_port*>/ibm/console; 9060 is the default port
   - https://<*was_host*>:<*was_port*>/ibm/console; the default port is 9443. Use this URL if Websphere Administrative Security is enabled.

2. Create a new variable, RBF_JDBC_DRIVER_PATH.

   Create the variable in **Environment > Websphere Variables** . Its scope should be the WAS node and server. Set the value to the directory that contains your database driver JAR files.

3. Save the change to the master configuration.

4. Stop and restart the Websphere server to make the new variable available.

5. Create a new shared library, RBF_JDBC_LIBRARY.

   Create the library in **Environment > Shared Libraries** . Add the JAR file names for your JDBC device driver, using the RBF_JDBC_DRIVER_PATH that you just created.

   The following example is for a MySQL database driver:

   ```
   ${RBF_JDBC_DRIVER_PATH}\mysql-connector-java-5.0.5-bin.jar
   ```

With Unix or Linux, in this example, use the forward slash (/) instead of backslash (\).

6. Save the change to the master configuration.
7. Install the Build Forge application WAR file.
   a. Open **Applications > Enterprise Applications** .
   b. Click **Install** .
   c. Browse to the `rbf-services.war` file in the Build Forge installation directory. Use `/rbf-services` as the context root.
   d. Click **Next** , and clear the following check box if it is selected:
      - Create MBeans for resources
   e. Click **Next** until you see a **Finish** button, then click **Finish** .
   f. Click **Save** at the bottom of the installation text.
8. Set RBF_JDBC_LIBRARY as a shared library reference.
   a. Open **Applications > Enterprise Applications** .
   b. Click the **rbf-services.war** link.
   c. Click **Shared library references** .
   d. Select the **A Services Layer Authentication** box.
   e. Click **Reference Shared Libraries** .
   f. Add RBF_JDBC_LIBRARY to the list.
9. Under **Manage Modules** , select **A Services Layer Authentication** and locate **Class loader order** in the drop down box. Change that value to **Classes loaded with application class loader first** .
10. Save the changes to the master configuration.

Starting the Build Forge services layer

Start Build Forge services layer (rbf-services.war) on the Websphere Application Server, as follows:

1. Choose **Applications > Enterprise Applications** .
2. Select **rbf-services.war** .
3. Click **Start** .

Enabling application support for Java 2 security in Websphere application server

If you are running Websphere Application Server with Java 2 Security enabled, you must configure the Build Forge services layer to use it.

Perform the following steps from the Websphere Admin Console:

1. Open **Applications > Enterprise Applications** .
2. Select **rbf-services_war** and click **Update** .
3. Select **Replace or Add a Single File** .
4. In the **Specify the path beginning with the installed application archive file to the file to be replaced or added** , enter META-INF/was.policy.
5. Select **Local file system** and browse to the directory `<`*`bfinstall`*`>/samples/projects/was.policy`. Click **Next** .
6. Click **OK** .
7. Save the changes to the master configuration, and then stop and start the application.

Prerequisite to enable WAS for Build Forge SSL and password encryption

If you intend to enable SSL or password encryption for Build Forge, you must first do the following:

1. Set the a property on the WAS server that hosts the Build Forge services component.
   - Windows

     ```
     -Dcom.buildforge.client.config=<bfinstall>/Platform/bfclient.conf
     ```

   - UNIX or Linux

     ```
     -Dcom.buildforge.client.config=<bfinstall>/bfclient.conf
     ```

2. Restart the WAS server.

You can now configure Build Forge to use the SSL or password encryption features.

Using IBM HTTP Server instead of Apache HTTP Server

Use the information in this section to configure IBM HTTP Server (IHS) for use with the Management Console instead of the Apache HTTP Server that is installed and configured by Installation Manager.

Prerequisite: due to restrictions in the license server, the Build Forge console and IHS must be running on the same operating system and hardware platform.

To use IBM HTTP Server (IHS), you must modify your installation as follows:

1. Modify your IBM HTTP Server configuration files to point to the Build Forge web application
2. Install PHP 5.2.4 and configure the PHP modules required for your Build Forge database and for OpenSSL.

Edit the IBM HTTP Server configuration file

Edit the IBM HTTP Server httpd.conf file to add information for the Build Forge web server.

1. Locate the httpd.conf file for the IBM HTTP Server (IHS) in the conf directory of your server installation.
2. Modify the DocumentRoot setting to point to the Build Forge web application, as shown in the example. In this example, the Build Forge installation directory is /usr/local/buildforge.

   ```
   <VirtualHost *:80>
        ServerAdmin build@yourdomain.com
        DocumentRoot /usr/local/buildforge/webroot/public
        ServerName ausbuild01.yourdomain.com
        ServerAlias build.yourdomain.com mc.yourdomain.com #optional server aliases
        ErrorLog logs/ausbuild.error_log
        CustomLog logs/ausbuild.access_log common
   </VirtualHost>
   ```

3. If necessary, change the IHS port number. The default port number is 80. Make any other necessary changes to httpd.conf.

Install and configure PHP for IHS

PHP is not installed with the IBM HTTP Server. You must install PHP 5.2.4 and configure it to point to the `httpd.conf` file for IHS.

- During PHP installation, select and install the PHP extensions for the database type that you are using as the Build Forge database.
- Optional: if you want to use OpenSSL to provide security for the Build Forge web server, you must install the PHP OpenSSL module provided by Build Forge and recompile PHP. Copy `php_openssl.dll` from the `<`*`bfinstall`*`>/Apache/php/ext` directory to the appropriate directory in IHS.

Configuring SSL for IHS

In addition to the normal SSL setup for IHS, there are additional requirements for it to work with Build Forge.

1. Convert the Build Forge keys from PKCS12 to CMS.

   Use the GSKIT tool. In gsk7bas/bin, run the following command (line breaks are added for clarity):

   ```
   gsk7cmd -keydb
           -convert
           -db bfinstall/Platform/keystore/buildForgeKeyStore.p12
   ```

```
        -pw buildForgeKeyStore_password
        -old_format pkcs12
        -new_format cms
```

2. Store the password in a stash file.

   IHS uses this file to get the password during startup. Without it, IHS prompts for the password. Use the GSKIT tool. In gsk7bas/bin, run the following command (line breaks are added for clarity):

   ```
   gsk7cmd -keydb
           -stashpw
           -db bfinstall/Platform/keystorebuildForgeKeyStore.kdb
           -pw buildForgeKeyStore_password
   ```

3. Modify httpd.conf.

   Include the following entries:

   ```
   Keyfile "bfinstall\keystore\buildForgeKeyStore.kdb"
   SSLStashFile "bfinstall\keystore\buildForgeKeyStore.sth"
   ```

Consult IHS documentation on setting up SSL for more information.

Security features

This section describes ways to enable security features in Build Forge® :

- Single sign-on (SSO)
- Enabling HTTPS and SSL
- Enabling password encryption
- `bfclient.conf` file for security configurations

These features are enabled through a combination of selections in the management console at **Administration** → **Security** and manual setup of configuration files. This section includes a reference section on `bfclient.conf`, a configuration file used to enable security features.

Note

this section is not for users who run Build Forge on z/Linux. See Installing the Management Console on SUSE Linux on System z for information about security features available in Build Forge on z/Linux.

Implementing single sign-on

A single sign-on framework is provided with Build Forge

Single sign-on is an authentication scheme that allows users to access an application without entering a user name and password each time. Build Forge ships a framework that can be used in combination with a third-party HTTP interceptor to implement single sign-on.

About the single sign-on framework

The Build Forge SSO framework provides the capability to integrate with many SSO solutions on the market. The SSO framework is interceptor-based, meaning that it intercepts an HTTP request and provides methods for handling it. You can write custom interceptors to receive and validate security artifacts in the HTTP request. In particular, the interceptor can set tokens in the HTTP response and then look for those tokens in a successive request.

Two SSO solutions are provided with Build Forge:

- Interceptor for SPNEGO (Simple and Protected Negotiation Protocol). See “Implementing single-signon using SPNEGO in an Active Directory domain” on page 108.
- Interceptor for an integration with Websphere SSO. See “Integrating with WAS security using a custom interceptor” on page 114.

SSO framework methods

An SSO interceptor is a Java class that implements an interface used by the Build Forge SSO framework:

`com.buildforge.services.server.sso.ISSOIntercaptor`

It is located in the services layer component:

`<`*`bfinstall`*`>/Apache/tomcat/webapps/rbf-services/WEB-INF/classes`

The interface provides the following methods.

initInterceptor	Called when the interceptor is loaded. A map of configuration properties is passed to the `initInterceptor()` method. Configuration properties are created in the Build Forge console at **Administration** → **Security** → **SSO** .
isTargetInterceptor	Reviews attributes in the inbound request to determine if this interceptor needs to act on them. If so, the interceptor is responsible for authenticating the request with the `authenticateRequest()` method. Otherwise this interceptor is skipped. Interceptor selection assumes that multiple interceptors are configured in running. They are addressed in order.
authenticateRequest	Authenticates the request using data in the request. It uses a response attribute to send data back to the client.
logoutRequest	Cleans up any user-related security information after the request is handled.

Interceptor configurations and ordering

Interceptor configurations are defined in **Administration** → **Security** → **SSO** . The following configurations are shipped with Build Forge:

- Form SSO Interceptor - active by default, implements a simple login form.
- SPNEGO SSO Interceptor - inactive by default, implements SPNEGO to perform authentication.

After you implement an interceptor class and place it in the Build Forge Apache Tomcat application server, you configure a new SSO configuration here. The class is one property of the SSO configuration.

The order of this list determines the order in which interceptors are consulted in order to handle requests. You can configure multiple interceptors to handle requests. During a login, each interceptor is consulted in order. The interceptor that handles the request is the first active interceptor whose attributes are appropriate for the attributes in the request. Only one interceptor handles the request. It is always the first one that responds true for `isTargetInterceptor`.

Note

The Form SSO Interceptor should be left active in order to provide a fallback in the case of error. Custom interceptors should be placed in the list before it.

Adding a custom SSO interceptor

To create a custom interceptor in Build Forge, do the following:

1. Create a custom Java class.

   The class must implement the `ISSOInterceptor` interface.

2. Deploy the custom class to the services layer component WAR.

   a. Create a JAR file containing the compiled custom SSO interceptor class.

   b. Copy the JAR file to the Build Forge services layer component in the following location:

      `<`*`bfinstall`*`>/Apache/tomcat/webapps/rbf-services/WEB-INF/classes`

   c. Unzip the JAR file in this directory. The SSOManager looks for the class to load here.

   d. Restart Build Forge.

3. Optional: define an environment. This environment can be passed to the `initInterceptor()` method as a Properties object.

   a. In the Management Console, go to **Environments** .

   b. Click **Add Environment** .

   c. Defined all of the properties needed by the SSO interceptor in order to initialize.

4. Add the SSO interceptor to Build Forge:

   a. In the Management Console, go to **Administration** → **Security** → **SSO** .

   b. Click **Add SSO Configuration** . Fill in its properties:

      - **Name** - enter a name for the SSO configuration.
      - **Active** - set to Yes. Active configurations are all accessed during an authentication request. They are accessed in the order in which they appear in this panel.
      - **Java Class** - enter the full package name of the class. A given class can be assigned to only one SSO interceptor.
      - **Environment** - if you defined an environment for use with this SSO interceptor, select it.

   c. Click **Save** .

   Your SSO interceptor now appears in the list.

5. Order the SSO configurations. Click the icon to the left of your SSO Interceptor, the select **Move to Top** .

   During a request, active SSO configurations are accessed in the order in which they appear in this panel. Your configuration must be placed before the **Form SSO** configuration, because it is active by default and always returns true when accessed. The **SPNEGO SSO** configuration is inactive by default.

Example authenticateRequest implementation

The following example is taken from the WebSphere SSO Interceptor, which is used to integrate WebSphere security with Build Forge.

The interceptor uses reflection to find the WebSphere class WSSubject. The class has a `getCallerPrincipal` method to return the principal used to log in to the AuthServlet. The AuthServlet needs to be protected before WAS will authenticate with it.

Other methods are available that can return even more information. Similar methods are available to work with any application server.

```
public Result authenticateRequest
      (Request requestAttributes, Response responseAttributes)
       throws SSOException {

  Result result = null;

try {
  Class<?> cl =
    Class.forName(“com.ibm.websphere.security.auth.WSSubject”);
      Method theMethod = cl.getMethod("getCallerPrincipal",
        (Class[])null);
    String principal = (String)theMethod.invoke((Object[])null,
          (Object[])null);
if (principal != null
      && principal.length() > 0
      && !principal.equals("UNAUTHENTICATED")) {
  result = new Result(Result.UseridOnlyOID, domain, principal);
 responseAttributes.setStatus(HttpServletResponse.SC_OK);
} catch (Exception e) {
  throw new SSOException(e);
}

return result;
}
```

During the implementation of `authenticateRequest`, you must set a response status before returning:

- If you do not need to perform any redirection and the information found is good, you can return the following:

  ```
  responseAttributes.setStatus(HttpServletResponse.SC_OK);
  ```

- If there is not enough information in the request to continue a valid login, return the following:

  ```
  responseAttributes.setStatus(HttpServletRespons,SC_FORBIDDEN);
  ```

- If you need to perform a redirection to gather additional information, return the following:

  ```
  responseAttributes.setStatus(HttpServletRespones.SC_MOVED_TEMPORARILY);
  responseattributes.sendRedirecct(url);
  ```

There are additional status values that can be used. See the JavaDoc for `HttpServletResponse`.

Recovering from a login error

If your custom interceptor does not work correctly when tested, the most likely issue is authentication. You see an error page with the following information:

```
Build Forge Error

     Access is denied to the Buil Forge console

     "Error authenticating:
     com.buildforge.services.common.api.APIException - API:
     Authentication Error."

     Please click here to try the same type of login again
     or click here to force a form login (user ID/password).
```

You have two options for recovery:

- Retry the login. It goes through the list of configured interceptors again in the same way.
- Force a form login. This choice bypasses the custom interceptor and uses the form login page.

Method source listing

The following comments and source listings provide more information about the methods in the `ISSOInterceptor` interface.

`initInterceptor`

```
 /**
  * This method is called when the interceptor is loaded.  A map of the configuration
     properties is passed into the init method.  You can create the configuration
    properties from a BuildForge Environment and associate it with the SSO configuration.

  *
  * @param initializationProps used to configure the implementation
  * @return true if successful, false if an error should be reported.
  * @throws SSOException if the initialization fails
  **/
 public boolean initInterceptor (Properties initializationProps) throws SSOException;
```

`isTargetInterceptor`

```
 /**
  * This methods will review the attributes in the requestAttributes Map
     to determine if there is something that this interceptor should
    act on.  If the interceptor return is "true", then the interceptor will be responsible
 for
     authenticating the request and the authenticateRequest method is
     invoked.   If the interceptor return is "false", then this interceptor is
     skipped and the next isTargetInterceptor in the list will be called.
     Ordering of the interceptors during the configuration will return
     which interceptor has the first shot at authenticating a request.
  *
  * @param requestAttributes attributes found in the inbound request
   * @return true if this interceptor will authenticate the request, false if it will not.

  * @throws SSOException
  *
  **/
 public boolean isTargetInterceptor(Request requestAttributes) throws SSOException;
```

`authenticateRequest`

```
 /**
  * This method is called on an interceptor that returns true for the
     isTargetInterceptor method.  The Request will contain data used
     to perform the authentication.  The Response is for the interceptor
     to send information back to the client.  The Result returned will contain
     the following information if the status code is 200:
  *
  * OID:  an object identifer of the SecurityContext that can process token
     information stored in this map when going to an Agent.
  * Domain:  a valid BF domain name or <default> if not known
     (the username must be valid in the configured realm).
  * Username:  a valid BF username.   This will be used to lookup BFUser attributes
     that are used in checking authorization policy.
  * @see com.buildforge.services.common.security.context.Result
  *
  * @param requestAttributes attributes found in the inbound request
  * @param responseAttributes sent back in the outbound response
  * @return com.buildforge.services.common.security.context.Result - result information
```

```
that tells BF how to handle the authentication request.
  * @throws com.buildforge.services.server.sso.SSOException
  **/
 public Result authenticateRequest(
   Request requestAttributes,
   Response responseAttributes)
  throws SSOException;
```

logoutRequest

```
 /**
  * This method is called to logout a request.  The first interceptor that
    returns true for the isTargetInterceptor method will perform the logout.
    The main point is to clean up any user-related security information that
    should not be kept.  The interceptor can inspect the request and response
    objects to determine what needs to be removed.
  *
  * @param requestAttributes attributes found in the inbound request
  * @param responseAttributes sent back in the outbound response
  * @return boolean - true if request redirect to exit page, false if redirect to login
page.
  * @throws com.buildforge.services.server.sso.SSOException
  **/
 public boolean logoutRequest(
   Request requestAttributes,
   Response responseAttributes)
  throws SSOException;
```

Implementing single-signon using SPNEGO in an Active Directory domain

A Simple and Protected GSS-API Negotation (SPNEGO) mechanism is provided to implement single sign-on in Active Directory domains

This task requires the following elements in your network:

- Active Directory domain
- Directory server host name
- Kerberos Key Distribution Center (KDC) host name
- Build Forge installation on a host in the Active Directory domain
- Client host in the Active Directory domain
- Kerberos configuration files on each client
- Windows Server 2003 SP2 resource toolkit installed on the directory server host
- Supported web browser

Note

Internet Explorer 6 is not supported for use with SPNEGO. Use another supported browser. See Web client requirements.

The following procedures include examples based on the following setup:

- `mycompany.com` is the name of the TCP/IP domain used by all hosts in the domain.
- `ITDEV.COM` is the name of the Active Directory domain
- `it_directory.mycompany.com` is the host where the directory server runs. It also runs the Kerberos KDC.
- `it_domain.mycompany.com` is the host where the Active Directory domain controller runs.
- `it_buildforge.mycompany.com` is the host where Build Forge is installed
- `bfuser` is the domain user name for the Build Forge system
- `happy_user` is the domain user name for an example user who will use SSO in a web browser to access Build Forge

Perform the following tasks to implement the SPNEGO SSO in an Active Directory domain and KDC. Each is given a section with detailed procedures.

Note

The SPNEGO interceptor can be used with KDCs other than Active Directory.

1. Set up Active Directory users and service principals.
2. Set up Kerberos files.
3. Configure Build Forge to use Active Directory and SPNEGO.
4. Configure browser clients for secure access
5. Access Build Forge through SSO

Setting up Active Directory users and service principals

The Build Forge server and Build Forge clients must be set up in an Active Directory domain.

Support Tools for Windows 2003 SP2 are required for the following procedure. They contain the `setspn` command, which is required to set a service principal in Active Directory. Install Support Tools from the Windows Server 2003 product CD or the Microsoft Download Center.

When the Build Forge client and server are in an Active Directory domain, a user generates a Kerberos credentials token when logging into a Windows host. When the user then attempts to access the Build Forge server, the SPNEGO interceptor receives the user token and validates it. The validated identity is passed to the Build Forge to perform a login through the configured Microsoft Active Directory LDAP server.

1. Log on to the domain controller host.

   In the example, the host is `it_example.mycompany.com`.

2. Add the Build Forge host to the Active Directory domain if it is not already a member.

   In this example, add host `it_buildforge` to the `ITDEV.COM` domain. The host now has a fully qualified name in the domain: `it_buildforge.ITDEV.COM`

3. Add a Build Forge user to the Active Directory domain.

   In this example, create user `bfuser`.

   Important

   - Select **Password never expires** . You may select other password management. However, you will need enter a new password for the Build Forge server each time it expires.
   - In the **Accounts** tab, select **Account is trusted for delegation**

4. If they do not exist, create user accounts in Microsoft Active Directory for all clients.

   In this example, there is one user to create, `happy_user`.

5. Create a service principal name (SPN) for Build Forge.

   In the example, the Active Directory user `bfuser` is associated with service name `HTTP/it_buildforge.mycompany.com` to create the SPN for the Build Forge server, `it_buildforge`..

   ```
   setspn -A HTTP/it_buildforge.mycompany.com bfuser
   ```

   HTTP is the service name for the Build Forge service.

Setting up files for Kerberos authentication

A startup file (Kerberos client configuration file) and a keytab file need to be set up on the Build Forge host.

1. Set up the startup file on the host where Build Forge runs.

   - Windows systems:
     - Name the file `krb.ini` and place it in `C:\winnt`. Create `C:\winnt` if it does not exist.
     - Set `default_keytab_name` to `FILE:C:\winnt\krb5.keytab`.
   - UNIX and Linux systems:
     - Name the file `krb.conf` and place it in C:\winnt.

     Set `default_keytab_name` to `FILE:/etc/krb5.keytab`.

   The following example file is set up for Windows using domain and realm settings from the example systems.

   ```
   [libdefaults]
    default_realm = ITDEV.COM
   ```

```
 default_keytab_name = FILE:C:\winnt\krb5.keytab
 default_tkt_enctypes = rc4_hmac
 default_tgs_enctypes = rc4_hmac
# kdc_default_options = 0x40800000
 forwardable  = true
 renewable  = true
 noaddresses = true
 clockskew  = 300
[realms]
 ITDEV.COM = {
  kdc = it_directory.itdev.com:88
  default_domain = mycompany.com
[domain_realm]
 .mycompany.com = ITDEV.COM
```

Note

Tokens will not work if the clock skew between client hosts and the Build Forge server host is more than 300 seconds. Set the time, date, and time zone to within skew limits on the client and server hosts.

2. Set up a Kerberos keytab file.

   The keytab file is used by the Build Forge server to validate Kerberos tokens when a client attempts to access the Build Forge server URL. Use the `ktpass` command on the Domain Controller host to create the file. The `ktpass` command is included in the prerequisite Windows resource toolkit. The following example uses the principal name of the Build Forge service and the Active Directory user name set up for Build Forge in the example scenario. Substitute your own password for `-pass Rat1onal`. Line breaks are shown in the example for clarity. Do not use them in your ktpass command.

   ```
   ktpass -out c:\it_buildforge.keytab
   -princ HTTP/it_buildforge.mycompany.com@ITDEV.COM
   -mapuser bfuser -mapop set
   -pass Rat1onal /crypto RC4-HMAC-NT /rndpass /ptype KRB5_NT_SRV_HST
   ```

   Rename it_buildforge.keytab to krb5.keytab and put it on the Build Forge host in the directory that contains the Kerberos startup file.

   - Windows: `C:\winnt\`
   - UNIX and Linux: `/etc`

Configuring Build Forge to use Active Directory and SPNEGO

1. In Build Forge, set up LDAP to point to the Active Directory domain controller.

   a. In Build Forge, click **Administration** → **LDAP**

   b. Set up access to your domain controller by creating a new LDAP configuration and setting properties as follows.

      - Name: set to the name of your Active Directory domain. In the example environment this would be `itdev`.

- Admin DN: set to an administrator user in the domain.
- Map Access Groups: No
- Host: set to the IP address of the domain controller host.
- Bind User Account: Yes
- Protocol: LDAP
- Display Name: displayname
- Distinguished Name: distinguishedname
- Group Name: memberof
- Mail Name: displayname
- Search Base: `on=users,do=`*`domainname`*`,do=`*`domainextension`* . In the example environment this would be `on=users,do=itdev,do=.com`
- Unique Identifier: `sAMAccountNames=%`

c. Click **Make Default** .

This configuration needs to be the default LDAP configuration.

2. Set Build Forge environment variables for SPNEGO.

a. In Build Forge, go to **Environments** → **Environment for SPNEGO SSO**

b. Set `bf_spnego_service_name` to HTTP

This matches the service principal name.

c. Set `bf_spnego_server_name` to `it_buildforge.mycompany.com`, the fully qualified host name for the Build Forge server host.

If this variable is not set, INetAddress APIs attempt to locate the host name.

d. Set `bf_spnego_realm` to ITDEV.COM, the Kerberos realm name.

If this variable is not set, the value in the Kerberos startup file is used.

3. Enable the SPNEGO interceptor.

a. In Build Forge, go to **Administration** → **Security** → **SSO** → **SPNEGO SSO Interceptor** .

b. Set the Active property to Yes, then click **Save** .

c. In **Administration** → **Security** → **SSO** , move **SPNEGO SSO Interceptor** to the top of the list.

Use the **Move to Top** selection in the SSO Options menu for SPNEGO SSO Interceptor, then click **Save** .

Configuring client browsers for SSO

Client browsers must have security settings set up to use SPNEGO.

Use the client setup instructions for the browser used to access Build Forge, either Microsoft Internet Explorer or Mozilla Firefox.

- For Internet Explorer, do the following.

  1. Log in to the Active Directory domain.

     In the example configuration, you would log in to itdev.com.

  2. Open Internet Explorer.
  3. Click **Tools** → **Internet Options** → **Security**
  4. Select **Local intranet** , then click **Sites** .
  5. In the **Local intranet** dialog, check **Include all local (intranet) sites not listed in other zones** , then click **Advanced** .
  6. Add the host where Build Forge runs to the **Web sites** list, then click **OK** .
  7. Click **OK** to close the **Local intranet** dialog.
  8. In the **Internet Options** window, click the **Advanced** tab.
  9. Scroll to the Security group and select **Enable Integrated Windows Authentication (requires restart)** if it is not already selected.
  10. Click **OK** .
  11. Restart Internet Explorer.

- For Mozilla Firefox, do the following.

  1. Log in to the Active Directory domain.

     In the example configuration, you would log in to itdev.com.

  2. Open Firefox.
  3. Type **about:config** in the address field.
  4. In the Filter box, type **network.n** .

     The list updates itself.

5. Double-click **network.negotiate-auth.trusted-uris** .

   Enter the list of trusted domains. It should include the directory server host and the Build Forge server host (`it_directory.mycompany.com` and `it_buildforge.mycompany.com`, in the example), then click **OK** .

6. Set up delegation.

   Double-click **network.negotiate-auth.delegation-uris** and enter the list of sites to which the browser can delegate user authentication.

Access Build Forge through SSO

Enter the server URL to test login through SSO.

1. Log on to a host that is in the Active Directory domain, using a user name that is in the Active Directory list of users.

2. Open your browser.

3. Enter the URL for the Build Forge server host.

   Using the example configuration, this would be `http://it_buildforge.mycompany.com`. If SSO is configured correctly, you see the Build Forge management console.

4. Verify that the user name shown on the upper right of the Build Forge console matches the client's Windows login name.

Integrating with WAS security using a custom interceptor

This section describes how to create an SSO interceptor to integrate with WebSphere Application Server (WAS) security.

The provided Form SSO Interceptor authenticates users with form-based login page. The following is an example of how to create a custom SSO interceptor. The custom interceptor uses a custom interceptor class.

The interceptor class accesses WAS to obtain authenticated user credentials. Once those credentials are obtained, they are cached. Subsequent logins use the cached credentials.

Prerequisite: A user must be configured in WAS with LDAP user credentials.

Note

You must make the Build Forge LDAP domain containing the WAS users the "default" LDAP server. To do this, go to this Build Forge LDAP domain, and select "Make Default."

Protecting your authorization service (AuthServlet)

Build Forge normally runs its services as an application in the provided Apache Tomcat application server.

The following are instructions to configure it to use WAS 6.1 instead of Tomcat. Follow the instructions in the Installation Guide for the section "Using Websphere Application Server instead of Apache Tomcat " on page 98, with one exception: the `rbf-services.war` file contains a file named `web.xml` that you need to extract and modify in order to add a security constraing. Before installing this application under WAS, the `war` file will need to be regenerated after modifying it to use the protected version of this file

In order to do this, follow these instructions:

1. Navigate to the directory containing your `rbf-services.war` file (the `webapps` directory in your Tomcat server root).

   Copy this file to a temporary location, such as `C:\rbf`.

2. Expand the WAR file.

   From command line, run the command: `%IBM_JAVA_HOME%\jar –xvf rbf-services.war` to expand the contents of the `war` file.

   Note

   Java must be available and the `IBM_JAVA_HOME` environment variable must already have been created.

3. Save the `rbf-services.war` file to retrieve later:

   a. Windows: `copy rbf-services.war rbf-services.war.bak`

   b. UNIX or Linux: `cp rbf-services.war rbf-services.war.bak`

4. Find the `web.xml` file in the `WEB-INF` directory (from the files expanded from the `war` file). Edit this file to add a security context. For example, add the following lines at the end of the file just before the `</web-app>` tag:

```
<security-constraint id="SecurityConstraint_1">
         <web-resource-collection id="WebResourceCollection_1">
            <web-resource-name>/*</web-resource-name>
            <url-pattern>/AuthServlet/*</url-pattern>
            <http-method>GET</http-method>
            <http-method>POST</http-method>
            <http-method>PUT</http-method>
            <http-method>DELETE</http-method>
         </web-resource-collection>
         <auth-constraint id="AuthConstraint_1">
            <description>myconstraint:+:</description>
            <role-name>User</role-name>
         </auth-constraint>
         <user-data-constraint id="UserDataConstraint_1">
            <transport-guarantee>NONE</transport-guarantee>
         </user-data-constraint>
      </security-constraint>
      <login-config id="LoginConfig_1">
         <auth-method>BASIC</auth-method>
         <realm-name>
                                 full-qualified-domain
                              </realm-name>
```

```
</login-config>
<security-role id="SecurityRole_1">
   <role-name>User</role-name>
</security-role>
```

Note

The <auth-method> can be any J2EE auth-method supported by WAS. The most common auth-method is FORM, which requires additional configuration parameters. Refer to your WebSphere documentation for instructions on configuring FORM in your application.

5. From command line, regenerate the `war` file (called from the same directory that it was extracted to) by running the following command:

   ```
   %IBM_JAVA_HOME%\jar –cvf  rbf-services.war
   ```

   You should now have a new version of `rbf-services` that has been modified to protect the AuthServlet with J2EE constraints. Complete the installation instructions for running with WAS and install this version of the `rbf-services.war` via **Applications ->Install New Application** . Make sure that Build Forge is not running while configuring WAS.

   Once this is installed, go to **Applications ->Enterprise Applications** . Click the application name to configure it. Under **Detail Properties** , click the link entitled **Security role to user/group mapping** . Select the **All Authenticated** checkbox for User. After making this change, be sure to save to the master configuration.

   Application Security also needs to be enabled under WAS. To do this, go to **Security ->Secure administration, applications, and infrastructure** . Make sure that **Enable application security** is checked.

   At this point, restart the WAS server, then restart the Build Forge server.

Note

The `rbf-services` should now start as part of the WAS startup process, so WAS needs to be started before Build Forge.

Creating a new SSO configuration

Create a new SSO configuration to use the interceptor.

1. In the Build Forge console, go to **Administration ->Security ->SSO** .
2. Click **Add SSO Configuration** .
3. Set properties for the configuration.
   - **Name** - Enter a name for this configuration.
   - **Java Class** - Enter `com.buildforge.services.server.sso.was.WebSphereSSOInterceptor`
   - **Active** - Select Yes.

4. Click **Save** .
5. Move this configuration to appear first in the list.

   In the menu to the left of the configuration name, select **Move to Top** .

Running the SSO custom interceptor

You may now log in using your new configurations.

This customized SSO interceptor will now allow WAS security techniques to authenticate the user, via an AuthServlet request to be passed to Build Forge as a user.

1. Open your web browser. Enter the address `http://localhost`.
2. Instead of the Build Forge login form, you will now see an authentication page. Enter your user credentials and hit **Enter** .
3. After authentication, note that login should occur automatically.
4. After logging out, it will display the default `jsp` page instead of the login form. Subsequent logins will be automatic as long as the user is still authenticated.

Reverting to form-based SSO

You may revert back to using the SSO login form.

In order to reconfigure the systems to use the Form Login, under WAS you must uninstall `rbf-services` and reinstall the original `rbf-services.war` file. Under Build Forge, ensure that the form-based SSO interceptor is enabled and listed as the top item. Disable the Custom WAS interceptor. WAS and Build Forge would need to be restarted for these changes to take place.

1. Uninstall `rbf-services` and reinstall the original `rbf-services.war` file.
2. Under Build Forge, ensure that the form SSO interceptor is enabled and listed as the first item (see "Build Forge SSO security configurations," above).
3. Disable the Custom WAS interceptor.
4. Restart WAS.
5. Restart Build Forge.

Enabling SSL and HTTPS

Configuring the Build Forge system to use SSL and HTTPS increases the security of the system. SSL includes endpoint authentication and data encryption.

By default the Build Forge system uses SSL only for the form login, which uses the authentication servlet on Apache Tomcat. Enabling additional SSL protection throughout the Build Forge system requires the following setup:

1. Enabling SSL on the Apache server. This step is needed only if you did not specify that SSL be enabled during installation.
2. Enabling SSL for client and internal communications
3. Enabling SSL for agents

Note

If you are integrating with Websphere components, be sure that the prerequisites for SSL support are met.

- For integration with Websphere Application Server: see Using Websphere Application Server instead of Apache Tomcat.

About default login security

Build Forge includes login security by default. When a user logs in, the request is redirected to an authentication servlet. You also have the option of installing a self-signed certificate to enable SSL.

Certificate messages about self-signed certificate

If you have Build Forge install a self-signed certificate, users accessing the system through a security-enabled browser get warning messages about the certificate.

To prevent those warnings, distribute the certificate to users for installation in their browser. The specifics of installing the certificate vary by browser. Consult the browser documentation.

The certificate is located in `<`*`bfinstall`*`>/keystore`.

Disabling default login security

To disable the authentication servlet, do the following:

1. Stop Build Forge if it is running.
2. Edit `<bfinstall>/buildforge.conf` to specify HTTP and port 8080 in communications with the services layer.

   Change this line:

   ```
   services_url https://hostname:8443/rbf-services
   ```

   Change the line to the following:

   ```
   services_url http://hostname:8080/rbf-services
   ```

3. Edit the services layer configuration file to turn off forced SSL. Edit `<`*`bfinstall`*`>/Apache/tomcat/webapps/rbf-services/WEB-INF/web.xml`. Change the `ForceHttps` setting to false.

   ```
   forceHttps=false
   ```

4. Start Build Forge.

Note

If the authentication servlet is disabled, user credentials are communicated in clear text over the network. This is a security risk.

About SSL and Build Forge components

Build Forge components are set up by default to use certain ports and security settings when SSL is enabled.

Default SSL Setup

Enabling SSL is relatively simple when you use the default certificates. Procedures in this section are based on that scenario.

However, it is generally not wise to use the same certificate (private key) on each system. If the private key on one system is compromised, then the entire infrastructure could be compromised. The chances of compromise can be reduced by enforcing physical security.

A more secure system uses a certificate for each process. In Build Forge that means you do the following:

- Create a certificate for every agent.
- Create a certificate for every engine. This is applicable when redundancy is set up. See Configuring redundancy.

This setup requires additional certificate management. You have choices:

- You can use a CA (Certificate Authority) for generating certificates. Doing this reduces the number of signer exchanges.
- You can ensure that every trust store or CA store has the signers needed for making connections.

The following sections identify the interfaces in the Build Forge system where SSL security is enforced.

Client interfaces

Users access the Build Forge system through client interfaces.

Web Client to Build Forge

Web clients access Build Forge through its Apache web server. When SSL is enabled and a security-enabled web browser is used, the following interfaces are used.

- **Apache web server port 443**

  Web clients access Build Forge through its URL. When SSL is enabled, the URL is the following:

`https://`*`host/`*

The *host* is the host where Build Forge runs. If you set up a port other than 443 for secure access to Apache, users must also specify the port:

`https://`*`host`*`:`*`port/`*

Web clients are redirected to an authentication servlet running on Apache Tomcat server.

- **Apache Tomcat application server port 8443**

  An authentication servlet accepts login credentials and authenticates the user. The servlet encrypts the credentials so they never appear in clear text over the wire.

The configuration for the listener port used by the Apache Tomcat servlet is managed through a configuration file. It is located in `<`*`bfinstall`*`>/Apache/tomcat/conf/server.xml`. Locate the following connector configuration.

```
<Connector port="8443" maxHttpHeaderSize="8192"
algorithm="IbmX509"
    maxThreads="150" minSpareThreads="25"
maxSpareThreads="75"
    enableLookups="false" disableUploadTimeout="true"
    acceptCount="100" scheme="https" secure="true"
    clientAuth="false" sslProtocol="SSL_TLS"

keystoreFile="C:\BuildForge71.536\keystore\buildForgeKeyStore.p12"

    keystorePass="password"
    keystoreType="PKCS12"

truststoreFile="C:\BuildForge71.536\keystore\buildForgeTrustStore.p12"

    truststorePass="password"
    truststoreType="PKCS12"/>
```

API Program Client to Build Forge

- **Apache Tomcat application server port 49150**

  API clients access Build Forge through its services layer component, an application running on the Apache Tomcat application server. API clients need to have a valid `bfclient.conf` file.

  The services layer component uses an SSL configuration for inbound communications. It is defined on the Build Forge console at **Administration** → **Security** → **SSL** . The default used is **Default JSSE Inbound SSL** .

Internal interfaces

Build Forge is made up of a web interface component (Apache web server and PHP), a services layer component, and an engine component. The web interface and engine components are clients of the services layer component. API program clients are also clients of the services layer component.

Services Layer inbound

Apache Tomcat application server port 49150

The services layer component uses an SSL configuration for inbound communications. It is defined on the Build Forge console at **Administration** → **Security** → **SSL** . The default used is **Default JSSE Inbound SSL** .

Services Layer client outbound

Apache Tomcat application server port 49150

The web interface component (through PHP) and the engine component both use an SSL configuration dedicated for outbound communications to the services layer component. It is defined on the Build Forge console at **Administration** → **Security** → **SSL** . The default used is **Default JSSE Outbound SSL** .

The SSL properties for the client outbound configuration and services layer inbound configuration need to be compatible for an SSL handshake to be successful. **Administration** → **Security** → **SSL** , the **Type** and **Handshake Protocol** properties must match.

Each SSL configuration has a reference keystore configurations:

- Keystore Configuration: defines the properties of a keystore that contains private certificates.
- Truststore Configuration: defines the properties of a keystore that contains trusted signers.

The configurations are specified by name. You define them in **Administration** → **Security** → **SSL** . Several defaults are provided.

External interfaces

External interfaces are those used by Build Forge to talk to external systems.

- The Build Forge engine communicates with agents.
- The Build Forge services layer component communicates with the database.

Build Forge engine to agent communications

Enabling SSL for this interface requires the following:

- Configuration of the agent. It requires a change in the agent configuration file, and placing certificates on the agent host.
- Enabling SSL communications for each Server resource that uses the agent. You do this on the console in the **Servers** panel.

	See Enabling SSL for agent communications.
Build Forge services layer component to database communications	The SSL configuration for this interface is defined in the device driver for your database.

Enabling SSL on the Apache server

Build Forge components are set up by default to use certain ports and security settings when SSL is enabled.

During installation you are asked the following question in the Web and Application Server panel: **Do you wish to use secure HTTP?**

If you answered yes, then SSL on the Apache server is enabled. Skip this section.

If you answered no, but you do want to enable SSL, then log on to the Build Forge host and configure the Apache server as follows:

- `httpd.conf` file

  Edit `<`*`bfinstall`*`>/Apache/conf/httpd.conf` to use these settings:

  ```
  Listen 0.0.0.0:443
  ServerName localhost:443
  ```

- `ssl.conf` file

  Edit `<`*`bfinstall`*`>/Apache/conf/ssl/ssl.conf` to use these settings:

  ```
  <VirtualHost *:443>
  ServerName localhost:443
  SSLEngine on
  ```

You may specify the fully qualified domain name for the host if you choose.

Enabling SSL for client and internal connections

Build Forge components are set up by default to use certain ports and security settings when SSL is enabled.

To enable SSL connections from clients to Build Forge and among Build Forge internal components, do the following in exactly the order specified:

- In the console, go to **Administration** → **Security** .
- Set **SSL Enabled** to Yes. Several additional properties are shown. *Leave them set to the default*. They can be customized later if required.
- Click **Save** . The SSL configuration is saved in the Build Forge database.

- Click **Update Master BFClient.conf** . The SSL configuration is used to update the `BFClient.conf` file. The settings must be in this file for Build Forge to use them.
- Stop Build Forge and restart. *This is required*.

Once the system is proven to work with the default settings, you may edit the properties that affect SSL.

Note

If you want to have clients use SSL but do not want to use SSL between internal clients (web interface component and engine component) and the services layer component, do the following:

- Edit the `BFClient.conf` file manually. It is in *`<bfinstall>`*.
- Change the `bf_services_preferred_protocol` setting tcp.

```
bf_services_preferred_protocol to tcp
```

This configuration improves performance at minimum security risk if the Build Forge host is physically secured.

Re-enabling TCP communications on a locked system

If there is a misconfiguration in SSL, the system locks you out.

To get access to a locked system, do the following:

- Stop Build Forge.
- In the *`<bfinstall>`* directory, open the `bfclient.conf` file with a text editor.
- Change the protocol property as follows:

```
bf_services_preferred_protocol=tcp
```

- Start Build Forge.

You should be able to log in.

Note

Changing the protocol does not disable secure login authentication, which is enabled by default. Users are redirected to a secure connection that allows secure communication of login credentials to Build Forge.

Enabling SSL for agent communications

Build Forge components are set up by default to use certain ports and security settings when SSL is enabled.

To enable SSL communication between Build Forge and agents, you need to do the following:

- Prerequisite: enable SSL for client and internal communications. See Enabling SSL for client and internal connections.
- Configure each agent. This task includes:
  - Adding certificates to the agent host
  - Editing the bfagent.conf file for the agent.
- In the console, enable SSL in each Server definition that connects to the agent.

Configuring agents for SSL

1. If the agent is running, stop it.
2. Place .PEM files for the certificates in the root installation directory for the agent.

   To implement and test SSL quickly you can copy the .PEM files from the Build Forge installation. The files are in `<`*`bfinstall`*`>/keystore`.

   The best practice for SSL is to use a separate certificate for each agent:

   a. Create separate keystore files (.PEM) for each of the following:
      - private key (key)
      - public certificate for the private key (cert)
      - trusted signers (ca or certificate authority)

   b. If you are using an unique certificate for the agent (instead of a copy of the Build Forge engine's certificate), then add the certificate for the agent to the certificate authority keystore for Build Forge, `<`*`bfinstall`*`>/keystore/buildForgeCA.pem`. If you are running multiple engines (redundant engines), add the certificate to each engine's certificate authority keystore.

3. Edit `BFAgent.conf`. The following lines are commented out in the file. Remove the comment prefix.

   ```
   ssl_key_location buildForgeKey.pem
   ssl_key_password password
   ssl_cert_location buildForgeCert.pem
   ssl_ca_location buildForgeCA.pem
   ssl_protocol TLSv1
   ssl_cipher_group ALL
   ```

   The *password* is for the buildForgeKey.pem keystore. If you want to encrypt it, see Encrypting passwords in buildforge.conf and bfagent.conf.

   If you want to require client authentication when a connection is made to the agent, uncomment the following line:

```
ssl_client_authentication true
```

This setting requires that the engine certificate be added in the agent's certificate authority keystore, `buildForgeCA.pem`.

If you want to use specific ciphers, uncomment this line and add your cipher list:

```
ssl_cipher_override cipher_list
```

4. Start the agent. The agent must be running in order to test the connection from the console.

Enabling SSL in Server definitions

The console uses Server definitions to connect to agents.

For each Server definition that is connected to an SSL-enabled agent, do the following:

- In the console, go to the **Servers** panel.
- Click the server definition name.
- In the **Details** tab for the server definition:
  - Set **SSL Enabled** to Yes.
- Click **Save** .
- Click **Test Connection** .

Troubleshooting SSL communication with agents

The following checklist describes common issues when enabling SSL:

- Agent
  - Agent SSL is not configured, though SSL is enabled in Security and in the Server definition.
  - The certificate for the agent is not trusted by the Build Forge engine. The agent certificate needs to be added to the engine's CA keystore:

    ```
    <bfinstall>/keystore/buildForgeCA.pem
    ```

  - Password for the keystore specified in `BFAgent.conf` is incorrect.
  - Client authentication is specified in `BFAgent.conf` but the engine's certificate has not been added to the agent's certificate authority, `buildForgeCA.pem`.
- Build Forge console
  - SSL was enabled on the console. It was not saved, or **Update Master Bfclient.conf** was clicked before saving, or **Update Master Bfclient.conf** was never clicked after saving.

- Server definition
  - SSL was not succesfully enabled. You must click **Save** before **Test Connection** .
- Engine and Agent settings match
  - Handshake protocol mismatch. The handshake protocol must be set to the same value for both the engine SSL configuration and agent SSL configuration, either TLSv1 or SSLv3. The default value is TLSv1.
  - Cipher suite mismatch. The cipher suites specified in the engine SSL configuration and agent SSL configuration must have ciphers in common. The default cipher suite group is ALL.

Enabling debugging messages

You can enable debugging in the engine and the agent. When debugging is enabled, additional detailed output is produced that can help identify problems in the configuration.

- Enabling engine debugging:
  1. In an environment used by an SSL-enabled server definition, add the following variable:

     ```
     BFDEBUG_SECURITY=1
     ```

  2. Stop the engine and restart it.
     - On Windows, start the engine in the foreground. Output appears in a command window.
     - On UNIX or Linux, start the engine with debugging turned on:

       ```
       cd <bfinstall>/rc
       ./buildforge start
       ```

       Output from the engine goes to the engine log file in `<bfinstall>/log`.
- Enabling agent debugging:
  1. Stop the agent.
  2. Add the following line to BFAgent.conf:

     ```
     activity_log bfagent.log
     ```

     In this example, the agent writes output to `bfagent.log`. You can specify a different filename.

     ### Note

     If the agent runs as a service, specify an absolute path.
  3. Start the agent.

Example engine debug output for a successful SSL connection

An engine produces the following output when it connects successfully to an agent.

```
SSL_ca_file: ./keystore/buildForgeCA.pem
SSL_cert_file: ./keystore/buildForgeCert.pem
SSL_key_file: ./keystore/buildForgeKey.pem
SSL_verify_mode: 0x01
SSL_version: TLSv1
SSL_cipher_list: ALL
SSL_use_cert: 1
Making as SSL connection using socket IO::Socket::INET=GLOB(0x1e8f0f4).
SSL connection to agent.
DEBUG: .../IO/Socket/SSL.pm:1387: new ctx 80662848
DEBUG: .../IO/Socket/SSL.pm:880: dont start handshake: IO::Socket::SSL=GLOB(0x1e
8f0f4)
DEBUG: .../IO/Socket/SSL.pm:284: ssl handshake not started
DEBUG: .../IO/Socket/SSL.pm:327: Net::SSLeay::connect -> 1
DEBUG: .../IO/Socket/SSL.pm:382: ssl handshake done
Socket is of type: ref(IO::Socket::SSL=GLOB(0x1e8f0f4))
ReadyLine: 202 HELLO TLS - BuildForge Agent v_VERSION_
.
Storing Agent Version [999.999.999.999-999-9999] for [08974C8E-6C3B-1014-972D-D9B2901D9F42]
cmd ping
username pbirk
encpass c1713f4a31af3f1300f7b2414a24559c4d6097e07310cf9c412e
go
Sending agent request...
```

Example agent debug output for a successful SSL connection

An agent running normally produces the following output when it establishes an SSL connection.

```
[2256] main.c          : 409: === NEW AGENT ===
[2256] io.c            : 264: In start_SSL
[2256] io.c            :  89: Key location: buildForgeKey.pem
[2256] bfpwdlocloader.c: 134: Looking for password locator: ssl_key_password_locator
[2256] bfpwdlocloader.c: 244: Looking for password for prop
                              ssl_key_password from bfagent.conf.
[2256] bfcryptloader.c : 202: Loading password encryption module.
[2256] bfcryptloader.c : 276: Password encryption property
                              password_encrypt_module is not configured.
[2256] bfcryptloader.c : 539: Password decoded.
[2256] io.c            :  98: Cert location: buildForgeCert.pem
[2256] bfpwdlocloader.c: 134: Looking for password locator:
                              ssl_cert_password_locator
[2256] bfpwdlocloader.c: 244: Looking for password for prop
                              ssl_cert_password from bfagent.conf.
[2256] io.c            : 153: Setting key password in default userdata.
[2256] io.c            : 160: Getting private key from PEM.
[2256] io.c            : 166: Checking private key from PEM.
[2256] io.c            : 172: Getting CA store information.
[2256] bfpwdlocloader.c: 134: Looking for password locator:
                              ssl_ca_password_locator
[2256] bfpwdlocloader.c: 244: Looking for password for prop
                              ssl_ca_password from bfagent.conf.
[2256] io.c            : 178: CA location: buildForgeCert.pem
[2256] io.c            : 184: Checking the CA store.
[2256] io.c            : 230: Returning from init_CTX.
[2256] io.c            : 281: Calling SSL_new
```

```
[2256] io.c            : 294: Calling SSL_accept.
[2256] io.c            : 346: Cipher chosen: AES256-SHA
[2256] io.c            : 367: ssl_state = SS_CERTIFIED
```

Example output for bad keystore password on the agent

If the keystore password configured on the agent side is wrong, it shows up in both engine and agent output.

Engine output (excerpt):

```
SSL_use_cert: 1
Making as SSL connection using socket IO::Socket::INET=GLOB(0x1e8f0f4).
SSL connection to agent.
DEBUG: .../IO/Socket/SSL.pm:1387: new ctx 80662848
DEBUG: .../IO/Socket/SSL.pm:880: dont start handshake: IO::Socket::SSL=GLOB(0x1e
8f0f4)
DEBUG: .../IO/Socket/SSL.pm:284: ssl handshake not started
DEBUG: .../IO/Socket/SSL.pm:327: Net::SSLeay::connect -> -1
DEBUG: .../IO/Socket/SSL.pm:1135: SSL connect attempt failed with unknown error
error:1408F10B:SSL routines:SSL3_GET_RECORD:wrong version number

DEBUG: .../IO/Socket/SSL.pm:333: fatal SSL error: SSL connect attempt failed wit
h unknown errorerror:1408F10B:SSL routines:SSL3_GET_RECORD:wrong version number
DEBUG: .../IO/Socket/SSL.pm:1422: free ctx 80662848 open=80662848 80566656
DEBUG: .../IO/Socket/SSL.pm:1425: OK free ctx 80662848
```

Agent output (excerpt):

```
[    5272] io.c            :  98: Cert location: buildForgeCert.pem
[    5272] bfpwdlocloader.c: 134: Looking for password locator: ssl_cert_passwor
d_locator
[    5272] bfpwdlocloader.c: 244: Looking for password for prop ssl_cert_passwor
d from bfagent.conf.
[    5272] io.c            : 153: Setting key password in default userdata.
[    5272] io.c            : 160: Getting private key from PEM.
[    5272] io.c            : 218: Failure reason: SSLErrorBadPKeyFile
[    5272] io.c            : 221: OpenSSL error string: error:00000000:lib(0):fu
nc(0):reason(0)
[    5272] io.c            : 281: Calling SSL_new
[    5272] platform.c      :2693: platform_release_credentials
[    5272] main.c          : 412: --- EXITING ---
```

Error codes in agent output

This list includes some of the other error codes you may encounter and their causes:

- SSLErrorBadCA: Problem loading the signers in `buildForgeCA.pem`. It could be a problem in the format of the file resulting from how the signers were added.
- SSLErrorBadCert: Problem loading the certificate in the `buildForgeCert.pem`. Either the certificate does not match the private key or it is corrupted in the PEM.
- SSLErrorBadPKeyFile: Problem with the `buildForgeKey.pem` password specified for the ssl_key_password property in BFAgent.conf.

- SSLErrorBadPKey: SSL_CTX_check_private_key returned a value other than 1. The private key format is not valid or it does not match the certificate.
- SSLErrorFIPSEnablement: An error occurred during FIPS enablement. This typically is due to an issue encountered during the FIPS self-check. This is likely an internal error.
- SSLErrorInvalidCipher: A cipher specification does not match what OpenSSL allows. Check the ciphers specified on properties ssl_cipher_group or ssl_cipher_override in BFAgent.conf.
- SSLErrorNoCtx: Problem creating a new SSL CTX object. This is likely an internal error.

Managing certificates

Certificates and keystores used by Build Forge can be modified after installation.

During installation you are given the opportunity to specify a certificate to use (either your own or one generated by Build forge) and a keystore password. This section describes procedures for the following:

- Converting PEM keystores to OpenSSL and JSSE keystores needed by Build Forge
- Changing the keystore password and modifying Build Forge to use the new password
- Creating a new self-signed certificate

Converting PEM keystores to Build Forge keystores

PEM keystores received from a Certificate Authority can be converted into keystores for use with Build Forge.

Download the unrestricted policy files for your SDK. This prerequisite applies only if your keysize is too large for the restricted policy files. Download the files from https://www14.software.ibm.com/webapp/iwm/web/reg/signup.do?source=jcesdk&lang=en_US&S_PKG=142ww

Note

You must use the keytool utility provided by IBM.

If you have a set of PEM files from a Certificate Authority, you must use them to create a set of OpenSSL and JSSE keystores for Build Forge.

1. Include Build Forge tool directories in your PATH.
   - `<`*`bfinstall`*`>/openssl`
   - `<`*`bfinstall`*`>/ibmjdk/bin` for Windows
   - `<`*`bfinstall`*`>server/ibmjdk/bin` for UNIX or Linux

   For UNIX and Linux, include the following directory in LD_LIBRARY_PATH:

```
<bfinstall>/openssl
```

2. Convert the PEM files into a PKCS12 keystore.

   Use the following command:

   ```
   openssl pkcs12
           -export
           -name "buildforge"
           -out buildForgeKeyStore.p12
           -inkey <key.pem>
           -passin pass:<pempassword>
           -in <crt.pem>
           -password pass:<bfpassword>
   ```

3. Verify that the certificate has been added and can be read.

   ```
   keytool -v
           -list
           -keystore buildForgeKeyStore.p12
           -storepass <bfpassword>
           -storetype pkcs12
   ```

   If you get an error about an invalid key size, download unrestricted policy files. Use the directions at the beginning of this section.

4. Export the public certificate.

   In a command window, go to `<bfinstall>/keystore`, then run this command:

   ```
   keytool -export
           -alias buildforge
           -file cert.der
           -keystore buildForgeKeyStore.p12
           -storepass <bfpassword>
           -storetype pkcs12
   ```

   - The certificate is stored in file `cert.der`.
   - Use the same *<bfpassword>* that was specified for keystores during installation. Otherwise you need to change the configuration.

5. Create the truststore and import the public certificate.

   In a command window, go to `<bfinstall>/keystore`, then run this command:

   ```
   keytool -import
           -noprompt -trustcacerts
           -alias buildforge
           -file cert.der
           -keystore buildForgeTrustStore.p12
           -storepass <bfpassword>
           -storetype pkcs12
   ```

6. Put the public client certificate in buildForgeCert.pem.

   In a command window, go to `<`*`bfinstall`*`>/keystore`, then run this command:

   ```
   openssl pkcs12 -clcerts -nokeys
           -in buildForgeKeyStore.p12
           -passin: pass:<bfpassword>
           -out buildForgeCert.pem
   ```

7. Put the certificate and keys in buildForgeKey.pem

   In a command window, go to `<`*`bfinstall`*`>/keystore`, then run this command:

   ```
   openssl pkcs12
           -in buildForgeKeyStore.p12
           -passin pass:<bfpassword>
           -passout pass:<bfpassword>
           -out buildForgeKey.pem
   ```

8. Create the PEM Certificate Authority buildForgeCA.pem.

   a. Download the CA root certificate to `<`*`bfinstall`*`>/keystore`.

      It is named CARootCert.crt. It needs to be added to your PEM keystores and can be imported into buildForgeTrustStore.p12.

   b. In a command window, go to `<`*`bfinstall`*`>/keystore`, then run these commands:

      ```
      cat CARootCert.crt > buildForgeCA.pem
      keytool -import -noprompt -v -trustcacerts
              -alias "CA Root"
              -file CARootCert.crt
              -keystore buildForgeTrustStore.p12
              -storepass <bfpassword>
              -storetype pkcs12
      ```

Build Forge uses a password-protected PEM keystore, buildForgeKey.pem. The Apache server prompts for the password during startup.

If you do not want to be prompted for a password during startup, then generate a PEM keystore that is not password-protected and have the Apache server use it. The following command is an example.

```
openssl rsa -in buildForgeKey.pem
            -passin pass:<password>
            -out buildForgeKeyForApache.pem
```

Be sure the unprotected PEM keystore is readable by any user who needs access to the ID of the process that runs Build Forge.

Changing keystore passwords

During installation you are given the opportunity to specify a keystore password. If you want to change that password, you need to do the following:

1. Modify the Build Forge keystores to use a new password.
2. Modify the Build Forge configuration to use the new password.

Modifying the keystore passwords

Build Forge has three default keystores that are password-protected, all installed on the host where you installed the Build Forge engine, in `<`*`bfinstall`*`>/keystore`:

- `buildForgeKey.pem` - used by OpenSSL, requires the `openssl` tool to change the password.
- `buildForgeKeyStore.p12`- used by JSSE, requires the `ibmjdk` tool to change the password.
- `buildForgeTrustStore.p12` - used by JSSE, requires the `ibmjdk` tool to change the password.

The tools are included with Build Forge software.

Note

Line breaks are used for clarity in the example commands. Do not use them in the command. Enter it as one string or use the line-continuation character (^ for Windows, \ for UNIX or Linux).

Important

The same password is used for all keystores. It is shown as *newpassword* in the examples.

1. Log on to the host where the Build Forge engine is installed.
2. Put the tool directories on your PATH.
   - `<`*`bfinstall`*`>/openssl`
   - `<`*`bfinstall`*`>/ibmjdk/bin`
3. Disable SSL.

   In the console, go to **Administration** → **Security** . Set **SSL Enabled** to No.
4. Click **Save** .
5. Click **Update Master BFClient.conf** .
6. Stop the Build Forge engine.
7. Back up the existing key stores.

   Copy the existing Build Forge keystores to a temporary directory. If the modified files get corrupted, you can use the backed up keystores.
8. Modify `buildForgeKey.pem`.

   In the directory `<`*`bfinstall`*`>/keystore`, run this command:

```
openssl rsa
-in buildForgeKey.pem
-passin pass:oldpassword
-out buildForgeKey.pem
-passout pass:newpassword -aes128
```

9. Modify `buildForgeKeyStore.p12`.

   In the directory `<bfinstall>/keystore`, run this command:

   ```
   keytool -storepasswd -all
   -new newpassword
   -keystore buildForgeKeyStore.p12
   -storepass oldpassword
   -storetype pkcs12
   ```

10. Modify `buildForgeTrustStore.p12`.

    In the directory `<bfinstall>/keystore`, run this command:

    ```
    keytool -storepasswd -all
    -new newpassword
    -keystore buildForgeTrustStore.p12
    -storepass oldpassword
    -storetype pkcs12
    ```

Once the passwords are changed, you need to modify the Build Forge configuration to use the new passwords.

Modifying the Build Forge configuration to use the new password

Build Forge configurations must be changed to use a changed keystore password.

Prerequisite:

- The Build Forge engine has not been started since you turned off SSL, stopped the Build Forge engine, and modified the keystore passwords.

The Apache Tomcat application server contains keystore passwords in the `server.xml` configuration file. They are stored as clear text. Apache Tomcat does not support encoded or encrypted passwords in this setting. In this procedure you modify `server.xml` and security properties in the Build Forge console.

1. Enter the new password in the Tomcat configuration.

   Edit `<bfinstall>/Apache/tomcat/conf/server.xml`. The Connector statement for SSL is located just under the comment `<!– Define a SSL HTTP/1.1 Connector on port 8443 -- >`.

   ```
   <Connector port="8443" maxHttpHeaderSize="8192" algorithm="IbmX509"
    maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
    enableLookups="false" disableUploadTimeout="true"
    acceptCount="100" scheme="https" secure="true"
    clientAuth="false" sslProtocol="TLS"
    keystoreFile="C:\Program Files\IBM\Build Forge\keystore\buildForgeTrustStore.p12"
    keystorePass="newpassword"
   ```

```
keystoreType="PKCS12"
truststoreFile="C:\Program Files\IBM\Build Forge\keystore\buildForgeTrustStore.p12"

truststorePass="newpassword"
truststoreType="PKCS12"/>
```

2. Start Build Forge.
3. Log in to the console.

   Use root or a login that has the Security access role.
4. Enter the new password for the keystores.

   In **Administration** → **Security** → **Keystore** , edit these entries to use *newpassword* in the Password property.

   - Default JSSE Key Store
   - Default JSSE Trust Store
   - Default OpenSSL Key Store
5. Enable SSL.
   a. In the console, go to **Administration** → **Security** .
   b. Set **SSL Enabled** to Yes.
   c. Click **Save**
6. Export the change to `BFClient.conf`.

   Click **Update Master BFClient.conf** .
7. Start the Build Forge engine.

Creating a new self-signed certificate

Use the tools provided to create a new self-signed certificate.

This procedure describes how to replace a certificate that was created automatically during a Build Forge installation. It creates a certificate with the following properties:

- Keystore: buildForgeKeyStore.p12
- Expiration: 15 years (set as 5475 days)
- Subject DN: CN=*hostname*, where *hostname* is the fully-qualified hostname.

Use the `openssl` and `ibmjdk` tools to create the certificate. The tools are included with Build Forge software.

Five keystores are needed:

- buildForgeKeyStore.p12 - keystore, container for certificates and keys
- buildForgeTrustStore.p12 - truststore, container for certificates and keys
- buildForgeKey.pem - PEM keystore
- buildForgeCert.pem - public certificate
- buildForgeCA.pem - PEM Certificate Authority (CA)

Note

Line breaks are used for clarity in the example commands. Do not use them in the command. Enter it as one string or use the line-continuation character (^ for Windows, \ for UNIX or Linux).

Important

The same password is used for all keystores. It is shown as *password* in the examples.

1. Log on to the host where the Build Forge engine is installed.
2. Put the tool directories on your PATH.
   - `<`*`bfinstall`*`>/openssl`
   - `<`*`bfinstall`*`>/ibmjdk/bin`
3. Create the keystore, `buildForgeKeyStore.p12`, certificate, and public-private key pair.

   In the directory `<`*`bfinstall`*`>/keystore`, execute this command:

   ```
   keytool -genkey -alias buildforge
   -keyalg RSA -keysize 1024 -validity 5475 -dname "CN=hostname"
   -keystore buildForgeKeyStore.p12
   -storepass password
   -storetype pkcs12
   ```

4. Export the public certificate.

   In the directory `<`*`bfinstall`*`>/keystore`, execute this command:

   ```
   keytool -export -alias buildforge
   -file cert.der -keystore buildForgeKeyStore.p12
   -storepass password
   -storetype pkcs12
   ```

5. Create the truststore and import the public certificate.

   In the directory `<`*`bfinstall`*`>/keystore`, execute this command:

```
keytool -import -noprompt -trustcacerts -alias buildforge
-file cert.der -keystore buildForgeTrustStore.p12
-storepass password
-storetype pkcs12
```

6. Put the public client certificate in `buildForgeCert.pem`

   In the directory `<bfinstall>/keystore`, execute this command:

   ```
   openssl pkcs12 -clcerts -nokeys
   -in buildForgeKeyStore.p12 -passin pass:password
   -out buildForgeCert.pem
   ```

7. Put the certificate and keys in `buildForgeKey.pem`

   In the directory `<bfinstall>/keystore`, execute this command:

   ```
   openssl pkcs12
   -in buildForgeKeyStore.p12 -passin pass:password
   -passout pass:password -out buildForgeKey.pem
   ```

8. Create the PEM Certificate Authority `buildForgeCA.pem`.

   It is a copy of `buildForgeKey.pem`. In the directory `<bfinstall>/keystore`, execute this command:

   ```
   cat buildForgeCert.pem > buildForgeCA.pem
   ```

The buildForgeKey.pem is a password protected PEM keystore. The Apache server prompts for the password during startup. If you do not want to be prompted for this password during startup, generate a PEM keystore that is not password-protected for the Apache server's use.

To remove the password from the private key, you can execute the following step. You should make sure the `buildForgeKeyForApache.pem` file is readable by those who need access to the ID of the process running Build Forge.

```
openssl rsa -in buildForgeKey.pem -passin pass:password
-out buildForgeKeyForApache.pem
```

Enabling password encryption

Configuring the Build Forge system to use password encryption increases the security of the system.

It is also critical to enforce physical security to prevent unauthorized access to the system.

Note

If you are integrating with Websphere components, be sure that the prerequisites for password encryption are met before configuring password encryption.

- For integration with Websphere Application Server: see Using Websphere Application Server instead of Apache Tomcat.

- For integration with IBM HTTP Server (IHS), see Using IBM HTTP Server instead of Apache HTTP Server.

About password security in Build Forge

The Build Forge system uses encoded passwords by default but can use encrypted passwords for additional security.

When password encryption is enabled, it is enabled as a symmetric key password scheme. The same key must be used by both the client using a password and the service that is accessed.

- Build Forge engine and Build Forge agents
- Build Forge services layer and the database used by Build Forge

In the Build Forge system, keys are kept in a `bfpwcrypt.conf` file. The file is located in the installation directory of Build Forge (for the engine) and the agent.

Password encryption uses symmetric keys. All systems that need to decrypt a common database password need the same key. Also, all agents that receive encrypted passwords from an engine need the engine’s key. If multiple engines are running (redundant configuration), the agent needs each engine's key.

For a simple installation of one Build Forge Management Console on one host and one agent on another host, enabling password encryption requires the following procedure:

- Enable password encryption on the Build Forge console (**Administration** → **Security**)
- Export the current key to a file. (This key is used by the agent and would also be used by other engines)
- Generate a new key for the agent. Export it to a file. (This key is used by the agent to encrypt its keystore password.)
- Update the agent's `bfpwcrypt.conf` with both keys. Put the new key last.
- In the Build Forge console, enable password encryption for all server definitions that use the agent. (**Servers** panel)
- On the Build Forge host, use the bfpwencrypt utility to encrypt the password that Build Forge uses to access the database. Replace the current password (encoded) with the encrypted password in `buildforge.conf`.
- Update the service layer's copy of `buildforge.conf`. See Build Forge configuration file (buildforge.conf).

Enabling password encryption for Management Console

Enabling password encryption in Management Console is a prerequisite for enabling password encryption in all other components:

- Redundant Management Consoles require export of the password key from the first Management Console to be included in their `bfpwcrypt.conf` files. All Management Consoles must use the same key. Typically they must also use the key to encrypt the database password used to access the database.
- Agents require the export of a password key if the engine is sending encrypted passwords to it. Both the engine and the agent should use the same key. The key exported from Management Console allows them to decrypt encrypted Server Auth passwords.

  If the agent is not receiving encrypted passwords, but needs to generate encrypted passwords for use in its `BFAgent.conf` file, then each agent should use a different password key. Generate each key individually from the Management Console.

To enable password encryption in the Management Console, do the following:

1. In the console, go to **Administration** → **Security** .
2. Set **Password Encryption Enabled** to Yes.
3. Click **Save** . This step saves the configuration in the Build Forge database.
4. Click **Update Master BFClient.conf** . This step saves the configuration in the Build Forge `bfclient.conf` file.
5. Restart Build Forge. This step is required so that the running Build Forge process uses the new settings in the `bfclient.conf` file.

When **Password Encryption Enabled** is Yes and in the configuration, you are able to do the following:

- Export the password key to a file
- Generate new password keys
- Run the `bfpwencrypt` and `bfagent` command to create encrypted passwords for inclusion in the console and agent configuration files.

Managing password keys

The Build Forge system uses encoded passwords by default but can use encrypted passwords for additional security.

Password encryption key file

The `bfpwcrypt.conf` file contains password encryption configuration properties.

When Build Forge is started for the first time, it automatically generates a `bfpwcrypt.conf` file in the same location as the `bfclient.conf` file.

- Windows: *`<bfinstall>`*
- UNIX or Linux: *`<bfinstall>`*`/Platform`

Important

Do not rename this file. It must always be named `bfpwcrypt.conf`.

The file contains these properties:

bfpwcrypt_key_alias	Alias of an encryption key. The alias is part of any password encrypted with this key. The system uses it to determine which key to use. There can be multiple definitions of this property, one for each key. The last definition is used to encrypt passwords. All others are used to decrypt an encrypted password when it is read. An encrypted password can be encountered in a configuration file, a database, or in a communication between an agent and the engine.
bfpwcrypt_key	The encrypted master key, encrypted using 128–bit AES encryption. This key is used to encrypt passwords.
bfpwcrypt_key_password	Password needed to decrypt `bfpwcrypt_key`. This password is encoded.

Example `bfpwcrypt.conf` file:

```
#**** Password Encryption Configuration Properties ****
bf_pwcrypt_key=MKuoiwD+MsWBFgl/2xeGOTEtpY+hAzXQu21fBcofM0M=
bf_pwcrypt_key_alias=8a679d430c401000b55e00007d1a7d1a
bf_pwcrypt_key_password=TqOeDXc4G/bdaWeatKTYUx6Sw4S3i6wX
# Creation date=Thu Nov 20 03:44:48 CST 2008
# Origination host=myhost.mycompany.com
```

Exporting the password key

Export the password key from the Build Forge console in order to place it in other locations.

The password key used on the Build Forge engine needs to be exported in order to place it in the following locations:

- Other Build Forge consoles that access the same database
- Agents that are identified in Server definitions that have password encryption enabled.

To export the password key, do the following:

1. In the console, go to **Administration** → **Security** .

   ### Note

   Password Encryption Enabled must already be set to Yes and the setting saved and updated in the master `BFClient.conf` file.

2. Click **Export Key File** . You are asked where to save the file.
3. Specify a location, then click **Save** .

Take the file to the hosts that need keys to decrypt passwords (other redundant management consoles, agents) or encrypt passwords (agents). Add the contents of the file (all three property settings) to the `bfpwcrypt.conf` file on that host:

- If placed at the end of the `bfpwcrypt.conf` file, the password key is used both to decrypt passwords that contain its alias and to encrypt new passwords.
- If placed anywhere else in the `bfpwcrypt.conf` file, the password key is used only to decrypt passwords that contain its alias.

Note

When you generate a new key, an additional key is placed at the end of the `bfpwcrypt.conf` file. When you export a key, only the latest key is exported.

Generating a new password key

You can generate new keys from Management Console for the purposes of changing the system-wide key. This should be done periodically to maintain good system security. The new key must be updated on all engines and agents.

Generating a new password key *adds* a new key to the `bfpwcrypt.conf` file:

- If user or console passwords have been generated or saved using the old key, they continue to work.
- **Important**: If a previous key has been exported and incorporated into an agent's `bfpwcrypt.conf` file, communications with the agent now fail until the newly generated key is added.
- Any new passwords saved in Build Forge or generated with `bfpwencrypt` use the new key.
- Exports of the key file export only the newest key.

To generate a new password key, do the following:

1. In the console, go to **Administration** → **Security** .

   ### Note

   Password Encryption Enabled must already be set to Yes and the setting saved and updated in the master `BFClient.conf` file.

2. Click **Generate New Key** . You are asked to confirm.
3. Click **Yes** .

Enabling password encryption for agents

Password encryption for agents is enabled in their configuration file.

To enable password encryption for an agent, do the following:

1. Go to the directory where the agent is installed.
2. Stop the agent if it is running.
3. Edit `bfagent.conf` and turn on the encryption setting:

   ```
   password_encrypt_module ./bfcrypt.dll;./bfpwcrypt.conf
   ```

   If the agent is launched from a directory other than where it is installed, change this path to refer to the files directly.
4. Get the Management Console encryption key. Export the key to a file. This key is required for the agent to decrypt an encrypted Server Auth password. It will also be used to encrypt local keystore passwords.
5. Place the key in the agent's `bfpwcrypt.conf` file. Put the generated key for the agent last in the file. The last entry in the file is used when you encrypt passwords manually.
6. Start the agent.
7. In the Management Console, go to **Servers** . For every server definition that uses this agent, set the **Password Encryption Configuration** property to Enabled.
8. Click **Test Connection** to make sure that the connection works with the encrypted password.

Encrypting passwords in buildforge.conf and bfagent.conf

Use an exported password key to build encrypted passwords for use in `buildforge.conf` and `bfagent.conf`.

The `buildforge.conf` file contains a username and password (db_password) that Build Forge uses to access the database. That password is normally encoded but can be encrypted. To encrypt a password for the Management Console, do the following:

1. Go to the Management Console root directory.
   - Windows: `<bfinstall>`
   - UNIX or Linux: `<bfinstall>/Platform`
2. Run the following command:

   ```
   bfpwencrypt -e password
   ```

   Use the plain-text password you want to encrypt for *password*.

   The encrypted password is sent to stdout.

The `bfagent.conf` file contains a the password key (ssl_key_password) that the agent uses to access the keystore. That password is normally clear text but can be encrypted. To encrypt a password for the agent, do the following:

1. Go to the agent root directory.
2. Run the following command:

   ```
   bfagent -e password
   ```

   Use the plain-text password you want to encrypt for *password*.

   The encrypted password is sent to stdout.

An encrypted password starts with the string `bfcrypt:` and the password key alias enclosed in braces, followed by the password, which is encrypted (AES 128 bit) and then encoded (Base63). Examples of encoded and encrypted passwords:

```
Encoded:
dd8b42eed5cc051500f5bffe2b82b1aa6a67baee028a85d0cefa

Encrypted:
{bfcrypt:7427ab360c4010008f9d000049664966}drAIT1zLDGX/xRcvw65+B8aFpTqvmAdbmnh6FpwkHjU=
```

Debugging problems with password encryption

If there are problems with encrypted passwords, these sections describe approaches to debugging them.

Debugging password encryption problems in the console

Services layer, the web interface, and engine share the same key file.

When all three components are installed on the same host, they use the same keyfile:

- Windows:

  `<bfinstall>/bfpwcrypt.conf`

- UNIX or Linux:

  `<bfinstall>/Platform/bfpwcrypt.conf`

Check the following issues if there are problems after you enable password encryption:

- Make sure you restarted Build Forge after enabling password encryption. Ensure that all processes got stopped and restarted properly (Apache, Apache Tomcat, engine).
- Redundant consoles: if you have multiple installations of management console using the same database, they must all use the same `bfpwcrypt.conf` file. The most secure method is to distribute it manually rather than over the network.
- Server definitions: if **Test Connection** fails in the console, be sure the key was exported and put in the `bfagent.conf` file correctly. To be sure it is a password problem, disable password encryption and try **Test Connection** .

- Login: if you cannot log in after enabling password encryption, make sure that Build Forge is using the correct `bfpwcrypt.conf` keys in both `bfclient.conf` and `buildforge.conf`. The `buildforge.conf` must be updated in the *<bfinstall>* directory and in the service layer's copy of it. See Build Forge configuration file (buildforge.conf).

If all of those checks are done but the problem persists, try enabling trace and examining the output logs.

- Web interface (UI): set environment variable BFDEBUG_SECURITY=1.

  Web interface: output appears in files.

  - Windows:

    `<bfinstall>/Apache/logs/php_error.log`

  - UNIX or Linux:
  - `<bfinstall>/server/apache/logs/php_error.log`

- Engine: start the engine in the foreground. In the installation directory, run `bfengine -d`. On UNIX or Linux you can pipe this to a file using `bfengine -d 2>&1 | tee out.txt`. On Windows, you can do the same if you obtain the tee utility.
- Services: do the following:

  1. Stop Build Forge.
  2. Open the log file in an editor.
     - Windows:

       `<bfinstall>/Apache/tomcat/common/classes/logging.properties`

     - UNIX or Linux:

       `<bfinstall>/server/apache/tomcat/common/classes/logging.properties`

  3. Add the following line to the end of the file.

     ```
     com.buildforge.services.common.security.level=ALL
     ```

  4. Start Build Forge.
  5. Inspect the output.
     - Windows:

       `<bfinstall>/Apache/tomcat/logs/catalina.out`

- UNIX or Linux:

  <*bfinstall*>/server/apache/tomcat/logs/catalina.out

Debugging password encryption problems in the agent

Debugging agent communications involves agent and engine components.

Check the following issues if there are problems after you enable password encryption:

- Check `bfagent.conf` The following line should be uncommented:

  ```
  password_encrypt_module ./bfcrypt.dll;./bfpwcrypt.conf
  ```

- Check that `bfpwcrypt.conf` is present in the directory where the agent is launched. It must contain at least one key entry.
- Check that the final entry in the engine's `bfpwcrypt.conf` is present somewhere in the agent's `bfpwcrypt.conf`.

If all of those checks are done but the problem persists, try enabling trace and examining the output logs. To enable trace, do the following:

1. Open `bfagent.conf`.
2. Uncomment the following line:

   ```
   activity_log bfagent.log
   ```

   You can specify another path instead of `bfagent.log`.

Path issues with bfcrypt.dll

In `bfagent.conf`, the `password_encrypt_module` property must point to the correct path to `bfcrypt.dll`. Example:

```
password_encrypt_module /opt/buildforge/bfcrypt.dll
```

With trace turned on, a problem with this path is indicated by output like the following:

```
[    8928] bfcryptloader.c : 208: Loading password encryption module.
[    8928] bfcryptloader.c : 223: module: bfcrypt
[    8928] bfcryptloader.c : 232: Loading module: C:/BuildForge71.181.Agent/bfcr
ypt.dll
[    8928] bfcryptloader.c : 262: Failed loading DLL, error code = 0
```

A successful load produces output like the following:

```
[   12248] bfpwdlocloader.c: 134: Looking for password locator: ssl_key_password_locator
[   12248] bfpwdlocloader.c: 244: Looking for password for prop ssl_key_password from
bfagent.conf.
[   12248] bfcryptloader.c : 208: Loading password encryption module.
[   12248] bfcryptloader.c : 223: module: bfcrypt
[   12248] bfcryptloader.c : 232: Loading module: ./bfcrypt.dll
[   12248] bfcryptloader.c : 269: Loading procedure bfcrypt_init.
```

Failed password decryption

When a password fails to decrypt because of the wrong key or some other reason, the log contains a line like the following:

```
[    4912] agent.c         : 237: AUTH failed
```

If you are sure that the password is correct, you can further diagnose the problem. Enable debugging for the bfcrypt.dll module. To enable debugging, set the following environment variable:

```
BFDEBUG_SECURITY=1
```

It needs to be set globally if the agent is running as a service.

Debug output is placed in `bfcrypt.txt` in the directory where the agent is launched.

The following output indicates that the correct key is not in `bfpwcrypt.conf` on the agent:

```
load_keys_from_file: Parsed 1 key configurations.decrypt: Looking for key matching info:
922492fe0c4010008304c3670e1e0e1e, length=32
decrypt: Comparing against: 4d553f110c401000ac08000051f651f6, length=32
decrypt: Warning!  No matching key found.
```

The following output indicates a correct key match:

```
load_keys_from_file: Parsed 2 key configurations.decrypt: Looking for key matching info:
922492fe0c4010008304c3670e1e0e1e, length=32
decrypt: Comparing against: 7427ab360c4010008f9d000049664966, length=32
decrypt: Comparing against: 922492fe0c4010008304c3670e1e0e1e, length=32
decrypt: Found match: 922492fe0c4010008304c3670e1e0e1e, length=32
```

bfclient.conf Reference

The bfclient.conf file stores settings Build Forge security. The file is in Build Forge installation root directory.

The `bfclient.conf` file contains settings used to enable secure communications (SSL) and password encryption. It contains the following sections:

- Connection properties
- Login properties
- SSL properties used by both OpenSSL and JSSE
- SSL properties used only with OpenSSL
- SSL properties used only with JSSE
- Keystore properties (`bf_keystore_*`)
- Cryptographic properties

Connection Properties

Attribute name	Default	Possible Values	Required	Description
`bf_services_hostname`	Specified during installation	n/a	Yes	Hostname where the Build Forge services layer is located
`bf_services_tcp_port`	Specified during installation	n/a	Yes	TCP port for connecting to Build Forge services. It is used when SSL is not specified.
`bf_services_ssl_port`	Specified during installation.	n/a	Yes	SSL port for connecting to BuildForge services securely
`bf_services_preferred_protocol`	tcp	ssl, tcp	Yes	For Perl or PHP clients, specifies SSL or TCP for making connections. For Java clients, the SecureAPIClientConnection object specifies SSL and APIClientConnection specifies TCP.

Login Properties

Attribute name	Default	Possible Values	Required	Description
`bf_login_user`	None	UserID in Build Forge users list.	No	Used as the login ID. The login ID can also be specified within a client program.
`bf_login_password`	None	Password for `bf_login_user`	Yes, if `bf_login_user` is used	Password for `bf_login_user`
`bf_login_realm`	None	LDAP domain name	No	LDAP domain to query if the user Is not already in the User table.

SSL properties used by both openSSL and JSSE

Attribute name	Default	Possible Values	Required	Description
`bf_ssl_usage`	None	jsse, openssl	Yes	Selects the SSL implementation. Depending on the selection, different properties are available.
`bf_ssl_cipher_group`	ALL	ALL, HIGH, MEDUM, LOW	No	Specifies the group of ciphers to be provided during the SSL handshake. HIGH is the most secure, LOW gives the best performance, ALL is the most interoperable.
`bf_ssl_cipher_override`	None	Cypher suites that you provide	No	Overrides `bf_ssl_cipher_group`. Can be used to choose a smaller set of ciphers to use during the SSL handshake.
`bf_ssl_protocol`	TLSv1	TLSv1, SSLv3. Can vary with implementation.	No	Handshake protocol used by SSL. TLSv1 is the preferred protocol.
`bf_ssl_cert_alias`	None	Valid certificate alias in the configured keystore	No	Specifies the certificate to use. This is possible when multiple certificates are in the same keystore.

SSL properties used only with openSSL

Attribute name	Default	Possible Values	Required	Description
`bf_ssl_key_ref`	openssl_key	Any valid PEM keystore reference which contains a private key	No	Reference to a keystore configuration that contains a private key for the client to use when connecting to a server. When used, you must also specify a valid certificate for this private key in `bf_ssl_cert_ref`. Used only when the server is set up to request personal certificates.
`bf_ssl_cert_ref`	openssl_cert	Any valid PEM keystore reference which contains a certificate for the specified private key	No	Reference to a keystore configuration that contains a certificate for the private key above. Used only when the server is set up to request personal certificates.
`bf_ssl_ca_ref`	openssl_ca	Any valid PEM keystore reference which contains one or more certificates used to validate the server certificates which this client is connecting to.	Yes	Reference to a keystore configuration that contains one or more signer certificates used to validate server certificates during an SSL handshake. The certificates can be CA root, intermediate, or self-signed.

SSL properties used only with JSSE

Attribute name	Default	Possible Values	Required	Description
`bf_ssl_keystore_ref`	jsse_keystore	Any valid PKCS12, JKS, or JCEKS keystore reference which contains a keyEntry (private key and certificate).	No	Reference to a keystore configuration that contains a personal certificate (private key and associated certificate) for the client to use when connecting to a server. This is necessary only when the server requests a personal certificate for client authentication.
`bf_ssl_truststore_ref`	jsse_truststore	Any valid PKCS12, JKS, or JCEKS keystore reference which contains a keyEntry (private key and certificate).	Yes	Reference to a keystore configuration that contains signer certificates used to validate server certificates during an SSL handshake. The keystore contains one or more trustedCertEntries, which are certificates used to validate other certificate signatures.

Keystore properties

Attribute name	Default	Possible Values	Required	Description
`bf_keystore_alias`	Various	String	Yes	This is the name which an SSL configuration will reference the keystore configuration by.
`bf_keystore_location`	Various	Relative or fully-qualified path to a keystore of the type specified.	Yes	This is the path and location to the keystore of the type specified. The path can be a relative path, but it must be correct with respect to the starting directory.
`bf_keystore_type`	PEM for openssl, PKCS12 for jsse.	PEM for openssl. PKCS12, JCEKS, or JKS for “jsse”.	Yes	The type of the keystore. Must match the actual keystore type referenced by the bf_keystore_location property.
`bf_keystore_password`	Specified during installation.	A string supported by the keystore type. Some keystores do not support non-ASCII strings.	No	The password for accessing the keystore. For OpenSSL, the password is usually not required for Cert and CA keystores containing just public keys.

Cryptographic Properties

Attribute name	Default	Possible Values	Required	Description
`bf_pw_crypt_enabled`	false	true, false	No	Specifies whether passwords are encoded (false) or encrypted (true). When enabled, the password encryption implementation uses a file called bfpwcrypt.conf located in the same directory as bfclient.conf.

Installing agents

This section describes how to install, run, configure and troubleshoot agents.

Install an agent on each host that you want to use as a server resource from the Management Console. An agent is a service that receives requests from the Management Console to run projects and steps.

Installing the agent on Windows platforms

To install the agent on Windows platforms:

1. Locate and start the agent installation program in the installation media. The filename for the installation program is `win-bfagent-`*`version`*`.exe`.

   ### Tip

   The Launch Pad starts this installation process when you choose **Rational Build Forge Agent Installation** .

2. If the installer detects an existing version of the agent, it prompts you to confirm to overwrite it. Click **OK** . Default: OK.

3. After the Welcome message opens, click **Next** .

4. If you agree to the terms of license agreement, click **I Agree** .

5. In the Choose Install Location window, set the **Destination Folder** , then click **Next** . Use the default location C:\Program Files\IBM\Build Forge\Agent, so that the file can be easily located.

6. In the Configuration window, choose the **Agent Options** that you want, then click **Install** .

7. Select one of the following installation methods:

   a. **Install as a service** .

   b. **Install User Mode Agent** Select a user mode agent only if the agent must be able to run a GUI application.

8. Optional: **Click Enable Cygwin Support**

   ### Tip

   If you use the Cygwin Linux® emulation environment, you can choose to install Cygwin support when you install the agent. If you install Cygwin support, do the following steps.

   a. During Cygwin installation, choose **DOS/text** line endings.

   b. In projects, use UNIX®-style syntax for commands.

Important

Cygwin works only with US ASCII. It does not support UTF-8, so it cannot be used with any other systems.

9. Specify the **Port** that the agent uses to communicate with the Management Console. The default port is 5555.

10. At the Completing Setup panel, click **Finish** .

Note

Do not close any pop-up windows during installation. Allow them to appear and disappear while the installation runs.

User mode agent

An agent installed as User Mode allows a user to interact with applications launched by a project.

User mode is an option offered for Windows agents only. The option is set during installation. It cannot be configured after installation. User mode has the following applications:

- Collect input manually through a GUI application as the job runs. This makes the job dependent on human input.
- Troubleshoot projects and steps. Output that is hidden during service mode operation is visible in user mode. Each step produces a console window while the step is running.

Keep in mind the following differences when setting up projects to use a machine in user mode:

- The user mode agent operates as the currently-logged-in user on the system. The agent is only active during the time that this user is logged into the machine. A server running a user-mode agent can't be used if the user is logged out.
- Steps run on a machine in user mode are visible to any users of the machine.
    - Each step opens a console window on the Windows machine where the agent is running. It displays command activity from the Management Console.
    - If you launch a GUI application from a step, the application window appears on the Windows machine where the agent is running. The Management Console waits until the application exits before continuing the step.

      Alternative: Use the start command if you want the job to continue without waiting.
- Do not use the _USE_BFCREDS variable. Any steps that use this variable will fail.
- The user must have the following privileges. They are typically not available by default. They must be added explicitly.

  ```
  SeInteractiveLogonRight
  SeAssignPrimaryTokenPrivilege
  ```

```
SeImpersonatePrivilege
SeIncreaseQuotaPrivilege
SeTcbPrivilege
```

Performing a silent agent installation on Windows operating systems

To perform an automatic and silent installation of an agent on Windows, use the `/S` (uppercase S) option. For example, at a command prompt enter the following command. The option is case-sensitive:

```
win-bfagent-7.0.1.2305.exe /S
```

The silent installation uses the following settings. They cannot be modified.

- Overwrite existing install: `yes`
- Install location: `C:\Program Files\IBM\Build Forge\Agent`
- Install as a Service: `yes`
- Cygwin support: `no`
- Port: `5555`

Installing the agent on UNIX and Linux systems

Follow the installation instructions for your platform:

- **AIX**

  1. Use the aix5-bfagent-<*version*>.tar.gz file or aix5np-bfagent-<*version*>.tar.gz file.

     The `aix5np` file does not contain support for PAM authentication.

     Important

     If you install the aixnp agent to run as root, at runtime the agent will use an AIX authentication call to authenticate, using the Server Authentication credentials that you specify. If you do not install the agent to run as root, then you must also use the magic_login setting in the bfagent.conf file to restrict access to it.

  2. Extract the file by entering this command:

     ```
     gzip -d gzipfilename.gz
     ```

  3. Extract the file by entering this command:

     ```
     tar xvf tarfilname.tar
     ```

  4. Install the agent by entering this command:

```
cd extracted-agent-directory
./install.sh
```

Important

If an agent is compiled for AIX with the configure option -–without-pam, authentication for the agent is turned off.

If it is then installed with root privilege, it allows people to connect as any valid user regardless of the password that they specify.

If you must compile an agent to run on an AIX system that does not use PAM, be sure to use a dedicated account for the agent, install it to run as that user, and use the magic_login setting in the bfagent.conf file to restrict access to it

- **HP-UX**

1. Set a PATH value if one is not already set. Although HP-UX supports operation without a PATH value, problems occur when the agent tries to add to the path. You can set the value by editing the system's `/etc/PATH` file, or by using one of the shell standard PATH set or append operations.
2. Get the `hpux11-bfagent-<`*`version`*`>.tar.gz` file from the installation media. Place it where you want the agent installed.
3. Extract the file.

   ```
   gzip -d gzipfilename.gz
   ```

4. Extract from the tar file.

   ```
   tar xvf tarfilename.tar
   ```

5. Install the agent.

   ```
   cd extracted-agent-directory
   ./install.sh
   ```

6. Modify the following line in `/etc/profile` to enable the agent to run commands in a non-interactive login shell.

   ```
   if [ ! $VUE]
   ```

   Change the line to the following:

   ```
   if [ -z "$VUE" -a -n "$PS1" ]
   ```

- **Mac OS**

1. Get the mac-bfagent-<*version*>.dmg file from the installation media and place it where you want it.

2. Double-click the file to extract its components.

- **Red Hat Linux**

1. Get the rh9-bfagent-<*version*>.rpm (Red Hat Enterprise Linux 4) or rhel5-bfagent-<*version*>.rpm file (Red Hat Enterprise Linux 5) from the installation media

2. Enter this command:

```
rpm -iUvh rh9-bfagent-<version>.rpm
```

- **Solaris**

1. Use the sol9-bfagent-<*version*>-sparc-opt.gz file for Solaris 9 or Solaris 10 on Sparc.

2. Extract the package:

```
gzip -d solN-bfagent-<version>-platform-opt.gz
```

3. At a command prompt, enter this command:

```
pkgadd -d ./unzipped-package
```

- **Other Platforms - Compiling from Source**

If you need an agent for another platform, use the src-bfagent-<*version*>.tar.gz file to compile the agent from the source:

1. Extract the downloaded tar file.

2. Run the configure script located in the src directory.

3. Run the **make** command in the src directory.

The source pack requires the GNU C compiler or the C compiler for your system. The source pack and prebuilt agents that do not include an installer for the local computer include an installer to install the agent in the system's inetd/xinetd configuration files of the computer.

Installing the agent on System i platforms

Use the following instructions to manually install the agent on System i.

The command script in step 7 creates a job description that runs at start up and starts the agent as the BFAGENT user with *ALLOBJ special authority.

- Any user with *ALLOBJ special authority or the QSECOFR user can be authenticated by using the server authentication credentials that you specify in the Management Console.

- To authenticate a user without these privileges, you must configure the magic_login setting in the bfagent.conf file. See the “bfagent.conf Reference” on page 165 for details.

To install the agent on System i platforms:

1. Using the product installation media or the download image, locate the iseries-bfagent-<*version*>.tar.gz file.
2. Extract the tar file from the archive by entering this command:

   ```
   gzip -d iseries-bfagent-<version>tar.gz
   ```

3. Extract the files from the tar file.

   ```
   tar xvf iseries-bfagent-<version>.tar
   ```

4. On the iSeries server, place the bfagent executable file in the agent installation directory, for example: /bin.
5. On the iSeries server, place the bfagent.conf file in /etc.
6. In the bfagent.conf file, uncomment the shell option and specify the default shell for PASE, as shown in the following example, or specify your preferred shell.

   ```
   shell /bin/sh
   ```

7. Configure System i to run as the BFAGENT user at startup.

   Enter the following commands to create the BFAGENT user with *ALLOBJ special authority and create a job description that runs at startup as the BFAGENT user. In the following example, the bfagent executable is installed in /bin.

   ```
   CRTLIB BFAGENT

   CRTSBSD SBSD(BFAGENT/BFAGENT) POOLS((1 *BASE)) TEXT('Build Forge Agent subsystem')

   CRTJOBQ JOBQ(BFAGENT/BFAJOBQ) TEXT('Build Forge Agent job queue')

   CRTUSRPRF USRPRF(BFAGENT) PASSWORD(*NONE) INLMNU(*SIGNOFF) LMTCPB(*YES)
    SPCAUT(*ALLOBJ) TEXT('Build Forge Agent user profile')

   CRTJOBD JOBD(BFAGENT/BFAJOBD) JOBQ(BFAGENT/BFAJOBQ)
    TEXT('Build Forge Agent autostart')USER(BFAGENT) RQSDTA('CALL PGM(QP2SHELL)
    PARM(''/bin/bfagent'' '' -s'')')

   CRTCLS CLS(BFAGENT/BFACLS) TEXT('Build Forge Agent job class')

   ADDRTGE SBSD(BFAGENT/BFAGENT) SEQNBR(1) CMPVAL(*ANY) PGM(QCMD) CLS(BFAGENT/BFACLS)

   ADDJOBQE SBSD(BFAGENT/BFAGENT) JOBQ(BFAGENT/BFAJOBQ) MAXACT(*NOMAX) SEQNBR(10)

   ADDAJE SBSD(BFAGENT/BFAGENT) JOB(BFAGENT) JOBD(BFAGENT/BFAJOBD)
   ```

Installing and running the agent on System z platforms

Follow these instructions to manually extract and compile the Build Forge agent source code on System z. The agent source code for z/OS is provided as uncompiled source only. A binary distribution is not available.

The following software and programs are required:

- The c89 compiler and Unix header files. On the z/OS system, the agent runs in the Unix System Services (USS) environment.
- The z/OS UNIX shell interface. During installation, you run all commands on z/OS in the z/OS UNIX shell.
- The gzip utility.

  Note

  If gzip is available on the z/OS system, you can extract the tar file on the z/OS system after transferring the source pack to z/OS. If it is not, you must first extract the files on a non-z/OS machine, then transfer them to the z/OS system.

- The Build Forge agent source pack for z/OS: src-bfagent-<*version*>.tar.gz.

To install the agent on System z platforms:

1. Using either the product installation media or the download product image, locate the file for the agent source pack: src-bfagent-<*version*>.tar.gz.

   Copy or download the source pack to a directory on the non-z/OS machine.

2. At a shell prompt on the non-z/OS machine, extract the tar file from the agent source pack by entering the command:

   ```
   gzip -d src-bfagent-<version>tar.gz
   ```

3. Using ftp or another transfer method, transfer the tar file to the z/OS system as a binary image and place it in a dedicated HFS subdirectory, usually the USS home directory for a user account.

4. On the z/OS system, run the following commands to build the agent source code:

   ```
   pax -rf src-bfagent-<version>.tar -ofrom=ISO8859-1,to=IBM-1047
   cd bfagent-<version>/src
   ./configure-zos
   ```

   After the ./configure-zos script completes, run the following command:

   ```
   ./build-zos
   ```

Note

If you receive errors after the ./build-zos script runs, see Troubleshooting the agent installation on z/OS.

5. On the z/OS system, place the bfagent.conf file in /etc.

   If bfagent.conf is not in /etc, the agent must be started with the -f option. See bfagent Reference.

6. On the z/OS system, place the bfagent executable file in an appropriate location, for example, /usr/bin or /usr/local/bin.

7. On the z/OS system, run the following command as root:

   ```
   # extattr +p -s bfagent
   ```

8. On the z/OS system, log in as root and start the agent manually by using the -s option:

   ```
   bfagent -s
   ```

   If security policy does not allow you to log in as root, see bfagent.conf Reference and see the instructions for the magic_login setting in bfagent.conf.

   The agent runs as a standalone daemon and uses the default agent port 5555. To change default port, use the port setting in bfagent.conf. See bfagent Reference.

   ### Note

   If the Unix TCP/IP daemon (inetd or xinetd) is installed and active on the z/OS system, you can set up the Build Forge agent to run as a service and start automatically. See Running an agent on UNIX, Linux, and MacOS.

9. On the z/OS system, use the telnet command to test the connection. See Testing the connection.

Troubleshooting the agent installation on z/OS

You might receive error messages after building the agent source code on z/OS. This topic describes fixes for some common errors.

The configure-zos script sets some common values and performs some basic checks to identify the headers and functions available on your system.

Because of variations in z/OS system configurations, the ./configure-zos script might run without errors but you might see the following errors when you run the ./build-zos script.

`CEE3501S The module CCNDRVR was not found.`	FSUM3066 The COMPILE step ended with the following return code: `-1: EDC5083I An error occurred attempting to load a module into storage.`

	This error indicates that a required dynamic library cannot be loaded by the compiler. Run the command: `% export STEPLIB="SYS1.SCCNCMP"` Rerun the ./build-zos command. If the command fails again, contact your system administrator for assistance in locating the required library.

`IKJ56228I DATA SET CEE.SCEEOBJ NOT IN CATALOG OR CATALOG CAN NOT BE ACCESSED`	FSUM3066 The COMPILE step ended with the following return code:
	
	 FSUM3052 The data definition name C8961 cannot be resolved. The data set was not found. Ensure that data set name CEE.SCEEOBJ is specified correctly.
	
	 This error indicates that the linker was unable to locate a system library that it needs to complete the compilation. Run the commands:
	
	 `% export _C89_LSYSLIB=SYS1.SCEELKED:SYS1.SCEELKEX`
	
	 `% export _C89_PSYSLIB=SYS1.SCEEOBJ`
	
	 Rerun the ./build-zos command. If the command fails again, contact your system administrator for assistance in locating the required libraries.

`IEW2456E 9207 SYMBOL xxx UNRESOLVED`	The unresolved symbol errors indicate that the build expected a symbol to be defined by your system C library that is not actually there. In most cases, this is a symbol that is often missing from other systems too, and there will be a setting in config.h to work around the problem. For example, your system might not define the unsetenv function. The configure-zos script should normally detect this; if it does not, edit the config.h file that is provided with the agent source pack, as follows: Change `#define HAVE_UNSETENV 1` to `#undef HAVE_UNSETENV`. Rerun the ./build-zos command to correct the problem. **Note** Similar #define statements exist for other functions.

Running an agent

This section describes how to set up an agent to run. It is normally run as an auto-starting service or daemon.

Running an agent on Windows

The agent is typically installed as a service and is set to Auto so that it starts when you turn on the computer. You must be logged on to the computer where the agent is installed to start and stop it.

To start and stop the agent, you can use the **Start** menu:

- To start the agent, click **Start** → **Programs** → **IBM Rational Build Forge** → **Start Agent Service** .
- To stop the agent, click **Start** → **Programs** → **IBM Rational Build Forge** → **Stop Agent Service** .

You can also use the following commands at a command prompt:

- `net start bfagent`
- `net stop bfagent`

Running an agent on UNIX, Linux, and MacOS

The agent is intended to be run as a service and needs to be restarted automatically at system restart.

Add a bfagent entry to your configuration of inetd or xinetd as appropriate. The following example is the bfagent entry in xinetd.d on a Linux system, where the agent is installed in /usr/local/bin:

```
# description: The IBM Rational Build Forge Agent serves build requests
#    from the IBM Rational Build Forge Management Console.
service bfagent
{
        disable         = no
        flags           = REUSE
        socket_type     = stream
        wait            = no
        user            = root
        server          = /usr/local/bin/bfagent
        log_on_failure  += USERID
}
```

The agent can be run outside of the inetd/xinetd environment if necessary. To run it as a standalone daemon, use the -s option.

```
bfagent -s
```

When you use this option, the agent moves into the background and begins listening for connections. Place this command in a startup script so that the agent starts automatically when the computer is started.

Running the agent on System i

Review the information in this topic if you plan to run the agent on a System i platform.

Verifying that the agent port number is unique

Port 5555, which is the standard Build Forge agent port, might be preassigned to other agents on System i servers. In this case, change the Build Forge agent port to an unassigned port before starting the agent. To do this, edit the bfagent.conf file directly. For details, see Changing the agent port.

Starting the agent manually

If you completed step 7 in the installation instructions, Installing the agent on System i platforms, the agent starts as the BFAGENT user when System i starts.

Alternatively, you can start the agent on System i manually, by using the following command.

```
bfagent -s
```

Note

If the bfagent.conf file is not installed in /etc (the default location), use the -f option to specify the bfagent.conf location.

When you issue the bfagent command and start the agent manually, the agent starts as the user who starts the agent.

- If the QSECOFR user or a user with the *ALLOBJ special authority starts the agent, the user is authenticated by using the server authentication that you specify in the Management Console.
- If another user starts the agent, authenticate the user by configuring the magic_login setting in the bfagent.conf file. See the "bfagent.conf Reference" on page 165.

Verifying that the i5/OS PASE program is installed

The agent runs as an i5/OS Portable Application Solution Environment (PASE) program. PASE is included in i5/OS and enables AIX binaries and commands to be run. PASE is typically installed by default.

To determine whether the PASE program is installed, run DSPSFWRSC at a command line.

If the PASE program is not installed, load it from the installation CD.

Using the agent in PASE

Most tasks necessary for building applications on i5/OS are accessible from the PASE environment. It is important to keep this in mind when planning and defining automation of processes targeted for the iSeries platform.

Commands in a step are interpreted by the PASE shell. You can also run native commands using the following syntax:

```
system -biOE "<native commands>"
```

Important

Each system command in a step runs its own process. This has implications for commands that work only within their own process.

For example, if you want to set library lists for a set of steps:

- You cannot use CHGSYSLIBL or ADDLIBLE as step commands because they are native commands (not recognized by PASE).
- You cannot use the supported native command syntax (for example, `system -biOE "ADDLIBLE FLGHT400"`) in a step, because it changes the library list only for the command's own process. Subsequent commands and steps are not affected by the change.

Although you cannot set library lists for just a step, a set of steps, or a project, you can set them in the startup command script for the BFAGENT user. See the example startup script in Installing the agent on System i platforms. Setting library lists in the startup command script, sets library lists for all projects and steps that are run in the example as the BFAGENT user. The user who runs the projects and steps must have access to the required libraries.

To set library lists, add a job description for the agent that lists the required libraries. The following example job description includes libraries `FLGHT400` and `FLGHT400M`.

```
10    UTLIB
20    QGPL
30    QTEMP
40    FLGHT400
50    FLGHT400M
```

The agent specifies this job description in its startup routine. For example, if the job description is BFAJOBD, the line in the system startup routine would be as follows:

```
ADDAJE SBSD(BFAGENT/BFAGENT) JOB(BFAGENT) JOBD(BFAGENT/BFAJOBD)
```

This solution affects all commands (from any step and project) that are run on the System i server associated with this agent.

bfagent Reference

The bfagent executable file starts the Build Forge agent. It reads its configuration from a BFAgent.conf file in the same directory.

The syntax for the command is:

```
bfagent [-f configfile | -s]
```

Options

-f configfile	Run using the configuration file in configfile, rather than `BFAgent.conf`. This is a runtime option on UNIX or Linux. It is a debugging option when running the agent manually on Windows. It cannot be used for starting the service on Windows.

-s	Start as a standalone service. Can be used on UNIX or Linux only. You can use this option only on UNIX or Linux. Running this way is an alternative to starting bfagent with either the **inetd** or **xinetd**.

Configuring the agent

This section describes how to configure the agent after installation.

Locating the agent configuration file

The agent configuration file, BFAgent.conf, provides runtime configuration of the agent's operation. It contains comments that explain all of the possible options. The file is located in the agent installation directory:

- Windows default: C:\Program Files\IBM\Build Forge\Agent\BFAgent.conf
- UNIX and Linux default: /etc/bfagent.conf

Important

If you make changes to the BFAgent.conf file in the installation directory, you must repeat the changes after you subsequently reinstall or upgrade the agent. The configuration file is overwritten during every installation.

You can specify an alternate configuration file:

- On UNIX or Linux systems, you can preserve agent configurations by using a configuration file outside of the agent installation directory. When you do this, use the `-f` command line option on the command that starts the agent. Example:

  ```
  bfagent -f /opt/bfagent.conf
  ```

- On Windows systems, you cannot start the service with this option. The service can be used only when you run the agent manually. It is a tool for debugging.

Changing the agent port

If the agent is being installed on a server where port 5555 is already occupied, the agent port can be changed after it is installed.

To change the port on Windows operating systems:

1. Open the registry editor: Click **Start** → **Run** , and then type `regedit`.
2. Go to HKEY_LOCAL_MACHINE\SOFTWARE\BuildForge Agent.
3. Change the value of AgentPort to the port number you want.

On UNIX, Linux, and Macintosh operating systems:

1. Open the /etc/services file.
2. Change the value of BuildForge Agent Port to the port number you want.

Configuring a different shell

You can configure an agent to use a shell other than the default shell by editing parameters in the BFAgent.conf file.

For example, to change a Windows system to use the Korn shell provided by MKSTools, you can change the shell parameter with this command: :

```
shell C:\MKSTools\mksnt\ksh.exe -L -c \"%s\"
```

The % in this command is replaced by the step command when the system sends a command to the server. In this case, use the backslash escape character to include quotation marks as literals in the command.

Running agent commands on a Network Share File System (Windows)

The Build Forge agent initially starts with Windows system account credentials. To run commands, the agent later authenticates with Windows using Build Forge server authentication credentials.

The server authentication credentials are accepted for local commands but might fail for some commands that the agent must run on external, networked shared drives. For example, to modify files in a ClearCase dynamic view, the agent must access ClearCase files on a network shared drive.

The commands fail because the external file system ignores the agent server authentication credentials; it recognizes only the agent's initial system account credentials.

If you experience problems running commands on a network shared drive, try the following actions:

Run commands using server authentication credentials.	To run commands using a Build Forge server authentication credentials with access to network shares, add the win_reexec_after_auth setting to the BFagent.conf file. If you want to use Build Forge server authentication credentials to establish access to a network share, adding this setting is prerequisite. The win_reexec_after_auth setting causes the agent to start a new process after authenticating with Windows. The new process forces the shared file system to recognize that the agent changed the user credentials. When win_reexec_after_auth is set, the agent runs as a service and does not distinguish between commands that access network shares and those that do not, so you might notice a performance impact.

Run the agent in single user mode	During agent installation, set up the agent to run commands in single user mode without Build Forge server authentication credentials. Select the **Install User Mode Agent** option. If the specified user is a member of the Administrator group, then the user's credentials must be specified using server authentication credentials. If the user is not an administrator, then use the magic_login setting in BFagent.conf to prevent unauthorized access to the agent. When you log on to the Management Console, the agent starts up and runs as the user name you provide, which immediately authorizes access to the network shares using that user's credentials.
Run the agent as a service with a dedicated user account	Set up the agent to run as a Windows service with a dedicated user account. This option restricts you to running the agent as a single user account, but does not require the agent to start a new process to re-authenticate, so performance is not affected. To run the agent as a service with a dedicated user account: 1. On the Build Forge server, click **Administration Tools** → **Services** to open the Windows Control panel. The list of services opens. 2. Open the service for the IBM Rational Build Forge Agent. 3. Provide the user account information for the user you want to run agent commands. For example, provide information for the ClearCase admin user or other user with access to ClearCase dynamic views and VOBs).

bfagent.conf Reference

The bfagent.conf file stores settings for how the Build Forge agent runs. The file is in the same directory as the bfagent executable file.

The file lists all settings and internal defaults. Inactive settings are commented out.

Settings

activity_log *path*	Turns on activity logging. The information is appended to the file specified by *path*. The path must exist and the agent user must have write permission for it. **Note** The agent does not report an error if the path does not exist or if it cannot write to the file.

Important

There is no limit on file size. The file must be manually deleted. This setting is intended to be used temporarily for debugging the agent. It is not intended as a permanent log for a working agent.

allow IP-address-or-range [...]

Use this setting for these conditions only: :

- Agents running on Windows
- Agents running in standalone mode on UNIX or Linux when the -s option on startup is used.

This setting limits connections to the agent. Connections are allowed only from the IP addresses that match IP-address-or-range. By default connections are allowed from all addresses.

Specify one or both of the items:

- **IP Address**: A fully-qualified IPv4 or IPv6 address. For example, for IPv4, 255.192.192.003. The specific IP address is allowed.
- **IP Address range**: A partially-qualified IPv4 or IPv6 address. These examples are correct for IPv4, 192.168 or 192.168.63. All IP addresses that match this qualification are allowed.

Note

If you are running the agent on a superserver such as inetd or xinetd, use another method to control access. You may want to use a firewall, TCP wrappers (hosts.allow and hosts.deny), or the built-in filtering capability of xinetd.

bind

This setting allows the user to specify an explicit bind address for the agent. This, together with the "port" setting, determines how the agent will listen for connections when it is started with the -s command line option. The value given in the bfagent.conf file will force the agent to bind to the IPv4 localhost address; thus, the agent will only receive connections from a console that is on the same machine. Example: `bind 255.192.192.003`

Note

It has no effect on Windows or Unix agents that are started by the system's service architecture, such as inetd, xinetd, or launchd.

ccviewroot root-path	This setting specifies the default view root for this host. See ClearCase documentation on init for more information. The internal defaults are as follows: • Windows: ccviewroot M: • UNIX or Linux: ccviewroot /view
command_output_cache size	This setting causes the agent to cache output until it reaches the specified size in bytes. The internal default is not to cache. Using a cache can significantly improve agent performance and reduce network overhead. The cache size depends on how much output that commands wproduce. Minimum value: 2048. A value of 2048 is used internally if the setting is less than that.

cygwin	This setting is used only with agents on Windows.
	This setting enables the agent to work on a Windows host using Cygwin, a Linux-like environment. When using Cygwin, a number of Linux tools are available to the agent.
	When you use this setting, you might need to set cygwin_script_magic and shell settings also. The example shows one way to configure these settings: :
	`cygwin`
	`shell C:\cygwin\bin\bash.exe --login -c "%s"`
	`cygwin_script_magic #!/bin/bash`
	The shell setting must match your installation of Cygwin.

cygwin_script_magic	This setting is used only with agents on Windows when **cygwin** is set. This setting specifies the `#!` line to use when executing steps. The default is `#!/bin/bash`.
default_logon_domain	Specifies the domain to use when an authentication request does not include a domain. If not specified, the agent machine's domain is used.
disable_telnet	For best results, use telnet to test the agent connection. For the agent, there is some built-in processing overhead associated with processing and correctly handling telnet control sequences. Use this setting to disable the agent from handling special telnet character codes which can slightly improve performance. In product environments, use this setting to benefit from the improved performance.

disable_transcode	Turns off processing that the agent performs to convert international data when the operating system is not using UTF-8 encoding. To avoid mixed encodings and data corruption, use UTF-8 for the agent operating system. If the operating system does not use UTF-8 encoding, the agent must convert data to the correct encoding for the operating system's locale settings. If your operating does not use UTF-8, use this setting for best results and improved performance of the agent.
enable_agent_dll	This setting enables DLL process tracing, which is a debugging tool.
env_recursion_limit number-of-recursions	Sets the variable-replacement recursion limit for pre-parsing. If not set, the limit is 32.
extensions	This setting specifies paths to external libraries of functions. The functions can be used as dot commands in a step. If this setting is not specified, external libraries are not loaded. During parsing, the first token in the step command is taken as the function name. The second token is a string, and the third is an integer timeout value (in seconds). Requirement: Dynamic loader support in the operating system. For example, in UNIX or Linux you need /usr/include/dlfcn.h. These defaults values are used internally. • UNIX or Linux: /usr/local/bin/bfextensions.so • Windows: c:\program files\ibm\build forge\agent\bfextensions.dll
getaddrinfo_using_addrconfig	This setting is used only for running the agent as a standalone service on UNIX or Linux operating systems (bfagent -s). This setting makes the agent use AI_ADDRCONFIG when calling getaddrinfo() to select a listening interface. By default AI_ADDRCONFIG is not used. If you use this setting, the agent ignores interfaces that do not have a properly configured address. It listens only for interfaces that have a properly configured address.
lang *lang-code*	Use this setting only when the Management Console does not provide a valid language. This setting specifies the language that the agent uses to write messages and command output. Typically it is not set explicitly because the agent uses the language that the Management Console specifies. However, setting the language can be useful if the desired locale is not available on the computer. The

	setting is also useful as a backup, in case the Management Console fails to communicate a language or communicates an invalid language. The internal default is `en`, as if it were explicitly set as follows: `lang en`
leave_tmp_file	Use this setting only while you are troubleshooting. This setting causes the temporary file that is used to hold step commands to be retained, rather than deleted after command execution. In troubleshooting, the file can be compared to the steps as they are displayed in the Management Console. **Note** Do not use this setting for typical operations.
locale locale-code.charset-code	This setting is used only with UNIX and Linux operating systems. Windows handles locales differently. This setting specifies the language and multibyte character set that localized applications use. This setting works by setting the LANG environment variable for the agent context. To set up the agent to treat command output as US English UTF-8, use the UTF-8 locale for your operating system. For example, on Linux use the following representation. `locale en_US.UTF-8` To determine the correct representation of the UTF-8 locale for your operating system, run the **locale -a** command. If this setting is not specified, the agent uses the locale of the operating system. This setting is a convenience. This setting is especially useful if the default locale of the operating system is not the locale that you want the agent to use. The setting is especially useful if changing the system locale to meet agent requirements is not practical.
magic_login user:encoded-password	The agent typically uses administrative privileges such as root or admin to log on to the operating system. The magic_login setting is an alternative to standard system authentication. With this setting, the system can authenticate your login with a single user name and password. If the agent is run as the root or admin user, this setting is ignored and normal authentication is attempted.

The agent runs all commands using the permissions of the user who started the agent, not the user name used to log in.

This setting is used in only these situations:

- When running the agent with administrative privileges is not possible. For example, use this setting with UNIX systems that do not work with PAM.
- When running the agent with administrative privileges is not permissible because of security policies.

To configure a login for the agent:

1. To create a server authentication that uses a user name and password. In the Management Console, click **Servers** → **Server Auth** .
2. For this example the user name is `build` and the password is `MySecretPassword.`
3. Create a server that uses the agent. Associate the server authentication with this server in the **Authentication** field.
4. Generate an encoded password for the agent. In the installation directory for the agent, run **bfagent -P** with the password that you choose.

   An SMD5 hash-encoded password is returned, as follows:

   ```
   bfagent -P "MySecretPassword"
   eca0b7f2f4fbf110f7df570c70df844e1658744a4871934a
   ```

5. In `BFAgent.conf`, set magic_login to use the desired user name and encoded password.

   ```
   magic_login
   build:eca0b7f2f4fbf110f7df570c70df844e1658744a4871934a
   ```

6. Start the agent.
7. Test the server connection. In **Servers** , select the server, and then click **Test Server** .

map *drive-and-user-spec*[; ...]

This setting specifies a mapped drive. Some systems might require drive mappings. For example, a drive mapping might be required because a shell is run from a shared drive. Mappings specified on the agent are performed before mappings specified by _MAP environment variables in the Management Console. This example illustrates two drive mappings:

	`map X:=//host1/share;Z:=//host2/share(username,password)`
map_drive_is_failure	When specified, this setting causes a step to fail upon encountering an unmapped drive specification, before step execution. If this setting is not specified, steps ignore drive failures and attempt to run the step. In that case ensure that the failure generates a meaningful error message.
no_preparse_command	This setting disables the variable-expansion parsing that the agent typically performs on a command before passing the command to the shell. See also the _NO_PREPARSE_COMMAND environment variable, which can be used for an single project or step.
no_pty	This setting is used only with agents that are running on UNIX or Linux systems. This setting can be used to help prevent the system shell from locking up when the shell interacts with the pseudoterminal of the agent. This setting is typically used with HP/UX and z/OS. You can also use two other methods to help prevent this kind of lockup: • Use an alternate shell. • Use the nologonshell setting The **no_pty** setting disables the pseudoterminal allocation. **Note** Using **no_pty** affects some commands. For example, the **ls** command returns output in a single column rather than three columns. If you use this setting, test thoroughly before you deploy the job to a production environment.
nologonshell	Use this setting only with agents that are running on UNIX or Linux. This setting causes the shell that the agent runs to be a normal shell, not a logon shell. This setting is often in these cases: • The logon shells provide verbose output. • The logon shells change environment settings in unwanted ways. • The logon shells attempt to communicate interactively with the user.

When set, standard methods of requesting that the shell be a normal shell rather than a logon shell are used. This may not work on all platforms and in such cases, the shellflag setting may be used to pass flags to the shell in order to modify its behaviour.

Those behaviors are not desirable for the agent, because it runs as a user without being an interactive user.

Note

The Mac OS X 10.5 system uses /bin/bash, which does not respond to nologonshell. Use `shellflag -l`.

Note

The z/OS operating system always uses the /etc/profile script for both logon shells and non-logon shells. You might need to change the contents of the script or use another shell if its behavior does not work well with the agent.

See also the **shellflag** setting. Flags can be used to change logon script behavior.

password_encrypt_module *dll_path*;*conf_path*

Required to enable SSL on the agent. It specifies paths to a DLL and configuration file.

- *dll_path* is the path to `bfcrypt.dll` (it is typically `./bfcyrpt.dll`).
- *conf_path* is the path to `bfpwcrypt.conf` (it is typically `./bfcrypt.conf`).

port *port-number-or-range* [...]

This setting is used only with agents that are running in standalone mode on UNIX or Linux when you issue the **-s** option on startup.

This setting specifies the port that the agent uses to listen for connections with the Management Console.

- Agents running in standalone mode on UNIX or Linux (-s option on startup).

Specifies the port that the agent uses to listen for connections with the Management Console.

Note

The port is set to 5555 by default. For UNIX or Linux the setting is in `/etc/services`.

shell *shell_name* [*options*]

This setting specifies the default shell. Internal defaults are as follows:

- Windows: `shell cmd.exe /q /c "%s"` unless the following settings are used:
    - If the `cygwin` setting is used, the default is `shell C:\cygwin\bin\bash.exe --login -c "%s"`
    - If the `cygwin` setting is not used, the default is `shell cmd.exe /u /q /c "%s"`
- UNIX or Linux: The shell set for the user account, or `/bin/sh` if the user's shell can not be determined. Note that you cannot specify parameters in this setting, but you can use the `shellflag` setting to pass them. The agent automatically forces the default to be a logon shell by inserting a hyphen. For example, `/bin/ksh` is sent as `-ksh`. If `shell` is set explicitly, then `nologonshell` is set implicitly. See `nologonshell`.
- *System i*: Set the shell value to /bin/sh

You can override this setting from within a step. A step that starts with a line containing `#!` overrides the shell setting and the `nologonshell` setting is used to run the step commands.

shell_compatible_undef_vars

This setting forces the representation of undefined variables to be an empty string. If not set, the representation is the variable name for variables of format $VAR, ${VAR}, or %VAR% and the empty string for $[VAR].

shellarg

This setting is used only with agents that are running on UNIX or Linux.

Use this setting if it seems that commands are being scrambled. Some shells on Red Hat Linux Enterprise require this setting.

The setting changes the way a command script is passed to the shell. Normally the script is passed through standard input:

```
   /bin/sh < /tmp/bfshellscript.sh
```

This setting causes scripts to be run by passing them as parameters:

```
/bin/sh /tmp/bfshellscript.sh
```

shellflag *flag*

This setting is used only with agents that are running on UNIX or Linux.

	This setting adds a flag when a shell is running. Only one flag can be specified. It is typically used to disable **rc** script processing in order to reduce output or undesired processing. Examples: • csh and derivatives: use `shellflag -f` to disable rc script processing. • bash: use `shellflag --noprofile` to disable profile script processing.
ssl_ca_location *path*	Specifies the keystore file that contains the certificate authority. If the agent runs as a service, use an absolute path.
ssl_cert_location *path*	Specifies the keystore that contains the private certificate. If the agent runs as a service, use an absolute path.
ssl_client_authentication [true | false]	Set to true to require client authentication when a connection is made to the agent. If true, the Build Forge engine's certificate must be added to the agent's certificate authority keystore.
ssl_cipher_group [*grouplist* | ALL]	Specifies individual cipher groups to use. Can be set to ALL.
ssl_cipher_override *cyphers*	Overrides the cipher group. Specify the ciphers to use.
ssl_key_location *path*	Specifies the keystore file that contains the key. If the agent runs as a service, use an absolute path.
ssl_key_password *password*	Password for the key. This property is stored in clear text by default. You can enable the agent to encrypt this password using its own key or the Build Forge server's key.
ssl_protocol *protocol*	The SSL handshake protocol to use, one of SSL, SSLv2, SSLv3, SSL_TLS, TLS, TLSv1. The protocol must match the protocol used by the Build Forge server.
update_path *path*	This setting identifies the full path to the Build Forge agent executable. The setting is established automatically during installation. The directory is a default directory for the operating system or the installation directory that you specify. **Note** This setting is ignored on Windows agents. The update path is taken from registry keys. The keys are set during agent installation.
win_reexec_after_auth	Add this setting if you need to run agent commands on a network share file system using Build Forge server authentication credentials. For example, to modify files in a ClearCase dynamic

view, the agent must access ClearCase files on a networked shared file system.

The Build Forge agent initially starts up with Windows system account credentials. To run commands, the agent later authenticates with Windows using Build Forge server authentication credentials.

Without this setting, the network share recognizes only the initial Windows system account credentials and ignores the subsequent server authentication credentials that are needed to access and write to files on the network share file system.

The win_reexec_after_auth starts a new process after authenticating with Windows again by using the server authentication credentials and forces the shared file system to recognize the changed credentials.

When you use the win_reexec_after_auth setting, the agent runs as a service and does not distinguish between commands that access network share files and those that do not, so you might notice a performance impact.

Troubleshooting agents

This section describes procedures that you can use to troubleshoot agents that are not working correctly. Go through the procedures in order. If you are unable to get an agent working by using procedures, then contact Support.

Testing host name resolution

Verify that the agent host can be reached from the Management Console host. Use the ping utility from the Management Console host to test the agent host:

```
ping hostname
```

This example session runs on Windows on which both the Management Console and the agent are installed.

```
C:\> ping localhost

Pinging somehost.city.company.com [127.0.0.1] with 32 bytes of data:

Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128

Ping statistics for 127.0.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

A message similar to the following one indicates a problem:

```
Unknown host
```

The problem lies in the network configuration of the Management Console host. Contact your network administrator.

Testing the connection

You can test the connection to the agent by using telnet or using a test from the Management Console.

To test the connection from the command line:

1. Connect to the agent with a telnet command. If you are logged on to the host where the agent is running, you can use localhost as the host name.

   ```
   telnet hostname 5555
   ```

   This response indicates a successful connection:

   ```
   200 HELLO - BuildForge Agent v7.0.1.buildnumber
   ```

2. Check authentication by issuing the following commands, using your login credentials:

   ```
   telnet localhost 5555
   username user name
   password  password
   cmd ping
   go
   ```

   The following message indicates success:

   ```
   AUTH: set user account to <user name>
   ```

   If the above tests work but jobs are failing, and a test of your server shows a user authentication error, check the pluggable authentication modules (PAM) configuration. If you see a message like the following, proceed to the next step.

   ```
   AUTH: unable to set user account to <user name>: unknown account (1)
   ```

3. Type the following command:

   ```
   cmd ping go
   ```

   The following output of a telnet session is typical. In particular look for`RESULT 0` at the end of the output as an indication of success. This test ran on an agent running on Windows.

   ```
   300 DATA s 67
   AUTH: Running as: [SYSTEM] in domain [NT AUTHORITY] SID Type [User]
   300 DATA s 52
   AUTH: Running with Privilege: [Lock pages in memory]
   300 DATA s 66
   AUTH: Running with Privilege: [Adjust memory quotas for a process]
   300 DATA s 63
   AUTH: Running with Privilege: [Create permanent shared objects]
   ```

```
    300 DATA s 46
    AUTH: Running with Privilege: [Debug programs]
    300 DATA s 56
    AUTH: Running with Privilege: [Bypass traverse checking]
    300 DATA s 61
    AUTH: Running with Privilege: [Back up files and directories]
    300 DATA s 54
    AUTH: Running with Privilege: [Change the system time]
    300 DATA s 68
    AUTH: Running with Privilege: [Remove computer from docking station]
    300 DATA s 73
    AUTH: Running with Privilege: [Impersonate a client after authentication]
    300 HEARTBEAT 1
    300 DATA s 16
    PLAT: Windows XP
    250 RESULT 0
    PING: internal loopback test complete
    260 EOR
```

To test the connection from the Management Console:

Note

Use this method only if a Server has been set up that uses the agent.

1. Go to **Servers** .
2. In the list of servers, click the one to test.
3. Click **Test Connection**

After the test, the results are available in the **Test Results** tab.

A message similar to the following output indicates a problem.

```
Could not open a connection to host on port 5555
```

Either the Management Console is unable to connect to the host or the agent is not functioning.

Troubleshooting an agent on Windows

To troubleshoot an agent on Windows, do the following:

1. Check for installed executable files. Verify that these files are present in the agent installation directory:
   - `bfagent.exe`
   - `bfdispatch.exe`

Note

Before you continue, determine whether you have a custom configuration. If you have a custom configuration, save BFAgent.conf outside of the installation directory, and then restore it after installation.

2. Reinstall the agent. You can solve most Windows agent problems by reinstallation the agent. It updates the executable files and restores registry keys.

Troubleshooting an agent on UNIX, Linux, or MacOS

To troubleshoot an agent on UNIX, Linux, or MacOS, try these procedures:

- Run **bfagent** from a shell. The correct response is similar to this message:

  ```
  200 HELLO - Build Forge Agent v7.0.1.122
  ```

  If you receive a message similar to the example and there are shared library problems, you will receive messages regarding those problems. You can resolve most shared library problems by setting the path correctly.

- Check that the agent is listening. Use the following command (assuming that the port is the default, 5555):

  ```
  telnet localhost 5555
  ```

  A 200 HELLO response indicates that the agent is listening. If you do not get this response, check your systems network configuration. Verify that the **inetd** configuration is correct, or check with your Linux or UNIX system administrator.

- Check authentication. Issue the following commands, using your login credentials:

  ```
  telnet localhost 5555
  username <user name>
  password <password>
  cmd ping
  go
  ```

  A message similar to the following message indicates that the authentication is working correctly:

  ```
  AUTH: set user account to <user name>
  ```

  If the previous tests work but builds are failing, and a test of your server shows a user authentication error, check the pluggable authentication modules (PAM) configuration. If you see a message similar to the following message, proceed to the next procedure.

  ```
  AUTH: unable to set user account to user name: unknown account (1)
  ```

- Check the PAM configuration. Problems with the PAM configuration are common issues on AIX platforms. Depending on your operating system, PAM is configured in one of two ways: with a line in the `pam.conf` file or with a file in the`pam.d` directory.

Tip

Solaris 10 is an exception in the following procedure: Delete any lines that specify a module of `pam_dial_auth`, for example `pam_dial_auth.so.1`. Agent authentication does not work if that module is included.

1. Verify that /etc/pam.conf exists. If it does not, go to the instructions for pam.d later in this topic. If the file does exist, continue to the next step.
2. In the file, create an entry for bfagent.
3. Copy the lines for another application, for example, sshd or login, and then substitute bfagent for the [application] field.

   ```
   [application] [when] [mode] [module]
   ```

   The fields are as follows:

   - *[application]* is the name of the application that needs to authenticate user
   - *[when]* is the type of authentication request
   - *[mode]* is the mode of the authentication request
   - *[module]* is the authentication module to invoke. The following example shows entries copied from login to bfagent. Note that the module names may be different from system to system.

     ```
     bfagent auth requisite          pam_authtok_get.so.1
     bfagent auth required           pam_dhkeys.so.1
     bfagent auth required           pam_unix_cred.so.1
     bfagent auth required           pam_unix_auth.so.1
     ```

4. After you set up the PAM entries, try to log in again as described in step 3.
5. For more information, see PAM documentation at http://www.sun.com/software/solaris/pam

- To troubleshoot PAM which is configured in pam.d:

  1. Find the /etc/pam.d directory and note that it contains several fills, each named for an application. Within each file, each line is formatted this way:

     ```
     [when]  [mode] [module]
     ```

  2. Copying a file from another application, such as sshd or login, and rename it `bfagent`.
  3. After you set up the PAM entries, try to login again as described in step 3.
  4. For more information, see PAM documentation at http://www.sun.com/software/solaris/pam/

Post-installation tasks

This section describes tasks to perform after a successful installation.

It discusses the following topics:

- Starting and stopping the engine
- Setting up users
- Verifying the installation
- Troubleshooting common problems

Starting and stopping the engine

You may want to stop the engine when you perform a backup of its database or when you install an updated version of the system. The following sections describe how to start and stop the engine.

Starting and stopping the engine on Windows

On Windows:

- At **Start** → **Programs** → **IBM Rational Build Forge Management Console** , choose either
  - **Start Engine Service**
  - **Stop Engine Service**

Control Panel: You can also use the **Administrative Tools > Services** control panel to start or stop the **IBM Rational Build Forge Management Console** service.

Running in the Foreground: If you have any problems getting the engine to run, run it in the foreground so that you can see the status and error messages it generates: **Start** → **Programs** → **IBM Rational Build Forge Management Console** → **Start Engine (Foreground)** . As the Management Console runs, log output is shown in a console window. To stop the engine in this mode, enter Ctrl-C in the console window.

Starting and Stopping the Engine on UNIX or Linux Systems

If you have an rc file installed (typically in `/etc/rc.d/init.d`):

```
$ /<path to rc file>/buildforge start
```

```
$ /<path to rc file>/buildforge stop
```

If you do not have an rc file installed, start engine using the following:

```
$ /<path to system installation>/Platform/buildforge.pl &
```

Stop it by determining its process ID and then issuing a kill command:

```
$ ps aux | grep buildforge
$ kill ${<PID>}
```

Setting Up Users

This section describes how to set up user accounts Build Forge® system.

The following topics are discussed:

- Root user
- Adding user accounts
- Read-only user for reports

Root user

The root user, the user whose login name is "root", has special characteristics:

- **Created upon installation**: The root user is the only default user that the installation program creates. The default password is "root". (Change the password immediately after installation).
- **No license required**: The root user does not consume a user license. No matter how many users are logged in, you can always log in as the root user. (When someone logs in as root, another user already logged in as root is logged off.)
- **System time zone**: The root user's time zone is the default time zone of the Management Console. The time zone for other users, both in-system and LDAP users, is taken from the root user's time zone by default. Users can set their own time zone after logging in once. All times and logs reported in the system are expressed in the user's time zone.

  By default, LDAP users are assigned the same time zone as the root user. However, they can edit the time zone assigned to them after they log in once. The system remembers the new preference.

  All permissions: The root user has all available permissions and can edit other users' properties. You cannot remove any access privileges from the root user. Although the root user is not a member of any access groups, the root user can view, edit, or use any data objects in the system.
- **Priority**: The root user is always a priority user.
- **Logging out current users**: The root user can log out users by clicking **Logout User** (Click **Administration** → **Users** and then click the user's name.
- By default, LDAP users are assigned the same time zone as the root user. However, they can edit the time zone assigned to them after they log in once. The system remembers the new preference.

Creating and editing users

You can create users and assign properties to them by using **Administration → Users** You can also connect your system to an LDAP/Active Directory database to get user information. You manage user security permissions by assigning users to groups, so you must create some users in order to test security features.

Click **Administration → Users** to display a list of current users. A user form is displayed after the list. The system displays the Name, Login, Email, Limit, Activity (elapsed time since last user activity), and Time Zone of each user.

- To edit a user, click the user's name and edit the properties in the user form, then click **Save**.
- To create a new user, specify properties in the user form when no user is selected. If a user is selected, click **Add User** to clear the form. Click **Save** when you are done editing user information.
- To log out a user, click the user's name and then click **Logout User** .
- To make a fixed license seat available, first log in as root. Click a user's name, then click **Purge Seat** . The console removes the user from the list of IDs that count toward the set of fixed licenses. The user is also logged off if he or she is logged in. For fixed licenses, the console counts the number of users who have ever logged in. When the limit is reached, no new users can obtain licenses. Existing users must be deleted or purged so that another user can use a fixed license. Purging a seat does not delete the user from the console. If the user logs back in, the fixed license count is increased. If you use **Purge Seat** on a floating-license user, the action has the same effect as clicking **Logout User** .
- To delete a user account, click the user's name and then click **Delete**.

  If **Delete** is disabled, the user account owns a scheduled job and you cannot delete the account. If you want to delete a user account that has scheduled jobs, first delete the scheduled jobs.

The user record sets default properties for the way that the user can interact with the system and controls the user's login name, password, and password expiration. You can provide the data for a user record to the system through the Management Console or the data can be derived from an LDAP/Active Directory database.

Note

When you edit a user whose record is derived from an LDAP database, many of the fields on the User page are disabled. You change these properties in the source database.

To add a new user, click **Add User**, edit the form, then click **Update**.

When you display a user record, three tabs are available:

- **Details** : Displays the user properties that you can edit. The available properties are described later in this topic.

- **Current Groups** : Displays the access groups that the user is a member of, either directly or through a group that is a member of another group.
- **Change Groups** : Displays the groups that the user is a direct member of, and allows you to add the user to groups or remove the user from them.

For each user, you can set the following properties on the **Details** tab:

Name	Specify the display name and label for the user.
Email	Specify the e-mail address where the system can send e-mail notifications to this user. **Note** e-mails are sent only to users explicitly selected for notification.
User Name	Specify the name that the user types to log on to the Management Console.
Password	Type the password that the user provides to log on to the Management Console. The field is not displayed for users who are logged on. Use this field to enter a new password or change one. Enter the same password in the **Verified** field.
Limit	Specify the maximum number of jobs the user can launch in a day. When the limit is reached, the system displays messages that indicate that the user's run quota has been exceeded. If the value is 0, the user can run any number of builds.
Time Zone	Specify the user's time zone. The system uses the time zone of the root user as the default time zone for all times posted. By default, in-system users and LDAP users are assigned the same time zone as the root user. Users can edit their assigned time zone. **Note** When upgrading to Build Forge 7.1 from a previous version, you will need to manually reset the time zone of the root user.
Verified	Repeat the password to verify that you type it correctly.
Priority Login	Specify whether the user is a priority user. A priority user can always log in to the system; if there are no more available user licenses, the system logs out the user with the oldest session so that the priority user can log in. The root user is always a priority user.

Date Format	Sets the user's preferred date format.
Language	Sets the user's preferred language.
Password Expires	Specify that the user's password will expire. If this option is checked, then the user's password expires after a number of days have elapsed, as specified in the **Password Expiration Days** system setting.
Uses screen reader	Enable the interface to support screen reader features such as dynamic highlighting and focusing.
Calendar Start Day of Week	Select the day of the week that the Schedule calendar displays first. The default is Sunday.

Verifying the Installation

This section describes how to test an installed and configured Build Forge® system.

It discusses the following topics:

- Creating a test project
- Setting up a server
- Running projects

Server Authentication

Use server authentications to associate login credentials to a server. You can use the same credentials for many servers, and update the credentials globally, by managing a set of server authentications.

A server authentication stores a login name and password as a single named object which you can associate with one or several servers. Use the Server Authentication page to create and edit server authentications.

Configuring Servers

To make best use of the system's dynamic server selection, you must set up a number of data objects, in a particular order. This topic outlines the minimum requirements for using servers.

1. Create server authentications.

   Server Authentications provide login names and passwords for servers. You can apply a server authentication to more than one server, so that you do not need to have unique logins for every server, and when you change a login, you can change it for a set of systems.

2. Create collectors for groups of servers.

Collectors both collect properties from servers, and assign properties to them. You can use a server without a collector. You must select a server based on default properties like BF_NAME.

3. Create selectors.

   Consider creating the following types of selectors:

   - Name-based: Create one selector for each server, which selects the server based on its host name. You can then choose servers by name.
   - Operating-system-based: Create one selector for each type of operating system in your environment, so that your projects can choose servers by operating system.
   - Capacity-based: You can get more specific by choosing servers based on available RAM or hard-disk space.

4. Install an agent on each machine that you intend to use as a Build Forge server.
5. Create a server in the Management Console for each machine you intend to use with Build Forge.
6. Test servers.

   Click **Test Connection** to test the connections to your servers. Review their manifests to make sure that they have the properties you expect.

Creating a Hello World Project

This topic describes how to create and execute a simple project to verify that Build Forge is set up properly.

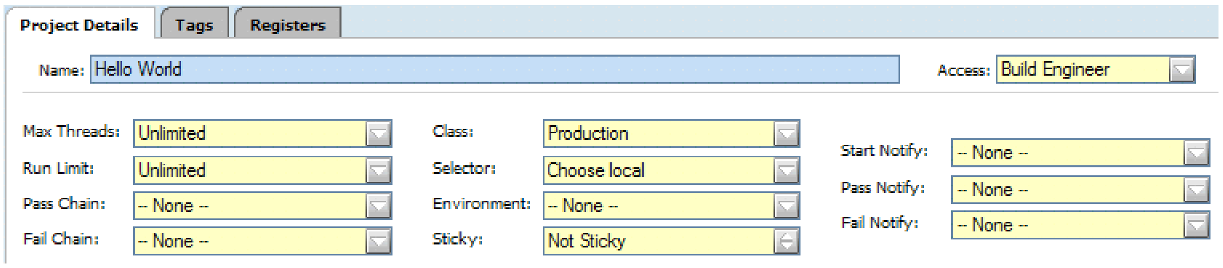
1. Create a server authentication, selector, and a server object.

   These objects are required before you can create a project.

2. Create a project named HelloWorld.

   Select **Projects** . In the Project Details form at the bottom of the main content pane, enter HelloWorld as the **Project Name** and choose a selector. Click **Save Project.**

Creating HelloWorld



Name: Hello World
Access: Build Engineer

Max Threads:	Unlimited	Class:	Production	Start Notify:	-- None --
Run Limit:	Unlimited	Selector:	Choose local	Pass Notify:	-- None --
Pass Chain:	-- None --	Environment:	-- None --	Fail Notify:	-- None --
Fail Chain:	-- None --	Sticky:	Not Sticky		

3. Add a step to the project named EchoHelloWorld.

   a. Select the HelloWorld project. The system displays the empty step list for the project and a blank Step Details form.

   b. In the Step Details form, enter a **Name** of EchoHelloWorld.

   c. In the Command field, enter a command line that will write “Hello World” to standard output on your chosen server.

      For example, the following command line works on Windows®, Solaris, Linux®, UNIX®, and Macintosh OS X systems:

      ```
      echo Hello World
      ```

      Then click **Save Step** .

4. Run the project.

   Select **Projects** to redisplay the project list, then click the ▶ icon next to the HelloWorld project. The system displays the **Running** tab of the **Jobs** module, with the HelloWorld project listed as running.

   ## Note

   If the HelloWorld project is not listed, skip to step 6.

5. Click the **Refresh** link at intervals until the Hello World project disappears from the Running list.

6. Click the **Completed** tab.

7. Click the job tag for the job.

   The default job tag for an initial job is BUILD_1. The system displays details for the run, with the step log at the bottom of the main pane.

8. Examine the log for the project.

   In most Hello World examples, you would see the "Hello World" text in a console window or pop-up window. The Management Console does its work by sending commands to the agent process on the targeted server; the agent then sends the output from those commands back to the Management Console, which stores them in the logs.

   The log has many sections; the relevant one is the final EXEC section. Click on it to display the results of your command:

   ```
   EXEC
   274 Jun 13, 2006 - 16:52 start [c:\BuildForgeTests\HelloWorld\BUILD_1@mcsystem] echo
    Hello World
   275 Jun 13, 2006 - 16:52 Performing variable expansion on command line
   276 Jun 13, 2006 - 16:52 Hello World
   277 Jun 13, 2006 - 16:52 end [c:\BuildForgeTests\HelloWorld\BUILD_1@mcsystem] echo
   Hello World (0)
   ```

This project demonstrates that you have configured your system correctly, that projects can successfully access a server, run and generate output on a server. The `echo` command can be replaced with any command that can be run on the target server.

Running Projects

There are several different ways to launch a project.

This task assumes that you have already created a selector, server, and project.

- While viewing the list of projects, click the ▶ icon in front of any project to start the project immediately.

  You cannot use this method if a project has no steps, or if it has any environment variables with the **Must Change** On Project action. Running a project this way uses its default values for selector, class, tags, and environment variables.

- While viewing the project's steps, click **Start Project** .

  This method displays the Start Project page for the project, where you can change project parameters, environment variable values, and select steps to exclude from the run:

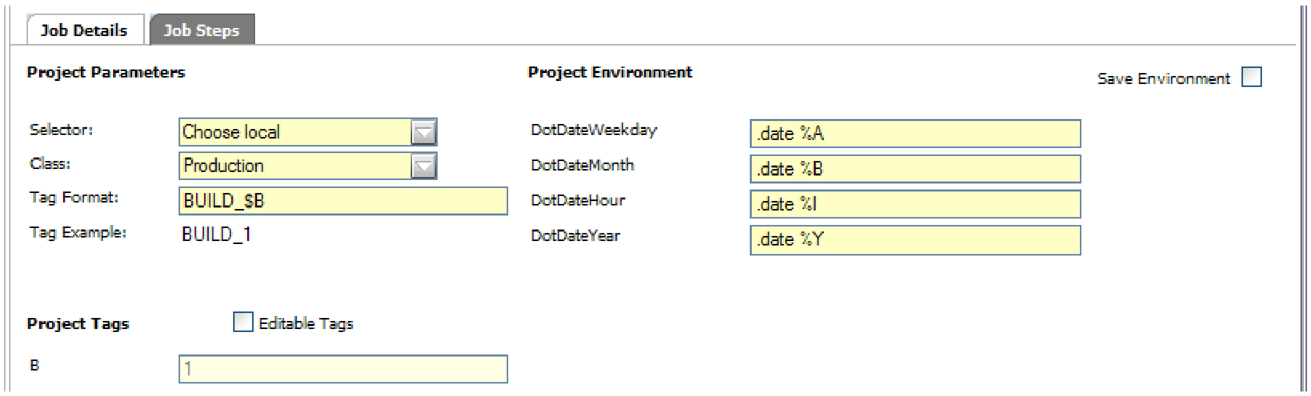
  - Select new values for project parameters.
  - Edit the project tag variable values.
  - Edit the project environment variable values. If you want your changes saved as the new defaults for these variables, click the **Save Environment** check box.
  - Select the Steps tab to display the list of project steps. You can select individual steps to exclude them from this run only.

  When you have made your choices, click **Execute** to start the project.

- Select **Jobs** → **Start** and then click the project name.

  As when you use **Start Project** , this method displays the Start Project page.

Start Project Page



Project Parameters

Selector:	Choose local
Class:	Production
Tag Format:	BUILD_$B
Tag Example:	BUILD_1

Project Environment

DotDateWeekday	.date %A
DotDateMonth	.date %B
DotDateHour	.date %I
DotDateYear	.date %Y

Project Tags

B	1

While a project is running, view the **Jobs → Running** page to check the project status.

To view job results, select **Jobs → Completed** to display the completed jobs. Click the Tag Name to access options for viewing job results.

Troubleshooting common problems

Upgrading an agent on Solaris requires running pkgrm command

Use the `pkgrm BFAgent` command to remove the existing Solaris Build Forge agent before running the `pkgadd` command.

URL for 7.0 notification templates might not work in later versions

The notification template URL opens the Build Forge job report when you click the URL link in the notification e-mail.

In versions 7.0.1 and 7.1, the URL in the notification templates changed; consequently, the URL might not work when you upgrade from 7.0 to a later version.

If you experience a link error, complete these steps to manually edit the notification templates:

1. Select **Project > Templates** .
2. Click the notification template name to display its properties on the Details tab.
3. In the Body field, locate the URL for the template. The URL should be similar to the one in the following example:

   ```
   http://${CONSOLEHOST}:${CONSOLEPORT}/fullcontrol/index.php?mod=projectruns&action=
   edit&bfid=${PID}&bfid=${BID}&bfid=${UID}
   ```

4. Replace the following URL elements with the appropriate 7.0.1 and later URL elements:

URL elements	7.0.1 (and later) URL elements
projectruns	jobs
&amp;	&
action=edit	action=build.view
&bfid=${PID} \| &bfid=${BID} | &bfid=${UID}	&bf_id=${BID}

Upgrade overview

The following sections describes how to upgrade components from a previous version to version 7.1.1.

The following topics are discussed:

- Upgrading Management Console
    - Upgrading a new version 7.1 console to version 7.1.1
    - Upgrading an upgraded version 7.1 console to version 7.1.1
    - Upgrading version 7.0.x console to version 7.1.1
    - bfdbdump reference
    - bfmigrateconfig reference
    - bfmigratehistorical reference
- Migration of pre-7.1 report designs for Quick Report
- Upgrading agents

Upgrading a new version 7.1 console to version 7.1.1

Use this section if you are upgrading a new version of 7.1 to version 7.1.1. A new version of 7.1 is one that is not an upgrade from 7.0.x

To upgrade, perform an update install. There are no other tasks to perform.

Performing an update installation

If you have modified the Tomcat configuration file `httpd.conf`, note that you must reapply those modifications after performing the update. The update process overwrites this file.

To perform an update installation, do the following:

1. Stop Build Forge if it is running.

   Then, if you are running on Windows, check the installation directory for a file named `buildforge.pid`. If it exists, remove it. Installation Manager is not be able to perform the update if it exists. It is most likely to exist if you have run Management Console in the foreground.

2. Start IBM Installation Manager.

3. Before updating, you must set your repository URL to the updated repository. See Specifying the repository URL for instructions.

4. In Installation Manager, click **Update** .

5. In Update Packages, select a package group for which to find updates, then click **Next** .

6. Select the updates or fixes to install. Select the 7.1.1 installation from the list, then click **Next** .

7. You are shown a list of versions to which you can update. Select the 7.1.1 installation from the list, then click **Next**

8. Select the *I accept the license terms* check box, then click **Next** .

9. The Update Packages page is displayed. Your features are selected for you. Click **Next** .

10. On the following page, you are presented with the information you edited in your buildforge.conf file. You are asked the following:

    a. *Do you want the installer to make the required database modifications?* Select this check box. It causes the installer to make the required changes to your schema. If this check box is not selected, you must run `bfschema -u` manually after installation in order to apply schema changes.

   b. *Do you wish to start the console after upgrade?*. Select this check box to have the console start automatically after installation is complete.

11. Click **Next** .
12. The features you have selected to install are listed.
13. Click **Update** to begin the installation.
14. When the update has been successfully performed, click **Finish** .
15. Close Installation Manager. Start Build Forge version 7.1.1.

Upgrading an upgraded version 7.1 console to version 7.1.1

Use this section if you are upgrading a previously upgraded version of 7.1 to version 7.1.1. An upgraded version of 7.1 is one that you upgraded from 7.0.x.

To upgrade to version 7.1.1, do the following:

- Perform an update installation.
- Migrate historical data.

Note

During your previous upgrade from version 7.0.x to version 7.1, configuration data was migrated to the version 7.1 system. If you have made changes to the version 7.0.x configuration data since it was migrated, you need to make those changes in the version 7.1.1 system manually . Make the changes after you complete the upgrade from version 7.1 to version 7.1.1.

Performing an update installation

If you have modified the Tomcat configuration file `httpd.conf`, note that you must reapply those modifications after performing the update. The update process overwrites this file.

To perform an update installation, do the following

1. Stop Build Forge if it is running.

   Then, if you are running on Windows, check the installation directory for a file named `buildforge.pid`. If it exists, remove it. Installation Manager is not be able to perform the update if it exists. It is most likely to exist if you have run Management Console in the foreground.

2. Start IBM Installation Manager.
3. Before updating, you must set your repository URL to the updated repository. See Specifying the repository URL for instructions.
4. In Installation Manager, click **Update** .
5. In Update Packages, select a package group for which to find updates, then click **Next** .
6. Select the updates or fixes to install. Select the 7.1.1 installation from the list, then click **Next** .
7. You are shown a list of versions to which you can update. Select the 7.1.1 installation from the list, then click **Next**
8. Select the *I accept the license terms* check box, then click **Next** .

9. The Update Packages page is displayed. Your features are selected for you. Click **Next** .
10. On the following page, you are presented with the information you edited in your buildforge.conf file. You are asked the following:
    a. *Do you want the installer to make the required database modifications?* Select this check box. It causes the installer to make the required changes to your schema. If this check box is not selected, you must run `bfschema -u` manually after installation in order to apply schema changes.
    b. *Do you wish to start the console after upgrade?*. Select this check box to have the console start automatically after installation is complete.
11. Click **Next** .
12. The features you have selected to install are listed.
13. Click **Update** to begin your install.
14. When the update has been successfully performed, click **Finish** .
15. Close Installation Manager. Start Build Forge version 7.1.1.

Migrating historical data

To migrate historical data, follow the instructions in Migrating historical data

Upgrading a version 7.0.x console to version 7.1.1

Use this section if you are upgrading any 7.0.x version to version 7.1.1.

Overview

To upgrade any version 7.0.x to version 7.1.1, you will perform the following tasks:

1. Create new databases or schemas on the database server that contains the database for version 7.0.x.
2. Install version 7.1.1 on the host where you are running version 7.0.x.
3. Create configuration files for configuration data migration
4. Migrate configuration data
5. Troubleshoot the configuration data migration
6. Migrate historical data. This requires that you do the following:
   - Create new databases or schemas on an offline database server.
   - Install version 7.1.1 on an offline host.

Note:

- *Configuration data* includes definitions of servers, environments, projects, users, and related objects.
- *Historical data* is data that is produced as a result of running jobs: job results, logs, and related objects.

The following information is not migrated during the upgrade process. It must be entered into the upgraded system manually.

- The root user's password
- Settings in **Administration** → **System**

Working with maintenance windows

The upgrade process can take a long time to run. However, it is designed to be accomplished in a few successive maintenance windows. A maintenance window of three hours is assumed. The list of steps to perform is as follows:

1. Create new databases

2. Install version 7.1.1 in two places: on the production system running version 7.0.x and on an offline system.
3. Create configuration files for configuration data migration
4. Migrate configuration data
5. Troubleshoot the configuration data migration
6. Migrate historical data

On most systems you should be able to perform the first five steps in a single maintenance window. You can also choose to perform steps 1–3 and steps 4–5 at different times.

Step 6 is made up of several sub-tasks. Three of them are performed on the version 7.0.2 database on the production database server.

- Export the version 7.0.x historical data from the database to files. You do this in order to take the files to the offline version 7.1.1 system for transformation.
- Import the transformed historical data back into the version 7.0.x database.
- Copy the historical data from the version 7.0.x database to the production version 7.1.1 database.

Each can be performed in one maintenance window. Smaller deployments might be able to accomplish more than one in a maintenance window.

Creating a new database or schema on the database server for version 7.0.x

This section describes how to create a new database or schema (depending on your database) and database users. During this process, you must do the following:

- Create the new database or schema BUILD711. This must be created in the same location as your current Build Forge database.
- Create a new database user BUILD711 for the new schema. The user must have GRANT OPTION for its tables.

If you are upgrading from a version earlier than 7.0.2 iFix2, you must also do the following:

- Create scratch database or schema BFSCRATCH.
- Create user BFSCRATCH for the BFSCRATCH database or schema. This user must have GRANT OPTION for its tables.

The instructions vary by database.

Note

All instructions assume that you used BUILD as the database name for your current Build Forge installation. Substitute the correct name if you changed it.

Important

You must install your new database or schema on the same system your current Build Forge schema resides. You may need to satisfy other database requirements, such as installing a database client on the host (DB2, Oracle) and performing other setup before and after installing Build Forge 7.1.1. See the following sections for your database:

- Pre-installation: Database setup
- Post-installation: Post-installation tasks

Creating a new database schema in DB2 and DB2 Express

The following commands creates the new schema, other needed objects, and grants. Substitute your current database name for BUILD if necessary.

```
db2 "CONNECT TO BUILD"
db2 "CREATE SCHEMA BUILD711"
db2 "GRANT  CREATETAB,CONNECT,IMPLICIT_SCHEMA ON DATABASE TO USER BUILD711"
db2 "GRANT CREATEIN,DROPIN,ALTERIN ON SCHEMA BUILD711 TO USER BUILD711 WITH GRANT OPTION"
db2 "CREATE BUFFERPOOL BFBP1 IMMEDIATE  SIZE 1000 PAGESIZE 16 K"
db2 "CREATE SYSTEM TEMPORARY TABLESPACE BFTMP2 PAGESIZE 16 K MANAGED BY SYSTEM USING
('[db2_path]\SQL0006.0') EXTENTSIZE 64 OVERHEAD 10.67 PREFETCHSIZE 64 TRANSFERRATE 0.04
BUFFERPOOL BFBP1"
db2 "CREATE  USER TEMPORARY  TABLESPACE BFUSE_TEMP2 PAGESIZE 16 K  MANAGED BY SYSTEM  USING
 ('[db2_path]/SQL0007.0' ) EXTENTSIZE 64 OVERHEAD 10.67 PREFETCHSIZE 64 TRANSFERRATE 0.04
 BUFFERPOOL  BFBP1"
db2 "GRANT USE OF TABLESPACE BFUSE_TEMP2 TO USER build711 WITH GRANT OPTION"
db2 "CREATE  REGULAR TABLESPACE BFUSERSPACE PAGESIZE 16 K  MANAGED BY SYSTEM  USING
('[db2_path]/SQL0008.0' ) EXTENTSIZE 64 OVERHEAD 10.67 PREFETCHSIZE 64 TRANSFERRATE 0.04
BUFFERPOOL  BFBP1"
db2 "GRANT USE OF TABLESPACE BFUSERSPACE TO PUBLIC WITH GRANT OPTION"
db2 "GRANT USE OF TABLESPACE BFUSERSPACE TO USER build711 WITH GRANT OPTION"
db2 "CONNECT RESET"
```

If you are upgrading from a version before 7.0.2 iFix2, perform these additional steps:

1. Create user BFSCRATCH on your operating system. During installation you are prompted for this user name and a password.
2. Create a database schema named BFSCRATCH.

The following commands creates the new schema, other needed objects, and grants:

```
db2 "CONNECT TO BUILD"
db2 "CREATE SCHEMA BFSCRATCH"
db2 "GRANT  CREATETAB,CONNECT,IMPLICIT_SCHEMA ON DATABASE  TO USER BFSCRATCH"
db2 "GRANT CREATEIN,DROPIN,ALTERIN ON SCHEMA BFSCRATCH TO USER BFSCRATCH WITH GRANT OPTION"
db2 "GRANT USE OF TABLESPACE BFUSE_TEMP2 TO USER BFSCRATCH WITH GRANT OPTION"
db2 "GRANT USE OF TABLESPACE BFUSERSPACE TO USER BFSCRATCH WITH GRANT OPTION"
```

Creating a new database in MySQL

To set up your existing MySQL database for upgrading to version 7.1.1, do the following:

1. Create a MySQL user named BUILD711. During installation you are prompted for this user name and its password.
2. Create a MySQL database named BUILD711.
3. Grant rights to grant to the current database user with grant option (BUILD, in the example) for all tables in the existing database (BUILD, in the example)

The following commands add the database, add the user, and add permissions for both the new user and the user for the previous version's database:

```
create database build711;

grant all on build711.* to 'build711'@'localhost' identified by 'password' WITH GRANT
OPTION;
grant all on build.* to 'build'@'localhost' identified by 'password' WITH GRANT OPTION;
grant all on build711.* to 'build'@'localhost' identified by 'password' WITH GRANT OPTION;
grant all on build.* to 'build711'@'localhost' identified by 'password' WITH GRANT OPTION;
```

If you are upgrading from a version before 7.0.2 iFix2, you need to create an additional databases and users:

```
create database bfscratch;
create database build;
create database build711;

grant all on bfscratch.* to 'bfscratch'@'localhost' identified by 'password' WITH GRANT
OPTION;
grant all on build.* to 'build'@'localhost' identified by 'password' WITH GRANT OPTION;
grant all on build711.* to 'build711'@'localhost' identified by 'password' WITH GRANT
OPTION;
```

Creating a new database schema in Microsoft SQL Server

To set up your existing SQL Server database for upgrading to version 7.1.1, do the following:

- Create a new SQL Server Login. Database Server Administrator's permissions are required in order to do this. The example script uses user name "BUILD711" and password "build". Substitute the name and password to use. The Build Forge version 7.1.1 installer asks for this information later.
- Create a new schema in the database you created for Build Forge 7.0.2. The example script uses database name "7025" and schema name "BUILD711". Substitute the name of your database. The 7.0.2 schema name is set to dbo by SQL Server unless you provided a schema name when you created the database.
- Assign the BUILD711 user to the 7.0.2 database
- Set the user's default schema to BUILD711.

- Give the BUILD711 user the role of "db_owner" on the 7.0.2 database. The role gives the user the permissions needed for both schemas during migration.
- Set READ_COMMITTED_SNAPSHOT to ON for the database. This setting is required.

The following example script illustrates the actions described above.

```
USE [master]
GO

CREATE LOGIN [BUILD711] WITH PASSWORD=N'build', DEFAULT_DATABASE=[7025],
DEFAULT_LANGUAGE=[us_english], CHECK_EXPIRATION=OFF, CHECK_POLICY=OFF
GO

USE [7025]
GO
CREATE USER [BUILD711] FOR LOGIN [BUILD711]
GO
CREATE SCHEMA [BUILD711] AUTHORIZATION [BUILD711]
GO
ALTER USER [BUILD711] WITH DEFAULT_SCHEMA=[BUILD711]
GO
EXEC sp_addrolemember N'db_owner', N'BUILD711'
GO
ALTER DATABASE [7025] READ_COMMITTED_SNAPSHOT ON
GO
```

If you are upgrading from a version before 7.0.2 iFix2, perform these additional steps:

1. Create user BFSCRATCH on your operating system. During installation you are prompted for this user name and a password.
2. Create a database schema named BFSCRATCH.

The following commands creates the new schema, other needed objects, and grants:

```
USE [master]
GO

CREATE LOGIN [BFSCRATCH] WITH PASSWORD=N'bfscratch', DEFAULT_DATABASE=[7025],
DEFAULT_LANGUAGE=[us_english], CHECK_EXPIRATION=OFF, CHECK_POLICY=OFF
GO

USE [7025]
GO
CREATE USER [BFSCRATCH] FOR LOGIN [BFSCRATCH]
GO
CREATE SCHEMA [BFSCRATCH] AUTHORIZATION [BFSCRATCH]
GO
ALTER USER [BFSCRATCH] WITH DEFAULT_SCHEMA=[BUILD711]
GO
EXEC sp_addrolemember N'db_owner', N'BFSCRATCH'
GO
ALTER DATABASE [7025] READ_COMMITTED_SNAPSHOT ON
GO
```

Creating a new database user and adjusting table sizes in Oracle

To set up your existing Oracle database for upgrading to version 7.1.1, do the following:

- Adjust TEMP and UNDO tablespace sizes to at least twice the size of the bf_store table. Additional adjustment of UNDO may be needed if bfmigrateconfig does not run successfully.
- Create an Oracle user named BFMIG711. During installation you are prompted for this user name and its password. Create it in the same schema used by Build Forge.

The following commands add the database user and the user's default schema, and adds permissions for the user:

```
create user bfmig711 identified by bfmig711
default tablespace db_migration
temporary tablespace temp;
grant create table to bfmig711;
grant create session to bfmig711;
alter user bfmig711 quota unlimited on db_migration;
```

If you are upgrading from a version before 7.0.x iFix2, perform these additional steps:

- Create an Oracle user named BFSCRATCH.

The following commands adds the user and and adds permissions for the user:

```
create user bfscratch
    identified by password
    default tablespace users
    quota unlimited on users;

grant create session, create table
    to bfscratch;
```

After you have used the instructions above to install a new database or schema, you are ready to install Build Forge version 7.1.1.

Installing version 7.1.1

This section describes how to install version 7.1.1 as part of the upgrade process.

When you install version 7.1.1, the following points are important:

- Shut down Build Forge if it is running.
- Use the IBM Installation Manager to locate the version 7.1.1 files and perform the installation. Use the installation instructions in this document. See Installing the Management Console.
- If you are installing on the same host as the one where your current version of Build Forge is installed, then you need to specify different port numbers for several components during installation. If your current version is running, the installer detects the ports in use and prompts you where different ports are needed.

- If you are running on the same host as your previous installation, then you need to specify a different installation directory than you used for the previous installation. For example, if you used the default `Build Forge` directory for the previous installation, you could specify `Build Forge 7.1.1` for the version 7.1.1 installation.
- Specify during installation that you are using an existing database (as opposed to creating a new one). Provide the information for the version 7.1.1 database you created:
  - Database name: build71
  - Database user name: build71
  - Database user password: as you specified
  - Other information may be required during installation.
- In the Console Start Options panel, check **Do Not Start Build Forge** .
- When the installation is complete, do not start the Management Console until you have migrated the data successfully.

Creating configuration files for configuration data migration

This section describes how to create configuration files that the configuration data migration program uses. You must do the following:

1. Copy the previous version's `buildforge.conf` file to `migrate.conf`.
2. Move `migrate.conf` to the version 7.1.1 installation directory:
   - Windows: by default, `C:\Program Files\IBM\BuildForge`
   - UNIX or Linux: by default, `/opt/buildforge/Platform`

If you are upgrading from a version earlier than 7.0.x iFix2, you must also do the following in the version 7.1.1 installation directory:

1. Copy `migrate.conf` to `scratch.conf`.
2. Edit `scratch.conf` as follows:
   - db_username: change to `bfscratch`
   - db_password: change to the password you specified when creating the `bfscratch` user.
   - db_schema: add this paramenter. Use the same value as db_username.

     Note

     db_username value must be typed in upper case (e.g. BUILD711, not build711).

Migrating configuration data

This section describes how to run a migration program to migrate configuration data from your old installation to your version 7.1.1 installation.

Do the following:

1. Stop Management Console on your old installation. Accordingly, you should not have started the Management Console on the version 7.1.1 installation.

2. Run the migration program (`bfmigrateconfig`) from the Build Forge installation directory. The program copies data from your existing Build Forge database and migrates it to the version 7.1.1 database.

   - Windows:

     ```
     > cd C:\Program Files\IBM\BuildForge
     > bfmigrateconfig.exe -ms
     ```

   - UNIX or Linux:

     ```
     $ cd /opt/buildforge/Platform   # default directory
     $ ./bfmigrateconfig -ms
     ```

   The migration takes several minutes to run. The time requirement scales up with the number of objects to be migrated (projects, users, servers, and so on). At the end of a successful migration a `Migration Successful` message is shown.

3. Start the version 7.1.1 Management Console.

4. Test the migration. Run several projects and view the migrated projects, users, and servers.

Note

If the migration fails or the file `migrateissues.log` is created during the migration, consult the following troubleshooting section. Fix any issues with the data before starting Management Console. The `migrateissues.log` file is created in the directory where you ran the migration program.

The migration program automatically resolves most issues that result from new uniqueness constraints placed on the data. In a small number of cases for some objects the issues cannot be resolved automatically. Some database rows may need to be removed by hand. Consult support for guidance.

Troubleshooting the configuration data migration

This section describes what to do if you encounter problems in the migration process.

If the the file `migrateissues.log` is created, examine it. It may contain entries like the following:

```
'Primary Key constraint [bf_id] for table [bf_sysconfig] is non-unique'
```

In this case, a number of uniqueness constraints were added in order to support data replication.

To resolve the problem, contact support. See “Support” on page 259.

Note

A failed migration does not affect its ability to run. You can safely restart your current installation of Build Forge. Do not attempt to start your version 7.1.1 installation until configuration migration is successful.

Migrating historical data

This section describes how to migrate historical data (information related to prior builds) from version 7.0.x to version 7.1.1.

If you have a small data set, you can decide not to set up Build Forge with an online database server instance and follow the instructions in Performing historical data migration in place.

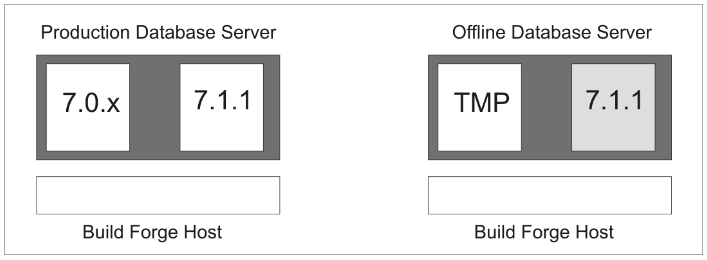
Prerequisites for migrating historical data

Historical data typically consumes a large amount of space due to the detail provided in job logs. Migrating historical data can take a significant amount of time, especially at sites where jobs are large and there have been a large number of jobs run. The migration is necessary in order to transform your existing historical data to use a new table architecture that was introduced in version 7.1.

The data transformation can take days or longer, depending on the size of the data. For this reason the historical migration process was designed to use offline resources, rather than operate on your production Build Forge installation. The prerequisites for running historical migration are as follows:

- Free space: exporting data from the version 7.0.x database requires a significant amount of free space, ranging from three to ten times the size of the existing database. Specific requirements are given in the historical data migration sections for each database.
- Offline database server instance: install a new database server instance on a host that runs nothing else.
- Build Forge 7.1.1: install Build Forge version 7.1.1 on a host that has access to the database server. This includes creating a new database or database schema on the new database server instance.
- Temporary database or schema: the historical migration instructions include instructions for creating a temporary database or database schema on the offline database server instance.

The following diagram shows the layout of Build Forge hosts, database server instances, and databases.



Production database server: on this server you have two databases.

- Version 7.0.x database. During configuration migration, the configuration data was migrated to the version 7.1.1 database. Staging tables were left in place. Those staging tables are used during the historical data migration process to map the historical data to the migrated projects in the version 7.1.1 database.
- Version 7.1.1 database. This database was created when you installed Build Forge version 7.1.1. All configuration data has been migrated to it and you started using it as your production system immediately after configuration data migration. You are currently using Build Forge 7.1.1 as your production system, running on the 7.1.1 database.

Important

If you ran jobs on the Build Forge 7.0.x system and 7.0.x database after configuration migration, the historical data for those jobs is not migrated.

Offline database server: on this server you have one database (version 7.1.1). You create a temporary database during historical data migration.

- Version 7.1.1 database. You create this database before starting the historical migration process. You install Build Forge version 7.1.1 on an offline host and configure it to use this database.

Note

This database is empty. No jobs are run on it. It is not used in the historical data migration process except to support the Build Forge 7.1.1 system.

- Temporary database. You create this database during the historical migration process. Depending on the database, it may be created explicitly in a separate step (example: DB2). It may also be created implicitly as an effect of data import (example: Microsoft SQLServer)/

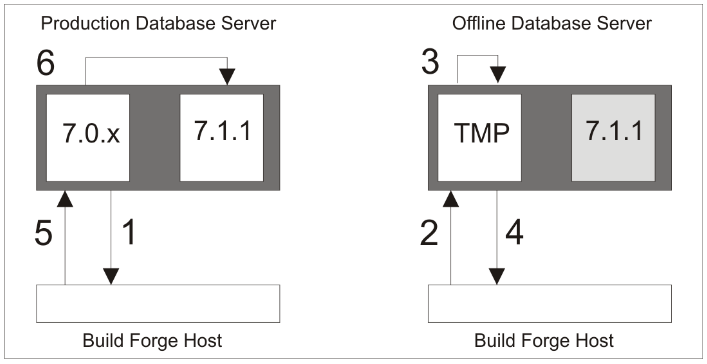
Note

If your database is small or it is acceptable to have the Build Forge system offline for an extended time, you can perform historical migration in place. See Performing historical data migration in place.

Overview of the historical data migration process

The following is an overview of the process for migrating historical data. Details within each step vary by database. Each database has its own section that gives the specific instructions for this process.

1. Export the historical migration data from your 7.0.x production schema to a file.
2. Import the file into a database on the offline database server.
3. Run the transform script `bfmigratehistorical -m` on the offline system. This process may take a significant amount of time to run, depending on database size.
4. Export the transformed data from the offline system to a file.
5. Import the file of transformed data into the 7.0.x database on the production database server.
6. Run `bfmigratehistorical -s` on the production system. It copies the data from the 7.0.x database to the 7.1.1 database.



The following tables are transformed into the new 7.1.1 table architecture:

- bf_builds
- bf_results
- bf_logs
- bf_filterevents
- bf_bom
- bf_interfacerelations
- bf_interfacecategories
- bf_interfacenotices
- bf_interfacesections

- bf_interfacefield
- bf_interfacedata
- bf_bom_manifests
- bf_logbuckets

Choice of scripts or database commands

For each export or import step, you have the choice of using the provided `bfdbdump` script or database commands.

- The `bfdbdump` can be used *only if Build Forge is installed on the same host as the database server*. You execute the command in the installation root of the Build Forge system.
- Database commands are executed the database server host. You need permissions to create files and perform the commands.

Performing historical data migration in place

If your database is small or it is acceptable to have the Build Forge system offline for an extended time, you can perform historical migration in place rather than set up an offline database server instance.

Do the following on the host where the Build Forge system is running.

1. Stop the Build Forge system.
2. Run the following command:

   ```
   bfmigratehistorical -ms
   ```

The command uses settings in the `migrate.conf` and `scratch.conf` files in order to access the database.

Note

Do not use this procedure is on production-scale systems. It is appropriate only for lab or test systems that have run very few jobs.

Migrating historical data in DB2

This section describes how to migrate historical data from the DB2 database used by your version 7.0.x system to the DB2 database used by your version 7.1.1 system.

Requirements

The following requirements must be met for migrating DB2 historical data:

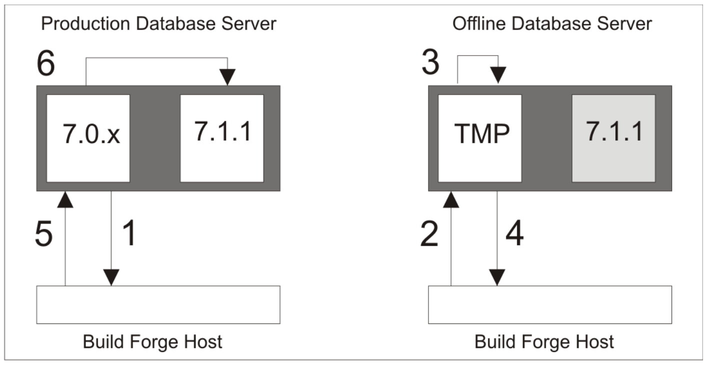
- Build Forge system: the system must be shut down.
- Offline system: A separate installation of Build Forge 7.1.1 and a DB2 database server must be set up on an offline system.
- Database server: the Build Forge database should not have any transactions running. The import commands attempt to lock each table. The process requires that you quiesce the database before each export and import. Database activity on other databases on the database server should be minimized.
- Disk space: the host where you perform the export must have enough free disk space. Up to 10 times the size of your Build Forge database may be required. Examine log policies and change them to minimize logging.
- Privileges: the user performing the import should have SYSADM or DBADM privileges. If the user does not have those privileges, the user needs privileges for individual steps:
    - Export: CONTROL or SELECT privileges on each of the tables to be export
    - Update: SELECT and INSERT privileges when updating an already existing table
    - Import: CREATETAB privileges to import data into an empty database

  See “DB2 setup” on page 24.

Migrating historical data in DB2 using scripts

This section describes how use scripts to migrate DB2 historical data.

You can use these steps only if Build Forge is installed on the DB2 database server host.



Before using the scripts, create a new database on the offline DB2 database server.

The following script creates a database named BFTEMP and a user named BFUSE_TEMP. A schema is created automatically using the default schema name, BFUSE_TEMP.

```
// Create database
db2 "CREATE DATABASE BFTEMP USING CODESET UTF-8 TERRITORY US AUTOCONFIGURE USING MEM_PERCENT
 40 APPLY DB ONLY"
db2 "CONNECT TO BFTEMP"
db2 "CREATE BUFFERPOOL BUFFP1 IMMEDIATE SIZE 1000 PAGESIZE 16384 NOT EXTENDED STORAGE"
```

```
db2 "CONNECT RESET"
db2 "CONNECT TO BFTEMP"

// Create tablespaces
db2 "CREATE SYSTEM TEMPORARY TABLESPACE TEMPSPACE2 PAGESIZE 16384 MANAGED BY  SYSTEM USING
 ('<path_to_database>/SQL003.0') EXTENTSIZE 64 PREFETCHSIZE 64 BUFFERPOOL BUFFP1"

db2 "CREATE USER TEMPORARY TABLESPACE BFUSE_TEMP PAGESIZE 16384 MANAGED BY SYSTEM USING
('<path_to_database>/SQL004.0') EXTENTSIZE 64 PREFETCHSIZE 64 BUFFERPOOL BUFFP1"

db2 "CREATE REGULAR TABLESPACE USERSPACE2 PAGESIZE 16384 MANAGED BY SYSTEM
USING ('<path_to_database>/SQL005.0') EXTENTSIZE 64 PREFETCHSIZE 64 BUFFERPOOL BUFFP1"

// User must be granted use of tablespaces
db2 "GRANT USE OF TABLESPACE BFUSE_TEMP TO USER BUILD WITH GRANT OPTION"
db2 "GRANT USE OF TABLESPACE USERSPACE2 TO USER BUILD WITH GRANT OPTION"
db2 "commit work;"
db2 "CONNECT RESET"
db2 "terminate"
```

1. **Export version 7.0.x data from DB2**

   a. Stop Build Forge if it is running.

   b. Run the following command in the `Build Forge` directory on the host where you are running the production Build Forge version 7.1.1:

   ```
   bfdbdump -e filename
   ```

   `filename` is the name of the *directory* where you want to place the exported files. Examples

   ```
   C:\temp\bf_alldata\   # Windows
   ```

   ```
   /usr/tmp/bf_alldata/  # UNIX and Linux
   ```

   c. Move the files to the host where you are running the offline Build Forge version 7.1.1 system.

2. **Import exported DB2 data to the offline database**

   a. Stop Build Forge if it is running.

   b. Copy configuration files.

   Copy `migrate.conf` and `scratch.conf` file (if it was used) from the Build Forge version 7.1.1 production host to the Build Forge version 7.1.1 offline host. Edit the following settings to match the settings you used when you created the offline database:

   - `db_hostname`
   - `db_user`
   - `db_password`

- `db_database`
- `db_schema`

c. Run the following command in the `Build Forge` directory on the host where you are running the offline Build Forge version 7.1.1:

```
bfdbdump -i filename
```

- *`filename`* is the absolute path to the *directory* where you placed the files. Examples:

```
C:\temp\bf_alldata\   # Windows

/usr/tmp/bf_alldata/  # UNIX and Linux
```

3. **Transform DB2 data**

a. Stop Build Forge if it is running.

b. Run the following command in the `Build Forge` directory on the host where you are running the offline Build Forge version 7.1.1:

```
bfmigratehistorical -m
```

This process may take a significant amount of time to run, depending on the amount of historical data. A progress indicator displays the number of records processed as the script runs.

4. **Export transformed DB2 historical data from the offline database**

a. Stop Build Forge if it is running.

b. Run the following command in the `Build Forge` directory on the host where you are running the offline Build Forge version 7.1.1:

```
bfdbdump -b -e filename
```

- `-b` specifies that *only* historical data is exported.
- *`filename`* is a full path name to a *directory* where the files are placed. Examples:

```
C:\temp\transformed\   # Windows

/usr/tmp/transformed/  # UNIX and Linux
```

c. Move the files to the host where you are running the production Build Forge 7.1.1 system. Place them in the directory you created in Step 1.

5. **Import transformed DB2 historical data to the version 7.0.x database**

a. Stop Build Forge if it is running.

b. Disable the limit on log space.

```
db2stop
db2set DB2_FORCE_APP_ON_MAX_LOG=false
db2start
```

This ensures that the operations will not fail once the log space limit is reached.

c. Run the following command in the `Build Forge` directory on the host where you are running the production Build Forge version 7.1.1:

```
bfdbdump -b -i filename
```

- `-b` specifies that *only* historical data is imported.
- *`filename`* is the absolute path to the *directory* containing the transformed historical data files. Examples:

```
C:\temp\transformed\   # Windows

/usr/tmp/transformed/  # UNIX and Linux
```

6. **Copy transformed DB2 historical data to the version 7.1.1 database**

a. Stop Build Forge if it is running.

b. Run the following command in the `Build Forge` directory on the host where you are running the production Build Forge version 7.1.1:

```
bfmigratehistorical -s
```

If the script fails, contact support.

Migrating historical data in DB2 using database commands

This section describes how to use DB2 commands to migrate historical data from the DB2 database used by your version 7.0.x system to the DB2 database used by your version 7.1.1 system.

Create an offline DB2 database

This section describes how to create a temporary database in an offline Build Forge 7.1.1 installation. The offline database is used to transform the data from your Build Forge version 7.0.x database.

1. Create a database on the offline DB2 database server.

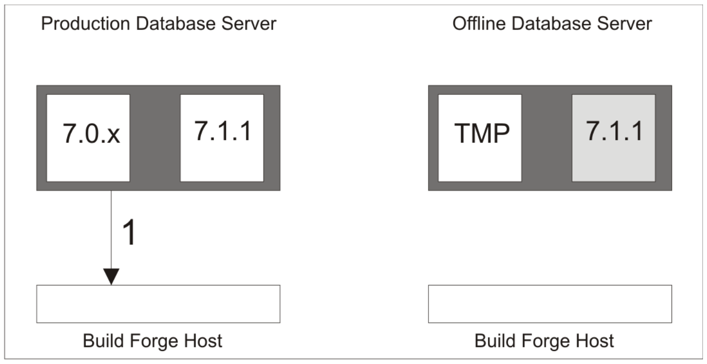
The following script creates a database named BFTEMP and a user named BFUSE_TEMP. A schema is created automatically using the default schema name, BFUSE_TEMP.

```
// Create database
db2 "CREATE DATABASE BFTEMP USING CODESET UTF-8 TERRITORY US AUTOCONFIGURE USING
MEM_PERCENT 40 APPLY DB ONLY"
```

```
    db2 "CONNECT TO BFTEMP"
    db2 "CREATE BUFFERPOOL BUFFP1 IMMEDIATE SIZE 1000 PAGESIZE 16384 NOT EXTENDED STORAGE"
    db2 "CONNECT RESET"
    db2 "CONNECT TO BFTEMP"

    // Create tablespaces
    db2 "CREATE SYSTEM TEMPORARY TABLESPACE TEMPSPACE2 PAGESIZE 16384 MANAGED BY  SYSTEM
     USING ('<path_to_database>/SQL003.0') EXTENTSIZE 64 PREFETCHSIZE 64 BUFFERPOOL BUFFP1"

    db2 "CREATE USER TEMPORARY TABLESPACE BFUSE_TEMP PAGESIZE 16384 MANAGED BY SYSTEM
    USING ('<path_to_database>/SQL004.0') EXTENTSIZE 64 PREFETCHSIZE 64 BUFFERPOOL BUFFP1"

    db2 "CREATE REGULAR TABLESPACE USERSPACE2 PAGESIZE 16384 MANAGED BY SYSTEM
    USING ('<path_to_database>/SQL005.0') EXTENTSIZE 64 PREFETCHSIZE 64 BUFFERPOOL BUFFP1"

    // User must be granted use of tablespaces
    db2 "GRANT USE OF TABLESPACE BFUSE_TEMP TO USER BUILD WITH GRANT OPTION"
    db2 "GRANT USE OF TABLESPACE USERSPACE2 TO USER BUILD WITH GRANT OPTION"
    db2 "commit work"
    db2 "CONNECT RESET"
    db2 "terminate"
```

Step 1: Export version 7.0.x data from DB2

Use these instructions to use DB2 commands to export Build Forge data from the version 7.0.x database.



You need the following information about the Build Forge version 7.0.x database to perform this step:

- Name of the database
- Name of the schema (by default it is the name of the user)
- User name and password for the database

Run the commands on the production database server.

1. Prepare the DB2 environment.
   - On Windows: `db2cmd`

- On UNIX or Linux: `DB2_HOME/db2profile`

2. List the Build Forge tables.

   Keep the list.

```
db2 "CONNECT TO db_name USER username USING password"
db2 "SELECT table_name FROM SYSIBM.tables_s WHERE table_name like 'BF_%' AND
is_insertable_into='YES' AND table_schema='schema_name'"
db2 "QUIESCE DATABASE IMMEDIATE FORCE CONNECTIONS"
```

   Example: if you installed build forge on Windows using the supplied DB2 Express database for 7.0.x and used the default names, the commands are as follows:

```
db2 "CONNECT TO BUILD USER build USING build"
db2 "SELECT table_name FROM SYSIBM.tables_s WHERE table_name like 'BF_%' AND
is_insertable_into=’YES’ AND table_schema='BUILD'"
db2 "QUIESCE DATABASE IMMEDIATE FORCE CONNECTIONS"
```

3. Export the tables.

   For each of the table names, use the following command to export the data from the given table:

```
db2 "EXPORT TO <filename> OF IXF MESSAGES log_filename SELECT * FROM
schema_name.table_name"
```

   For example:

```
db2 "EXPORT TO BF_JOBCOUNT.ixf OF IXF MESSAGES BF_JOBCOUNT-msgs.txt SELECT * FROM
BUILD.BF_JOBCOUNT"
```

   The *filename* and *log_filename* parameters must be absolute paths. The *log_filename* file contains log messages for the export command being run. The *schema_name* parameter specifies the name of the schema to which the tables belong. The *table_name* parameter specifies is the name of the table you are currently exporting.

```
db2 "EXPORT TO BF_JOBCOUNT.ixf OF IXF MESSAGES BF_JOBCOUNT-msgs.txt SELECT * FROM
BUILD.BF_JOBCOUNT"
```

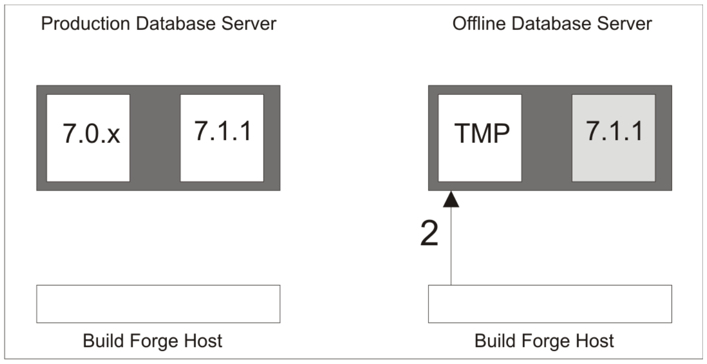
4. When you finish exporting the tables, unquiesce the database and log off:

```
db2 "UNQUIESCE DATABASE"
db2 "connect reset"
```

5. Move the exported files to the host where you are running the off-line Build Forge version 7.1.1 system.

Step 2: Import exported DB2 data to the offline database

This section describes how to use DB2 commands to import the data you exported from Build Forge 7.0.x to the offline temporary database.



You need the following information about the offline Build Forge version 7.1.1 database to perform this step:

- Name of the database
- Name of the schema
- User name and password for the database

Run the commands on the offline database server host.

1. Stop Build Forge if it is running.
2. Disable the limit on log space.

    ```
    db2stop
    db2set DB2_FORCE_APP_ON_MAX_LOG=false
    db2start
    ```

    This ensures that the operations will not fail once the log space limit is reached.

3. Connect to the database and quiesce it:

    ```
    db2 "CONNECT TO db_name USER username USING password"
    db2 "QUIESCE DATABASE IMMEDIATE FORCE CONNECTIONS"
    ```

4. Import the data.

    Run the following command on each .ixf file.

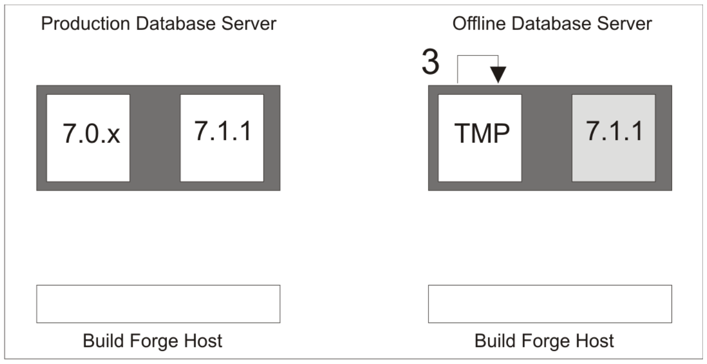
    ```
     db2 "IMPORT FROM filename of filetype COMMITCOUNT AUTOMATIC MESSAGES log_filename
    CREATE INTO schema_name.table_name"
    ```

    For example:

    ```
     db2 "IMPORT FROM BF_JOBCOUNT.ixf OF IXF COMMITCOUNT AUTOMATIC MESSAGES
    BF_JOBCOUNT-IMP-msgs.txt CREATE INTO BUILD.BF_JOBCOUNT"
    ```

Step 3: Transform DB2 data

This section describes how to transform the data that you imported into the offline temporary database.



1. Stop Build Forge if it is running.
2. Copy configuration files.

   Copy `migrate.conf` and `scratch.conf` file (if it was used) from the Build Forge version 7.1.1 production host to the Build Forge version 7.1.1 offline host. Edit the following settings to match the settings you used when you created the offline database:

   - `db_hostname`
   - `db_user`
   - `db_password`
   - `db_database`
   - `db_schema`

3. Run the following command in the `Build Forge` directory on the host where you are running the offline Build Forge version 7.1.1:

   ```
   bfmigratehistorical -m
   ```

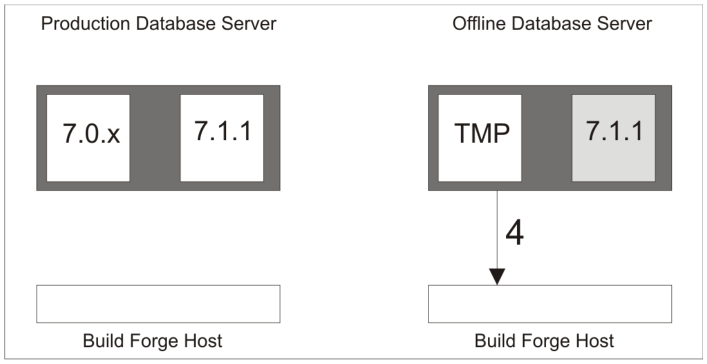
   This process may take a significant amount of time to run, depending on the amount of historical data. A progress indicator displays the number of records processed as the script runs.

   Note

   In case of failure, this process is recoverable. If the script terminates because of an error, re-run the script. It will continue without reprocessing data that has already been processed.

Step 4: Export transformed DB2 historical data from the offline database

This section describes how to use DB2 commands to extract and export transformed historical data from the offline temporary database. Only the historical data is exported.



You need the following information about the offline temporary database to perform this step:

- Name of the database
- Name of the schema
- User name and password for the database

Run the commands on the offline database server host.

1. Stop Build Forge if it is running.
2. Connect to the database and quiesce it.

   Run the following command on the offline database server host:

   ```
   db2 "CONNECT TO db_name USER username USING password"
   db2 "QUIESCE DATABASE IMMEDIATE FORCE CONNECTIONS"
   ```

   Above, *db_name* is the name of your off-line database, and *username* and *password* are the authentication credentials to connect to that database. The following example uses a database named BFTEMP, a user named build, and a user password of build.

   ```
   db2 "CONNECT TO BFTEMP USER build USING build"
   db2 "QUIESCE DATABASE IMMEDIATE FORCE CONNECTIONS"
   ```

3. Export the tables.

   On the offline database server host, run the following command on each table:

   ```
   db2 "EXPORT TO filename OF IXF MESSAGES log_filename SELECT * FROM
   schema_name.table_name"
   ```

   For example:

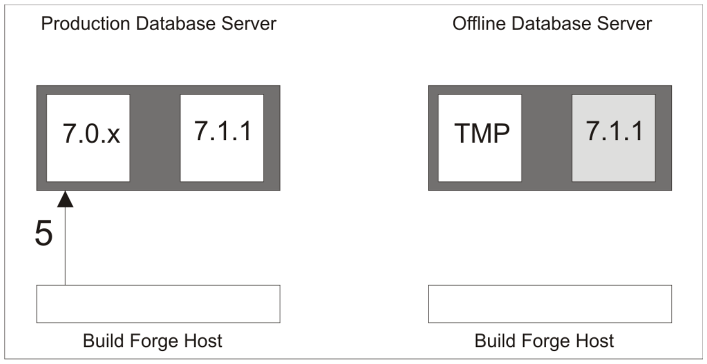
   ```
   db2 "EXPORT TO b7_messages.ixf OF IXF MESSAGES b7_messages-msgs.txt SELECT * FROM
   BUILD.b7_messages"
   ```

   The *filename* and *log_filename* file must be absolute paths. The *log_filename* file contains log messages for the export command being run. The *schema_name* parameter specifies the name of the schema to which the tables belong. The *table_name* parameter specifies is the name of the table you are currently exporting.

4. Move the exported files to production database server host.

Step 5: Import transformed DB2 historical data to the version 7.0.x database

This section describes how to use DB2 commands to import transformed historical data into the database for Build Forge version 7.0.x. Only historical data is imported.



You need the following information about the offline Build Forge version 7.0.x database to perform this step:

- Name of the database
- Name of the schema
- User name and password for the database

Run the commands on the production database server.

1. Stop Build Forge if it is running.
2. Disable the limit on log space.

```
db2stop
db2set DB2_FORCE_APP_ON_MAX_LOG=false
db2start
```

   This ensures that the operations will not fail once the log space limit is reached.

3. Connect to the version 7.0.x database and quiesce it:

```
db2 "CONNECT TO db_name USER username USING password"
db2 "QUIESCE DATABASE IMMEDIATE FORCE CONNECTIONS"
```

   Example: If you created BUILD as the schema and BUILD user to install the 7.0.x management console on a database named BUILD, then the commands are as follows:

```
db2 "CONNECT TO BUILD USER BUILD USING BUILD"
db2 "QUIESCE DATABASE IMMEDIATE FORCE CONNECTIONS"
```

4. Import the tables.

Run the following command on each .ixf file.

```
db2 "IMPORT FROM filename.ixf of IXF COMMITCOUNT AUTOMATIC MESSAGES log_filename
REPLACE_CREATE INTO schema_name.table_name"
```

For example:

```
db2 "IMPORT FROM b7_messages.ixf OF IXF COMMITCOUNT AUTOMATIC MESSAGES
b7_messages-IMP-msgs.txt REPLACE_CREATE INTO BUILD.b7_messages"
```

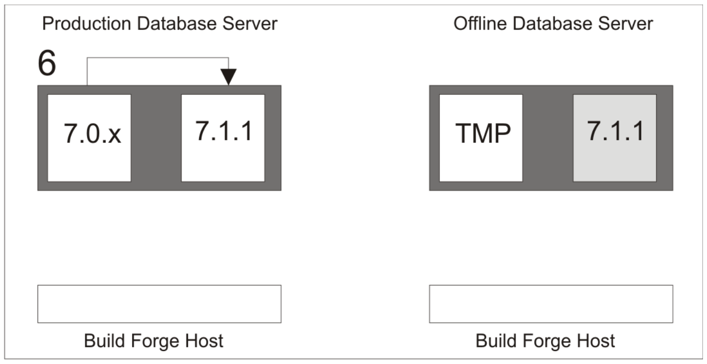
5. Check the messages file (`*-IMP-msgs.txt`) after each command to ensure the command was successful.

6. When you finished importing tables, unquiesce the database and log off:

```
db2 "UNQUIESCE DATABASE"
db2 "connect reset"
```

Step 6: Copy transformed DB2 historical data to the version 7.1.1 database

This section describes how to copy transformed historical data from the version 7.0.x schema to the version 7.1.1 schema. The database server must be running.



1. Stop Build Forge if it is running.
2. Run the following command in the `Build Forge` directory on the host where you are running the production Build Forge version 7.1.1:

```
bfmigratehistorical -s
```

Note

If the script fails, contact support.

Migrating historical data in Microsoft SQL Server

This section describes how to migrate historical data from the Microsoft SQL Server database used by your version 7.0.x system to the Microsoft SQL Server database used by your version 7.1.1 system.

Requirements

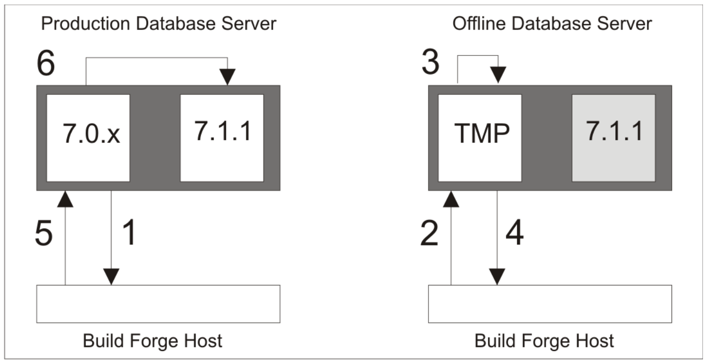
The following requirements must be met:

- Build Forge system: the system must be shut down.
- Offline system: A separate installation of Build Forge 7.1.1 and a Microsoft SQLServer database server must be set up on an offline system. .
- Database server: the Build Forge database should not have any transactions running. Database activity on other databases on the database server should be minimized.
- Disk space: you need free disk space that is at least five times the size of the Build Forge database.
- Privileges: the user performing the import must have full permission to manipulate databases and their contents (grant, create, drop). See “Microsoft SQL Server setup” on page 27
- Database size: perform a shrink on the data and delete the transaction logs. Do this before migration and after migration. Refer to Microsoft SQL Server documentation for information on performing a shrink.

Migrating historical data in Microsoft SQL Server using scripts

This section describes how use scripts to migrate Microsoft SQLServer historical data.

You can use these steps only if Build Forge is installed on the Microsoft SQLServer database server host.



1. **Export version 7.0.x data from Microsoft SQLServer**

   a. Stop Build Forge if it is running.

   b. Run the following command in the `Build Forge` directory on the host where you are running the production Build Forge version 7.1.1:

   ```
   bfdbdump -e filename -d dbname
   ```

   - *`filename`* is an absolute path name to the file, including the file extension. Example:

```
C:\temp\bf_alldata.bak
```

- *`dbname`* is the name of the Build Forge database, for example build.

c. Move the file to the host where you are running the offline Build Forge 7.1.1 system.

2. **Import exported Microsoft SQL Server data to the offline database**

   a. Stop Build Forge if it is running.

   b. Copy configuration files.

   Copy `migrate.conf` and `scratch.conf` file (if it was used) from the Build Forge version 7.1.1 production host to the Build Forge version 7.1.1 offline host. Edit the following settings to match the settings you used when you created the offline database:

   - `db_hostname`
   - `db_user`
   - `db_password`
   - `db_database`
   - `db_schema`

   c. Run the following command in the `Build Forge` directory on the host where you are running the offline Build Forge version 7.1.1:

   ```
   bfdbdump -i filename -d dbname
   ```

   - *`filename`* is an absolute path to the data file being imported. It must include the file extension. Example:

     ```
     C:\temp\bf_alldata.dump
     ```

   - *`dbname`* is the name of a new database to create, for example offline_build.

3. **Transform Microsoft SQL Server data**

   a. Stop Build Forge if it is running.

   b. Run the following command in the `Build Forge` directory on the host where you are running the offline Build Forge version 7.1.1:

   ```
   bfmigratehistorical -m
   ```

   This process may take a significant amount of time to run, depending on the amount of historical data. A progress indicator displays the number of records processed as the script runs.

4. **Export transformed Microsoft SQL Server historical data from the offline database**

a. Stop Build Forge if it is running.

b. Run the following command in the `Build Forge` directory on the host where you are running the offline Build Forge version 7.1.1:

```
bfdbdump -b -e filename -d dbname
```

- `-b` specifies that *only* historical data is exported.
- *`filename`* is a full path name for the exported data file, including the file extension. Example:

  ```
  C:\temp\bf_transformed.bak
  ```

- *dbname* is the name of the database you are dumping. Example: offline_build.

c. Move the file to the host where you are running the production Build Forge 7.1.1 system.

5. **Import transformed Microsoft SQL Server historical data to the version 7.0.x database**

a. Stop Build Forge if it is running.

b. Run the following command in the `Build Forge` directory on the host where you are running the production Build Forge version 7.1.1:

```
bfdbdump -b -i filename -d dbname
```

- `-b` specifies that *only* historical data is imported.
- *`filename`* is the absolute path to the backup file containing the transformed historical data. It should include the file extension. Example:

  ```
  C:\temp\bf_transformed.bak
  ```

- *`dbname`* is the name of the database where the imported data is placed. It should match the name of the database from which you exported data in Step 1.

When the script executes, it does the following:

- Restores the transformed data from the file to a temporary database
- Selects the needed tables from the temporary database to the database you specify.

Example:

```
bfdbdump -b -i bf_transformed.bak -d build
```

6. **Copy transformed Microsoft SQL Server historical data to the version 7.1.1 databas**

a. Stop Build Forge if it is running.

b. Run the following command in the `Build Forge` directory on the host where you are running the production Build Forge version 7.1.1:

```
bfmigratehistorical -s
```

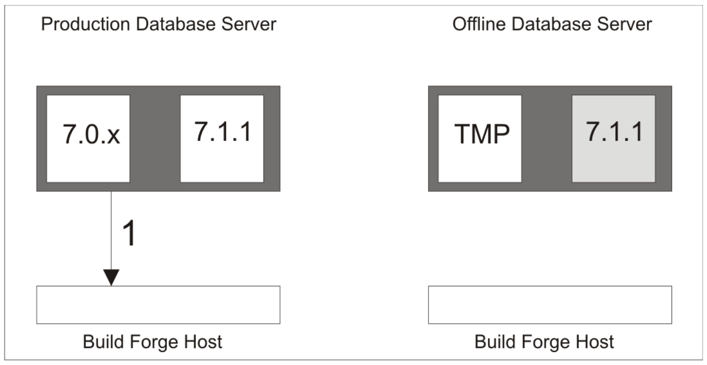
If the script fails, contact support.

Migrating historical data in Microsoft SQL Server using database commands

This section describes how to migrate historical data from the Microsoft SQL Server database used by your version 7.0.x system to the Microsoft SQL Server database used by your version 7.1.1 system.

Step 1: Export version 7.0.x data from Microsoft SQL Server

Use these instructions to use commands to export Build Forge data from the version 7.0.x database.



You need the following information about the Build Forge version 7.0.x database to perform this step:

- Name of the database
- Name of the schema (dbo by default)
- User name and password for the database

Run the commands on the production database server host.

1. Stop Build Forge if it is running.
2. Log into SQL Server Management Studio as a user with full permissions (grant, create, drop).
3. Click **New Query**.
4. In the query window that opens, type the following:

   ```
   BACKUP DATABASE [database] TO DISK = 'path/to/backup_file'
   ```

   For example:

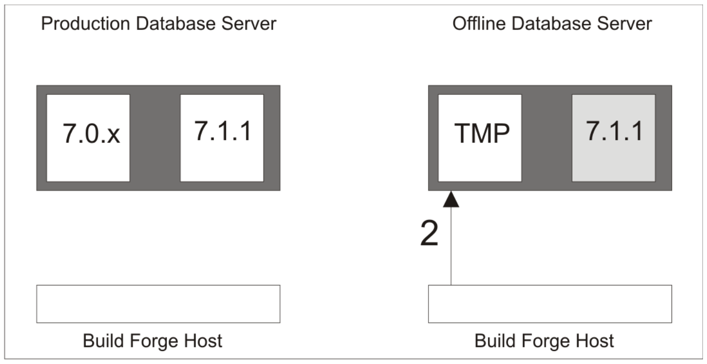
```
BACKUP DATABASE [build] TO DISK = 'C:\bf_alldata.bak'
```

5. Click **Execute**.

6. Move the file to the offline database server host.

Step 2: Import exported Microsoft SQL Server data to the offline database

This section describes how use commands to import the data you exported from Build Forge 7.0.x to the offline temporary database.



You need the following information about the offline Build Forge version 7.1.1 database to perform this step:

- Name of the database
- Name of the schema (dbo by default)
- User name and password for the database

Run the commands on the offline database server host.

1. Stop Build Forge if it is running.

2. Log into SQL Server Management Studio as a user with full permissions (grant, create, drop).

3. Click **New Query**.

4. Execute the following query:

   RESTORE FILELISTONLY FROM DISK='*path/to/database/backup_file*'

   This query only lists two files and the paths where these two files are located. One file is an `.mdf` file (the database itself). The other is an `.ldf` file (a log file for the database). Example:

   ```
   build.mdf
   build_log.ldf
   ```

5. Clear the query window.

6. Execute the following query:

```
RESTORE DATABASE [database] FROM DISK='absolute_path_for_backup_file' WITH MOVE
'old_mdf_name' TO 'new_mdf_absolute_path', MOVE 'old_ldf_name' TO
'new_ldf_absolute_path'
```

Note

You must specify a new database name and change the name of the `.mdf` and `.ldf` files (see previous step above). Leaving any of these fields with the original values will result in an error instead of a successful copy. If you have backed up database `build`, you can restore to a database named `offline_build`. The files `build.mdf` and `build_log.ldf` are associated with the database. Specify new names for them, such as the following:

```
offline_build.mdf
offline_build_log.ldf
```

Example:

```
RESTORE DATABASE [offlinebuild] FROM DISK='C:\temp\bf_alldata.bak' WITH REPLACE,
 MOVE 'build' TO 'C:\Program Files\Microsoft
SQLServer\MSSQL.1\MSSQL\Data\offline_build.mdf', MOVE 'build_log' TO 'C:\Program
 Files\Microsoft SQL Server\MSSQL.1\MSSQL\Data\offline_build_log.ldf'
```

7. Clear the query window.

8. Execute the following query:

```
ALTER DATABASE [database] SET READ_COMMITTED_SNAPSHOT ON
```

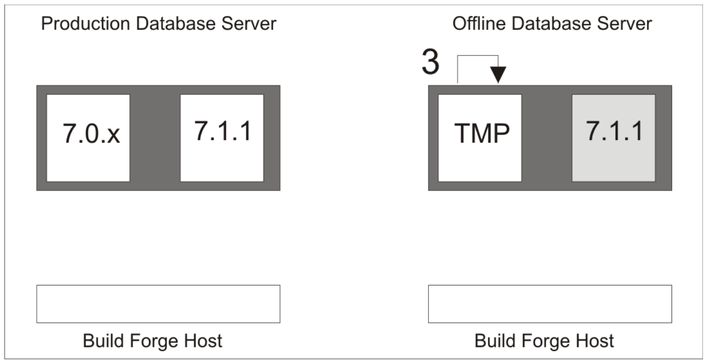
Replace `[database]` with your database name.

Note

This setting is required. The Build Forge services component cannot connect to the database without this setting.

Step 3: Transform Microsoft SQL Server data

This section describes how to transform the data that you imported into the offline temporary database.



1. Stop Build Forge if it is running.
2. Copy configuration files.

   Copy `migrate.conf` and `scratch.conf` file (if it was used) from the Build Forge version 7.1.1 production host to the Build Forge version 7.1.1 offline host. Edit the following settings to match the settings you used when you created the offline database:

   - `db_hostname`
   - `db_user`
   - `db_password`
   - `db_database`
   - `db_schema`
3. Run the following command in the `Build Forge` directory on the host where you are running the offline Build Forge version 7.1.1:

   ```
   bfmigratehistorical -m
   ```

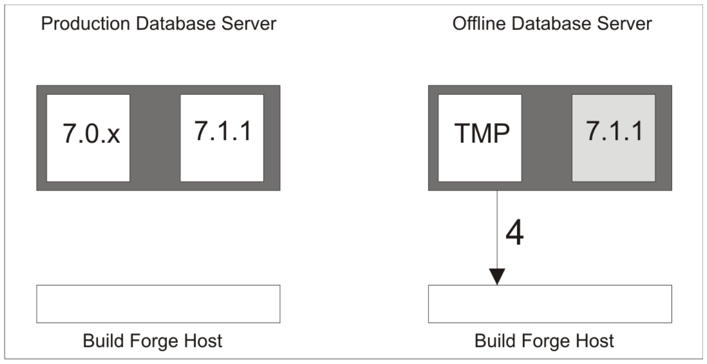
   This process may take a significant amount of time to run, depending on the amount of historical data. A progress indicator displays the number records processed as the script runs.

Note

If the script terminates because of an error, re-run the script. It continues from where it stopped.

Step 4: Export transformed Microsoft SQL Server historical data from the offline database

This section describes how to use commands extract and export transformed historical data from the offline temporary database. Only the historical data is exported.



You need the following information about the offline Build Forge version 7.1.1 database to perform this step:

- Name of the database
- Name of the schema
- User name and password for the database

Run the commands on the offline database server host. You use the commands to create a temporary database, select only historical data tables into it, and then dump the temporary database to a file. In the following commands, the following information is assumed:

- `bf711hmtemp` is the name of the temporary database to create. SELECT statements copy data into it.
- `build` is the name of the database that was exported from the 7.0.x database and imported into the temporary database. SELECT statements copy data from it.
- `bf_transformed.bak` is the name of the file where the dumped data is placed.

1. Stop Build Forge if it is running.
2. Log into SQL Server Management Studio as a user with full permissions (grant, create, drop).
3. Click **New Query**.
4. In the query window, enter the following:

```
CREATE DATABASE bf711hmtemp

SELECT * INTO bf711hmtemp.dbo.b7_builds FROM build.dbo.b7_builds
SELECT * INTO bf711hmtemp.dbo.b7_results FROM build.dbo.b7_results
SELECT * INTO bf711hmtemp.dbo.b7_filterevents FROM build.dbo.b7_filterevents
SELECT * INTO bf711hmtemp.dbo.b7_bom FROM build.dbo.b7_bom
SELECT * INTO bf711hmtemp.dbo.b7_interfacerelations FROM build.dbo.b7_interfacerelations
SELECT * INTO bf711hmtemp.dbo.b7_interfacecategories FROM
build.dbo.b7_interfacecategories
SELECT * INTO bf711hmtemp.dbo.b7_interfacenotices FROM build.dbo.b7_interfacenotices
SELECT * INTO bf711hmtemp.dbo.b7_interfacesections FROM build.dbo.b7_interfacesections
SELECT * INTO bf711hmtemp.dbo.b7_interfacedata FROM build.dbo.b7_interfacedata
SELECT * INTO bf711hmtemp.dbo.b7_interfacefield FROM build.dbo.b7_interfacefield
SELECT * INTO bf711hmtemp.dbo.b7_bom_manifests FROM build.dbo.b7_bom_manifests
SELECT * INTO bf711hmtemp.dbo.b7_logbuckets FROM build.dbo.b7_logbuckets
SELECT * INTO bf711hmtemp.dbo.b7_logs FROM build.dbo.b7_logs
SELECT * INTO bf711hmtemp.dbo.b7_store FROM build.dbo.b7_store
```

Note

The example assumes that you are using the default schema name, dbo. If you are using another schema name, run the following commands after you create the database and before you run the SELCECT statemends:

```
USE [711hmdatatemp]
CREATE SCHEMA [myschema] AUTHORIZATION [db_owner]
```

Substitute your schema name for *myschema* and the database owner's name for *db_owner*.

5. Click **Execute**.

6. Click **New Query**.

7. In the query window, enter the following:

   ```
   BACKUP DATABASE [database] TO DISK = 'path/to/backup_file'
   ```

   For example:

   ```
   BACKUP DATABASE [build] TO DISK = 'C:\bf_transformed.bak'
   ```

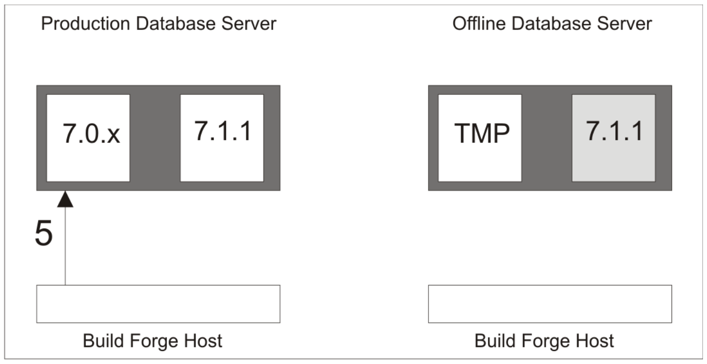
   Two files are included with the data. Using the database name of `bf711hmtemp`, they are `bf711hmtemp.mdf` and `bf711hmtemp_log.ldf`.

8. Click **Execute**.

9. Move the `bf_transformed.bak` file to the production database server host.

Step 5: Import transformed Microsoft SQL Server historical data to the version 7.0.x database

This section describes how use commands to import transformed historical data into the database for Build Forge version 7.0.x. Only historical data is imported.



You need the following information about the Build Forge version 7.0.x database to perform this step:

- Name of the database
- Name of the schema
- User name and password for the database

This section describes how to do the following:

- Restore the transformed data to a new database.

- Select the needed tables from the new database to the 7.0.x database.

1. Stop Build Forge version 7.0.x if it is running.
2. Log into SQL Server Management Studio as a user with full permissions (grant, create, drop).
3. Click **New Query**.
4. Execute the following query:

   ```
   RESTORE DATABASE [bf711hmtemp] FROM DISK='C:\temp\bf_transformed.bak' WITH REPLACE,
   MOVE 'bf711hmtemp' TO 'C:\Program Files\Microsoft SQL
   Server\MSSQL.1\MSSQL\Data\bf711hmtemp.mdf', MOVE 'bf711hmtemp_log' TO 'C:\Program
   Files\Microsoft SQL Server\MSSQL.1\MSSQL\Data\bf711hmtemp_log.ldf'
   ```

   - `bf711hmdata` is the database name to create. This name is used in the next step.
   - The Microsot SQLServer directory is `C:\Program Files\Microsoft SQL Server\`. During the restore, the `bf711hmdata.mdf` and `bf711hmdata_log.ldf` are placed there.

5. Clear the query window.
6. Execute the following commands:

   ### Important

   substitute names as follows, if required:

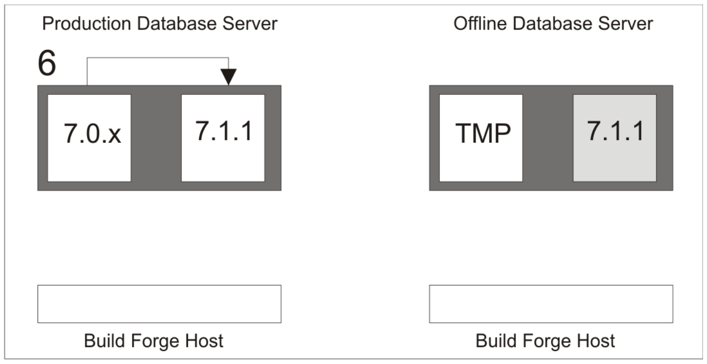
   - Substitute your version 7.0.x database name for `build`.
   - `bf711hmtemp` is used as the database name used to perform the restore. It was used to export data to a file in Step 4.

   ```
   SELECT * INTO build.dbo.b7_builds FROM bf711hmtemp.dbo.b7_builds
   SELECT * INTO build.dbo.b7_results FROM bf711hmtemp.dbo.b7_results
   SELECT * INTO build.dbo.b7_filterevents FROM bf711hmtemp.dbo.b7_filterevents
   SELECT * INTO build.dbo.b7_bom FROM bf711hmtemp.dbo.b7_bom
   SELECT * INTO build.dbo.b7_interfacerelations FROM bf711hmtemp.dbo.b7_interfacerelations
   SELECT * INTO build.dbo.b7_interfacecategories FROM
   bf711hmtemp.dbo.b7_interfacecategories
   SELECT * INTO build.dbo.b7_interfacenotices FROM bf711hmtemp.dbo.b7_interfacenotices
   SELECT * INTO build.dbo.b7_interfacesections FROM bf711hmtemp.dbo.b7_interfacesections
   SELECT * INTO build.dbo.b7_interfacedata FROM bf711hmtemp.dbo.b7_interfacedata
   SELECT * INTO build.dbo.b7_interfacefield FROM bf711hmtemp.dbo.b7_interfacefield
   SELECT * INTO build.dbo.b7_bom_manifests FROM bf711hmtemp.dbo.b7_bom_manifests
   SELECT * INTO build.dbo.b7_logbuckets FROM bf711hmtemp.dbo.b7_logbuckets
   SELECT * INTO build.dbo.b7_logs FROM bf711hmtemp.dbo.b7_logs
   SELECT * INTO build.dbo.b7_store FROM bf711hmtemp.dbo.b7_store
   ```

   Replace dbo with your schema name if you used a different schema name.

Step 6: Copy transformed Microsoft SQL Server historical data to the version 7.1.1 database

This section describes how to copy transformed historical data from the version 7.0.x database to the version 7.1.1 database. The database server must be running.



1. Stop Build Forge if it is running.
2. Run the following command in the `Build Forge` directory on the host where you are running the production Build Forge version 7.1.1:

   ```
   bfmigratehistorical -s
   ```

Note

If the script fails, contact support.

Migrating historical data in MySQL

This section describes how to migrate historical data from the MySQL database used by your version 7.0.x system to the MySQL database used by your version 7.1.1 system.

Requirements

The following requirements must be met:

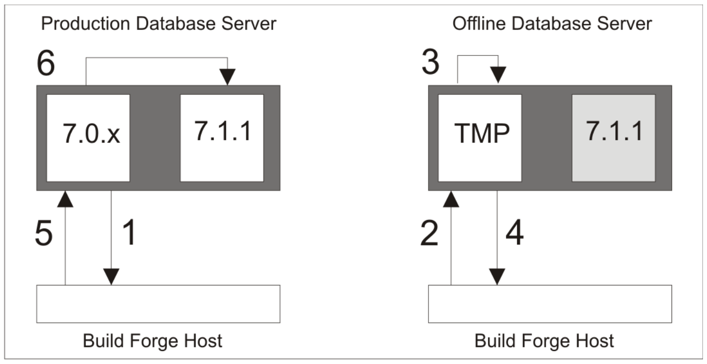
- Build Forge system: the system must be shut down.
- Offline system: A separate installation of Build Forge 7.1.1 and a MySQL database server must be set up on an offline system. .
- Database server: the Build Forge database should not have any transactions running. The import commands attempt to lock each table when doing an import. Database activity on other databases on the database server should be minimized.
- Disk space: you need free disk space that is at least five times the size of the Build Forge database.

- Privileges: The users for both the source and destination databases must have all privileges. See “MySQL setup” on page 32
- Database information: You need the database name and login credentials for the database containing the 7.0.x data.

Migrating historical data in MySQL using scripts

This section describes how use scripts to migrate MySQL historical data.

You can use these steps only if Build Forge is installed on the MySQL database server host.



1. **Export version 7.0.x data from MySQL**

   a. Run the following command in the `Build Forge` directory on the host where you are running the production Build Forge version 7.1.1:

   ```
   bfdbdump -e filename
   ```

   - *`filename`* is an absolute path for the file, including the file extension. Examples:

     ```
     C:\temp\bf_alldata.dump    # Windows
     /usr/tmp/bf_alldata.dump   # UNIX and Linux
     ```

   b. Move the file to the host where you are running Build Forge 7.1.1 offline.

2. **Import exported MySQL data to the offline database**

   a. Stop Build Forge if it is running.

   b. Copy configuration files.

   Copy `migrate.conf` and `scratch.conf` file (if it was used) from the Build Forge version 7.1.1 production host to the Build Forge version 7.1.1 offline host. Edit the following settings to match the settings you used when you created the offline database:

   - `db_hostname`
   - `db_user`

- `db_password`
- `db_database`

c. Run the following command in the `Build Forge` directory on the host where you are running the offline Build Forge version 7.1.1:

```
 bfdbdump -i filename
```

- *`filename`* is an absolute path to the data file being imported. It must include the file extension. Examples:

  ```
  C:\temp\bf_alldata.dump    # Windows
  
  /usr/tmp/bf_alldata.dump   # UNIX and Linux
  ```

3. **Transform MySQL data**

a. Stop Build Forge if it is running.

b. Run the following command in the `Build Forge` directory on the host where you are running the offline Build Forge version 7.1.1:

```
bfmigratehistorical -m
```

This process may take a significant amount of time to run, depending on the amount of historical data. A progress indicator displays the number of records processed as the script runs.

4. **Export transformed MySQL historical data from the offline database**

a. Stop Build Forge if it is running.

b. Run the following command in the `Build Forge` directory on the host where you are running the offline Build Forge version 7.1.1 system:

```
bfdbdump -b -e filename
```

- `-b` specifies that *only* historical data is exported.
- *`filename`* is a full path name for the exported data file, including the file extension. Examples:

  ```
  C:\temp\bf_transformed.dump   # Windows
  
  /usr/tmp/bf_transformed.dump  # UNIX and Linux
  ```

c. Move the file to the host where you are running the production Build Forge 7.1.1 system.

5. **Import transformed MySQL historical data to the version 7.0.x database**

a. Stop Build Forge if it is running.

b. Run the following command in the `Build Forge` directory on the host where you are running the production Build Forge version 7.1.1:

```
bfdbdump -b -i filename
```

- `-b` specifies that *only* historical data is imported.
- *`filename`* is the absolute path to the backup file containing the transformed historical data. It should include the file extension. Example:

```
C:\temp\bf_transformed.dump          # Windows

/usr/tmp/bf_transformed.dump         # UNIX and Linux
```

6. **Copy transformed MySQL historical data to the version 7.1.1 database**

a. Stop Build Forge if it is running.

b. Run the following command in the `Build Forge` directory on the host where you are running the production Build Forge version 7.1.1:

```
bfmigratehistorical -s
```

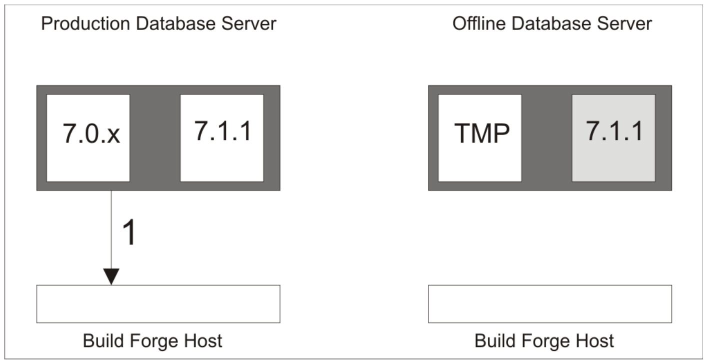
If the script fails, contact support.

Migrating historical data in MySQL using database commands

This section describes how to migrate historical data from the MySQL database used by your version 7.0.x system to the MySQL database used by your version 7.1.1 system.

Step 1: Export version 7.0.x data from MySQL

Use these instructions to use commands to export Build Forge data from the version 7.0.x database.



You need the following information about the Build Forge version 7.0.x database to perform this step:

- Name of the database

- User name and password for the database

1. On the production database server host, run the following command:

```
mysqldump --user=username --password=password --no-create-db database_name > filename
```

   - *filename* is an absolute path for the exported file, including the file extension. Examples:

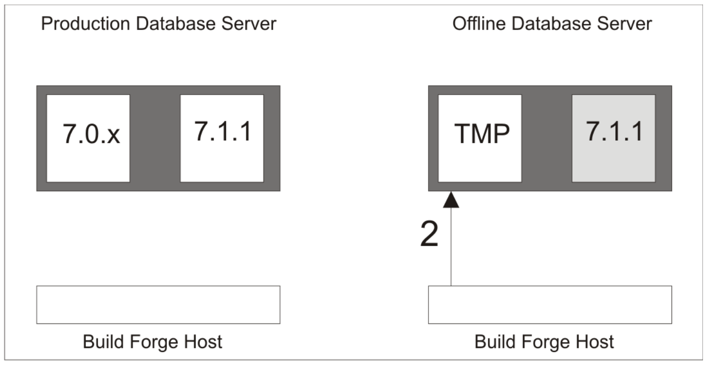
     ```
     C:\temp\bf_alldata.dump   # Windows
     /usr/tmp/bf_alldata.dump  # UNIX and Linux
     ```

   Use the database name and credentials that are used for Build Forge. Example: the `build` database.

2. Move the file to the offline database server host.

Step 2: Import exported MySQL data to the offline database

This section describes how to use commands to import the data you exported from Build Forge 7.0.x to the offline temporary database.



You need the following information about the offline Build Forge version 7.0.x database to perform this step:

- Name of the database
- User name and password for the database

1. Run the following commands on the offline database server host:

```
mysql -u root
mysql create database database_name;
mysql grant all on database_name.* to 'user'@'host' identified by "password" with
grant option;
mysql --user username --password password
                              database_name < filename
```

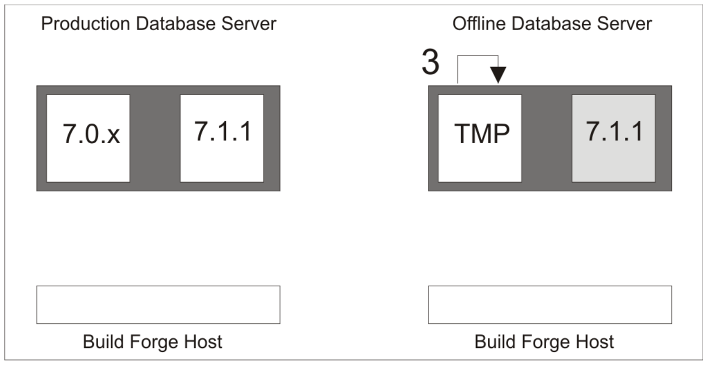
   Use the same file name, database name, and credentials that were used in Step 1.

Example, for a dump file named `bf_alldata.dump`database named `build`, user named `builder`, with password `buildpass`:

```
mysql -u root
mysql create database build;
mysql grant all on build.* to 'builder'@'localhost' identified by "buildpass";
mysql --user builder --password buildpass build < bf_alldata.dump;
```

Step 3: Transform MySQL data

This section describes how to transform the data that you imported into the offline temporary database.



1. Stop Build Forge if it is running.
2. Copy configuration files.

   Copy `migrate.conf` and `scratch.conf` file (if it was used) from the Build Forge version 7.1.1 production host to the Build Forge version 7.1.1 offline host. Edit the following settings to match the settings you used when you created the offline database:

   - `db_hostname`
   - `db_user`
   - `db_password`
   - `db_database`

3. Run the following command in the `Build Forge` directory on the host where you are running the offline Build Forge version 7.1.1:

   ```
   bfmigratehistorical -m
   ```

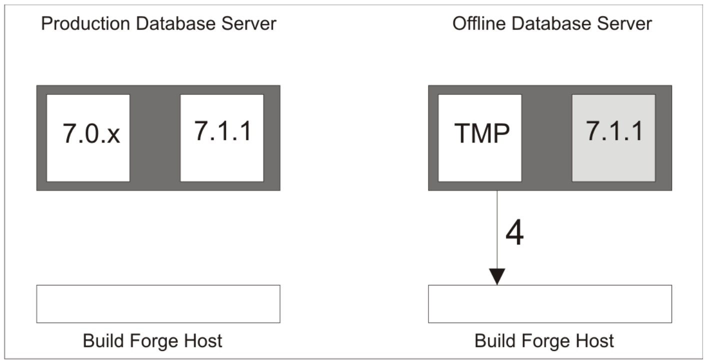
   This process may take a significant amount of time to run, depending on the amount of historical data. A progress indicator displays the number of records processed as the script runs.

Note

If the script terminates because of an error, re-run the script. It continues from where it stopped.

Step 4: Export transformed MySQL historical data from the offline database

This section describes how to extract and export transformed historical data from the offline temporary database. Only the historical data is exported.



You need the following information about the database to perform this step:

- Name of the database
- User name and password for the database

1. Stop Build Forge if it is running.
2. Run the following command on the offline database server host:

   ```
   mysqldump --user=username --password=password --no-create-db database_name > filename
   ```

   - *filename* is a full path name for the exported file, including the file extension. Examples:

     ```
     C:\temp\bf_transformed.dump   # Windows
     ```

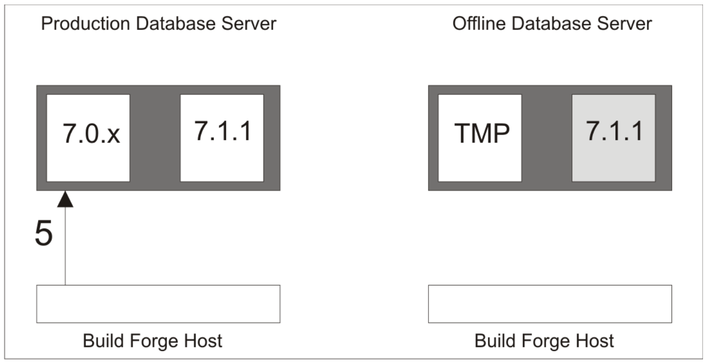
     ```
     \usr\tmp\bf_transformed.dump  # UNIX and Linux
     ```

   The database name and credentials should match those used in steps 1 and 2.

3. Move the exported file to the production database server host.

Step 5: Import transformed MySQL historical data to the version 7.0.x database

This section describes how to use commands to import transformed historical data into the database for Build Forge version 7.0.x. Only historical data is imported.



You need the following information about the 7.0.x database to perform this step:

- Name of the database
- User name and password for the database

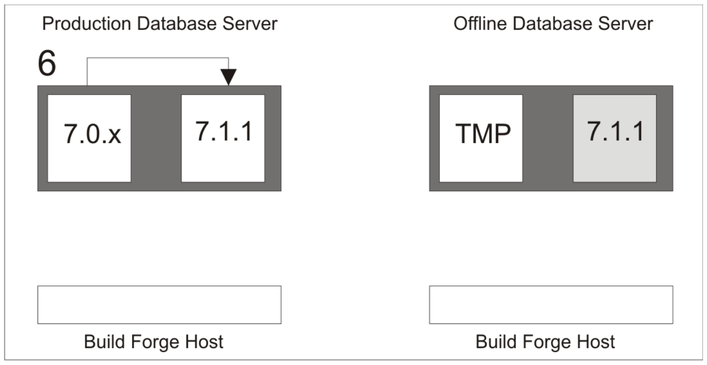
1. Run the following command on the production database server:

```
mysql --user username --password password
                              database_name  < filename
```

   The file name, database name, and credentials should match those used in step 4.

Step 6: Copy transformed MySQL historical data to the version 7.1.1 database

This section describes how to copy transformed historical data from the version 7.0.x database to the version 7.1.1 database. The database server must be running.



1. Stop Build Forge if it is running.
2. Run the following command in the `Build Forge` directory on the host where you are running the production Build Forge version 7.1.1 :

```
bfmigratehistorical -s
```

Note

If the script fails, contact support.

Migrating historical data in Oracle

This section describes how to migrate historical data from the Oracle database used by your version 7.0.x system to the Oracle database used by your version 7.1.1 system.

The data import and export steps must be performed on the database server host.

Requirements

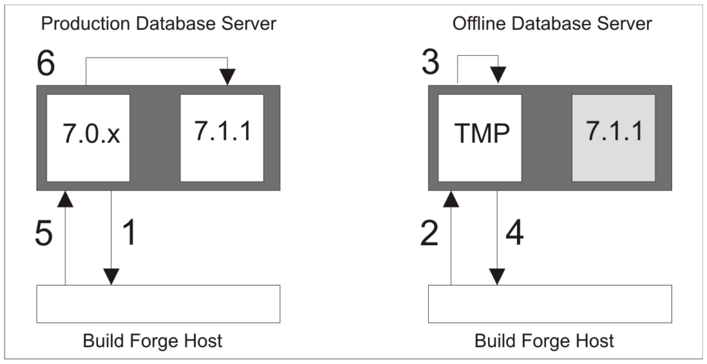
The following requirements must be met:

- Build Forge system: the system must be shut down.
- Offline system: A separate installation of Build Forge 7.1.1 and Oracle database server must be set up on an offline system. .
- Database server: the Build Forge database should not have any transactions running. The import commands attempt to lock each table when doing an import. Database activity on other databases on the database server should be minimized.
- Disk space: you need free disk space that is at least five times the size of the Build Forge database.
- Privileges: The users for both the source and destination databases must have sufficient privileges: CREATE TABLE, CREATE SESSION, and CREATE DIRECTORY. See "Oracle setup" on page 34. If it is not allowed for the user to have CREATE DIRECTORY, then your dba must create the directory, as described in Step 1: Export version 7.0.x data from Oracle. This requirement applies to both migrating historical data using scripts and migrating historical data using database commands.
- The CHARSET specified for both the source and destination databases must match.
- Database information: the database to be used. You need the database name, login credentials, and schema name for the database containing the 7.0.x data.

Migrating historical data in Oracle using scripts

This section describes how use scripts to migrate Oracle historical data.

You can use these steps only if Build Forge is installed on the Oracle database server host.



Before starting the process, create a user on the offline Oracle database server. Example: create user name **offline_build** and give it a password. Do not use the same user that you created for the offline Build Forge 7.1.1 installation.

- Add appropriate grants, including CREATE SESSION and CREATE TABLE.
- Add an appropriate QUOTA size in the DEFAULT TABLESPACE, to provide enough space for the system to store data.

```
create user offline_build
    identified by password
    default tablespace users
    quota unlimited on users;

grant create session, create table
    to offline_build;
```

1. **Export version 7.0.x data from Oracle**
   a. Stop Build Forge if it is running.
   b. Run the following command in the `Build Forge` directory on the host where you are running the production Build Forge version 7.1.1:

      ```
      bfdbdump -e filename
      ```

      - *`filename`* is an absolute path name prefix for the exported data files. token. Examples:

        ```
        C:\temp\bf_alldata    # Windows
        ```

        ```
        /usr/tmp/bf_alldata   # UNIX and Linux
        ```

        Dump files are numbered automatically. Example: `bf_alldata01`. The default FILESIZE= value of 2G is used.
   c. Move the files to the host where you are running the offline Build Forge version 7.1.1.

      Do not delete the directory. You re-use it in Step 5, when you import transformed historical data into the version 7.0.x database.
2. **Import exported Oracle data to the offline database**

a. Stop Build Forge if it is running.

b. Copy configuration files.

Copy `migrate.conf` and `scratch.conf` file (if it was used) from the Build Forge version 7.1.1 production host to the Build Forge version 7.1.1 offline host. Edit the following settings to match the settings you used when you created the offline database:

- `db_hostname`
- `db_user`
- `db_password`
- `db_database`
- `db_schema`

c. Run the following command in the `Build Forge` directory on the host where you are running the offline Build Forge version 7.1.1:

```
bfdbdump -i filename -s username
```

- *`filename`* is an absolute path name prefix for the data files to import. The prefix must match the prefix specified in the export step. Examples:

  ```
  C:\temp\bf_alldata    # Windows
  /usr/tmp/bf_alldata   # UNIX and Linux
  ```

  All files that match the prefix are imported.

- *username* is the user name for the Build Forge database.

3. **Transform Oracle data**

a. Stop Build Forge if it is running.

b. Run the following command in the `Build Forge` directory on the host where you are running the offline Build Forge version 7.1.1:

```
bfmigratehistorical -m
```

This process may take a significant amount of time to run, depending on the amount of historical data. A progress indicator displays the number of records processed as the script runs.

4. **Export transformed Oracle historical data from the offline database**

a. Stop Build Forge if it is running.

b. Run the following command in the `Build Forge` directory on the host where you are running the offline Build Forge version 7.1.1 system:

```
bfdbdump -b -e filename
```

- `-b` specifies that *only* historical data is exported.
- *`filename`* is an absolute path name prefix for the exported data files. Examples:

  ```
  C:\temp\bf_transformed    # Windows
  /usr/tmp/bf_transformed   # UNIX and Linux
  ```

  Dump files are numbered automatically. Example: `bf_trasnformed01`. The default FILESIZE= value of 2G is used.

c. Move the files to the host where you are running the production Build Forge 7.1.1 system. Place them in the directory you created in Step 1.

5. **Import transformed Oracle historical data to the version 7.0.x database**

a. Stop Build Forge if it is running.

b. Run the following command in the `Build Forge` directory on the host where you are running the production Build Forge version 7.1.1:

```
bfdbdump -b -i filename -s username
```

- `-b` specifies that *only* historical data is imported.
- *`filename`* is an absolute path name prefix for the data files to import. The prefix name part of the path must match the prefix used for export in Step 4. Examples:

  ```
  C:\temp\bf_transformed    # Windows
  /usr/tmp/bf_transformed   # UNIX and Linux
  ```

  Dump files are numbered.

- *username* is the user name for the offline Build Forge database.

All files that match the prefix are imported.

6. **Copy transformed Oracle historical data to the version 7.1.1 database**

a. Stop Build Forge if it is running.

b. Run the following command in the `Build Forge` directory on the host where you are running the production Build Forge version 7.1.1:

```
bfmigratehistorical -s
```

If the script fails, contact support.

Migrating historical data in Oracle using database commands

This section describes how to migrate historical data from the Oracle database used by your version 7.0.x system to the Oracle database used by your version 7.1.1 system.

The data import and export steps must be performed on the database server host.

Create an offline Oracle user

On the offline Oracle database server, create a user. Example user name **offline_build**. Give it a password. Do not use the same user that you created for the offline Build Forge 7.1.1 installation.

- Add appropriate grants, including CREATE SESSION, CREATE TABLE, and CREATE DIRECTORY.

  Note

  If you are not allowed to grant CREATE DIRECTORY, you need to have a dba create the directory when needed. See Step 1: Export version 7.0.x data from Oracle.

- Add an appropriate QUOTA size in the DEFAULT TABLESPACE, to provide enough space for the system to store data.

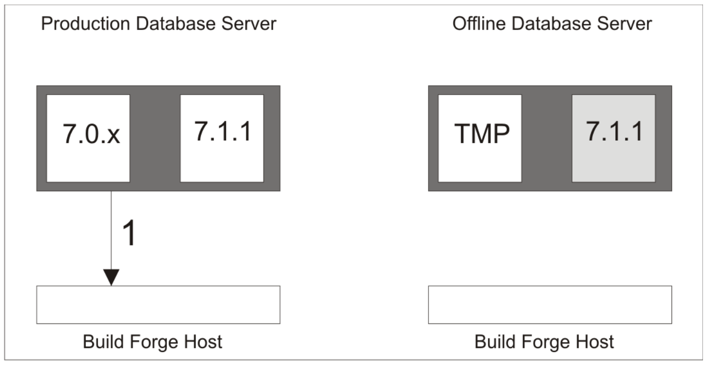
```
create user offline_build
    identified by password
    default tablespace users
    quota unlimited on users;

grant create session, create table, create directory
    to build;
```

Important

During installation, the same value is used for the database name and the Oracle SID.

Step 1: Export version 7.0.x data from Oracle

Use these instructions to use commands to export Build Forge data from the version 7.0.x database.



You need the following information about the Build Forge version 7.0.x database to perform this step:

- User name and password for the Build Forge schema

Run the commands on the production database server host.

1. Create a directory object for the files.

   The command must be on a single line.

   ```
   CREATE DIRECTORY directory_object_name as 'path'
   GRANT READ, WRITE ON DIRECTORY directory_object TO user
   ```

2. Create a directory on the file system that matches the path specified for the directory object.
3. To get an estimate on how much disk space is needed for the export, run the following commands:

   ```
   expdp user/password TABLES=BF% DIRECTORY=directory_object_name ESTIMATE_ONLY=y

   expdp user/password TABLES=B7% DIRECTORY=directory_object_name ESTIMATE_ONLY=y
   ```

4. Dump the data from the database to files in the directory. Run the following commands:

   ```
   expdp user/password DUMPFILE=bf_dump_%U.dat TABLES=BF% DIRECTORY=directory_object_name
   expdp user/password DUMPFILE=b7_dump_%U.dat TABLES=B7% DIRECTORY=directory_object_name
   ```

   The files are generated and placed into the directory you created. The `DUMPFILE` parameter specifies only the file name. Paths are not allowed.

   - You can also set FILESIZE= if you do not want to use the default of `"2G"`. The file size used must be smaller than the maximum size for a file in the file system. The dump creates multiple files if a table dump is larger than FILESIZE=.
   - Using `%U` in the dump file name causes files to be numbered if multiple files are created.

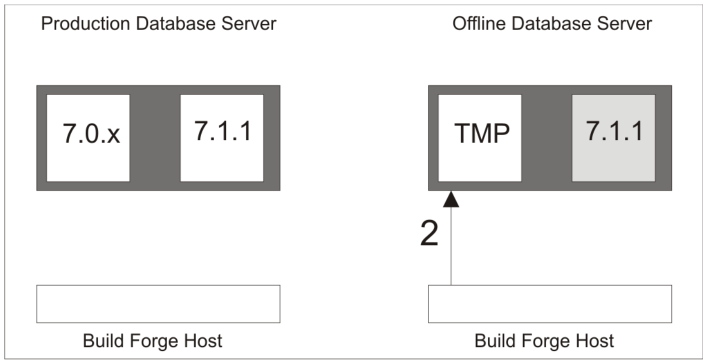
5. Move the files to the offline database server host.

Note

Do not delete the directory object and the file system directory. They are used again in Step 5: Import transformed Oracle historical data to the version 7.0.x database.

Step 2: Import exported Oracle data to the offline database

This section describes how to use commands to import the data you exported from Build Forge 7.0.x to the offline temporary database.



You need the following information about the Build Forge version 7.0.x database to perform this step:

- User name and password for the Build Forge schema

Run the commands on the production database server host.

1. Stop Build Forge if it is running.
2. Create a directory object for the files.

```
CREATE DIRECTORY directory_object_name as 'path'
GRANT READ, WRITE ON DIRECTORY directory_object TO user
```

3. Create the directory on the file system. Use the path you specified when you created the directory object.
4. Move all Build Forge dump files from the production database server host into the directory you created.
5. Run the following commands on the destination server:

```
impdp user/password DUMPFILE=bf_dump_file_list
DIRECTORY=directory_object REMAP_SCHEMA=source_user:destination_user

impdp user/password DUMPFILE=b7_dump_file_list
DIRECTORY=directory_object REMAP_SCHEMA=source_user:destination_user
```

Do not use spaces after the commas in the DUMPFILE= lists.

The *bf_dumpfile_list* is a comma-separated list of dump files produced by the expdb command in the previous step. If you used `bf_dump_%U.dat` as the pattern, then the first dump file is named `bf_dump_01.dat`.

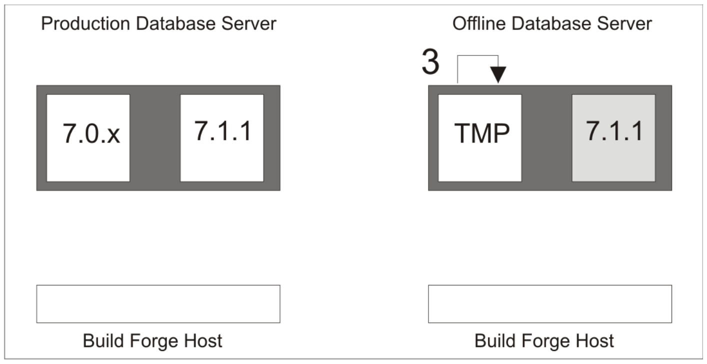
The *b7_dumpfile_list* is a comma-separated list of dump files produced by the expdb command in the previous step. If you used `b7_dump_%U.dat` as the pattern, then the first dump file is named `b7_dump_01.dat`.

Note

The `REMAP_SCHEMA` parameter is optional. It is only used if the database user for this step is different from the database user for the export from the version 7.0.x database.

Step 3: Transform Oracle data

This section describes how to transform the data that you imported into the offline temporary database.



1. Stop Build Forge if it is running.
2. Copy configuration files.

   Copy `migrate.conf` and `scratch.conf` file (if it was used) from the Build Forge version 7.1.1 production host to the Build Forge version 7.1.1 offline host. Edit the following settings to match the settings you used when you created the offline database:

   - `db_hostname`
   - `db_user`
   - `db_password`
   - `db_database`
   - `db_schema`
3. Run the following command in the `Build Forge` directory on the host where you are running the offline Build Forge version 7.1.1:

   ```
   bfmigratehistorical -m
   ```

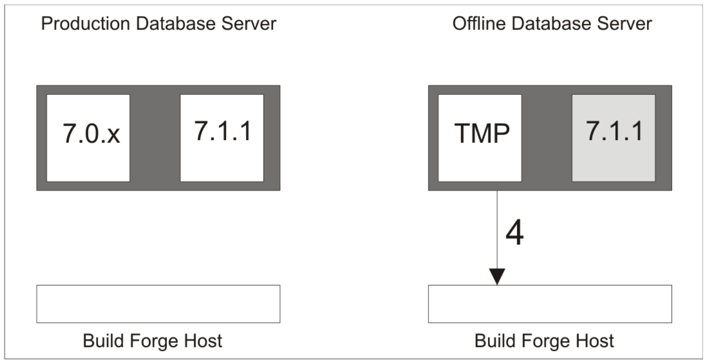
   This process may take a significant amount of time to run, depending on the amount of historical data. A progress indicator displays the number of records processed as the script runs.

Note

If the script terminates because of an error, re-run the script. It continues from where it stopped.

Step 4: Export transformed Oracle historical data from the offline database

This section describes how to use commands to extract and export transformed historical data from the offline temporary database. Only the historical data is exported.



You need the following information about the Build Forge version 7.0.x database to perform this step:

- User name and password for the Build Forge schema

Run the commands on the production database server host.

1. Stop Build Forge if it is running.
2. To get an estimate on how much disk space is needed for the export, run the following command on the offline database server host:

   ```
   expdp user/password TABLES=B7% DIRECTORY=directory_object ESTIMATE_ONLY=y
   ```

3. Dump the data from the source database to the file system. Run the following command on the offline database server host:

   Note

   Enter the command as a single line, without line breaks.

   ```
   expdp user/password
   DUMPFILE=b7_histdump_%U.dat TABLES=b7_builds,b7_results,
   b7_filterevents,b7_bom,b7_interfacerelations,b7_interfacecategories,
   b7_interfacenotices,b7_interfacesections,b7_interfacedata,b7_bom_manifests,
   b7_logbuckets,b7_logs,b7_store
   DIRECTORY=directory_object
   ```

   Do not use spaces after the commas in the TABLES= list. The files are generated and placed into the directory you created. The `DUMPFILE` parameter specifies only the file name. Paths are not allowed.

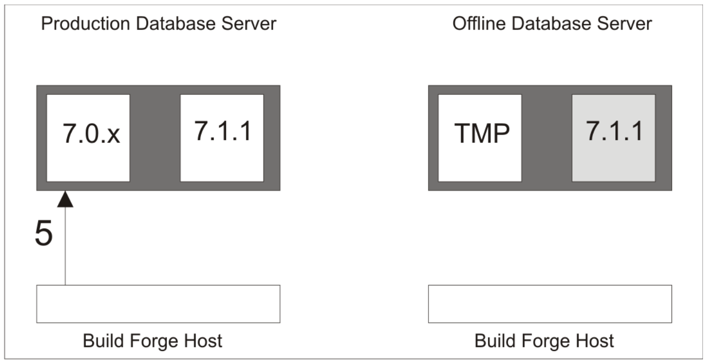
   - You can also set FILESIZE= if you do not want to use the default of `"2G"`. The file size used must be smaller than the maximum size for a file in the file system. The dump creates multiple files if a table dump is larger than FILESIZE=.

- Using `%U` in the dump file name causes files to be numbered if multiple files are created.

4. Move the files to production database server host. Place them in the directory you created in Step 1.

Step 5: Import transformed Oracle historical data to the version 7.0.x database

This section describes how to use commands to import transformed historical data into the database for Build Forge version 7.0.x. Only historical data is imported.



Run the commands on the production database server host. This step assumes that you placed the files containing transformed data into the directory you created in Step 1.

1. Stop Build Forge if it is running.
2. Make the directory containing the dumped files the current working directory.
3. Run the following command:

```
impdp user/password DUMPFILE=b7_dumpfile_list DIRECTORY=directory_object
REMAP_SCHEMA=source_user:destination_user TABLE_EXISTS_ACTION=REPLACE
```

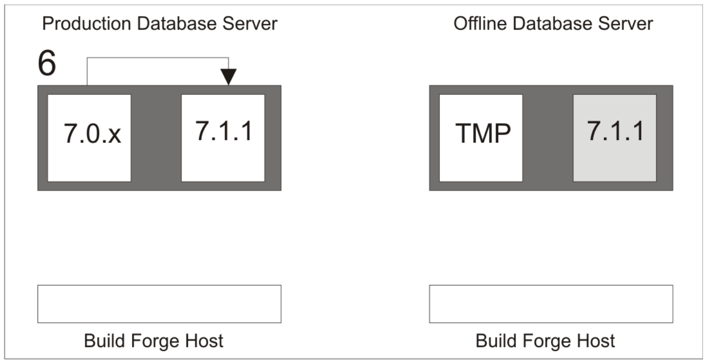
The *b7_dumpfile_list* is a comma-separated list of dump files produced by the expdb command in the previous step. Do not use spaces after the commas. If you used `b7_histdump_%U.dat` as the pattern, then the first dump file is named `b7_dump_01.dat`.

Note

Use the directory object you created in step 1. The `REMAP_SCHEMA` parameter is optional. It is only used if the database users for the source and destination are different.

Step 6: Copy transformed Oracle historical data to the version 7.1.1 database

This section describes how to copy transformed historical data from the version 7.0.x schema to the version 7.1.1 schema. The database server must be running.



1. Stop Build Forge if it is running.
2. Run the following command in the `Build Forge` directory on the host where you are running the production Build Forge version 7.1.1:

   ```
   bfmigratehistorical -s
   ```

Note

If the script fails, contact support.

Removing the staging tables

Staging tables are created in order to migrate configuration data. They can be removed. Do not remove them until you are confident that your migration has been successful and your production Build Forge system is stable. Back up the Build Forge database before removing the staging tables.

To remove the staging tables, execute `bfmigrateconfig` and `bfmigratehistorical`with the `-d` option. At the prompt "Do you still want to delete your migration staging tables?" enter y to confirm.

- Windows:

  ```
  > cd C:\Program Files\IBM\BuildForge
  > bfmigrateconfig.exe -d
  > bfmigratehistorical.exe -dp
  > bfmigratehistorical.exe -d
  ```

- UNIX or Linux:

  ```
  $ cd /opt/buildforge/Platform   # default directory
  $ ./bfmigrateconfig -d
  $ ./bfmigratehistorical -dp$
  $ ./bfmigratehistorical -d
  ```

bfdbdump reference

The bfdbdump program imports and exports data from databases.

It is used in migrating historical data from Build Forge version 7.0.x to Build Forge version 7.1.1.

Syntax

```
bfdbdump.pl [-h] [-i filename | e filename] [-d dbname] [-b] [-c configfile]
```

Options

-h	Displays help text.
-i *filename*	Imports *filename* into the database. For DB2, *filename* must be a directory.
-e *filename*	Exports the database to *filename*. For DB2, *filename* must be a directory.
-d *dbname*	Specifies a database (required for Microsoft SQL Server only).
-b	Restricts the list of tables to those containing historical data.
-c *configfile*	Specifies a different configuration file to use. The default is `migrate.conf`. The database name, schema name, user name, and user password are used to access the database.
-s *username*	Specifies a user name to use as the source user. It is used only when importing data into an Oracle schema. It is required when the user for the destination (imported) schema is different than the user for the source schema.

bfmigrateconfig reference

The migration program copies data from a 7.0.x database to a version 7.1.1 database and migrates it to the new version 7.1.1 schema.

Syntax

```
bfmigrateconfig -[d | h | m | s]
```

This command copies a 7.0.x database to a version 7.1.1 database.

Note

Your database must be running before you run the `bfmigrate` command.

Options

-d	Deletes staging tables (all tables with the prefix b7_). You normally delete the staging tables after you have tested a successful migration. It may also be necessary to delete the staging tables as part of the recovery from a failed migration. Do not combine -d with other options.
-h	Displays help text.

-m	Moves a version 7.0.x database to staging tables in a version 7.1.1 database. For example, bf_users becomes b7_users. This option can be combined with -s, as -ms.
-s	Saves data from the staging tables to the new database. At this stage the command does a transform of the data from the old schema to the version 7.1.1 schema. This option can be combined with -m, as -ms.

Prerequisites

The `bfmigrateconfig` program requires the following:

- A new schema that is created for version 7.1.1
- An installation of version 7.1.1. *Do not start Build Forge after installing it*.
- migrate.conf: - a copy of the buildforge.conf file for your current installation, placed in the installation directory of your version 7.1.1 installation
- If you are migrating from a version before 7.0.2 iFix2, the following is also required:
    - scratch.conf - a copy of migrate.conf with different users. It is placed in the installation directory of your version 7.1.1 installation. This file is required only if you are migrating from a version before 7.0.2 iFix2.

Description

The `bfmigrateconfig` program is shipped in the Build Forge installation directory:

- Windows default: `C:\Program Files\IBM\BuildForge`
- UNIX and Linux default: `/opt/buildforge/Platform`

Start the program from a command line.

The program migrates configuration data, which includes data for all of the following objects:

- Environments
- Projects
- Users
- Servers

The program does not migrate historical data from jobs:

- Job logs
- Job data (BOM, adaptor data returned with a job)

bfmigratehistorical reference

The bfmigrateconfig program processes data in databases.

It is used in the context of migrating historical data from Build Forge versions 7.0.x to Build Forge version 7.1.1.

Syntax

```
bfmigratehistorical [-f migrate.conf] [-d[x][p]] [-m] [-s]
```

Options

-f	Path to the migrate.conf file or scratch.conf file
-d	Deletes 7.0.x historical staging tables.
-p	Deletes 7.1.1 historical staging tables.
-x	Deletes data in the 7.1.1 tables corresponding to the ids in the 7.1.1 staging tables.
-m	Migrates the data in the staging tables.
-s	Sends data from the 7.0.x staging tables to the 7.1.1 staging tables, then inserts those tables into the final destination tables in an ordered fashion.
-h	Displays this help text.

Migrating pre-7.1 report designs for Quick Report to 7.1

If you want to use pre-7.1 Quick Report designs in the 7.1 Quick Report tool, you must migrate your report designs from their file system locations to the Build Forge database.

The report migration process is managed by the services layer component using Quick Report system settings. The services layer component attempts to migrate reports each time it starts up.

To successfully migrate your reports, you might need to modify your 7.1 system settings for Quick Report. This section describes how report migration works and what to do to enable migration of your pre-7.1 Quick Report report designs to 7.1.

Prerequisites for migrating report designs

Before you begin report migration, complete the database upgrade tasks described in "Upgrade overview" on page 189.

Complete the following report migration prerequisites before or after you start the 7.1 Management Console for the first time.

1. The services layer must be able to access the file system where the pre-7.1 report designs to be migrated are stored:

    - Report designs must be on the same host where the 7.1 services layer is installed. Use FTP or SSH to transfer the report designs (.rpt files) to the services layer host.

    - Report designs must be in the directory locations specified by the 7.1 Quick Report Pub dir and Quick Report Users system settings.

      See "Quick Report system settings required for migration" on page 249.

2. The working directory for migration, specified by 7.1 Quick Report Temp dir, must be on the same host where the 7.1 services layer is installed.

   See "Quick Report system settings required for migration" on page 249.

Quick Report system settings required for migration

Report migration is attempted each time that the services layer component starts. The service layer checks the Quick Report Pub dir and Quick Report Users dir file system locations to see if there are any reports to be migrated to the database. The services layer records each report design successfully migrated in the Quick Report Temp dir directory.

To migrate report designs, the services layer component uses the following system settings. These settings must have the correct values to successfully migrate report designs. Use the following information to check your Quick Report systems settings.

Report system setting	Description
Restart Report Migration	In 7.1, if you want to start migration without restarting the services layer component, set this value to Yes.
Quick Report Pub dir	In 7.1, use this system setting to specify the fully-qualified location to public reports. Your report designs must be in this directory to automatically migrate them. In earlier releases, the default file location (../../reports/public) for public reports is relative to the application server installation directory, for example: <*bf-install*>/Apache/tomcat/webapps/quickReport.
QuickReport Users dir	In 7.1, use this system setting to specify the fully-qualified location to private reports. Your report designs must be in this directory to automatically migrate them. In earlier releases, the default file location (../../reports/users) for private reports is relative to the application server installation directory, for example: <*bf-install*>/Apache/tomcat/webapps/quickReport.
QuickReport Temp dir	In 7.1, use this directory to specify a fully-qualified directory on the same host as the services layer component. The services layer uses this working directory to list the report designs that have been successfully migrated to the database. In earlier releases, this directory was used to temporarily store Quick Report report designs before they were saved to the public or private directory on the file system.

Verifying system settings for report migration

Report designs are automatically migrated if the system settings are correctly set when the services layer component starts.

In the 7.1 Management Console UI, use the following procedure to locate the system settings for Quick Report and verify that they have the correct values.

1. Select **Administration** > **System Settings** .
2. Locate the system setting for **Quick Report Pub dir** .

   Enter the full path to the public reports and click **Save** . The public reports must be located on the host where the 7.1 services layer component is running.
3. Locate the system setting for **Quick Report Users dir** and enter the correct file system location.

   Enter the full path to the private reports and click **Save** . The private reports must be located on the host where the 7.1 services layer component is running.

4. Locate the system setting for **Quick Report Temp dir** and enter a fully-qualified directory on the host where the 7.1 services layer component is running.

5. To restart report migration without restarting the services layer, locate the system setting **Restart Report Migration** . Set it to **Yes** and click **Save** .

Restarting report migration without restarting the services layer

Report designs for Quick Report are automatically migrated when the services layer component starts. The services layer starts when the Build Forge engine is started, but you can force migration to start without restarting the services layer, as follows:

1. Select **Administration** > **System Settings** .

2. Locate the system setting for **Restart Report Migration** .

   Set the value to **Yes** and click **Save** .

Upgrading Agents

Upgrade agents by installing new agents over the old ones.

You upgrade an agent by installing a new agent on top of the old one. Follow the installation instructions for installing an agent.

Uninstalling product components

Use IBM Installation Manager to uninstall product components that you installed using Installation Manager. To uninstall the Build Forge Agent software, use the operating system tools and commands described in this section.

Using Installation Manager to uninstall the product

Use this scenario to uninstall Build Forge product components that you installed with Installation Manger, except the provided DB2 Express database.

To uninstall product components, complete the following steps:

1. Log in to the operating system using the same user account that you used to install the product package.

   Close any running programs that you installed with Installation Manager.

2. If the Build Forge engine and service are running, stop them as follows:

Windows	• **If the engine is running in foreground**: Press **Ctrl + c** to stop the engine and Build Forge services. **Important** The most reliable method of stopping the engine and services is to use Ctrl + c. If the engine or any Build Forge services are still running when you uninstall, the uninstall will fail. • **If the engine is running in background**: Open a command window and enter the following command to stop the engine and services: `net stop bfengine71` These commands stops the Build Forge engine, Apache HTTP server, and Apache Tomcat application server.

UNIX/Linux	a. In a command window, go to the Build Forge rc directory.
	`cd /opt/buildforge/rc`
	b. Stop the engine.
	`./buildforge stop`
	This command stops the Build Forge engine, Apache HTTP server, and Apache Tomcat application server.

3. Start IBM Installation Manager.

4. On the Start page click **Uninstall** .
5. On the Uninstall Packages page, from the Installation Packages list, select the product package that you want to uninstall and click **Next** .
6. On the Summary page, review the list of packages that will be uninstalled and click **Uninstall** . The Complete page is displayed after the packages are removed.
7. Click **Finish** .
8. Close the Installation Manager window and exit Installation Manager.

   You must exit Installation Manager before cleaning up Build Forge files, optionally removing the DB2 Express database, or reinstalling the product.

Manually uninstalling the product if Installation Manager install fails

If you start IBM Installation Manager before the Build Forge engine and services have stopped, the uninstall fails.

To manually uninstall product components after Installation Manager fails:

1. In a command window, change to the Manager directory in the Build Forge installation directory, for example:

Windows	C:\Program Files\IBM\Build Forge\Manager
UNIX/Linux	/usr/local/Build Forge/Manager

2. At the command prompt, type `main.exe uninstall main.res`.
3. When the command finishes running, delete the Build Forge installation directory.
4. Reinstall the product using Installation Manager.

Post-uninstallation cleanup of Build Forge files

Some manual cleanup is needed after an uninstall.

After you uninstall product components through Installation Manager, manually delete the following directories. If these directories are present, you cannot reinstall Build Forge using Installation Manager.

Important

Deleting the Build Forge root directory also deletes the rafw product directory. Rational Automation Framework for WebSphere users should review the uninstall instructions in the *Rational Automation Framework for WebSphere Installation Guide* before deleting the Build Forge installation directory.

Directory	Default location
Build Forge installation directory	**Windows**: C:\Program Files\IBM\Build Forge **UNIX/Linux**: /usr/local/buildforge
Shared files directory	**Windows**: C:\Program Files\IBM\SDPShared **UNIX/Linux**: /opt/IBM/SDPShared

Post-installation removal of the provided DB2 Express database

Optionally remove the provided DB2 Express database. Removing the DB2 Express database deletes the Build Forge database schema and tables.

Remove the DB2 Express database to:

- Reinstall the product and use a database other than DB2 Express as your Build Forge database
- Reinstall the product and reinstall a new, empty DB2 Express database.

To remove DB2 Express, do the following:

1. Open the Windows Add/Remove Programs tool, locate IBM DB2 Express Edition in the list of programs and remove it.
2. Delete the DB2 installation directory. The default installation location is C:\DB2.
3. Delete the DB2 library directory. The default installation location is C:\Program Files\IBM\SQLLIB.

Using Installation Manager to reinstall the product and use an existing DB2 Express

You can reinstall the product and use the existing DB2 Express database, if you did not remove it when you uninstalled other product components. This scenario gives you access to your existing Build Forge projects and job logs.

Before you start Installation Manager, use the check list in the following steps to obtain the required DB2 Express information.

To reinstall and use an existing DB2 Express database:

1. Start Installation Manager.
2. On the Start page click **Install** .
3. Follow the instructions in the Installation Manager wizard to reinstall product components.
4. On the Database Configuration page, make the following selections:

a. For *Install DB2 Express*, select **No** .

b. For *Do you wish to populate this database at install time?*, select **No** .

c. To complete the remaining fields, use the information in the following table:

✓	**Field**	**Description**
	Database host	The provided DB2 Express database is installed on the local host name (127.0.0.1).
	Database name	The database name is BUILD, all uppercase.
	Database schema name	The name of the database schema is BUILD, all uppercase.
	Database user name	The database user name that you provided to create the DB2 Express database.
	Database password	The password that you provided for the database user name.
	Path to the DB2 client libraries	The path to the DB2 client libraries (db2cli.dll) for DB2 Express is C:\Program Files\IBM\SQLLIB\bin.
	JDBC driver location	The path to the JDBC driver (db2jcc.jar) for DB2 Express is C:\Program Files\IBM\SQLLIB\java.

5. Click **Test Connection** .

6. Click **Next** to continue and complete the installation.

Uninstalling the Windows Build Forge Agent

Use the Add or Remove Programs tool for Windows to remove the Build Forge agent software

1. In Windows, locate Add or Remove Programs. For example, select **Start** → **All Programs** → **Control Panel** → **Add or Remove Programs** .

2. Locate the IBM Rational Build Forge Agent in the list of currently installed programs.

3. Click **Change/Remove** .

4. Follow the instructions to uninstall the agent.

Uninstalling a UNIX/Linux Build Forge agent

Use the following instructions to uninstall the agent software from UNIX/Linux platforms.

Linux	To remove agent software that was installed using the rpm package:

1. Find the agent software and list package names and versions:

   ```
   rpm -qa | grep bfagent
   ```

2. Delete the agent software:

   ```
   rpm -e bfagent-<version_number>
   ```

Solaris

To remove agent software that was installed using the pkgadd program, run the following command:

```
pkgrm BFAgent
```

Other platforms: AIX, HP-UX, Mac OS

For other platforms, the uninstall process is manual and varies by platform. Follow the instructions that apply to your platform and super server implementation.

Note

To run most commands, you need root access and the /sbin and /usr/sbin directories must be set in your current PATH environment variable.

1. Remove the agent service daemon, bfagent. Use the instructions for the super server implementation (inetd, xinetd, launchd, or SMF) that applies to your platform.

Super server	**Procedure**

inetd, common on older UNIX systems	a. Edit the /etc/inetd.conf file and remove the line for the bfagent.
	
	b. Find the process ID for inetd.
	`ps -ef | grep [i]netd`
	For BSD-derived systems, such as FreeBSD and Mac OS/X 10.4 and earlier, substitute `ps auwwwx` for `ps -ef`
	
	c. Read the updated inetd.conf and start inetd.
	`kill -HUP <PID>`

xinetd, common on newer UNIX systems	a. To remove the agent service, run the following command: `rm /etc/xinetd.d/bfagent` b. Find the process ID for inetd.

Super server	Procedure

	`ps -ef | grep [i]netd`
	
	For BSD-derived systems, such as FreeBSD and Mac OS/X 10.4 and earlier, substitute `ps auwwwx` for `ps -ef`
	
	c. Read the updated inetd.conf and start inetd.
	
	`kill -HUP <PID>`

launchd for Mac OS/X and OpenBSD systems	a. Run `launchctl`. b. Enter `stop com.ibm.rational.bfagent`. c. Enter the following command: `unload /Library/LaunchDaemons/com.ibm.rational.bfagent.plist` d. Enter `quit`. e. Run the following command: `rm Library/LaunchDaemons/com.ibm.rational.bfagent.plist`
Solaris System Management Facility (SMF) for Solaris 10	a. Run `inetadm -d network /bfagent/tcp` b. Run `svccfg delete -f network/bfagent/tcp`

2. Remove the agent service from the PAM interface.

   a. Edit /etc/pam.conf and remove all lines that begin with bfaent.

   b. Run `rm /etc/pam.d/bfagent`

3. Remove the protocol entry from the etc/services file.

   Edit /etc/services and remove the line for bfagent.

4. Remove the following files installed by the agent:

```
/etc/bfagent.conf
/etc/bfagent.conf-example
/usr/local/bin/bfagent
/usr/local/bin/bfcrypt.dll
```

Support

This topic describes how to get support for the Build Forge® system.

The following topics are discussed:

- Contacting IBM support
- Executables list

Executable commands installed with the product

The following table lists the executable commands provided and used by Rational® Build Forge® and describes each command.

On Windows, the command files are in the Build Forge installation directory, for example C:\Program Files\IBM\Build Forge.

On UNIX/Linux, the command files are in the <*bf-install*>/Platform directory. The recommended installation directory on UNIX/Linux is /usr/local/buildforge.

Note

If you are running the Management Console on z/Linux, you must specify the .pl extension to run a command.

To display the version number for any executable command, use the -v option. You must run the command from the directory where the executable commands are installed.

```
bfproject -v
```

The - v option for any command displays the command name and its version number, as shown in the following example:

```
bfproject.exe 7.0.351
```

Executable	**Service?**	**Description**
bfdbmigrate	N	Use this migration script used to convert a default database from version 3.8 (MySQL) to DB2® after the schema has been converted to version 7.0. See your *Installation Guide* for instructions. This script is intended to be used *after* using `bfmigrate.exe/bfmigrate.pl`
bfproject	N	buildforge.exe issues this command to start a job.
bfengine	Y	This command starts buildforge.exe and the web server. Windows only.
bfexport	N	Use this utility to export data from the database.
bfbomexport	N	Use this utility to used to export the BOM from the database.
bfimport	N	Use this utility to import project data into the database.

Executable	Service?	Description
bfmigrate	N	Use this migration script to upgrade a version 3.8 database to version 7.0. See your *Installation Guide* for instructions.
bfpurge	N	buildforge.exe issues this command to purge builds.
bfrefresh	N	The build system issues this command to update the manifests for servers.
bfsched.exe or bfsched.pl	N	buildforge.exe issues this command to check the database for scheduled jobs and start them when appropriate.
bfstepcmd	N	bfproject issues this command for long-running steps, to create a separate process for them.
buildforge	N	This command manages build, purge, and schedule processes.
console_uninst	N	Use this command to uninstall the Management Console. Windows only.
bfdispatch	Y	This command starts the agent service. Windows only.
bfagent	N	Agent executable
bfpwencrypt	N	Utility for encrypting passwords

Contacting IBM Customer Support for Rational products

If you have questions about installing, using, or maintaining this product, contact IBM Customer Support as follows:

The IBM software support Internet site provides you with self-help resources and electronic problem submission. The IBM Software Support Home page for Rational products can be found at http://www.ibm.com/software/rational/support/.

Voice Support is available to all current contract holders by dialing a telephone number in your country (where available). For specific country phone numbers, go to http://www.ibm.com/planetwide/.

Note

When you contact IBM Customer Support, please be prepared to supply the following information:

- Your name, company name, ICN number, telephone number, and e-mail address
- Your operating system, version number, and any service packs or patches you have applied
- Product name and release number
- Your PMR number (if you are following up on a previously reported problem)

Downloading the IBM Support Assistant

The IBM Support Assistant (ISA) is a locally installed serviceability workbench that makes it both easier and simpler to resolve software product problems. ISA is a free, stand-alone application that you download from IBM and install on any number of computers. It runs on AIX, (RedHat Enterprise Linux® AS), HP-UX, Solaris, and Windows® platforms.

ISA includes these features:

- Federated search
- Data collection
- Problem submission
- Education roadmaps

For more information about ISA, including instructions for downloading and installing ISA and product plug-ins, go to the ISA Software Support page.

IBM Support Assistant: http://www.ibm.com/software/support/isa/.

Notices for IBM Rational Build Forge Installation Guide

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

```
IBM Corporation
Department BCF
20 Maguire Road
Lexington, MA 02421
U.S.A.
```

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

IBM, the IBM logo, Rational, ClearCase, ClearQuest, and DB2 are registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published.

Other company, product or service names may be trademarks or service marks of others.

For other IBM trademark attributions, see http://www.ibm.com/legal/copytrade.shtml