IBM Rational Build Forge Online Help Version 7.1.1.1

Document # SC23-7681-08

Table of Contents

Document # SC23-7681-08

Document # SC23-7681-08

Document # SC23-7681-08

List of Figures

Document # SC23-7681-08

Welcome to Build Forge help

This help system provides how-to and reference information for the Management Console.

To use the help:

- Select topics in the table of contents at the left of this window. Click on + signs to expand chapters.
- Click the or button in the upper right corner of this window to hide or show the table of contents.
- When you are using the application, click the **Help** button to display context-sensitive help.

Help (HTML) and documentation is available in the **Help** topic:

- **Help** : this help system (HTML)
- **Help** → **Installation** : *Build Forge Installation Guide* as a help system (HTML)
- **Help** → **Installation (PDF)** : *Build Forge Installation Guide* opened in a separate window as a PDF file.
- **Help** → **Help (PDF)** : this help system opened in a separate window as a PDF file.
- **Help** → **What's New (PDF)** : *What's New in the Build Forge System* opened in a separate window as a PDF file.

PDF files are useful for printing multiple topics or for storing for use outside of a running Build Forge system. Help files (HTML) offer more accessibility options. Use your web browser settings to control font size, color, and other accessibility features.

Note

Before using this information and the product it represents, read the information in “Notices for IBM Rational Build Forge Online Help” on page 298.

First Edition August 2009

This edition applies to version 7.1.1.1 of IBM® Rational® Build Forge® .

Document version and build: 7.1.1.1.0.0009.

Accessibility

The help files are designed to be accessible.

Document # SC23-7681-08

The help files are provided in PDF and HTML, each of which have different accessibility features.

1. You can magnify PDF files in Adobe Acrobat products. They are also useful for printing multiple topics or for storing for use outside of a running Build Forge system.

2. Help files (HTML) support several accessibility options. The specific options available vary by web browser. Use web browser settings to control font size, color, and other accessibility features. The diagrams and figures in the help HTML provide alternate text for screen readers.

Document # SC23-7681-08

Getting started with Build Forge®

These topics provide a quick introduction to the system.

To begin, install the system if you have not already. You can install the system (Management Console and an agent) on a single host.

Accessing and using the console

This topic provides basic information about using system menus and tabs.

Accessing the console

To access the console, do the following:

1. Open a browser window.
2. Enter the URL for the console.

   `http://`*`host`*`:`*`port`*`/`

   Host is the fully qualified network name or IP address of the host running the console.

   Port is required only if the HTTP default port is not used. The default port is 80 unless SSL is enabled. The default HTTPS port is 443. HTTPS is used only if the console is configured to use SSL.

   Examples:

   - `http://localhost/` or `http://127.0.0.1`: can be used if you are running a browser on the host where the console is installed.
   - `http://my.company.com/`
   - `http://my.company.com:81`: the port number must be specified because the console is installed to use port 81.
3. Log in. Provide the following information, then click **Login** :
   - User name
   - Password
   - Domain

   If you log in as root or another user that is defined only in Build Forge, then you are not prompted for Domain. The best practice is to configure Build Forge to use LDAP to authenticate users. When that is set up, the Domain field is presented. It refers to the LDAP domain that is used to authenticate users.

   See also “LDAP integration” on page 209.

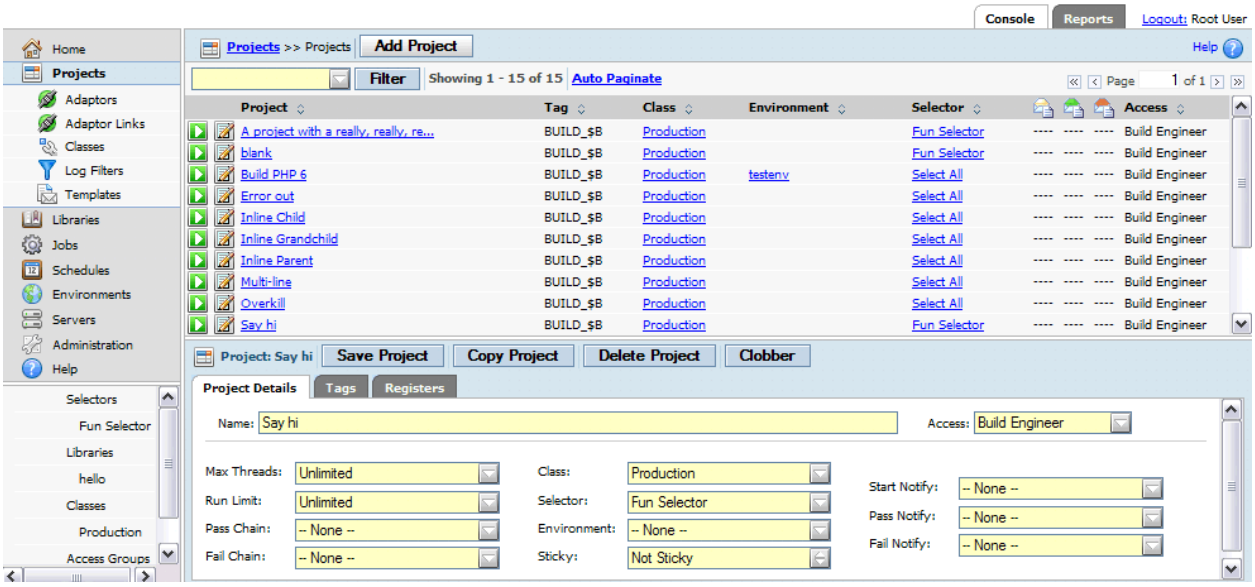
Constraints and tips

Be aware of the following constraints and tips for use:

- Set your display no lower than 1024 x 768. For best results, use 1280 x 1024 or higher.
- Do not shrink the browser window smaller than 1024 x 768.
- Use browser settings to control font size, color, and other accessibility features. You may need to refresh the page to properly display new browser settings.
- You can have only one active login session at the console under your user name. If you attempt to log in from a different host before you log out of the first login session, the first session is logged out automatically.
- When you are logged in, you can have more than one browser window or tab accessing the console. Session information is stored in cookies.
    - The browsers must be of the same type (for example, FireFox or Internet Explorer).
    - The browsers must be running on the same host.

Using the console window

Console window



Click a tab in the upper right to view an application. You can click **Console** (the default) or **Reports** .

Navigate in the **Console** application as follows:

- A menu is shown on the left and a main viewing panel is shown on the right.

- Click a menu item to see a panel or open a submenu of panels.
- For panels that are longer than the viewing area, use the paging controls at the upper left: « ‹ Page 1 of 4 › »
- Use the Filter field when viewing lists. When you enter a string and click **Filter** , the list is updated. It shows only items that have the string in their names.
- Drag the right edge of the menu to resize it.
- Panels that allow you to create or edit data typically have these controls:
  - The upper part of the panel lets you view and select items. Click an item name to see its contents. Click the edit icon  to edit the item.
  - The lower part of the panel shows the contents of a selected item.
- In some cases, when you select an item in a list, additional information is shown below the menu on the left. Example:

  1. Open **Administration** → **Users**
  2. Click on a user.

  The permissions for that user appear below the menu on the left.

Filtering lists

You can quickly filter lists to display just the items you want to work with, by entering a string in a **Filter** field.

Filter boxes are displayed with lists of objects, such as projects or servers. When a **Filter** box is available, you can use it to filter the contents of the list that is associated with it. Type text in the box, and click **Filter** and the system updates the list to display only items that match your entry.

- Type text in the Filter box and press Enter or click the **Filter** button to list items that match your entry. The system lists only the items that contain the text you entered (in any of their listed columns).

  ### Tip

  The filter is case-sensitive.

- You can filter a list based on the contents of a single column by placing the column name before the filtering string. For example, you can base a filter on the Selector column: **`Selector: Select All`**.
- The system retains the filter strings that you enter. Click the arrow beside the **Filter** box to display a list of filters you or others have entered. You can delete a filter string by highlighting a string, and clicking the **trash can** icon to the right of the string.

Document # SC23-7681-08

- The system always provides a **Display All** option in the list of filter strings.

Creating a Hello world project

This topic describes how to create and run a simple project to verify that the build system is set up properly.

Create a server authentication, selector, and a server object.

1. To create a project named HelloWorld, complete these steps:
   a. Select **Projects**.
   b. On the **Project Details** page at the bottom of the main content pane, type **`HelloWorld`** in the **Name** field, and choose a selector. Click **Save Project**.
2. 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** page.
   b. On the **Step Details** page, type **`EchoHelloWorld`** in the **Name** field.
   c. In the **Command** field, enter a command that writes `HelloWorld` to standard output on your chosen server.

      For example, this command works on Windows®, Solaris, Linux®, UNIX®, and Apple Macintosh OS X operating systems:

      ```
      echo hello world
      ```
3. Click **Save Step** .
4. Run the project:
   a. Select **Projects** to redisplay the project list.
   b. Click the **run** icon next to the HelloWorld project. The system displays the **Running** tab of the **Jobs** module, with the HelloWorld project listed as running.
   c. In the Step Details form, enter a **Name** of EchoHelloWorld.
   d. 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 the **Save Step** button.

Document # SC23-7681-08

5. 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.

6. Click the **Refresh** link at intervals until the Hello World project disappears from the Running list.

7. Click the **Completed** tab.

8. 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.

9. 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 target 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.

Project samples

Samples of projects are included to help you get familiar with the system.

Project samples are included in the following directory:

```
<bf-install>/samples/projects/
```

To use a sample project, do the following:

- Import it into the Management Console, using the bfimport comman.

Document # SC23-7681-08

- Execute the project.

Document # SC23-7681-08

About the Home panel

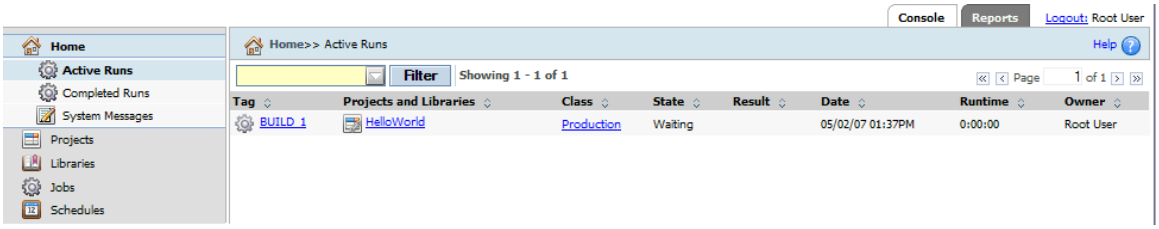
The Home panel provides information about recent jobs and system messages.

Use the Home module to view your recent or current jobs. You also use it to view notifications and system messages. Select a menu item to view the following:

- **Active Runs** - your currently executing jobs
- **Completed Runs** - your completed jobs
- **System Messages** - the system message log

Use the Jobs module to view all system jobs (if you have sufficient permissions).

The Home Module



Setting up server resources

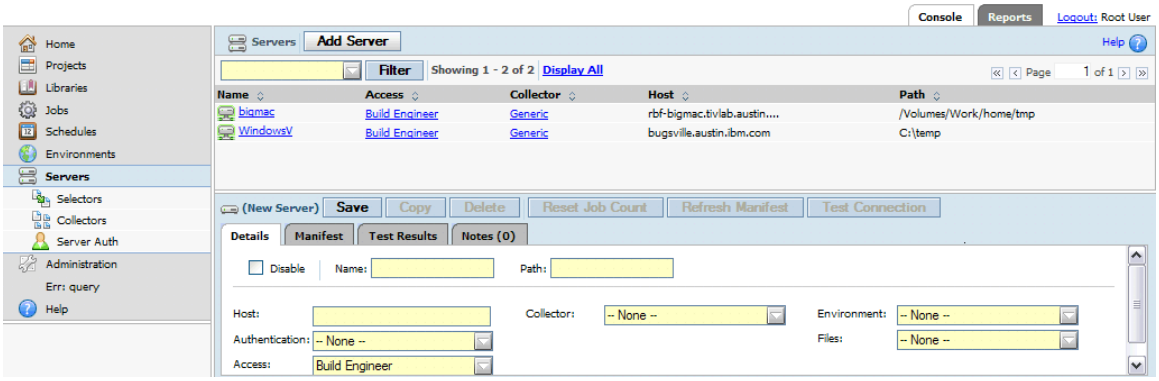
This topic describes how to set up and manage server resources in the Management Console.

About server resources

A server resource in the console represents a host where you can run projects or steps.

To set up a computer to be available as a server resource in the system:

- Install an agent on the server. See the *Build Forge Installation Guide* for more information.
- Create a server resource by using the Management Console.



- Servers have *manifests*. A manifest is a list of server properties. A manifest is populated when a collector runs. If a server does not have a collector assigned to it, some properties are automatically populated in the server's manifest.
- Manifests are populated by *collectors*. A collector is assigned to a server. A collector both sets property values and collects values for properties from the agent for a server.
- Selector reads server properties from manifests. Projects can use selectors to determine which server runs a step. A *selector* reads server properties from the manifest.

As an administrator setting up the system: first, you create servers. You then create collectors that you can assign to servers. You run the collectors to populate the server manifests. After you complete those tasks, build engineers can create projects that use selectors to determine where project steps run.

Creating server resources

A server resource represents a host on which you can run projects and steps.

Each server resource points to a host that has an agent installed on it. When you add a server resource, you are describing how the Management Console accesses and uses a specific host.

Document # SC23-7681-08

Before you create a server, make sure that the data objects that the server depends on already exist. Assign these items to a server:

Item	Required or option	Description
Server authentication	Required	Specifies the login name and password to use with the server.
Collector	Optional	Defines the properties the system collects from the server, in addition to default properties
Environment	Optional	Specifies environment variables to be assigned whenever a project is executed on the server.

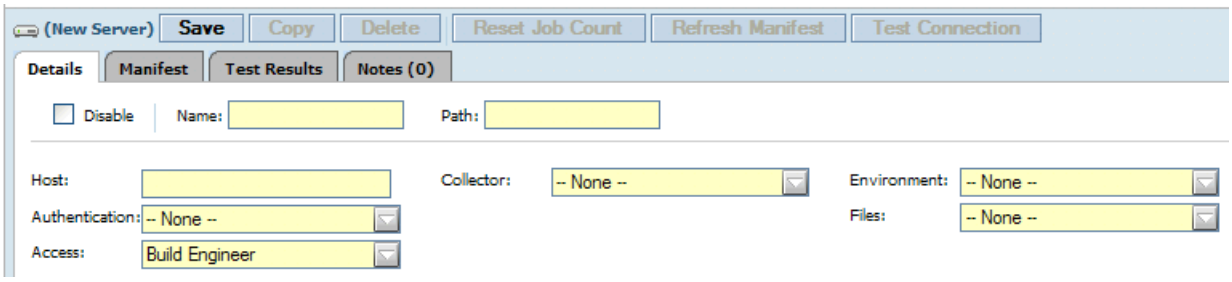
Note

If desired, you can create more than one server object within the Management Console for a single physical server. These server objects are called *logical servers*. Logical servers are typically used so that projects can access the same hardware with different properties. For example, two logical servers might use different paths or different environments:

- Two logical servers with different paths would create separate working directories on the same server. You can distinguish work that is performed using one server from work performed with another because all output is saved in different directories.
- Two logical servers with different environments run their steps with different starting environment values.

To create a server:

1. Select **Servers** in the left pane. The system displays the New Server form at the bottom of the main content pane. If you have selected an existing server, click **Add Server** to erase the form so that you can add a new server.
2. Provide the server details:



- **Name** : Give the server a name. This name is the BF_NAME property of the server. You refer to this name in selectors to specify a server by name.
- **Path** : Specify a directory that the server uses when it creates project and job directories, such as `c:/buildforgeprojects`. The system uses this path value as a starting point when it creates the build directory.

Document # SC23-7681-08

Tip

The system does not create the server path. The path must exist before a build attempts to access the server. If the path does not exist, the build fails.

- **Host** : Provide the host name for a physical computer that is running the agent. Use value localhost if you are defining the Management Console computer as a server. (The agent must also be installed on the Management Console.)

Note

You can include a port number with the host name; for example, *host_name: port_number*. If you specify a port number with the host name, it overrides the port number that the Default Agent Port system setting defines. (Click **Administration> System > Default Agent Port** .)

Note

Do not precede the host name with a protocol. For example, do not use http://.

- **Authentication** : Select the server authentication to use with this server.
- **Access** : Select an access group of users who can use this server.
- **Collector** : Select the collector to use with this server.
- **Environment** : Select a group of environment variables to be applied whenever this server is used to run a project. These variables are applied before all other variables. They set parameters specific to the server.
- **Files** : Define the types of file transfers for this server that can be used with the .get and .put commands. You can choose no transfers (None), file reads (.get), file writes (.put), or both (.get and .put).
- **Max Jobs** : enter the maximum number of jobs that should be run simultaneously. Default is 3.
- **Password Encryption Configuration** : select Enabled if you want the agent password to be encrypted. Default is Disabled.
- **SSL Enabled** : select Yes if you have configured the Build Forge system to use SSL and you want this server resource to communicate with the agent through SSL. Default is No.

Note

If you select Yes but Build Forge is not correctly configured for SSL, this server resource is not able to communicate with the agent at all.

3. Click **Save** . Your new server is displayed in the server list at the top of the content pane. To verify that you have correctly configured the server, select your server in the list and then click **Test Server** . The system reports errors if it cannot communicate with the server.

Document # SC23-7681-08

Server authentication

Use server authentications to associate login credentials to a server.

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.

You can use the same credentials for many servers, and update the credentials globally by managing a set of server authentications.

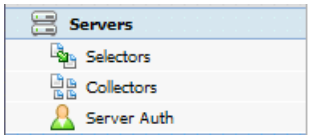
Creating server authentications

Use server authentications to store the login information for sets of servers.

Assign each server a server authentication so that the Management Console can log in to the server with appropriate privileges. Server authentications separate the login information from the server records so that you can apply the same login information to more than one server.

To create a server authentication:

1. In the left panel of Build Forge, click **Servers** → **Server Auth** .



   The build system displays the list of existing server authentications at the top of the main content pane, and a blank Server Auth Details form at the bottom.

Tip

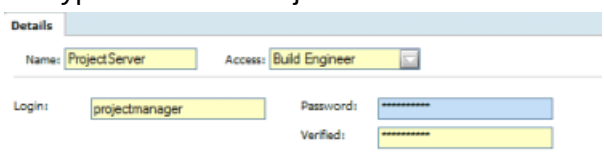
When you select a server authentication, the system populates the Server Auth Details form with the selected server's authentication information. To clear the form so that you can create a new authentication, click **Add Server Authentication** .

2. In **Name** , type an authentication name, a logical name to identify the server authentication in the system.

3. In **Login** , specify the server login name.

Note

If the login name is from a domain user, include the domain in this field. For example, type: MYDOMAIN/joeuser.



Document # SC23-7681-08

4. In **Password** , type the password.
5. In **Verified** , retype the password.
6. Click **Save Server Authentication** . The system stores a new server authentication with the name that you select.

The build system stores a new server authentication with the name that you select.

Overriding server authentication

With a special environment variable, you can force the server to use your Management Console login credentials instead of the server authentication assigned to the server, by using a special environment variable. To override the assigned authentication, add this variable named _USE_BFCREDS, with a value of 1, to an environment that your project or step uses. If you add the variable to the project environment, the build system uses the override on every step in the project.

When the build system attempts to run a step with an environment that contains _USE_BFCREDS=1, the system uses the console login credentials of the user who started the project to execute the step's command.

Note

If you are using LDAP/Active Directory Authentication, the **Store User Authentication Locally** system setting must be set to Yes (its default value) for the _USE_BFCREDS function to work. When the setting is Yes, the system caches user authentication information in encrypted form, and can then access the user authentication information for use with _USE_BFCREDS. Otherwise, the system does not store the LDAP information and cannot use it.

Allowing the use of restricted server authentication

Use the Execute Inaccessible Server Auths permission to allow a user to execute a step on a server with a server authentication to which a user does not have access.

As a prerequisite, the user must already have or be granted access to the server. (To grant access to servers, click **Servers > Access** .)

You use server authentications to log in and access servers. Server authentications are associated with an access group (**Servers > Server Auth > Access**).

You might create the following server authentications for a server:

- A dev/dev server authentication. Associate this server authentication with the developer access group for the server.
- A qa/qa server authentication. Associate this server authentication with the QA access group.
- A prod/prod server authentication. Associate this server authentication with the Build access group.

Document # SC23-7681-08

To permit a user who has access only to the qa/qa server authentication to run a step as the prod/prod server authentication, add the Execute Inaccessible Server Auths permission to the QA access group.

Note

If the user has access to the server but does not have access to server authentication through the Execute Inaccessible Server Auths permission, the step will still run but only if the environment variable _USE_BFCREDS is set to override server authentication.

Collectors

Collectors define what properties are collected and assigned to server resources.

The Collectors section of the Servers module lists the available collectors and allows you to create new collectors.

To open the Collectors section for the Servers module, click **Servers** → **Collectors** .

A collector consists of one or more properties. Each property specifies information to be included in the manifest.

Note

Collectors enable the use of dynamic selectors. Enterprise Edition is required in order to use collectors.

Collector property types

Collector properties specify how to collect information for the server manifest.

You can define the following types of properties in a collector:

Set value	These properties assign a named, static value to the server. You specify the property name and the value. Special values can be used in the value to get predefined responses. Special values begin with the underscore (_) character. See “Special set value properties in collectors” on page 33.
Built-in	These properties return information about the host assigned to the server resource. For a list of built-in properties, see “Built-in properties reference ” on page 25.
Run command	This type of property specifies a command for the system to run. The property value is set to the output from running the command. By default the first 255 characters of output is used. You can use a regular expression to extract specified parts of the output. • **Property** : name of the property • **Command** : the command to run on the host assigned to the server resource.

- **Regular Expression** : optional, a regular expression to use to filter the output. If specified, the build system attempts to match the regular expression with *each line* of output from the command. The first time a line matches, the system retrieves the value of $1, which is a Perl convention, and uses $1 as the value for that property. The regular expression must include at least one set of parentheses so that it returns a value. Consult Perl documentation for more information on constructing Perl regular expressions.

Include — This type of property specifies a list of collectors. You can nest collectors. When you create a collector of type Include, you specify the name of another collector as its value. When the build system creates or updates the manifest, the system inserts the properties from the referenced collector.

Tip

The system applies collector properties in the order that they are listed in the collector; later properties of the same name *override* earlier ones. Use this feature when you include one collector within another one. If you want to use some of the properties of a collector but not all, override the ones you do not want to use.

The system also applies a few properties automatically, such as the BF_NAME property that contains the logical name of the server. These are considered as part of the special manifest properties. See “Pre-set manifest properties reference” on page 31.

Selectors

Selectors choose a server resource to run a project or step.

A selector contains a list of variables. For each variable, you specify a value and a comparison. For example, you can specify a property CompilerVersion = 1.1 to select only server resources that have that property. You can also specify CompilerVersion >= 1.1 to select server resources that have versions 1.1, 1.3, 2, or 2.0.

When a project or step runs, the selector assigned to it is used to determine the server resource it runs on.

- Static selectors identify a server resource by name using the BFNAME variable.
- Dynamic selectors choose a server resource from all server resources in the system, using the criteria specified by variables in the selector.

Selector setup practices

A selector describes the kind of server that is appropriate for the project or step. It can specify a server by name or by a property that a collector collects and stores in the manifest. It can specify multiple properties, both required and optional.

There are multiple approaches possible for setting up selectors:

- Select server resources by name. Create selectors and name the selectors after your server resources. The selector specifies the server resource by its BF_NAME value, the unique name

Document # SC23-7681-08

used in the system. Use one of these selectors when you want to be specify the server resource to run a project or step.

- Select servers by server pool. You can organize servers into named pools and create a collector for each pool. Define a pool name as a property in the collector (a set-value property). Then create a selector for each pool name. The server resource for a project or step is selected based on its current load.
- Select servers by server attributes. You can choose servers based on functional properties, such as available hard-disk space, operating system, or number of CPUs. To implement dynamic selection, do the following:

  1. Create collectors that collect and assign appropriate properties.
  2. Assign the collectors to the appropriate servers.
  3. Create a selector for each property or set of properties that represent a server choice

  For example, you can create selectors to make selections according to these criteria:

  - Server resources with an operating system that includes "Windows®".
  - Server resources with more than one CPU.
  - Server resources running at less than a specified load.
- Select server resources by nested collectors. Use the Type property of Include to create a collector that points to another collector. A collector itself can be made up of a set of collector pointers. You may want to create individual collectors for each server, for example, so that each server can have some unique properties that you specify. You can use the Include type to point to utility collectors. For example, you can create a collector called Version that specifies the version numbers for key resources in your environment, such as Perl and Java.

Selector variable types

Selectors define how to choose a server resource for a project or step at run time.

The following types of variable are available to define in a selector:

- **Standard Property** : for this type of variable you specify the following:
  - **Name** : the name of a property to use. You choose from an automatically generated list that is made up of all built-in properties plus any set-value properties defined by collectors.

    ### Note

    To build selectors out of set-value properties, you must first define the set-value properties in one or more collectors.

  - **Operator** : one of the following comparison operators:
    - EQ - Equals. The value must match exactly. Value can be a number or a string.

Document # SC23-7681-08

    - NE - Not Equals. The value must not equal the value specified. Value can be a number or a string.
    - GT - Greater Than.
    - GE - Greater than or Equal.
    - LT - Less Than.
    - LE - Less than or Equal.
    - Contains.
  - **Value** : the value that the operator uses to compare.
  - **Required** : Yes if the selector is required to match this variable, No if it is optional.
- **Include** : The Include type allows you to define complex selectors built up out of simple selectors through nesting. You specify the following
  - **Selector** : choose a selector to include. All properties specified by the selector are included. You can build up complex selectors by including multiple simple selectors.

Selector variable comparison rules

When the system is choosing a server resource to use for a project or step, it compares the value of a selector variable and the value of the manifest property of the same name.

The system performs a string comparison unless *both* values match the following criteria for numbers:

- If the value starts with a digit and contains only digits and decimal points followed by at least one digit, the system performs a numeric comparison.
  - 5, 5.5, 0.5, 5.0, and 5.5.5 are considered numbers.
  - 5., .5, 5.., 5..5, 5.4.6_05, and 5.6i5 are all considered strings
- A numeric value that contains more than one decimal point causes a sub-version numeric comparison, where the system compares each decimal-separated field. Although 5.21 is less than 5.3 in an ordinary numeric comparison, 5.21.0 is greater than 5.3 in a sub-version numeric comparison.

Note

For the Contains operator, the system always performs a *case-insensitive* string comparison.

The following table shows examples of how the comparison rules are applied.

Document # SC23-7681-08

Property name	Manifest property value	Operator	Selector variable value	Comparison type	Match?
PerlVersion	v5.8.4	>=	5.2.1	String	Yes
PerlVersion	v5.8.4	>=	v.5.2.1	String	Yes
PerlVersion	v5.8.4	>=	v5.22.1	String	Yes
OS_VERSION	1.15	>=	1.1	Numeric	Yes
OS_VERSION	1.10	>=	1.1.0	Sub-version numeric	Yes
BF_NAME	WinServer1	contains	win	String	Yes
BF_NAME	Server123	=	123	String	No

Selector assessment of eligible server resources

To choose a server resource for a project or step, the system uses the selector to assess all eligible server resources:

1. The system compiles a list of the servers that contain all the *required* variables in the selector. If no server resource matches the required selector criteria, then the project or step fails and the system creates a note.

2. If more than one server resource meets the required criteria, the system rates each eligible server resource and assigns point as follows:

   - One point is given for each *optional* variable it matches. If the selector contains more than one copy of the same variable, the system assigns one point for each copy.

   - One point to the server resource that has the lowest BF_LOADRATIO value.

3. The system chooses the server that received the most points. If more than one server resource has the most points, they system chooses from among them.

You can repeat optional variables in a selector. Doing so increases the score of a server that matches them. For example, to require memory of 1 GB but strongly prefer memory of 2 GB or more, you can set selector variables as follows:

- Specify a required MEM_TOTAL GE 1GB.

- Specify an optional MEM_TOTAL GE 2 GB three times.

Managing server resources

This section contains information on managing server resources.

Testing server resources

You can perform a set of diagnostic tests on a server resource.

To perform a diagnostic test:

Document # SC23-7681-08

1. Click **Servers–>** *server_name*.
2. Click **Test Server** .

If the server fails the test, try one of these actions:

- Verify that the username and password in the server authentication that you are using for the server are correct.
- Verify that you are using the correct host name.
- Reinstall the agent on the server, or verify that it is installed.

Enabling and disabling server resources

You can temporarily disable a server resource. If it is disabled, jobs cannot run on it.

To disable or enable a server resource:

1. Click **Servers** to display the list of servers.
2. Click the name of server resource to modify. The system displays the details for that server.
3. Select **Disable** in the Details form at the bottom of the main pane.
4. Click **Save Server** .

Limiting concurrent jobs on a server

Use a server property to specify the maximum number of jobs that a particular server can run simultaneously. Set the default maximum with a system setting.

The system limits how many processes it attempts to run on any one server. If the server has a **Max Jobs** property, the build system limits the number of processes to its value. If the server does not have a **Max Jobs** property, the build system uses the default value, which comes from the **Default _MAXJOBS** system setting. To change the default value, click **Administration → System → Default _MAXJOBS** and set a different value.

Note

Other programs can run on the server. The system limit applies only to Build Forge jobs.

To give a server a non-default value, do the following:

1. Click **Servers** .
2. Select an existing server or create a new one.
3. Set the **Max Jobs** property to the desired value.
4. Click **Save** .

Document # SC23-7681-08

Resetting the job count to zero on a server

On the Servers page, use **Reset Job Count** to reset the job count (BF_JOBS) for the selected server to zero. BF_JOBS is the number of steps or jobs currently running on the server.

This selection allows you to reset BF_JOBS if it does not correctly reset when a job completes, fails, or is cancelled.

For example, cancelling multiple jobs occasionally fails to reset BF_JOBS. If BF_JOBS is not reset, it can reach the limit for the Max Jobs settings, causing steps or jobs to not run.

1. Select **Servers** to display the list of servers.
2. Select a server.
3. Click **Reset Job Count** .

Resetting the job count to zero on all servers

Use the **Reset Server Job-Count** page to simultaneously reset to zero the job count (BF_JOBS) for all servers. The BF_JOBS property is the number of steps or jobs that are running on the server.

The reset occurs when the manifest check interval runs. The default time is every 10 seconds.

After you reset the BF_JOBS value for all servers, the Reset Server Job-Count value reverts to No, the default setting.

For example, when you cancel several jobs, occasionally the build system fails to reset the BF_JOBS value. If the BF_JOBS value is not reset, the value can reach the limit for the default Max Jobs system settings, causing steps or jobs to not run.

1. Click **Administration** → **System** to display the list of system configuration settings.
2. Locate Reset Server Job-Count.
3. Click **Reset Server Job-Count** .
4. Click the Details tab, and click **Yes** as the value.
5. Click **Save**.

Manifests and dynamic server selection

You can use collectors, manifests, and selectors together to choose a server resource at run time for a project or step.

Three different data objects allow the system to dynamically choose servers:

- A *collector* is an object that defines the set of properties that the system collects from or assigns to a server resource. The system runs a collector when it checks a server resource's properties. The collected property values are stored in a manifest.

Document # SC23-7681-08

- A *manifest* is a list of the properties for a specific server. It contains the results from running a collector.
- A *selector* is a list of properties and comparisons such as MEM_TOTAL = 512. The system can compare the properties of a selector with a manifest to see if a server meets the requirements for a particular selector. Projects and steps specify a selector as one of their properties. When the project or step runs, the selector is compared to the manifest of all defined server resources in order to choose the server resource to run on.

The following example shows how to create and use a simple selector:

1. Create a server resource named Mercury and associate it with an agent.
2. Create a selector named Mercury. Set it to selects servers with BF_NAME = Mercury.
3. Create a project named Lincoln. Assign the Mercury selector to it.

When you run the Lincoln project, the system selects the server resource named Mercury. If that server resource is not available, the project fails.

The following example shows how to set up dynamic server selection in a set of servers:

1. Create a collector named RAMSIZE. Set it to collect the Built-in property MEM_TOTAL.
2. Create server resources to associate with hosts. Set each one to use collector RAMSIZE.
    - Mercury, a host with 512 MB RAM
    - Mars, a host with 1 GB RAM
    - Jupiter, a host with 3 GB RAM
3. Create a selector named BigRam. Set it to select a Standard Property, property=MEM_TOTAL, Operator=GE (Greater than or Equal), and Value=2048. MEM_TOTAL is expressed in MB. This selector selects only hosts that have 2 GB of RAM or more.
4. Create a selector named SmallRam. Set it to select a Standard Property, property MEM_TOTAL, Operator GE (Greater than or Equal), and Value 256. This selector selects only hosts that have 2 GB of RAM or more.
5. Create two projects:
    - HighMaint: set this project to use selector BigRam.
    - LowMaint: set this project to use selector SmallRam

When you run HighMaint, the system chooses the server Jupiter, because it is the only one that meets the selector requirement of having 2 GB of RAM.

When you run LowMaint, the system chooses any of the three server resources that is available.

Document # SC23-7681-08

If you later add a server resource named Neptune for a host that has 2 GB RAM, then the next project HighMaint runs, one of either Neptune or Jupiter is selected for the project. If Jupiter is down for some reason, then Neptune is selected. It is the only one left that fits the selector.

Viewing manifests

To view the manifest for a server, do the following:

- Click **Servers** .
- Click the name of the server resource.
- Click the **Manifest** tab.

You cannot directly change the manifest for a server. The manifest content is defined by the collector.

The manifest is refreshed automatically at intervals determined by server properties or system settings. To refresh it immediately, click **Refresh Manifest** .

Refreshing the manifest manually

To refresh the manifest manually, do the following:

1. Click **Servers** to display the list of servers.
2. Select a server.
3. Click **Refresh Manifest** .

This control is provided in addition to system settings that control how frequently server manifest properties get updated.

Setting the update frequency of a server manifest

You can use system settings to control how often the system queries a server for its manifest properties.

By default, the system checks Built-in and Set Value properties more frequently than Run Command properties. It uses several system settings to determine when to perform updates:

- The settings **Active server refresh interval** and **Inactive server refresh interval** set the number of seconds between updates of Built-in and Set Value properties. The settings discriminate between active (currently running a project) and inactive (not currently running a project) servers, so that you can have properties refresh more frequently on servers when they are being used.
- The setting **Default _AGE** sets the number of seconds between updates of Run Command properties. These properties, which require that the system run a command on the server to get updated data, create more network traffic during their updates. The default value for this

Document # SC23-7681-08

setting is 86400, which provides for updates once per day. Unless the values that you retrieve through Run Command properties change frequently, do not set a value lower than this setting.

You can use a collector to set a value for the _AGE setting that applies only to the servers that use that collector. When you use the special name _AGE setting for a variable, the build system interprets its value as the desired number of seconds between updates. To set this up:

1. Click **Servers** → **Collectors** .
2. Select an existing collector, or create a new one.
3. Add a property to the collector.
    a. Select the type **Set Value.**
    b. Type the **Property** name `_AGE`.
    c. Specify a **Value** in seconds.
    d. Click the **Save Variable** button.
4. If necessary, change the collector properties for more than one server to match the collector you just edited.

Example setups for static and dynamic server selection

The following example shows how to create and use a simple static selector:

1. Create a server resource named Mercury and associate it with an agent.
2. Create a selector named Mercury. Set it to select a Standard Property, property=BFNAME, Operator=EQ (Equal), Value=Mercury, and Required.
3. Create a project named Lincoln. Assign the Mercury selector to it.

When you run the Lincoln project, the system selects the server resource named Mercury. If that server resource is not available, the project fails.

The following example shows how to set up dynamic server selection in a set of servers:

1. Create a collector named RAMSIZE. Set it to collect the Built-in property MEM_TOTAL.
2. Create server resources to associate with hosts. Set each one to use collector RAMSIZE.
    - Mercury, a host with 512 MB RAM
    - Mars, a host with 1 GB RAM
    - Jupiter, a host with 3 GB RAM

Document # SC23-7681-08

3. Create a selector named BigRam. Set it to select a Standard Property, property MEM_TOTAL, Operator GE (Greater than or Equal), Value 2048, and Required. MEM_TOTAL is expressed in MB. This selector selects only hosts that have 2 GB of RAM or more.

4. Create a selector named SmallRam. Set it to select a Standard Property, property MEM_TOTAL, Operator GE (Greater than or Equal), and Value 256. This selector selects only hosts that have 2 GB of RAM or more.

5. Create two projects:

    - HighMaint: set this project to use selector BigRam.

    - LowMaint: set this project to use selector SmallRam

When you run HighMaint, the system chooses the server Jupiter, because it is the only one that meets the selector requirement of having 2 GB of RAM.

When you run LowMaint, the system chooses any of the three server resources that is available.

If you later add a server resource named Neptune for a host that has 2 GB RAM, then the next project HighMaint runs, one of either Neptune or Jupiter is selected for the project. If Jupiter is down for some reason, then Neptune is selected. It is the only one left that fits the selector.

Built-in properties reference

The Management Console collects built-in properties from servers and then assigns the values to the server manifest.

List of built-in properties

Built-in properties are used by several different data objects in the build system:

- **Selectors** can use built-in properties as selector variables to match servers with certain values in those properties.

- **Collectors** use built-in properties to collect data from servers.

- **Manifests** store the values of built-in properties if they have been collected.

Built-in properties are not automatically added. You must add a built-in property to a collector for the property to be displayed in the manifest. This table lists and describes the built-in properties.

Document # SC23-7681-08

Built-in Properties for Collectors and Manifests

Property	Description
CPU_ARCH	The returned value is a `label` for an architecture name, as shown: • `HP-PA`: HP Precision Architecture • `IA-64`: Intel Itanium • `MVS`: IBM S/390 • `PPC`: PowerPC • `PPC-64`: PowerPC 64 • `SPARC`: Sun SPARC • `X86`: x86-compliant architecture used by Intel, AMD, Cyrix, and others.
CPU_LOAD (Windows only)	The CPU load, or CPU usage, is expressed as a percentage of capacity (between 0 and 100).
CPU_LOAD1	The returned value reports the average number of processes (load average) that were running or waiting to run over the last minute. CPU_LOAD1 is a measure of CPU activity. An idle computer has a load number of 0. Each process that is using the CPU or waiting for the CPU adds to the load number by 1. **Note** On Windows, every process adds to the load number, active or not. Also, this information is gathered by the bfdispatch process and published to the agent using a shared memory segment. If the user credentials that are used for connecting to the agent are unprivileged, these statistics will not be available.
CPU_LOAD5	The returned value reports the average number of processes (load average) that were waiting to run over the last 5 minutes. CPU_LOAD5 is a measure of CPU activity. An idle computer has a load number of 0. Each process that is using CPU or waiting for CPU adds to the load number by 1. **Note** On Windows, every process adds to the load number, active or not. Also, this information is gathered by the bfdispatch process and published to the agent using a shared memory segment. If the user credentials that are used for connecting to the agent are unprivileged, these statistics will not be available.

Document # SC23-7681-08

Property	Description
CPU_LOAD15	The average number of processes (load average) that were waiting to run over the last 15 minutes. CPU_LOAD15 is a measure of CPU activity. An idle computer has a load number of 0. Each process that is using CPU or waiting for CPU adds to the load number by 1. **Note** On Windows, every process adds to the load number, active or not. Also, this information is gathered by the bfdispatch process and published to the agent using a shared memory segment. If the user credentials that are used for connecting to the agent are unprivileged, these statistics will not be available.
CPU_MHZ	This property reports the processor speed in megahertz. Certain conditions are required for this property to be filled in successfully: • In Linux: frequency scaling must be enabled. • In Windows: the ~MHz registry entry must exist and be filled in. • For x86 and x86-64 processors inline assembly must work.

Document # SC23-7681-08

Property	Description
CPU_MANUFACTURER	This property returns the company name of the processor manufacturer. If the information is not directly available, the names are assumed based on architecture if the information is not directly available. No value is returned if there is insufficient processor information available. These values are supported: • `AMD`: for their x86 and AMD64 processors • `Cyrix`: for their x86-compliant processors • `DEC`: for Alpha and VAX • `HP`: Hewlett-Packard Precision Architecture • `IBM`: IBM S/390 and PowerPC G5 • `Intel`: Intel x86 (including Intel64), IA-64 Itanium • `Motorola`: PowerPC G4 • `NexGen`: x86-compliant processors • `National`: National Semiconductor x86-compliant processors • `Rise`: Rise x86-compliant processor • `Sis`: Sis x86-compliant processor • `Sun`: Sun Microsystems SPARC • `TransMeta`: TransMeta x86-compliant processor • `UMC`: UMC x86-compliant processor • `VIA`: VIA Technologies x86-compliant processor

Document # SC23-7681-08

Property	Description

CPU_MODEL	This property returns the manufacturer-specific CPU model numbers. These values are reported as follows:
	• x86 architecture
	  • 386
	  • 486
	  • 586
	  • 686
	  • X86_64
	• PowerPC architecture
	  • 6xx
	  • POWER
	  • RS64
	  • G3
	  • G4
	  • G5
	  • Cell

CPU_SERIAL	This property returns the serial number of the CPU or computer. Currently, this functionality is limited to these architectures: • x86: Intel or Transmeta serial numbers only. Note: Most x86 processors will not report a serial number. No value is returned in such cases. • MacOS/X: The serial number assigned is retrieved from an I/O registry. To report the property, it must able to find the CoreFoundation and IOKit frameworks.
DISK_FREE	For UNIX and Linux, this property returns the amount of free space (in MB) on the file system that the server Path specifies. For Windows, the free disk space (in MBs) on the drive that the server Path specifies. For example, a disk with 4 GB of free space is reported as 4096 MB.

Document # SC23-7681-08

Property	Description
DISK_TOTAL	This property returns the total free disk space available. This value is reported for the base path of the agent, which might have a separate allocation that is smaller than the entire remaining disk or partition. Disk space management varies significantly between operating systems.
MEM_LOAD (UNIX/Linux only)	For UNIX and Linux, the amount of RAM or system memory that is currently in use is expressed as a percentage of total real memory (between 0 and 100).
MEM_FREE	This property returns the amount of RAM or system memory (in MB). For example, 2 GB of free RAM is reported as 1183 MB.
MEM_PAGESIZE	This property returns the RAM or system memory page size (in MB). This figure represents the standard page size for the host system. For example, a host system with a page size of 4 KB is reported as 4096 MB.
MEM_TOTAL	This property reports total RAM, or system memory (in MB). For example, a computer with 2 GB of RAM would be reported as 2048 MB.
NET_FQDN	This property returns the fully qualified domain name (FQDN) of the computer that is running the agent. The FQDN is reported based on the address that the agent is using to communicate. The returned address can be based on IPv4 or IPv6, depending on the address actually being used. See also NET_IPV, NET_IPV4, and NET_IPV6.
NET_HWADDR	This property returns the hardware address for the interface reported in NET_IFACE.
NET_IFACE	This property returns the name of the interface that the agent uses to communicate. • In Windows, the reported name is the name that ipconfig command returns, for example, `Intel(R) PRO/100 VE Network Connection - Packet Scheduler Miniport` • In other operating systems, the reported name is the name that `ifconfig` returns, for example, `en0` or `eth0` or `OSA1`.
NET_IPV	This property returns the type of IP connection that is used to communicate with the agent, either 4 for IPv4 or 6 for IPv6.
NET_IPV4	This property returns the IPv4 address that the agent uses to communicate. On connections over IPv6, if the agent is able to identify an IPv4 address for the same interface, that address is reported.
NET_IPV6	This property returns the IPv6 address that the agent uses to communicate.
NET_SPEED (Windows only)	This property returns the speed of the interface in Mb/sec, for example,`1000` for Gigabit Ethernet.
NUM_CPU	This property returns the number of CPUs on the computer.

Document # SC23-7681-08

Property	Description
OS_HOSTID	This property returns the result of the `gethostid()` system call. Typically, this result is not informative unless a system administrator has set `/etc/hostid` to an informative value.
OS_SYSNAME	This property returns the operating system name of the server. Examples: Microsoft Windows XP, AIX, Macintosh OS.
OS_RELEASE	This property returns the operating system release level of the server. For example, if the server operating system is Microsoft XP Version 5.1.2600, this returned value is 5.
OS_VERSION	This property returns the operating system version of the server. For example, if the server operating system is Microsoft XP Version 5.1.2600, this returned value is 1.
WIN_SERVICEPACK (Windows only)	This property returns the version number of the Windows service pack installed on the server. For example, for Service Pack 2, this value is 2.

Pre-set manifest properties reference

Some manifest properties are set automatically.

The following manifest properties are set automatically. Unlike built-in properties, these properties do not have to be added to a collector in order to populate them.

The properties marked as **Selector** in the table description can be uses in a selector. The others serve only to provide information in the manifest.

Document # SC23-7681-08

Automatically set manifest properties

Property	Description
BF_AGENT_VERSION	The version number of the agent that is installed on the server.
BF_EXCLUSIVE	**Selector**. It is a flag that takes no operator or value. If a selector includes this property, all slots on the selected server are reserved for the duration of the job. If a step in the job specifies a different server to run on, all current slots on the current server continue to be reserved while the other server runs the step.
BF_JOBS	**Selector**. The number of jobs (steps) that the server resource is running at the same time. This value is updated every time the console assigns a step to the server, independent of other manifest property updates.
BF_LAST_REFRESH	The time of the last update of built-in properties in the manifest. The value is reported as a UNIX® timestamp: the number of seconds since January 1, 1970.
BF_LASTJOBS	The number of jobs that the server was running the last time the manifest was refreshed.
BF_LAST_UPDATE	The time of the last update of run-command properties to the manifest. The value is reported, such as a UNIX® timestamp, as the number of seconds since January 1, 1970.
BF_LOADFACTOR	**Selector**. A calculated value, reported as a ratio: the number of jobs (BF_JOBS) divided by the maximum number of jobs allowed for the server (Max Jobs setting). A server that has 1 job running and 4 Max Jobs has a load ratio of .25.
BF_NAME	**Selector**. Specifies the server resource to run on. The value is the name of the server resource. The BF_NAME property is not displayed in the manifest list.
BF_RESERVE	**Selector**. It is a flag that takes no operator or value. If a selector includes this property, a slot is reserved on the selected server for the duration of the job. • If a step in a job specifies a different server to run on, then the slot on the selected server is reserved while the other server runs the step. • If a step specifies the selected server explicitly, the server uses the reserved slot for that step. This flag prevents project delays that occur when projects lose their slot on a server when one or more of their steps are run on other servers.

Document # SC23-7681-08

Special set value properties in collectors

When some set value properties are named in a collector, the properties cause certain behaviors in the system. These properties begin with the underscore character. The build system uses the values of these properties to apply behavior to servers that acquire these properties from a collector.

Note

You cannot create properties that begin with the string "BF_" because these names are reserved for use by the system.

Special set value properties for collectors and manifests

Property	Description
_AGE	This property defines, in seconds, how often a manifest should be refreshed, in seconds. The default value, 86400, provides a refresh once per day. A value of 3600 causes the system to update the manifest every hour. (You can modify the default setting by using the system setting **Default _AGE**).

Subscribing to RSS data feed for server status

The Management Console runs a server status check to verify that the server can pass a functional test and verify that the agent can log in. The Test Results tab displays the results of the status check. The Management Console automatically checks server status whenever a server is created or edited. Also, you can initiate a server status check at any time, for example, before running a project.

The Build Forge RSS data feed for server status displays the same information as the Test Results tab in the Build Forge Management Console.

To subscribe to the RSS data feed for server status, do the following:

1. In the Build Forge Management console, select **Servers** .

   The Web browser detects the RSS feed and displays an RSS icon in the browser address bar.

2. In the RSS aggregator tool, load the Build Forge RSS data feed.

   For example, copy the URL to add it to the list of RSS data feeds or drag-and-drop the RSS icon to add the URL to the list of RSS data feeds.

3. Subscribe to the RSS data feed to save the URL and be notified when updates occur.

Note

- For information about loading URLs and subscribing to RSS data feeds, see the documentation for your RSS aggregator tool.

Document # SC23-7681-08

- To view Build Forge system messages or server status through an RSS data feed in languages other than English, your RSS aggregator tool must support UTF-8 multibyte character encoding.

Subscribing to RSS data feed for jobs status

You can now track and filter the status of individual jobs using RSS data feeds. The Build Forge RSS data feed for jobs displays the same information as the server status in the Build Forge Management Console.

To subscribe to the RSS data feed for jobs status, do the following:

1. In the Build Forge Management console, select **Jobs** .

   The Web browser detects the RSS feed and displays an RSS icon in the browser address bar.

2. In the RSS aggregator tool, load the Build Forge RSS data feed.

   For example, copy the URL to add it to the list of RSS data feeds or drag-and-drop the RSS icon to add the URL to the list of RSS data feeds.

3. Subscribe to the RSS data feed to save the URL and be notified when updates occur.

Note

- For information about loading URLs and subscribing to RSS data feeds, see the documentation for your RSS aggregator tool.
- To view Build Forge jobs status, system messages, or server status through an RSS data feed in languages other than English, your RSS aggregator tool must support UTF-8 multibyte character encoding.

Using snapshots to create new instances of a selector

Snapshot a selector to quickly create a new instance of a selector that you want to change or modify.

Selector snapshots overview

Review these topics to learn about selector snapshots and understand how to use them.

Selector snapshot use cases

The following examples describe some common use cases for selector snapshots:

- Snapshot a selector to make changes to the selector configuration or perform testing of new tools or scripts while continuing to run jobs with the existing selector.
- Store a snapshot of a selector as an temporary backup or as part of an official archive.

Document # SC23-7681-08

- Snapshot a selector to capture a point-in-time selector configuration that corresponds with a milestone, such as an external or internal release.

Selector snapshot concepts and terms

In the UI, snapshots introduce some new concepts and terms for working with selectors.

Selector snapshot : A snapshot is a new instance of an existing selector. Some key points to remember about snapshots are as follows:

- A snapshot is a separate selector object. Making a change to one snapshot in a snapshot set does not affect the other snapshots in the set.
- A snapshot is not a copy. If you snapshot an object associated with a selector, snapshot creates a separate instance of the object. Copy maps relationships between objects, it does not create new objects.
- A snapshot is not a revision of a selector:
  - Snapshot does not support comparing changes between two selector snapshots.
  - Changes to selector snapshots are not tracked or identified with a version number as in a source control system. However, you can correlate selector snapshots to milestones by using a snapshot naming scheme that includes version numbers, for example, 7.5.0, 3.4.01.

Snapshot set : A snapshot set is the set of all the selector snapshots that are descendants of one base snapshot. At a minimum, the set includes the base or parent snapshot and a child snapshot. In the UI, the Snapshot icon beside the selector name indicates that a snapshot set for the selector exists.

Base snapshot : Initially, all selectors have a snapshot name of Base Snapshot. You can change Base Snapshot to another name. The base snapshot is the parent of the snapshot set.

Default selector snapshot : The default selector snapshot is the current, working selector. Only one snapshot in the set can be the default. If you do not specify a default snapshot, the base snapshot is the default.

- In the UI, the default snapshot is displayed at the top-level of the selectors list. Select **Servers > Selectors** to display the selectors list.
- When you select a selector with snapshots, the default selector snapshot is used unless you select a different selector snapshot in the list box.
- To access and work with other snapshots in the selector snapshot set, you must click the Snapshot icon.

Selector snapshot views

Select the Snapshot icon to display the Snapshot view. In the UI, the Snapshot view shows the hierarchy of the snapshots in a set:

Document # SC23-7681-08

- The base snapshot is at the top level and has the name Base Snapshot, if you do not assign it a unique name.
- All selector snapshots are children of a base snapshot. Children of the same base snapshot are indented at the same level in the Snapshot column.
- Selector snapshots that are created from a child snapshot become children of the child snapshot and are indented at the next level in the Snapshot column.

Selector snapshot planning

Review some best practices for selecting a default selector snapshot and naming selector snapshots.

- **Strategy for selecting the default snapshot in a set**

  The UI recognizes only one default or current selector snapshot for a snapshot set. Use a consistent strategy for selecting a default snapshot:

  - Use the base snapshot as the default snapshot

    Use snapshots as backups. Make changes only to the base snapshot. Do not make changes to the backed up selector snapshot.

  - Use the latest snapshot as the default snapshot

    Each time you create a new selector snapshot, make it the new default selector snapshot. Do not make changes to the base selector snapshot or earlier selector snapshots.

- **Identifying a snapshot naming scheme for the set**

  The selector snapshot name must be unique within a selector snapshot set.

  Use the following criteria to help you create selector snapshot names:

  - The name should be descriptive: it should indicate snapshot usage or purpose.
  - The naming scheme should follow a defined standard. You can use the Comment box on the Snapshot tab to describe the naming scheme.

- **Using a single selector name for the set**

  After you create a selector snapshot, you have the option to change the name of the selector. If you change the selector name, it is updated for every selector snapshot.

Creating a selector snapshot from an existing selector or selector snapshot

Creates a new instance of a selector. A snapshot is not a copy; it is a new selector.

Document # SC23-7681-08

Tip

To create snapshots and change the default snapshot, users must have the required permissions. See “Verifying and editing access groups for snapshot permissions” on page 91.

1. Click the **Edit** icon beside the selector or selector snapshot that you want to snapshot:
   - To snapshot the default selector snapshot, in the selectors list (**Servers > Selectors**), click the Edit icon beside the top-level snapshot.
   - To snapshot a nondefault selector snapshot, click the Snapshot icon. The Snapshot view displays the selector snapshots in the set. Click the Edit icon beside the nondefault selector snapshot.
2. Click **Create New Snapshot** .
3. At **Name** on the Snapshot tab, enter the snapshot name. The name is assigned to all the objects that you snapshot with the selector.

   The name must be unique within a selector snapshot set.
4. Select the Build Forge objects to snapshot when you create the selector snapshot. The objects that you can select are described in the following table.

Object	Description
Default	In the UI, the default snapshot is displayed at the top-level of the selectors list. Select **Servers > Selectors** to display the selectors list.
Change References To Old Default	Updates references from the old default selector snapshot to the new default if the Default option is selected.
Follow Selector Includes	If the selector uses the **Include** property type to include other selectors, a snapshot is created for those selectors. **Note** The **Include** variable type replaces the .include functionality provided in prior releases.

5. Click **Save** to save the selector snapshot.

Changing the default selector snapshot

The default selector snapshot is the top-level snapshot in a selector snapshot set and is displayed in the selectors list (**Servers > Selectors**).

Tip

To create snapshots and change the default snapshot, users must have the required permissions. See “Verifying and editing access groups for snapshot permissions” on page 91.

To change the default selector snapshot, edit the snapshot definition for the snapshot that you want to be the new default:

1. Select **Servers > Selectors** .
2. In the selectors list, click the **Snapshot** icon for the default selector snapshot.
3. In the snapshots list, click the **Edit** icon for the selector snapshot to be the new default.
4. Click **Make Default** .
5. **Important:** On the popup, choose OK or Cancel.

OK	**Update references**: If any objects reference the previous default selector, update the objects to use the new default selector.
Cancel	**Do not update references**: For any objects that reference the previous default, do not update references to the new default selector snapshot.

Changing the snapshot name for a selector snapshot

You can change the snapshot name for a selector snapshot and also for the objects that you selected to snapshot when you created the selector snapshot.

For the base snapshot, you can use this option to change its default name of Base Snapshot to another snapshot name for a single selector snapshot only or for all current and future selectors.

To change the snapshot name:

1. Select **Servers > Selectors** .
2. In the selectors list, click the **Snapshot** icon for the default selector snapshot.
3. In the snapshots list, click the **Edit** icon for the selector snapshot.
4. Select the **Snapshot** tab.
5. At **Name** , enter the new name.
6. **Optional:** At **Comment** , enter a comment.
7. **Important:** On the popup, choose OK or Cancel.

Document # SC23-7681-08

OK	**Change the selector snapshot name and other snapshot object names**: For objects that you selected to snapshot at the time that you created the selector snapshot, change the names of these objects and the selector snapshot. **For the Base Snapshot**: Changes the name of Base Snapshot for all current selector snapshots and all future selector snapshots.
Cancel	**Change the selector snapshot name but do not change other snapshot object names**: For objects that you selected to snapshot at the time that you created the selector snapshot, do not change the names of these objects. Change the selector snapshot name only. **For the Base Snapshot**: Retains the name Base Snapshot for all current selector snapshots and all future selector snapshots.

Accessing and viewing snapshots in a selector snapshot set

Snapshot a project to quickly create a new instance of a selector that you want to change or modify.

Creating a selector snapshot creates a snapshot set that contains a minimum of two environments: the base selector and the new selector snapshot.

To view all the selector snapshots in a snapshot set:

1. Select **Servers > Selectors** .

   The selectors list displays a list of selectors and selector snapshots. The top-level snapshot is the default selector snapshot.

2. Click the **Snapshot** icon to display the selector snapshots in the snapshot set.

   In the Snapshot view, you can:

   - Create a new selector snapshot. To begin, click the Edit icon.
   - Change the default snapshot for a selector. Click the Edit icon and click **Make Default** .
   - Edit the selector snapshot definition just like you would for a standard selector.

Deleting a selector snapshot

You can delete a selector snapshot by using the Delete option.

A selector cannot be deleted if it being used by another object. For example, it cannot be deleted if it is included by another selector or used by a project, a step, or a schedule.

To delete a selector snapshot:

1. Select **Servers > Selectors** .

Document # SC23-7681-08

2. In the selectors list, click the **Snapshot** icon for the base snapshot.

   The Snapshot view displays the selector snapshots in the set.

3. Click the **Edit** icon beside the selector snapshot to be deleted.

4. Click **Delete** .

Document # SC23-7681-08

Working with environments

This section describes how to set up and manage environments.

About environments

An environment is a named set of variables.

Once defined, environments are available to do the following:

- Set variables to be used by steps within in jobs. Environments can be assigned to servers, projects, and steps. During job execution, a running step inherits variable values from all three:
  - server environment associated with the server where the job is running
  - project environment associated with the project where the step is defined
  - step environment associated explicitly with the step
- Set variables to be used by scheduled jobs. An environment set for a scheduled job replaces the environment specified for the project.
- Set variables to be used by adaptors. An environment can be assigned to an adaptor link. It is used by the initial adaptor step of the project.

The Environments panel

To create a new environment:

1. Click **Add Environment** .
2. Specify a name for the environment.
3. Click **Save** . A variables panel is presented where you can add variables to the environment.

In the Environments panel, you can also click an environment to display and edit its variables.

How environments are applied to steps

This topic describes how environments are applied to steps in a running job.

Document # SC23-7681-08

Environment inheritance

Before the system executes a step, it creates the step environment. The step environment consists of all variables applicable to the step. The variables are inherited from the server environment, project environment, and step environment in order. The following is the basic case.

1. Server environment: server environment variables are copied to the step environment.
2. Project environment: project environment variables are applied to the step environment. If the project environment contains a variable of the same name as a variable in the server environment, then the value is updated according to the Variable Action in both variable definitions.
3. Step environment: step environment variables are applied to the step environment. If a variable in the step environment has the same name as a variable inherited from the server and project environments, then the value is updated according to the Variable Action.

The variable action for a variable directly affects how values are applied as they are inherited. For example:

- Case 1: values overwritten through inheritance when variable action is Set
  - Server environment: X = 1, action: Set
  - Project environment: X = 100, action: Set
  - Step environment: X = 3, action: Set
  - Final value during step execution: X = 3

  Variable X is set to 1, then 100, then 3. The variable action of Set replaces the variable value each time a new value is applied.
- Case 2: values inherited because of variable action Set if Not Set
  - Server environment: Y = 1, action: Set
  - Project environment: Y = 100, action: Set
  - Step environment: Y = 3, action: **Set if Not Set**
  - Final value during step execution: x = 100

  Variable X is set to 1, then 100 due to the Set action on X in the server and project environments. Because X uses the variable action **Set if Not Set** in the step environment, the value set in the project environment is inherited.

Special cases for inheritance

The following cases affect inheritance.

Document # SC23-7681-08

Inline projects	A step inlines a project by specifying a project in the **Inline** property for the step. When a step inlines a project, the called project's server environment and project environment are not used. Inheritance goes in this order: 1. The server environment for the calling step. 2. The project environment for the calling step. 3. The step environment for the calling step. 4. For each step in the called project, the step environment (if specified).
Chained projects	A project or step can specify project as a Pass Chain or Fail Chain. When a project is called in that way, it runs in its own environment. In addition, it has access to all of the variables from the calling project or calling step. Those variables are copied to new names using the prefix BF_CALLER_. Example: the variable BF_NAME in the calling project or step is available as BF_CALLER_BF_NAME in the called project and steps.
Scheduled jobs	When a project is on a schedule in **Schedules** , you can choose to apply a different environment to the project than what is set by default. Once the environment is specified, the **Environment** tab can be used to set values for variables in that environment. The variables are presented in the **Environment** tab according to their **On Project** property setting. They follow the same rules as they would if presented for a non-scheduled job start.
Overriding inheritance order	Use system setting **Apply server environment last** to override inheritance order. If its value is Yes, then the inheritance order is set as follows: 1. Project environment 2. Step environment 3. Server environment

Project variable changes made when starting a job

When users start jobs, they can change project variables, overriding variable values set in the project environment.

When a user starts a job, variables from the project environment are presented on a job start page. Depending on the On Project property of each variable, the user may change the value presented.

The changes made at job start are subject to the same inheritance rules as variables defined in a project environment.

Document # SC23-7681-08

For example:

1. You define the JavaEnv environment to have a variable with an initial value (JavaVersion = 1.4).
2. You define project MyBuild to use the JavaEnv environment.
3. You launch a job to execute project MyBuild. On the Job Start panel you change the value of JavaVersion to 1.5.

As a result:

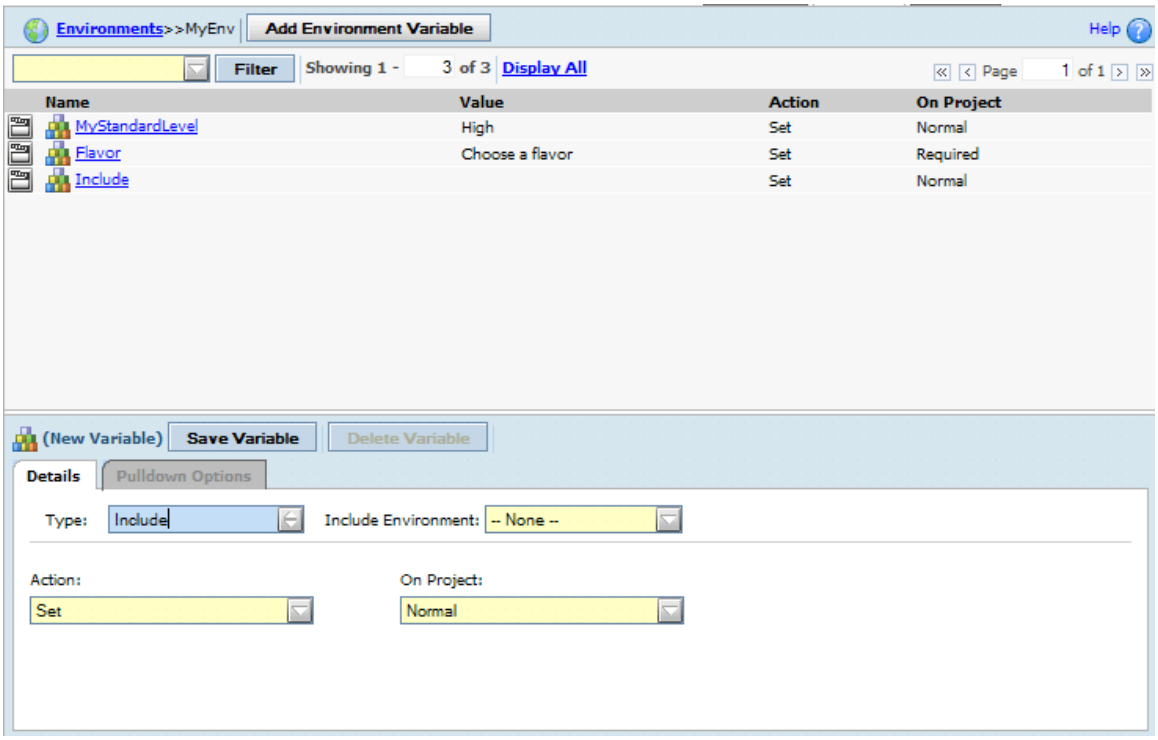
- Steps that do not override the project environment (JavaEnv) inherit the modified JavaVersion value of 1.5.
- Steps that explicitly use the JavaEnv environment as the step envirronment use the value of JavaVersion defined in the project environment: 1.4.

About variables

Variables are defined within environments.

In addition to a value, a variable has additional properties that govern its behavior when it is interpreted.

Variables in an environment



To create a new variable:

Document # SC23-7681-08

1. Click **Add Environment Variable** .
2. Specify a name for the variable.
3. Specify other properties for the variable as desired. See Variable properties for more information.
4. Click **Save** .

Once it is created, you can then click on the variable to edit it.

Variable properties

Use the variables panel to specify the following properties:

Name — Variable names can use only alphanumeric characters (a-z, A-Z, 0-9) and the underscore character (_) in a name. The maximum length is 255 bytes:

- Single-byte character sets: 255 characters
- Double-byte character sets: 127 characters

Value — Variable values have the following characteristics:

- **Length**: Values can be any length (up to the operating system limit, if there is one).
- **Special characters**: The characters %, $, [,], {, }, \, ", and ' have special meanings for the pre-parser (before OS evaluation) and for evaluation on the operating system. Avoid using them. Escape them with a backslash (\) to pass them. See Calling variables in steps.
- **Variables in values**: if a variable is in a variable value, that variable is interpreted when a step using the variable is executed.
- **Pulldown values**: To specify items for a pulldown variable, set the variable type to Pulldown List, save the variable, then edit it. Click the Pulldown Options tab to add items to the pulldown. See Creating pulldowns for a variable.
- **Dot commands as values**: Some dot commands can be used as the value of an environment variable; in these cases, the system replaces the dot command with other values. See “Using dot commands in variables” on page 53.
- **Carriage returns**: Variables do not store carriage returns. You can assign a multiple line value to a variable, as shown in the following example. The contents of the file `text.txt` are assigned to the variable *test*:

  ```
  .bset env "test = `type text.txt`"
  ```

  The lines in the file are concatenated. For example, suppose the file's contents were as follows:

```
A first line
And a second line
```

The variable's value then becomes the following:

```
A first lineAnd a second line
```

Type — Variables are assigned one of these types:

- **Standard** - the default. The variable can have a value and action assigned.
- **Include** - the variable value points to another environment to include. All variables in the environment are included.

  ### Note

  The Include variable type replaces the .include functionality provided in prior releases.

- **Pulldown List** - the variable contains a set of values that users can choose from. Once a pulldown list variable is created, you can select it and click on the **Pulldown Options** tab to add values.

Action — One of the following:

- **Set** : The default option. The specified value is assigned to the variable. The variable is created if it does not exist.
- **Set if not set** : This action assigns the value to the variable only if the variable does not already have a value. See “How environments are applied to steps” on page 41.
- **Append** : the value is appended to the current value for the variable. The OS-specific PATH delimiter is added between the values:
  - Windows: semicolon (;)
  - UNIX or Linux: colon (:)
- **Prepend** : the value is inserted in front of current value. The OS-specific PATH delimiter is added between the values:
  - Windows: semicolon (;)
  - UNIX or Linux: colon (:)
- **Clear** : the value is set it to an empty string. If the Value property contains a value, it is not used.
- **Delete / Unset** : the variable is deleted from the current applied environment. If the Value property contains a value, it is not used.

Document # SC23-7681-08

- **Assign Hidden** : The system assigns the variable, but hides the value in the logs, showing it as “*****”. Use this option when you are including a password or other sensitive information in a variable. Example: you need to include password information in an _MAP variable in order to map a drive. You want to hide the password from users who run the project.

  ### Note

  The system normally changes the syntax of a variable in a command line to the appropriate form for your operating system (%VAR% for Windows®, $VAR for Linux® and UNIX® systems). It cannot do this for a hidden variable.

On Project Defines how a variable is used when a job is run. This property affects only variables that are used in environments assigned to a project. The value can be one of the following:

- **Normal** : The variable behaves normally when assigned to a project.
- **Required** : A value must exist for the variable. Variables with this property are highlighted in the Start panel. A value defined in the variable definition is sufficient. If a value is not defined, a job cannot be quick-started or started.

  If a job containing Required variables is started by the scheduler rather than a user, the variables are left unchanged if they currently have a value or blank if they do not have a value.
- **Read-Only** : The value cannot be changed.
- **Suppress Display** : The variable is not displayed on the Start Job panel. However, the variable exists and can be used in steps.
- **Must Change** : The variable value must be changed. Variables with this property are highlighted in the Start panel. If a new value is not entered, the job cannot be quick-started or started.
- If a job containing Must Change variables is started by the scheduler rather than a user, the variable values are not changed.

Calling variables in steps

You can use either a UNIX®-style or Windows®-style variable syntax in step commands or environment variables definitions.

The system uses a preprocessor to interpret both UNIX-style ($VAR) or Windows-style (%VAR%) syntax into an appropriate format for the server where the step is run. The preparsing can enable a step to run on either a Windows-based server or a UNIX-based server.

Examples:

- The following two assignment statements are equivalent in a step:

Document # SC23-7681-08

```
echo %fooVar%      # Windows syntax
echo $fooVar       # UNIX or Linux syntax
```

- Variable assignments are not pre-processed. Therefore, avoid using direct assignments in a command line, especially in a scenario where a server may be selected without an operating system restriction. Use variables in environments.

```
set fooVar=100     # Windows
fooVar=200         # UNIX or Linux syntax
```

How variables are parsed:

1. The preparser evaluates the variable assignment. Special characters are consumed unless escaped by the backslash character ($, %, {, }, ", '). If preparsing is turned off, all characters are passed.
2. Each side of the variable assignment is evaluated by the target environment.
3. The evaluated variable assignment is executed.

The preparser, the Windows environment, and the various UNIX and Linux shells interpret special characters differently. Take care when using special characters and the backslash escape character.

Interpretation of undeclared variables

If a variable is called in a step but is undefined, the value returned depends on whether pre-parsing is turned on (default) or off. Pre-parsing behavior is set by editing the no_preparse command in the bfagent.conf file or the _NO_PREPARSE_COMMAND environment variable. See Trigger variables reference.

Variable format	Value returned - Pre-parsing on (default)	Value returned - pre-parsing off
echo %foo%	foo	Windows: %foo% UNIX or Linux: blank
echo $foo	foo	Windows: $foo UNIX or Linux: blank
echo ${foo}	foo	Windows: ${foo} UNIX or Linux: blank
echo $[foo]	empty string	Windows: $[foo] UNIX or Linux: system error

Tasks

The following sections describe procedures for accomplishing common tasks.

Document # SC23-7681-08

Creating pulldowns for a variable

You can define multiple possible values for a variable. The values you provide are displayed as selectable options in a pulldown.

To create a pulldown for an environment variable:

1. Select **Projects → Environments.**
2. In the list, select the environment.
3. Click **Add Environment Variable** . Fill in its properties as follows.
   - Name: enter the environment variable name.
   - Type: select **Pulldown List** .
   - Action: select an action.
   - On Project: select a property.
4. Click **Save** .
5. Click the variable.
6. Click the **Pulldown Options** tab.
7. Add values for the pulldown as follows:
   a. Specify a name for the pulldown option. The user sees this as a choice name in the pulldown. Pulldown names can use only alphanumeric characters (a-z, A-Z, 0-9) and the underscore character (_). The maximum length is 255 bytes:
      - Single-byte character sets: 255 characters
      - Double-byte character sets: 127 characters

   b. Specify a value for the pulldown option. You can set a value that is the same as the variable name if you want the user to see the value that is being used. If a variable is used in the value, it is interpreted on the operating system where the step executes. It is not preprocessed or evaluated before execution.

   c. Click **Create** . The option is added to the list.

   d. Repeat for each desired value.
8. Click **Save Variable** .

You can further work with options as follows:

- Use the edit icon to the left of each option name to position options in the list or delete them.

Document # SC23-7681-08

- Click on an option to edit it. You can edit both the **Name** and **Value** fields. Click **Save** when done.
- Click **Clear** to clear the **Name** and **Value** fields. You normally do this after viewing an existing option in order to create a new option.
- Set the default option presented. Do this after you have populated **Pulldown Options** . Click the environment. In the **Details** tab, set the **Default Option** property to the value you want to be presented by default. If **Default Option** is not set, the first item in the **Pulldown Options** list is presented by default.

Including other environments

You can include all variables from another existing environment using the Include variable type.

1. Select **Projects → Environments.**
2. Create a new environment, then click **Save** .
3. Click the environment name.
4. Click **Add Environment Variable** . Fill in its properties as follows.
   - Name: enter the environment variable name.
   - Type: select **Include** . The user interface changes to show an **Include Environment** pulldown. The Action and On Project properties are removed.
   - Include Environment: select the environment to include from the list.
5. Click **Save Variable** .

Changing variable values during step execution

Variables can be changed during execution with step, project, or permanent scope.

- Step scope: using a command in a step can override variable values by using explicit assignments. Those values are in effect only during the current step.
- Project scope: using the .bset command in a step changes the variable value for the scope of the running job. You can create new variables using .bset. They are in scope for the remainder of the job. Changes made using .bset take effect in the step *after* the step where .bset is used.
- Permanent scope: using the .set command in a step changes the environment variable definition. Variables are defined in server environments, project environments, and step environments. Changing a server variable or project variable using .set does not change the current job's copy of the variable. Jobs run after the current job get the changed variable. However, if a .set command changes a step environment variable, later steps that use the step environment get the changed variable. The .set command cannot create new variables.

Document # SC23-7681-08

For example, if you launch a project with a project environment named Java that includes a variable JavaVersion = 1.4, and you use .bset to change the value to 1.5, any steps that inherit that project environment get the value 1.5, while any steps that reference the Java environment specifically get the original value of 1.4.

Note that when the system starts a job, it copies the project environment variables to a database record set aside for the job, and refers to this job environment thereafter when getting project default values. If the user modifies the starting values of any project variables when the user starts the job, the values are recorded in the job record.

Mapping Windows drives

Windows® 2000 and XP operating systems manage mapped drives differently. Windows 2000 uses a common namespace for drive mappings while Windows XP uses separate namespaces for each user session. The agent attempts to remap remembered connections for user accounts, but may not be able to successfully complete the mapping at runtime. You can use a special environment variable to assist with drive mapping on Windows: the _MAP variable. When you set this variable, the Windows Agent maps drives before executing your steps.

A typical practice when using the _MAP variable is to assign it in the project environment, so that the same drive mapping is passed to all the step environments through environment variable inheritance. Note that if you also define a _MAP variable in a step environment, the step environment's value overrides the project environment, because only one _MAP value can be defined for a particular step.

Although it is intended for Windows environments, use forward slashes to separate directory path names in the _MAP variable. When the paths are used, the agent automatically corrects them as needed.

For example, setting _MAP to

```
X:=//server/share
```

defines a runtime mapping that connects the X: drive to the Windows UNC path name \\server\\share.

Multiple drives can be mapped by providing additional mapping specifications in the _MAP variable, with semicolons to separate them:

```
X:=//server/share;Y:=//server/share2
```

By default, drive mappings on Windows are performed using the same user name and password as defined for the logical server. You can map a drive for a different user name by adding the user name and password in parentheses after the mapping, as in the following example:

```
X:=//server/share(alternateusername,password)
```

Note

If your password contains a $ character, escape it with another $ character. Example: enter the password `pas$word` as `pas$$word`. Avoid using the following special characters in passwords: %, [,], {, }, ", or '.

Document # SC23-7681-08

Drives mapped via the _MAP variable are unmapped on command completion.

Even if the drives map successfully, drive mappings on Windows 2000 might still be inaccessible if a user logged onto the system's console is using the drive or share in question.

Mapping a Windows drive using the next available drive letter

You can have the system pick the next available drive letter. Use the following syntax:

```
<driveletter>?=//<directory path>
```

For example, you can set _MAP as follows:

```
X?=//server/share
```

In this case the system does *not* map a drive to X. Instead, it maps to the next available drive, and stores the *drive letter it selected* in a variable named _MAP_X. If the selected drive is F, the _MAP_X variable's value is “F:”. You can use the variable to access the mapping.

You can use multiple mappings as follows:

```
X?=//server/share; Y?=//server/public
```

You can use any letter you want, and you can even use multiple mappings, as in the following example:

- F: mapped to //server/share
- G: mapped to //server/public

The example also creates the following variables:

- _MAP_X with a value “F:”
- _MAP_Y with a value “G:”

Note

If you use the next available drive syntax when targeting a Windows® system that uses Cygwin, you must escape the question mark with a backslash as follows:

```
Y\?=//server/share
```

Agent-based drive mapping

You can map drives by using a configuration parameter in the agent. The *map* parameter, when added to the `BFAgent.conf` file, uses a syntax identical to that of the *_MAP* variable. You can use this parameter to create drive mappings for specific servers. If you also use the _MAP variable, its mappings override the agent mappings.

Document # SC23-7681-08

Using dot commands in variables

Some dot commands can be used as the name or value of environment variables.

Running scripts before a command with .source

The system provides the ability to execute a script on the server before the system runs the command by defining a special environment variable named .source. This allows you to load a set of environment variables from a source file on the server, or execute a custom preparation command.

To try this feature out:

1. Create a batch file on the system called mybatch.bat that echoes some sentence. Save the batch file to `c:\temp`.
2. Create a new environment called `Step Variables`.
3. Add a variable named `.source` with a value of `c:\temp\mybatch.bat`.
4. Set a step's environment to the newly created `Step Variables` environment.
5. Run the project, and examine the log output for the step.

Notice the additional log data showing that the mybatch.bat file was executed before the step command. Some important notes on .source:

- The path specified cannot have arguments.
- On Windows® platforms, the script is invoked via call.
- On UNIX® platforms, the script must be in the native shell syntax as it is sourced in the running shell.

Storing the date or time in a variable with the .date command

You can use the .date environment dot command to supply a variable with the current date or time. Use the command as the value of a variable. The system updates the variable with the result of the .date command when you run a project that uses the variable.

For example, a variable named MONTH with a value .date %B, when included in a project, is supplied to a job during May with the value “May”.

See “.date” on page 124 for more information on using this command, including a list of date format codes.

System variables reference

System-defined variables are available to use in variables.

Document # SC23-7681-08

The system automatically sets values for the following variables in each step of a job. These variables are read-only. Their values for the job are listed in the ENV lines of the step log. The first four are project-level notifications. All other BF_ variables are used at step-level.

Project-Level Variable	**Value**
BF_D	Date. Can be used in tags. Format is determined by the Tag: Date Format system setting.
BF_J	Day of the year. Can be used in tags.
BF_T	Time. Can be used in tags. Format is determined by the Tag: Time Format system setting.
BF_W	Day of the week, represented by a number from 0 (Sunday) to 6 (Saturday).

Step-Level Variable	**Value**
BF_D	Date. Can be used in tags. Format is determined by the Tag: Date Format system setting.
BF_J	Day of the year. Can be used in tags.
BF_T	Time. Can be used in tags. Format is determined by the Tag: Time Format system setting.
BF_W	Day of the week, represented by a number from 0 (Sunday) to 6 (Saturday).
BF_AGENT_PLATFORM	String identifying the operating system platform that the agent is running on.
BF_AGENT_VERSION	Version number of the agent for the current server.
B	Default tag variable, which starts at 1 and gets incremented for every job. Can be used in tags, which are represented by BF_TAG.
BF_BID	Job ID number, unique for jobs of the same project.
BF_CALLER_	Prefix applied to variables passed into a chained project from a calling project.
BF_CLASS	Build Forge class for the project
BF_ENGINE	A string that uniquely identifies the engine. This value is also stored in a file stored in the installation directory: `engine.id`. Example: D8531015-6C07-1014-8CA0-BD58317220B3.
BF_HOST	Host name of the logical server (TCP/IP hostname)
BF_ITERATION	Number of times a step in a While Loop has been started successfully. It is incremented when the Condition for the step evaluates to true. A

Document # SC23-7681-08

Step-Level Variable	Value
	job restart uses the value of this variable as the iteration to restart.
BF_ITERATION_MAX	Maximum number of times a While Loop can be executed. It is set in the step properties. If this number of iterations is reached, then BF_ITERATION_MAX_REACHED is set to Yes.
BF_ITERATION_MAX_REACHED	Not created or set by default. The step ID (BF_SID) of the While Loop step is *appended* to this variable when while loop iterations reach BF_ITERATION_MAX. If multiple While Loop steps in a project reach their BF_ITERATION_MAX, this variable contains multiple values, one for each step that reaches the maximum iterations.
BF_LASTGOODRUN	Date of the last passing job of the same project, or the last job if no passing job exists.
BF_LASTGOODTAG	Tag for the last passing job (or last job, if no passing jobs stored of the same project).
BF_LASTGOODUNIX	Same as BF_LASTGOODRUN, but expresses the date in UNIX® format.
BF_LASTRUN	Date of the previous run of the current job.
BF_LASTTAG	Tag string for the previous job of the same project.
BF_LASTUNIX	Same as BF_LASTRUN, but expresses the date in UNIX format.
BF_ONFAIL	Halt/Continue flag for the step
BF_PID	Project ID number.
BF_PROJECTNAME	Project name of this job.
BF_PROJECTNAME_PHYS	Project name as used to create the project directory. The system changes characters specified in the **Invalid Relative Dir Characters** system setting into underscore characters to create the project directory. For example, if the setting includes a space, then a project named `My Project` receives a project directory named `My_Project`.
BF_ROOT	Base working directory for the job, taken from job properties. See also BF_STEP_ROOT.
BF_SERVER	Server name that the current job is running on
BF_SERVER_ROOT	Path assigned to the logical server in the server properties

Document # SC23-7681-08

Step-Level Variable	Value
BF_SID	Step ID, unique identifier for the current step in the project.
BF_SPID	Contains the calling project ID if the current job was called by another job. If no the value is the same as BF_PID.
BF_STEPNAME	Step name. Set in the step properties.
BF_STEP_ROOT	Base working directory for the step, taken from step properties. See also BF_ROOT.
BF_SSID	Step ID of the calling step, if the current project was called by another project. If not, the value is the same as BF_SID.
BF_STATUS	Current status of a job, used internally to categorize jobs and display their state. BF_STATUS is not logged. However, BF_CALLER_STATUS appears in the logs of chained projects.
BF_TAG	Tag for the job. Tag definitions can contain variables. This variable contains the value resulting from interpreting those variables at the time the job starts.
BF_TAG_PHYS	Tag for the job, with underscores replacing any spaces that were in the BF_TAG value. If a step has the Absolute option select, then BF_TAG_PHYS is the same as BF_TAG.
BF_USER	User name of the job owner

Trigger variables reference

The system watches for the following variable names. When a step's environment contains one of them (either specifically or inherited from a project or server), actions are performed.

Variable	Contents
_CI_BUILD_DELETE	Set this variable to any value to delete the build and associated build data after the job runs. (The tag variable is reset to its initial value, prior to the deleted build, if no other project builds executed.)
_CI_BUILD_KEEP	Set this variable to any value to keep the build and associated build data after the job runs. For example, if your job includes an adaptor link and the adaptor step fails, the other project steps do not execute. You might want to keep a copy of

Document # SC23-7681-08

Variable	Contents
	the build records for the job, for example, for debugging.
_CLEARCASE_VIEWS	Specifies a list of ClearCase views to start before command execution. Set the value to a comma-separated list of views; for example, “View1,View2,View3”.
_CLEARCASE_VOBS	Specifies a list of ClearCase VOBs to mount before command execution. Set the value to a comma-separated list of VOBs; for example, “\Vob1,\Vob2,\Vob3”.
_CONTEXT_LOG_RANGE	Use this variable to limit log output to lines near filter matches. It takes a positive integer value, and causes the system to omit log output except for a range of lines around each filter string hit whose size is equal to the variable's value. For example, if you set the variable to 5, your logs show only lines with filter matches, plus the 5 lines preceding and 5 lines following those matches.
_ERROR_THRESHOLD	Establishes the maximum numbers of errors (caught by the “Set Fail” filters you have defined) allowed. Using this variable, you can establish thresholds for individual steps or for a project, and either fail the step or project when the threshold value is met, or merely note the fact that the threshold was met in the job notes. When you set the _ERROR_THRESHOLD variable for a project, you can use one of the following forms: • A value 5 or F5 indicates a maximum of 5 failures are allowed before failing the job. • A value like N7 indicates that the system should add a message to the job notes that the threshold was met when 7 errors occur. When you use the variable in a step, the system counts the errors in the individual step. Additional forms are available: • A value such as W9 indicates that after 9 errors, the step is put in a warning state, regardless of future errors caught by filters.

Document # SC23-7681-08

Variable	Contents
	• A value such as C8 indicates that after 8 errors, the step is set to failure status, but any Clear Fail filter can clear the failure. **NOTE:** The errors counted by this variable are defined as strings that match filters with Set Fail actions and which are assigned to steps in the project. Each string identified as a failure by a filter counts as one error toward the step total, and one toward the project total.
_EXITCODE_MAP	Specifies a list of numbers (separated by commas, spaces, semicolons, or colons) that the system should accept as indicators of step success. By default, an exit code of 0 indicates success; when this variable is specified, any values listed in it also indicate success.
_InterfaceLoggingLevel	Controls how much log data Build Forge logs when it runs an adaptor step. Create an environment variable (in your adaptor environment) with the name _InterfaceLoggingLevel. Assign it an integer value from 0 to 8. Logging levels are inclusive, for example, level 2 includes information from levels 1 and 0. • 0: Exec line plus server connection errors or cancel notification; nothing else • 1: Parsed commands (commands as they will be sent to the server) • 2: Unparsed commands (commands prior to having their local variables set) • 3: Build and environment variable SET lines • 4: Temp and internal variable SET lines • 5: Environment evaluations, e-mail group additions, BOM text logging lines • 6: Block & Sub-block start/end lines • 7: (Default logging level) Agent output that is checked against match patterns, plus the lines that matched the patterns. • 8: All agent output

Document # SC23-7681-08

Variable	Contents
_LOG	Specifies a path name to create a copy of the command output in. Use this variable to save a copy of the job log on the server. If the file exists, the system appends to it.
_MAP	See “Mapping Windows drives” on page 51 for a discussion of how to use this variable.
_NO_PREPARSE_COMMAND	The system normally attempts to resolve the values of environment variables before sending commands to agents. When the _NO_PREPARSE_COMMAND variable is defined (with any value), the system sends variables to agents without resolving them. Use this variable to ensure that your operating system shell handles the variables.
_PRISM_DIR_POSTCMD	Used with plug-ins for IDEs. Specifies a command to be run on directories after the project step has executed. See “Special variables for test projects” on page 278.
_PRISM_DIR_PRECMD	Used with plug-ins for IDEs. Specifies a command to be run on directories before they are copied to the server for a project step. See “Special variables for test projects” on page 278.
_PRISM_FILE_POSTCMD	Used with plug-ins for IDEs. Specifies a command to be run on files after the project step has executed. See “Special variables for test projects” on page 278.
_PRISM_FILE_PRECMD	Used with plug-ins for IDEs. Specifies a command to be run on files before they are copied to the server for a project step. See “Special variables for test projects” on page 278.
_SUPPRESS_LOG_OUTPUT	When defined (with any value), causes the system to omit nearly all of the log output normally produced by the agent. Some Management Console log messages remain, and filter matches are also shown.
_TIMEOUT	A value that overrides the Timeout property for one or all of the steps in your project.
_TRAP	A string to be executed if the current step fails; the string can be set to the name of an executable file or command. **NOTE:** The output of the command is not returned to the console because the connection between the console and the agent is closed when the step fails; if you want to retain output from a command

Document # SC23-7681-08

Variable	Contents
	executed via _TRAP, have the command write its output to a file for later retrieval.
_USE_BFCREDS	When set to 1, the system uses the *user's* login credentials to log in to servers, rather than the credentials stored in the server authorization attached to server. The system uses the Management Console login credentials of the user who started the project to execute the commands in the project. You can set this variable for a single step, or for an entire project. **Note** If you are using LDAP/Active Directory authentication, the **Store User Authentication Locally** system setting must be set to Yes (its default value) for the _USE_BFCREDS function to work. When the setting is Yes, the system caches user authentication information in encrypted form, and can then access the user authentication information for use with _USE_BFCREDS.

Snapshots of an environment

Snapshot an environment to quickly create a new instance of an environment that you want to change or modify.

Environment snapshot overview

Review these topics to learn about environment snapshots and understand how to use them.

Environment snapshot use cases

The following examples describe some common use cases for environment snapshots:

- Snapshot an environment to make changes to the environment configuration or perform testing of new tools or scripts.
- Store a snapshot of an environment as an temporary backup or as part of an official archive.
- Snapshot an environment to capture a point-in-time environment configuration that corresponds with a milestone, such as an external or internal release.

Document # SC23-7681-08

Environment snapshot concepts and terms

In the UI, snapshots introduce some new concepts and terms for working with environments.

Environment snapshot : A snapshot is a new instance of an existing environment. Some key points to remember about snapshots are as follows:

- A snapshot is a separate environment object. Making a change to one snapshot in a snapshot set does not affect the other snapshots in the set.
- A snapshot is not a copy.

  If you snapshot an object associated with an environment, snapshot creates a separate instance of the object. Copy maps relationships between objects, it does not create new objects.
- A snapshot is not a revision of an environment:
  - Snapshot does not support comparing changes between two environment snapshots.
  - Changes to environment snapshots are not tracked or identified with a version number as in a source control system. However, you can correlate environment snapshots to milestones by using a snapshot naming scheme that includes version numbers, for example, 7.5.0, 3.4.01.

Snapshot set : A snapshot set is the set of all the environment snapshots that are descendants of one base snapshot. At a minimum, the set includes the base or parent snapshot and a child snapshot. In the UI, the Snapshot icon beside the environment name indicates that a snapshot set for the environment exists.

Base snapshot : Initially, all environments have a snapshot name of Base Snapshot. You can change Base Snapshot to another name. The base snapshot is the parent of the snapshot set.

Default environment snapshot : The default environment snapshot is the current, working environment. Only one snapshot in the set can be the default. If you do not specify a default snapshot, the base snapshot is the default.

- In the UI, the default snapshot is displayed at the top-level of the environments list. Select **Environments** to display the environments list.
- When you select an environment with snapshots, the default environment snapshot is used unless you select a different environment snapshot in the list box.
- To access and work with other snapshots in the environment snapshot set, you must click the Snapshot icon.

Environment snapshot views

Select the Snapshot icon to display the Snapshot view. In the UI, the Snapshot view shows the hierarchy of the snapshots in a set:

Document # SC23-7681-08

- The base snapshot is at the top level and has the name Base Snapshot, if you do not assign it a unique name.
- All environment snapshots are children of a base snapshot. Children of the same base snapshot are indented at the same level in the Snapshot column.
- Environment snapshots that are created from a child snapshot become children of the child snapshot and are indented at the next level in the Snapshot column.

Environment snapshot planning

Review some best practices for selecting a default environment snapshot and naming environment snapshots.

- **Strategy for selecting the default snapshot in a set**

  The UI recognizes only one default or current environment snapshot for a snapshot set. Use a consistent strategy for selecting a default snapshot:

  - Use the base snapshot as the default snapshot

    Using this strategy, you create snapshots as point-in-time backups and do not make changes to the backed up environment snapshot. You make changes to the base snapshot.

  - Use the latest snapshot as the default snapshot

    Using this strategy, when you create a new environment, you do so with the intent of making it the new default environment snapshot. You do not make changes to the base snapshot or earlier environment snapshots.

- **Identifying a snapshot naming scheme for the set**

  The environment snapshot name must be unique within an environment snapshot set.

  Use the following criteria to help you create environment snapshot names:

  - The name should be descriptive: it should indicate snapshot usage or purpose.
  - The naming scheme should follow a defined standard. You can use the Comment box on the Snapshot tab to describe the naming scheme.

- **Using a single environment name for the set**

  After you create a environment snapshot, you have the option to change the name of the environment. If you change the environment name, it is updated for every environment snapshot.

Creating an environment snapshot

Creates a new instance of a environment. A snapshot is not a copy; it is a new environment.

You can create an environment snapshot from an environment or from an environment snapshot.

Document # SC23-7681-08

Tip

To create snapshots and change the default snapshot, users must have the required permissions. See “Verifying and editing access groups for snapshot permissions” on page 91.

1. Click the **Edit** icon beside the environment or environment snapshot that you want to snapshot:
   - To snapshot the default environment snapshot, in the environments list (**Environments**), click the Edit icon beside the top-level snapshot.
   - To snapshot a nondefault environment snapshot, click the Snapshot icon. The Snapshot view displays the environment snapshots in the set. Click the Edit icon beside the nondefault environment snapshot.
2. Click **Create New Snapshot** .
3. Enter **Name** on the Snapshot tab. The name is assigned to all the objects that you snapshot with the environment.

   The name must be unique within a environment snapshot set.
4. Select the Build Forge objects to snapshot when you create the environment snapshot. The objects that you can select are described in the following table.

Object	Description
Default	In the UI, the default snapshot is displayed at the top-level of the environments list. Select **Environments** to display the environments list.
Change References To Old Default	Updates references from the old default environment snapshot to the new default if the Default option is selected.
Follow Environment Includes	Snapshots the environments that the environment includes by using the Include environment variable type.

5. Click **Save** to save the environment snapshot.

Changing the default environment snapshot

The default environment snapshot is the top-level snapshot in the snapshot set and is displayed in the environments list (**Environments**).

Tip

To create snapshots and change the default snapshot, users must have the required permissions. See “Verifying and editing access groups for snapshot permissions” on page 91.

Document # SC23-7681-08

To change the default environment snapshot, edit the snapshot definition for the snapshot that you want to be the new default:

1. Select **Environments** .
2. In the environments list, click the **Snapshot** icon for the default environment snapshot.
3. In the snapshots list, click the **Edit** icon for the environment snapshot to be the new default.
4. Click **Make Default** .
5. **Important:** On the popup, choose OK or Cancel.

OK	**Update references**: For any objects that reference the previous default, update references from the previous default environment snapshot to the new default.
Cancel	**Do not update references**: For any objects that reference the previous default, do not update references to the new default environment snapshot.

Changing an environment snapshot name

You can change the snapshot name for an environment snapshot and also for the objects that you selected to snapshot when you created the environment snapshot.

For the base snapshot, you can use this option to change its default name of Base Snapshot to another snapshot name for a single environment snapshot only or for all current and future environments.

To change the snapshot name:

1. Select **Environments** .
2. In the environments list, click the **Snapshot** icon for the default environment snapshot.
3. In the snapshots list, click the **Edit** icon for the environment snapshot.
4. Select the **Snapshot** tab.
5. At **Name** , enter the new name.
6. **Optional:** At **Comment** , enter a comment.
7. **Important:** On the popup, choose OK or Cancel.

OK	**Change the environment snapshot name and other snapshot object names**: For objects that you selected to snapshot at the time that you created the environment snapshot, change the names of these objects and the environment snapshot. **For the Base Snapshot**: Changes the name of Base Snapshot for all current environment snapshots and all future environment snapshots.

Document # SC23-7681-08

Cancel	**Change the environment snapshot name but do not change other snapshot object names**: For objects that you selected to snapshot at the time that you created the environment snapshot, do not change the names of these objects. Change the environment snapshot name only. **For the Base Snapshot**: Retains the name Base Snapshot for all current environment snapshots and all future environment snapshots.

Accessing and viewing snapshots

Creating an environment snapshot creates a snapshot set that contains at least two environments: the base snapshot environment and the new environment snapshot.

To view all the environment snapshots in a snapshot set:

1. Select **Environments** .

   The environments list displays a list of environments and environment snapshots. The top-level snapshot is the default environment snapshot.

2. Click the **Snapshot** icon to display the environment snapshots in the snapshot set.

   In the Snapshot view, you can:

   - Create a new environment snapshot. To begin, click the Edit icon.
   - Change the default snapshot for an environment. Click the Edit icon for the new default snapshot and click Make Default.
   - Edit the environment snapshot definition just like you would for a standard environment.

Deleting an environment snapshot

You can delete an environment snapshot by using the Delete Environment option.

An environment cannot be deleted if it being used by another object; for example, if it is included by another environment or used by a project, a step, a schedule, or a server.

To delete an environment snapshot:

1. Select **Environments** .
2. In the environments list, click the **Snapshot** icon for the base snapshot.

   The Snapshot view displays the environment snapshots in the set.

3. Click the **Edit** icon beside the environment snapshot to be deleted.
4. Click **Delete Environment** .

Document # SC23-7681-08

Working with projects

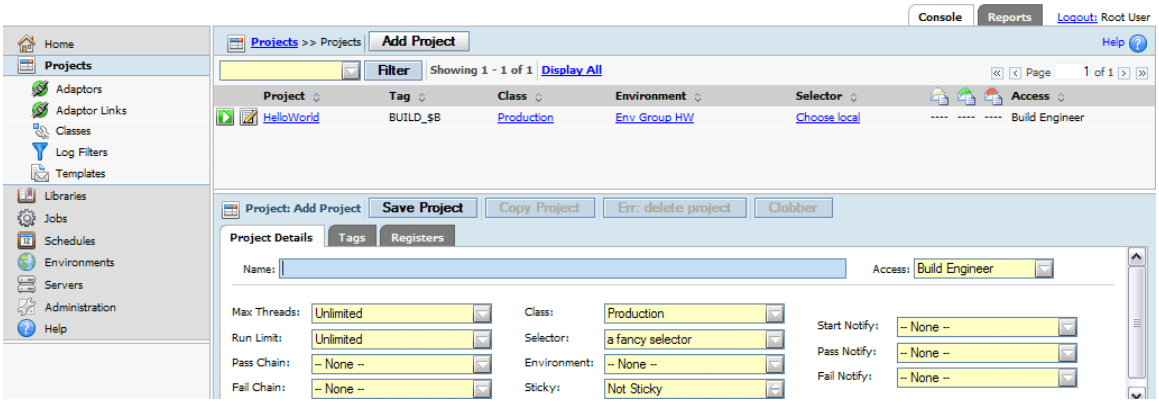
This topic describes how to create and manage projects in the Management Console.

About projects

Use the Projects module to create new projects and edit or view existing projects. Projects are executable sets of steps, with their own environment group and server properties.

Within the Projects module you can view a project, edit a project, or add a new project.

The Projects module



Changing project properties

To change project-level properties, select **Projects** , then click the icon next to the desired project's name.

Project Name

The name of the project; used to refer to the project in lists and in the database.

The project name is used to construct the project directory when the project is executed. You can use any characters in the project name, but you must make sure that the project name does not contain characters that are invalid. Because a project may contain steps that run on different operating systems, avoid special characters and symbols in your project names.

Access

The access group that is allowed to view and use the project. The **Access** property is used along with permissions to determine what a user can do. For example, to launch a job, you must be a member of the access group specified for the project and you must also be a member of a group that has the Execute Jobs permission. For more information on access groups, see “Access overview” on page 202.

Document # SC23-7681-08

Tag Format	A string that defines the tags for the project, using plain text and tag variable references. For more information on tag formats, see “Changing the build tag during a job” on page 114.
Tag Sync	Synchronize the tag variables for two projects. Select the project whose tag variable you want to synchronize with the current project. When two projects are synchronized, their variables are drawn from the same pool, so that when they run in sequence, one project gets the value 1, the next gets the value 2, and so on. For more information, see “Synchronizing tags” on page 72.
Max Threads	The maximum number of parallel processes the project is allowed to launch. Use this field to keep a project from using too many system resources. Each thread-enabled step and any inline projects (which themselves might launch thread-enabled steps) can result in parallel processes, but all of those processes are counted against the maximum for the parent project. The system stops launching new parallel processes when it reaches the Max Threads value, and waits until the number of parallel processes for the project drops below the Max Threads value before continuing. For more information about threading, see “Threading: running steps in parallel” on page 105.
Run Limit	The **Run Limit** property sets the maximum number of jobs of the project that are allowed at one time. If you launch a project, but the currently active jobs already equal the limit, the new job stays in the Waiting queue until one or more of the jobs completes. If a schedule attempts to launch a project when the number of running projects equals the run limit, the system does not launch a new job at all. Also, projects that are launched through an inline chain are not considered instances of the original project and do not count toward its run limits.
Class	Each project must be assigned to a class, which assigns global properties to groups of projects. For more information, see “Using classes” on page 80.
Selector	The name of the selector to use when choosing a server for the project. The system uses this selector as the default for any steps within the project that do not specify their own selectors. See “Selectors” on page 16. If a selector is not specified, the project is added to the Libraries module instead of the Projects module. Library projects use the selector of the calling project.
Pass/fail chain	Select the project that is executed when the project build passes or fails. Setting a pass/fail chain at the project level allows you to invoke separate pass/fail actions based on the pass/fail

Document # SC23-7681-08

	status of the project. This capability is similar to setting pass/fail actions at the step level within a project. At the project level, the pass/fail actions are triggered by the project run status not the step status.
Environment	An environment to apply after the Server environment and before the Step environment. For more information on how environments work together, see “How environments are applied to steps” on page 41.
Sticky	Enable the Sticky check box to force all the steps of the project that use the default project server to stay on the same server, and to wait for it to become available if it is busy. For more information on this option, see “Making steps stick with a server” on page 68.
Start Notify, Pass Notify, Fail Notify	Use these fields to direct the system to send notification e-mails on project start, pass, and/or fail, by selecting an access group in one or all of these fields.

Copying a project

To make a copy of an existing project, display the existing project's list of steps by selecting **Projects** → **<ProjectName>** , then click **Copy Project** .

Select similar options from the **Library** module to make a copy of a library.

The system displays a dialog box to prompt you for a name for the new project or library.

When you copy a project or library, the system copies the following references from the existing object to the new object:

- The steps and all of their properties;
- All the project properties listed on the **Project Details** tab, such as the project's tag format, class, selector, and other properties.

The system does not copy the following properties:

- Tag variables (found in the **Tags** tab on the project properties form);
- Project registers (found in the **Registers** tab on the project properties form).

Making steps stick with a server

Steps within a project can run on different servers if their selectors allow it. But you may want all or most of the steps of a project to run on the same server, whether or not you specify that server in advance. The project-level **Sticky** property gives you that option.

Document # SC23-7681-08

To view project-level properties, select **Projects** , then click the  icon next to the desired project's name.

The Sticky property applies only to the steps in a project that do not specify a selector of their own. If a step has a selector option other than None, the system uses that selector to choose a server for the step—even if the selector is the same as the project's selector.

When the Sticky property is set, the project uses the same server for every step whose selector field is set to None.

When the system starts an inline project, it chooses a server for the project based on the inline project's settings. The Sticky property of the calling project does not affect the inline project, and the inline project obeys its own Sticky property if it is set.

When the system starts an inline library, it obeys the following rules:

- An inline library, with Sticky property not checked: uses the selector of the calling step as the default selector for the inline steps.
- An inline library, with Sticky property checked: uses the *server* of the calling step as the default server for the inline steps.

Note

You can use the .bset server command to change the default server for a project during the job. Steps that occur after the .bset command use the new default set by that command, and stick to that new server.

Chaining projects together

You can link projects together using a feature called chaining. You can use this feature to maintain frequently-used groups of steps separately from projects that depend on them. Other uses include executing automated test and deployment projects upon completion of certain steps. You can also use chaining to clean up files that are no longer needed by development teams, by assigning a project to be run on upon completion of a job of a specific class.

There are two ways to chain:

- Pass Chain: specifies a project or library to run when a step passes.
- Fail Chain: specifies a project or library to run when a step fails.

A conditionally chained project inherits some characteristics from the calling project but otherwise runs according to its own properties and environments.

- The called project uses its own properties, including its own notification settings and chain settings.
- The called project applies its own project environments *after* applying the calling project's environments. See also “Environment variable inheritance in chained projects” on page 70.

Document # SC23-7681-08

- The chained project inherits the *class of the calling project* by default. This behavior can be changed in **Administration → System Configuration** by setting Override Class when Chaining to N or n.
- The called project uses the server specified by its own selector.

Note that if the chain is to a library (selector is set to None), it looks for the following, in this order:

- 1st: the calling step's selector
- 2nd: the project's selector

If the calling step is using the project's selector, then the chain uses that selector as well. Steps of the chained project use the project selector, even if a different selector is specified for those steps.

If you want to include a platform-independent step as a placeholder step for chaining a project, you can use `.sleep 0` as the step command.

If you use a .break command within a chained project, the system stops the chained project but returns control to the calling project, which then continues. See ".break" on page 123.

You can create nested sets of chained projects. When you chain (launch) one project from another, the chained project can itself be chained to another project. The chain of projects you can create can be no more than 32 levels deep, but is otherwise limited only by the available memory of the host running Management Console.

Environment variable inheritance in chained projects

When a project is launched through a chain, the system applies environment variables from the calling project. They are applied differently depending on whether the called project is called as an Inline chain or as a Conditional chain (Pass Chain or Fail Chain).

- Inline chain: the steps of the chained project are executed as though they are *part of* the calling project. If Step 1 in Project 1 calls Project 2, variables are applied in this order:

  1. Server environment for Project 2, if specified. If not specified, the server environment variables for the Project 1 server are used.
  2. Project 1 environment
  3. Step 1 environment
  4. Step environments for each step in Project 2, if specified

  In each step, a variable's value is controlled by the last environment to set it.

- The system applies all the environments that are relevant. The called project sets up variables from the calling project's environment and its own environment in the following order:

  1. Called project server environment.

Document # SC23-7681-08

2. Calling project's variables, in a set, with BF_ variable names changed to BF_CALLER.
3. Called project server environment (applied a second time in case it was modified by the caller's variables).
4. Called project environment.
5. Step environments (if specified) as they are executed.

Canceling chained projects when wait enabled

Typically the system does not cancel chained projects. Set the Pass Wait or Fail Wait attribute to Yes to have the system automatically cancel the called projects for a Pass Chain or Fail Chain. The system cancels the called project when the calling project or the calling step is canceled.

Defining tags

The system uses *tags* to identify specific jobs of a project, and to construct the name of the job directory in which process activity takes place by default. The system makes the tag for a job from the *tag format* property for the project, which can contain static text as well as numeric *tag variables*.

The default tag format for projects is BUILD_$B, which uses the default tag variable B, an automatically incremented value that is defined by the system for every project. This default tag format results in a stream of build tags as follows:

BUILD_1

BUILD_2

BUILD_3

You are not limited to these tags, however. You can define your own tag variables and set up your own tag formats to produce a variety of tag types. You can also use the .retag command during a job to change the tag to an arbitrary string. (For more information, see ".retag" on page 136.

The current job's tag is available as an environment variable (BF_TAG) defined by the system during a job, so that you can access and use it to label source repositories, or for other tracking or labeling purposes. (For more information on these variables, see "System variables reference" on page 53.)

You can synchronize the tag variables from two projects; this creates a link so that when either one runs, the same tag variable values are used. (For more information, see "Synchronizing tags" on page 72 for details.

The topics in this section describe how to set up tag formats and tag variables to produce dynamic tags that reflect the values you want.

Editing the tag format for a project

The tag format defines how the system constructs the tag. The tag format consists of plain text and variable references indicated by the $ symbol. Any variables you use in the tag format must

Document # SC23-7681-08

be from the list of system-defined tag variables in the preceding section, or you must define these variables for the project before it runs. Variables that are not defined are treated as static text.

Tag format is a project property. To edit it, click the Project button to display the list of projects, then click the project name for the project you want to edit. The system displays the list of steps in the project; click the project name at the top of the page to display the project properties.

In your tag format, use a $ symbol to indicate the start of a tag variable. You can include several tag variables if desired. For example, you could define a non-incrementing variable for a project's major revision ($MAJ) and an incrementing variable for its minor revision ($MIN), then have a tag format that reflects the project's version number, for example, Version$MAJ.$MIN. This allows you to manually control the major version number, but automatically increment the minor version number with every release, producing tags like the following:

Version1.1

Version1.2

Synchronizing tags

You can synchronize tags across different projects, so that two or more projects increment the same variable, with the project-level Tag Sync property. When you set a Tag Sync property for Project B equal to Project A, you establish a parent-child relationship between Project A (the parent) and Project B (the child).

When you run a project with a Tag Sync property, the system checks to see if any of the tag variables in the child project match variables in the parent project. If the $B variable is used in both tags, for example, and the parent project is on run 7, the child project uses the next value, 8. After run 8, the next run of either project is run 9.

If no variables in the child project's tag format match variables in the parent project's tag format, the Tag Sync property has no effect.

Only the variables in the tag are synchronized, so you can still distinguish between different projects.

For example, if you define two projects as shown in the following table:

Project	Tag format	Tag sync
Project A	Project_A_$B	-- None --
Project B	Project_B_$B	Project A

And you then run the projects alternately (Project A, then Project B, then A and B again), your completed jobs list shows the tags as follows (with the last run shown first):

Project	Tag
Project B	Project_B_4
Project A	Project_A_3
Project B	Project_B_2
Project A	Project_A_1

Document # SC23-7681-08

System-defined variables for tags

You can use the following predefined variables in your job tags:

Variable	Value
B	The job number: an integer value that starts at 1 and is incremented with every new job.
BF_D	Date, in the format set by the Tag: Date Format system setting.
BF_ENGINE	Identifier of the engine.
BF_J	Day of the year.
BF_T	Time, in the format set by the Tag: Time Format system setting.
BF_W	Day of the week (a numeric value, from 0 to 6).

These variables are standard variables populated by the system every time it creates the environment for a step. For example, a tag format value of BUILD_$B.$BF_T produces tags like the following for successive jobs:

BUILD_9.120529

BUILD_10.120533

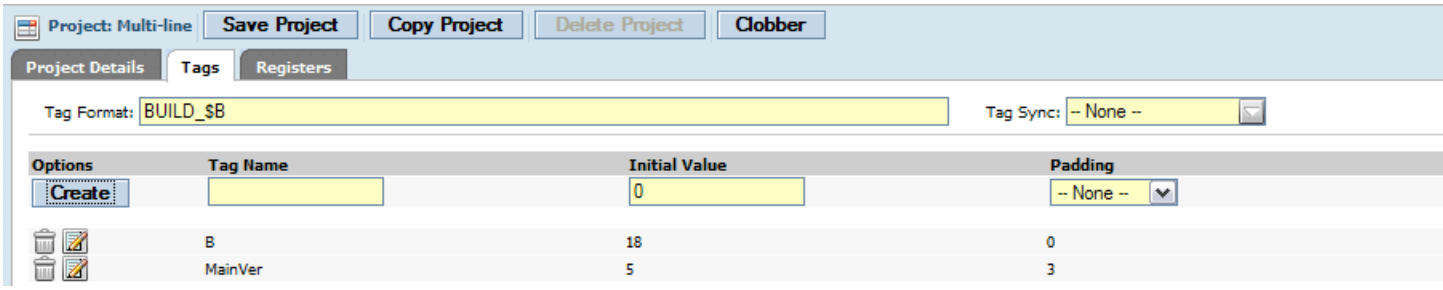
Creating or editing tag variables

You can define your own tag variables to include in tag formats. Tag variables take numeric values and can be incremented by the system automatically with each job, if desired.

To add new variables or edit exist ones for a project, select **Projects** , then click the icon next to the desired project's name. The project properties appear in the lower pane; click the **Tags** tab to display the project's tag variables.

The system displays a list of tag variables for the project.

Tags tab, showing tag variable form and tag variable list



- To edit a tag variable, click the icon next to its name. The system fills the tag form on the left side of the tab with the tag variable's values and changes **Create New Tag** to a **Save Edited Tag** button. Change the values and click **Save Edited Tag** to store your changes.
- To delete a tag variable, click the trash can icon next to its name.

Document # SC23-7681-08

- To add a new variable, enter properties for the variable and click the **Create New Tag** button.

Each tag variable has the following properties:

Tag Name	The name of the variable. When you use a tag variable in a tag format, reference its name using the form $<Tag Name>. For example, to create a tag that uses the MainVer and B variables, use a tag format "Build_$MainVer.$B" to get tags like Build_005.1.
Initial Value	Sets the value for the tag variable. If you do not use the **Auto Inc** option, the variable retains this value until you change it.
Padding	If you select a **Padding** value other than None, the system adds leading zeroes to the value of the variable when it is used in a tag if needed to make sure the number of digits equals the **Padding** value. For example, if the variable is current at 2, and it has a Padding of 3, then the system renders the value as 002. Padding can range from 1 to 8.
Auto Inc	If set to Yes, the system increments the variable's value by 1 for every job of the project.

Libraries

A library is any project whose Selector property is set to None. Libraries are intended to be run within other projects. They run on the server resource of the step that calls them.

When you save a project that has a selector of None, the system warns you that it will be saved as a libaryr. Libraries are listed in the **Libraries** panel.

To call a library from a step, choose it in the Inline, Pass Chain, or Fail Chain properties of the step.

About libraries

The Libraries panel contains libraries, which are projects that do not have a selector specified.

Libraries use the selector of any project that calls them. Libraries are typically called by other projects as an Inline for a step or as a Pass chain or Fail chain for a step.

From the Libraries module, you can view, edit, create, or launch library projects. You can execute a library project by itself, but you must specify a selector when you do this.

Note

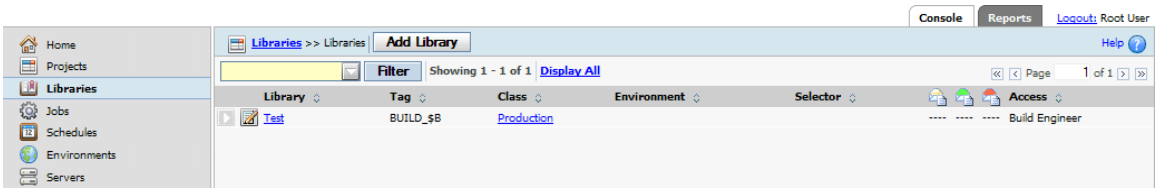
The first selector in the list is assigned by default when launching a library. If you want to specify a different default selector, make it a project.

You can change a library project into a normal project by editing the project and choosing a selector for it. When you save a library project with a selector, it becomes an ordinary project, disappearing from the Libraries list.

Document # SC23-7681-08

Aside from the lack of a selector, libraries are treated just like any other project.

The Libraries Module



Copying a library

To make a copy of an existing project, display the existing project's list of steps by selecting **Projects** → **<ProjectName>** , then click **Copy Project** .

Select similar options from the **Library** module to make a copy of a library.

The system displays a dialog box to prompt you for a name for the new project or library.

When you copy a project or library, the system copies the following aspects of the existing object to the new object:

- The steps and all of their properties;
- All the project properties listed on the **Project Details** tab, such as the project's tag format, class, selector, and other properties.

The system does not copy the following properties:

- Tag variables (found in the **Tags** tab on the project properties form);
- Project registers (found in the **Registers** tab on the project properties form).

Deleting projects

There are two ways to delete projects, depending on whether the projects has any jobs associated with it.

Choose one of the following options for deleting projects:

Delete Project button

The **Delete Project** button can delete projects that have no jobs. The button appears on the project property editing page and the project step list. To view project-level properties, select **Projects** , then click the icon next to the desired project's name. Deleting a project cannot be undone. If you want to delete a project with this button, make sure all of the project's jobs are deleted first.

Clobber button

The **Clobber** button deletes a project and all of its associated jobs from the Build Forge database. The system asks for

Document # SC23-7681-08

confirmation before clobbering a project. After you clobber a project, you cannot undo it. The button appears on the project property editing page; select **Projects** , then click the icon next to the desired project's name.

Log filters

Use log filters to specify the success criteria for a step. A filter stores one or several regular expressions.

If filtering is not set up, Build Forge determines the success or failure of a step command by the command's exit status, where 0 is success and 1 is failure. If multiple commands are used in the Command property, only the exit status of the last command executed affects the step result status. Note that some commands always return a 0 exit status. A reporting command like `net use` prints a list of mapped network drives. The command always succeeds, even if the list does not contain the desired drive.

Log filters allow you to evaluate the output from the commands in a step rather than the exit status.

For example, for the `net use` command, you can use a log filter to look for a specific drive and mark the step as successful if it is found.

You can create new filters, add criteria to them, and edit existing criteria at **Projects** → **Log Filters** .

Creating a log filter

Log filters may contain one or more filter patterns. Each filter pattern is associated with an action, and optionally an access group for notification. You define a log filter first and then associate the log filter with a step in the project.

To create a log filter, do the following:

1. Select **Projects** → **Log Filters** . The Management Console displays the list of Log Filters and the New Log Filter form.
2. At **Name** , enter a name for the log filter, then click **Save** . The Management Console saves the log filter and displays the New Pattern form.
3. For each filter pattern that you define for the log filter, do the following:
    a. Enter a regular expression in the **Pattern** field. (The regular expression must be Perl-compatible.) Build Forge searches step output for the pattern when the project runs. For details, see “Filter patterns” on page 77.
    b. At **Action** , choose a filter action to take when the filter pattern is found. The default property, Set Fail, sets the step status to Fail. For details, see “Filter actions” on page 78.
    c. At Notify, optionally select an access group to send members a notification e-mail when the filter is activated.

Document # SC23-7681-08

d. Click **Save** .

To use the log filter, choose a project step and set the step's **Result** property to the new log filter. See “Assigning a log filter to a step” on page 77.

Assigning a log filter to a step

To use the log filter, you must assign the log filter to a project step using the step Result property. When you assign a log filter to a step, the filter patterns in the log filter are run on the step output whenever the project runs.

Note that when you assign a log filter to a step, the step result that is set by the log filter overrides all other criteria for determining the success or failure of the step. This includes the exit status for the step commands or any step properties. For example, if the step run time exceeds the time specified by the step Timeout property, the step stops. But, its status is not considered a failure unless its associated log filter action causes it to be set to Fail.

To assign a log filter to a step, do the following:

1. Select **Projects** or **Libraries** to access the step.
2. Select the project or library that contains the step.
3. Select the step to open the step Details form.
4. At **Results**, select the log filter that you want to run each time the step executes.

Filter patterns

A filter pattern defines the character string or expression that you want to match in step output. Each filter pattern you create is associated with a single filter action. Both filter patterns and actions are defined in filter log sets. The ability to include multiple filter patterns in a log filter and apply it to output from a single step allows you to use multiple search criteria without constructing complex expressions.

To create a log filter, select **Projects** → **Log Filters** . For details, see “Log filters” on page 76.

Filter pattern syntax

Review these guidelines for creating filter patterns:

- The filter pattern is defined as a regular expression and must use Perl-compatible syntax. For details about constructing Perl-compatible expressions, refer to Perl documentation.
- The system adds the delimiting forward slash characters (/<expression>/), so specify the expression **without** surrounding forward slash delimiters (expression).
- If your expression includes a metacharacter (for example, a /b), the metacharacter must be preceded by a backslash escape character (a\/b).

Syntax for some standard regular expressions are shown in the following table.

Expression	Matches
Production	Matches the *Production* anywhere in the string.
^Production	Matches *Production* at the beginning of the string.
Error:.*[0-9]$	Matches a line that contains *Error* followed by any set of characters terminated by a number at the end of the string.
[Ww]arning	Matches *Warning* or *warning*.
.*	Matches 0 or more of any character. The dot (.) matches any character, and the asterisk (*) matches any character 0 or more times.

Multiple pattern matches on the same line

To construct a pattern filter, it is important to understand how the system searches for pattern matches.

For each line of output, the system checks for matches against all the filter patterns in order; it stops when it finds a match, and moves on to the next pattern. So, if the pattern occurs twice on one line, the system may not find it.

```
exception retrying exception
```

For example, using the previous line of step output and the filter patterns in the following table, the system would match the first *exception*, set the step result to Fail, match *retrying* and set the step result to Pass, and move on to the next line without matching the second *exception*.

One way to resolve this problem is to replace the filter patterns in the table with the following filter pattern:

```
retrying.*exception
```

Filter patterns	Filter actions	Example description
[Ee]xception [Rr]etrying	Set Fail - Fail Clear Fail - Pass	This is useful for Java projects; it fails the step on exceptions, but clears the failure on a retry. If the retry fails, a new exception will be generated, so that the final state of the command is valid.

Filter actions

Filter actions define what action is taken when a filter pattern is found in step output. Each filter pattern you create is associated with a single filter action. Both filter actions and patterns are defined in log filters.

To create a log filter, select **Projects** → **Log Filters** . For details, see “Log filters” on page 76.

Filter Action	Definition	Step Results
Set Fail (the default)	When the system finds the filter pattern, it sets the step results status to Fail and continues searching the current line for filter patterns in the set.	Fail

Document # SC23-7681-08

Filter Action	Definition	Step Results
Set Fail/Halt	When the system finds the filter pattern, it sets the step results status to Fail, stops searching the current line for the filter patterns in the set, skips to the next line, and starts the pattern search again.	Fail
Clear Fail	When the system finds the filter pattern, it sets the step results status to Pass and continues searching the current line for the filter patterns in the set.	Pass
Clear Fail/Halt	When the system finds the filter pattern, it sets the step results status to Pass, stops searching the current line for the filter patterns in the set, skips to the next line, and starts the pattern search again.	Pass
Halt	When the system finds the filter pattern, it stops searching the current line for the filter patterns in the set, skips to the next line, and starts the pattern search again. It does not change the step results status.	not applicable
Include	Include allows you to reference one or more log filters in another log filter. You specify the log filter you want to include in the Pattern field and select Include in the Action field.	not applicable
Warning	When the system finds the filter pattern, it sets the step results status to Warn and continues searching the current line for the filter patterns in the set.	Warning
Clear Warning	When the system finds the filter pattern, it sets the step results status to Pass and continues searching the current line for filter patterns in the set.	Pass
Clear Warning/Halt	When the system finds the filter pattern, it sets the step results status to Pass, stops searching the current line for the filter patterns in the set, skips to the next line, and starts the pattern search again.	Pass
Notify Changers	To use Notify Changers, an adaptor that creates a relationship list must be included in the project and the adaptor step must run before the step that contains the Notify Changers log filter. The adaptor relationship list pairs users and objects (such as changed files). For details, see the Adaptor XML Reference. After the adaptor runs and creates the relationship list, if a log filter with the Notify Changers action matches its filter pattern in a step output line, the line is scanned again to try to match objects in the relationship list. If an object match is found, the users paired with the object are sent e-mail notification. For example, in the following line of step output, the object match for the filter pattern `Error` is MyFile.c. So, the users paired with	not applicable

Document # SC23-7681-08

Filter Action	Definition	Step Results
	the MyFile.c object in the relationship list are sent e-mail notification of the error. `Error: Invalid token on line 55 of MyFile.c`	
Stop Build with Fail Result	When the system finds the filter pattern, it sets the job result to Fail and exits the job. The step result is set according to its results. No further steps are run.	result
Stop Build with Pass Result	When the system finds the filter pattern, it sets the job result to Pass and exits the job. The step result is set according to its results. No further steps are run.	result
Stop Build with Warning Result	When the system finds the filter pattern, it sets the job result to Warning and exits the job. The step result is set according to its results. No further steps are run.	result

Filter notification

For every filter pattern in the log filter, you can optionally set notification to send e-mail to an access group to notify members that a pattern filter for a step has been activated.

Error thresholds

You can use a special environment variable, _ERROR_THRESHOLD, to establish thresholds for individual steps and/or for a project. The system then counts the number of filter matches, and either fails the step or project when the threshold value is met, or notes the fact that the threshold was reached in the job notes.

For more information, see “Trigger variables reference” on page 56.

Error and warning counts

If filters are associated with steps to determine whether the steps succeed or fail, the system displays the number of errors and warnings caught by the filters. In the **Jobs → Completed**) tab, the numbers appear in the **Results** column in parentheses after the job result. The format is **(<** *fail matches* **> / <** *warning matches* **>)**.

Example: A job result of **Failed (1 / 0)** shows that the job failed, 1 Fail filter was matched, and no Warning filters were matched.

Using classes

A class is a group of jobs; each job must be a member of one and only one class. You can use classes to set up types of jobs, and apply behavior to each type globally. A job gets its default class from the properties of its project, but you can manually choose a different class for a job when you launch it from the **Jobs → Start** page.

Classes have properties to manage the following activities:

Document # SC23-7681-08

- Deleting jobs automatically
- Launching jobs when the system purges job of this class, or when an existing job is changed to or from this class.

Note

You can change the class of a job after it completes. To change the class of a job, view the job by selecting **Jobs → Completed** and then click the job tag (BUILD_5, for example). Select a different class in the **Class** field.

To edit the properties of a class, select the **Projects → Classes** page. The system displays a list of classes; click an existing class, or click the **Add Class** button.

The **Access** property of a class controls which users can view or change it, based on the access group you assign.

About classes

A class is a group of jobs. Each job must be a member of one and only one class. You can use classes to apply different global management behaviors to each job in a class. A job gets its default class from the properties of its project. You can also manually choose a different class for a job when you launch it from the **Jobs → Start** page.

Classes have properties to manage the following activities:

- Deleting jobs automatically
- Launching jobs when the system purges job of this class, or when an existing job is changed to or from this class

Note

You can change the class of a job after it completes. To change the class of a job, view the job by selecting **Jobs → Completed** and then click the job tag (BUILD_5, for example). Select a different class in the **Class** field.

To edit the properties of a class, select **Projects → Classes** . The system displays a list of classes; click an existing class, or click **Add Class** .

The **Access** property of a class controls which users can view or change it, based on the access group you assign.

Deletion properties of classes

Most of the properties for classes control what kinds of project data are deleted and under what conditions they are deleted.

The system checks for jobs to delete at an interval defined by the Purge Check Time system setting, which defaults to 15 minutes.

Document # SC23-7681-08

Note

You can also use schedules to denote when purges should be performed, so that the system does not try to run purges when the system is otherwise occupied. You can use this feature to have purges occur only at night, or once a week, for example. See “Using classes” on page 80.

When a purge job runs, the system archives the job and deletes data according to settings in the class.

Delete Files: Determines what kinds of data are deleted. It has the following options:

Everything — Deletes all information about the job from the database and deletes the job directory from the server(s) that ran it.

Console Data — Deletes all information about the job from the database, but leaves the job directory on the server intact.

Logs And Files — Deletes the job directory and the logs, but retains step pass/failure information on the **Jobs → Archived** page.

Logs Only — Deletes only the job logs.

Files Only — Deletes the job directory on the server(s) that ran the job. Logs and some other information (such as step pass/fail status) remain within the database; the job record moves to the **Jobs → Archived** page.

Days: The number of days old a job must be before it is deleted.

Count: The maximum number of jobs allowed. When the number of jobs exceeds the Count value, the system schedules purge jobs to delete the extra builds. The default value, Unlimited, prevents the system from deleting jobs because of the number of jobs that exist.

Note: The system deletes jobs when **either** the Days or Count values are exceeded. For example, if you have Count set to 10 and Days set to 2, and there are 8 jobs, but 3 are more than 2 days old, those three jobs would be deleted. Similarly, if you had 12 jobs, all less than 2 days old, the two oldest jobs would be deleted.

Which: The Which property sets additional conditions that must be met before a job can be deleted. It has the following options:

Any Build — When this option is selected, the Which property has no effect on job deletion.

Only Failed — The system deletes only failed jobs.

Only Passed — The system deletes only passed jobs.

Keep 1 Pass — The system always retains the most recent passed job, even if it meets other deletion criteria.

Document # SC23-7681-08

Launching projects on class events

You can launch (chain) projects when certain events occur that are relevant to classes. Using these properties, you can model a progression of states in your processes.

The following properties of classes allow you to launch jobs when certain events occur:

Start on purge	This property launches the specified project when any job in the class is purged (that is, whenever the system starts a purge job for a job with this class). You can use this property to make sure that some specific files are deleted, which are not automatically deleted along with the purge.
Start on entry	This property launches the specified project when a job's class property is changed to this class. You can use this property to tie a process to the reclassification of a job; for example, you could create a Test class, and launch some standard tests when a job is promoted to the Test class.
Start on exit	This property launches the specified project when a job's class property is changed from this class to another class.

These properties launch projects as chains.

Setting up notification

The system can send out e-mail notifications when projects or steps pass or fail, or when certain other events occur. This section describes how to configure e-mail notifications and how to modify the notification templates that control what e-mail notifications look like.

Notifications are sent to access groups, so design your access groups with notifications as well as security in mind.

Note

Notifications are always sent to groups, not individual users directly, but you can set up groups that contain only one user, if needed.

To create a notification event, select an access group for a notification property.

- For projects, you can choose Start Notify, Pass Notify, and Fail Notify groups.
- For steps, you can choose Pass Notify and Fail Notify groups.

Whenever these properties have access groups selected, the system sends e-mail to the group members when the appropriate event occurs.

When a project includes another project as an inline project, the inlined project's start, pass, and fail notification settings are ignored, but any notification settings for its steps are honored. See “Notification for inlined projects” on page 87 for details.

Before you can use notification, you must:

Document # SC23-7681-08

- Configure the SMTP Server system setting so that the system knows what SMTP server to use to send e-mail. The default is localhost. You might also need to set the system setting for **System Alert Source** . This address is used as the source address and most SMTP servers require a valid source address. The default is root@localhost.
- Create one or more notification groups and assign users to them.
- Select groups to notify for individual projects and/or steps.

In addition, you can configure the notification e-mails that the system sends out by editing notification templates. See “Customizing notification templates” on page 85.

About notification templates

Notification templates provide a means of sending customized messages to user about events in the system. The system includes a number of templates which you can customize for your organization's needs. You can also create templates for specific projects using the **Projects → Templates** page. Plain text templates are sent as plain text e-mail messages; templates that contain common HTML markup are sent as MIME messages. The system parses template bodies for a number of variables (see “Using environment variables and register variables in templates” on page 86).

Configuring your SMTP server

To configure your SMTP server, select **Administration → System → SMTP Server** . The system displays an edit form for the SMTP Server value. Enter the name of your site's SMTP Server. The default is localhost.

You might also need to set the **System Alert Source** parameter as this is used as the source address and most SMTP servers require a valid source address. The default is root@localhost.

For more information, see “System configuration settings” on page 216.

Setting notification properties of projects and steps

When the SMTP and group configuration for notification is in place, you can configure projects and steps to send notifications when certain events occur.

- For projects, you can set Start, Pass, and Fail Notify properties. These are project properties.
- For steps, you can set Pass or Fail Notify properties. These are step properties.

Notification exercise

The following procedure describes how to set up and try out e-mail notification. The exercise requires an SMTP server and an e-mail account.

1. Set up your SMTP server as described in “Configuring your SMTP server” on page 84

   Make sure you have a user account that sends e-mail to an e-mail account you can access.

2. Select **Administration → Access Groups**
3. Create a new access group called Email Test with your chosen user as the Initial Member.
4. Select a project (for example, the Hello World project) and edit its project properties. Select the Email Test group in the Start Notify, Pass Notify, and Fail Notify fields.
5. Run the project.
6. Verify that you received two e-mails: one to indicate the start of the project, and one to indicate its success or failure. If you do not receive the e-mails, verify that you used the correct SMTP server value.

Important

If the `<notify>` directive fails (for example, if the user the e-mail is addressed to doesn't exist), the .xml will fail, and all subsequent notifications will fail.

Customizing notification templates

Notification templates provide a means of sending customized messages to user about events in the system. The system includes a number of templates which you can customize for your organization's needs. You can also create templates for specific projects using the **Projects → Templates** page. Plain text templates are sent as plain text e-mail messages; templates that contain common HTML markup are sent as MIME messages. The system parses template bodies for a number of variables. For more information, see “Using environment variables and register variables in templates” on page 86).

Creating new templates for specific projects and steps

The system comes with templates for many events that can occur in the system. You can create new ones that are specific to a particular project and/or step. When you do this, the more specific template is used instead of the global project template. For example, you can create a variant of the normal Project Run Start message that applies only to your FinalProductionWidget project.

To create a new template, select **Projects → Templates** . The system displays the current list of templates. Click **Add Template** to add a new one.

When you click **Add Template** , the system presents some choices. You can select a project, step, and type of notification:

- Project: Choose the project that the new template applies to. The template is only used on notification messages that are generated for runs of the selected project.

  ### Note

  A list of snapshots appears if more than one snapshot is defined for the project. Choose the snapshot to use. The notification applies only to projects run using that snapshot. If you specify Default Snapshot, the snapshot assigned as the default is used.

Document # SC23-7681-08

- Step: You can choose a specific step (so that the template only applies to notifications for that step) or select Project Events to have the template apply to all notifications for the selected project.
- Notification: If you selected Project Events as the Step option, you can choose from a list of project events. If you chose a specific step, you can choose from Step Pass, Step Warn, and Step fail messages for your notification type.

Editing notification templates

To edit a notification template, select **Projects → Templates** , then click on the name of the template you want to edit. Newly-created templates default to the same text as standard templates, until you edit them.

Using environment variables and register variables in templates

You can reference environment variables (ones defined by you as well as standard system variables) in notification templates, as long as you use the ${VAR} syntax.

You can also include register variables for a project in notification templates. If you reference an empty register, the system returns an empty string.

Special notification template variables

The following table lists the special variables available to notification templates. Some variables are context sensitive and are only available when relevant (for example, the STEPNAME variable is not set for project level notifications, only step level notifications).

Variable	Contains
ACTION	For purge templates, describes the type of deletion performed.
BID	Specifies the job ID number. Used to construct links back to the Management Console to access reports.
CMD	For source adaptors, identifies link command specific error strings.
CONSOLEHOST	The host name of the Management Console machine.
CONSOLEPORT	The port number used by the Management Console. Allows you to construct valid URLs within a notification template.
CONTEXTLOGLINKS	Lists lines from the log that begin with “FILT:”, with three lines of context per entry. The system provides links to the Management Console log entries in the message.
DURATION	For steps, specifies the number of seconds the step ran.
EID	Specifies the Environment ID number. Used to construct links back to the Management Console to access reports.
FULLNORMALLOG	Shows the log information for each step in the job, excluding the environment setup actions that appear in the detailed log.
LINK	For source adaptors, specifies the link name.
MESSAGE	Contains the error or message text for failure or alert messages.

Document # SC23-7681-08

Variable	Contains
ONFAIL	For steps, holds the continue property for the step.
PATH	Specifies paths where appropriate, for data items like servers or steps.
PID	Specifies the Project ID number. Used to construct links back to the Management Console to access reports.
PROJECTNAME	Contains the name of the project.
RUNACTION	Specifies the variable that the e-mail template leverages.
SELECTOR	Contains the selector name for a step or project.
SERVER	Contains the logical server name for a step.
SID	Specifies the Step ID number. Used to construct links back to the Management Console to access reports.
SRVRHOST	Contains the TCP/IP host name of the server for a step.
START	Contains the date/time a job started.
STEPNAME	For steps, contains the name of the step.
STEPNORMALLOG	Shows the log information for the current step in the job, excluding the environment setup actions that appear in the detailed log.
TAG	Contains the tag string for a job. The same value as $BF_TAG.
TAILNORMALLOG	Works like FULLNORMALLOG, but only displays the end of the log. The number of lines displayed is controlled by the **Tail Log Amount for Mail Template** system setting. This variable must be used in a step notification template.
UID	Specifies the User ID number. Used to construct links back to the Management Console to access reports.
USEREMAIL	Contains the e-mail address for the owner of a job/event.
USERNAME	Contains the full name for the owner of a job/event.

Notification for inlined projects

The system handles notifications for inlined projects as if the steps from the inlined project were embedded in the calling project:

- When a project contains an inlined project, the inlined project's project-level notification settings are ignored. When Project A inlines Project B, no messages about the start, passing, or failure of Project B are sent.
- Step-level notification settings are unaffected. If a step has a Pass or Fail Notify access group set, the appropriate messages get sent, whether the step is in a top-level project or an inlined project.
- The inlined steps contribute to determining whether the calling project succeeds or fails. A failure in an inlined step, for example, either causes the calling project to fail or, if the step is set to Continue On Failure, changes the calling project's state to Passed with Warnings.

Document # SC23-7681-08

Using snapshots to create new instances of a project

Snapshot a project to quickly create a new instance of a project that you want to change or modify. A project snapshot is a separate and executable project. You can also use snapshot to create new a new instance of a library.

Project snapshot overview

Review these topics to learn about project snapshots and understand how to use them.

Project snapshot use cases

The following examples describe some common use cases for project snapshots:

- Snapshot a project to make changes to the project configuration or perform testing of new tools or scripts while continuing to run jobs with the existing project.
- Store a snapshot of a project as an temporary backup or as part of an official archive.
- Snapshot a project to capture a point-in-time project configuration that corresponds with a milestone, such as an external or internal release.

Project snapshot concepts and terms

In the UI, snapshots introduce some new concepts and terms for working with projects.

Project snapshot : A snapshot is a new instance of an existing project. Some key points to remember about snapshots are as follows:

- A snapshot is a separate project. Making a change to one snapshot in a snapshot set does not affect the other snapshots in the set.
- A snapshot is an executable project. It runs with the objects that you also select to snapshot when you create the project snapshot or with the objects associated with the source project, also called the base snapshot.
- A snapshot is not a copy.

  If you snapshot an object associated with a project, a separate instance of the object is created. Copying a project copies the relationships between objects, it does not create a new instance of a selector, environment, or an inline or chained project.
- A snapshot is not a revision of a project:
  - Snapshot does not support comparing changes between two project snapshots.
  - Changes to project snapshots are not tracked or identified with a version number as in a source control system. However, you can correlate project snapshots to milestones by using a snapshot naming scheme that includes version numbers, for example, 7.5.0, 3.4.01.

Document # SC23-7681-08

Snapshot set : A snapshot set is the set of all the project snapshots that are descendants of one base snapshot. At a minimum, the set includes the base or parent snapshot and a child snapshot. In the UI, the Snapshot icon beside the project name indicates that a snapshot set has been created for the project.

Base snapshot : Initially, all projects have a snapshot name of Base Snapshot. You can change Base Snapshot to another name. The base snapshot is the parent of the snapshot set.

Default project snapshot : The default project snapshot is the current, working project. Only one snapshot in the set can be the default. If you do not specify a default snapshot, the base snapshot is the default.

- In the UI, the default snapshot is displayed at the top-level of the projects list. Select **Projects** or **Jobs** > **Start** to display the projects list.
- When you select a project with snapshots as an inline project or chained project, the default project snapshot is used unless you select a different project snapshot in the list box.
- To access and work with other snapshots in the project snapshot set, you must click the Snapshot icon.

Project snapshot views

Select the Snapshot icon to display the Snapshot view. In the UI, the Snapshot view shows the hierarchy of the snapshots in a set:

- The base snapshot is at the top level and has the name Base Snapshot, if you do not assign it a unique name.
- All project snapshots are children of a base snapshot. Children of the same base snapshot are indented at the same level in the Snapshot column.
- Project snapshots that are created from a child snapshot become children of the child snapshot and are indented at the next level in the Snapshot column.

Project snapshot planning

Review some best practices for selecting a default project snapshot and naming project snapshots.

- **Strategy for selecting the default snapshot in a set**

  The UI recognizes only one default or current project snapshot for a snapshot set. Use a consistent strategy for selecting a default snapshot:

  - Use the base snapshot as the default snapshot

    Using this strategy, you create snapshots as point-in-time backups and do not make changes to the backed up project snapshot. You make changes to the base snapshot and continue to run jobs using the base snapshot project only.

  - Use the latest snapshot as the default snapshot

Document # SC23-7681-08

Using this strategy, when you create a new project, you do so with the intent of making it the new default project snapshot. You do not make changes to the base snapshot or earlier project snapshots. The latest snapshot is the one used to run jobs.

- **Identifying a snapshot naming scheme for the set**

  The project snapshot name must be unique within a project snapshot set.

  Use the following criteria to help you create project snapshot names:

  - The name should be descriptive: it should indicate snapshot usage or purpose.
  - The naming scheme should follow a defined standard. You can use the Comment box on the Snapshot tab to describe the naming scheme.

- **Using a single project name for the set**

  After you create a project snapshot, you have the option to change the name of the project. If you change the project name, it is updated for every project snapshot.

Project snapshot options

When you create a snapshot, you must select which objects to include in the snapshot. The following table describes the options available for Build Forge objects.

- Objects that are automatically included in the project snapshot.
- Objects that are optionally created and included if you select them when you create the project snapshot.
  - For these objects, a new object is created in the UI with the same snapshot name as the project snapshot.
  - For example, if the name of your project snapshot is release_7.1, the snapshot name of the environment, selector, inline projects or libraries, and chained projects or libraries is also release_7.1
- Objects not included in the project snapshot; you must manually create these objects and add them to the project.

Automatically included in project snapshot	**Optionally included with project snapshot** *** Copied only, a separate instance is not created**	**Not included with project snapshot**
project steps	environments of the project and its steps	log filters for steps
project tags	environments added by an environment variable of type Include for snapshot environments	
project notes	inline projects or libraries and their steps	
	inline projects or libraries and their steps	
	chained projects or libraries and their steps	

Document # SC23-7681-08

Automatically included in project snapshot	Optionally included with project snapshot * Copied only, a separate instance is not created	Not included with project snapshot
	selectors of the project and its steps	
	selectors added by a selector property of type Include for snapshot selectors	
	* project registers (copied)	
	* project tag variable values (copied)	
	* templates for notification (copied)	
	* adaptor links (copied)	

Verifying and editing access groups for snapshot permissions

Verify that users have the permissions required to create snapshots and to set the default snapshot. If not, assign permissions to users by using access groups.

Permissions are assigned to users through access groups, which can be either provided by Build Forge or created by a Build Forge administrator.

To verify and edit access groups assigned to snapshot permissions:

1. Select **Administration > Permissions** .
2. In the permissions list, select **Display All** to list all permissions.
3. Verify that the correct access groups and users have access to the following snapshot permissions:

Create Snapshots	User permission required to create a snapshot for projects, environments, and selectors.
Set Default Snapshots	User permission required to set or change the default snapshot for projects, environments, and selectors.

Creating a project snapshot from an existing project or project snapshot

Creates a new instance of a project and the objects that you choose to snapshot. A snapshot is not a copy; it is a new, executable instance of a project.

1. Click the **Edit** icon beside the project or project snapshot that you want to snapshot:
   - To snapshot the default project snapshot, in the projects list (**Projects**), click the Edit icon beside the top-level snapshot.

Document # SC23-7681-08

- To snapshot a nondefault project snapshot, click the Snapshot icon. The Snapshot view displays the project snapshots in the set. Click the Edit icon beside the nondefault project snapshot.

2. Click **Create New Snapshot** .

3. At **Name** on the Snapshot tab, enter the snapshot name.

   The name must be unique within a project snapshot set. The name is assigned to all the objects that you snapshot with the project.

4. Select the Build Forge objects to snapshot when you create the project snapshot. The objects that you can select are described in the following table.

Object	Description
Default	In the UI, the default project snapshot is displayed at the top-level of the projects list. Select **Projects** or **Jobs** > **Start** to display the projects list.
Change References To Old Default	Updates references from the old default project snapshot to the new default if the Default option is selected.
Include Project Environments	Snapshots the project and step environments in the project.
Follow Environment Includes	If Include Project Environments is selected, also snapshots any other environments that are included by an environment variable of type Include.
Include Project Selectors	Snapshots the project and step selectors that you have included in the project.
Follow Selector Includes	If Include Project Selectors is selected, also snapshots any other selectors that are included by a selector property of type Include.
Clone Project Adaptor Links	Copies the adaptor link as part of the snapshot. The adaptor link adds an adaptor into the project. The adaptor runs as the first step (step 0) in the project.
Clone Project Registers	Copies the project registers as part of the snapshot.
Clone Project Tag Variable Values	Copies the tag values for the project tag variables. The tag variables are automatically copied, but their values are not. If you do not copy the tag values, they are reset to 1.
Clone Project Templates	Copies the notification templates for pass and fail notification events that are set at the project level and step level.
Include Chained Projects	Snapshots chained projects or libraries and their steps that are referenced at the project level or step level. Chains are triggered by a project pass/fail or a step pass/fail condition.

Document # SC23-7681-08

Object	Description
Include Project Inlines	Snapshots inline projects or libraries and their steps that are referenced at the step level. Inlines are triggered by a step and run after the step completes.

5. Click **Save** to save the project snapshot.

Changing the default project snapshot

The default project snapshot is the top-level snapshot in a project snapshot set and is displayed in the projects list (**Projects**).

To change the default project snapshot, edit the snapshot definition for the snapshot that you want to be the new default:

1. Select **Projects** .
2. In the projects list, click the **Snapshot** icon for the default project snapshot.
3. In the snapshots list, click the **Edit** icon for the project snapshot to be the new default.
4. Click **Make Default** .
5. **Important:** On the popup, choose OK or Cancel.

OK	**Update references**: For any objects that reference the previous default, update references from the previous default project snapshot to the new default.
Cancel	**Do not update references**: For any objects that reference the previous default, do not update references to the new default project snapshot.

Changing the snapshot name for a project snapshot

You can change the snapshot name for a project snapshot and also for the objects that you selected to snapshot when you created the project snapshot.

For the base snapshot, you can use this option to change its default name of Base Snapshot to another snapshot name for a single project snapshot only or for all current and future projects.

To change the snapshot name:

1. Select **Projects** .
2. In the projects list, click the **Snapshot** icon for the default project snapshot.
3. In the snapshots list, click the **Edit** icon for the project snapshot.
4. Select the **Snapshot** tab.

Document # SC23-7681-08

5. At **Name** , enter the new name.
6. **Optional:** At **Comment** , enter a comment.
7. **Important:** On the popup, choose OK or Cancel.

OK	**Change the project snapshot name and other snapshot object names**: For objects that you selected to snapshot at the time that you created the project snapshot, change the names of these objects and the project snapshot. **For the Base Snapshot**: Changes the name of Base Snapshot for all current project snapshots and all future project snapshots.
Cancel	**Change the project snapshot name but do not change other snapshot object names**: For objects that you selected to snapshot at the time that you created the project snapshot, do not change the names of these objects. Change the project snapshot name only. **For the Base Snapshot**: Retains the name Base Snapshot for all current project snapshots and all future project snapshots.

Accessing and viewing snapshots in a project snapshot set

Creating a project snapshot creates a snapshot set that contains a minimum of two projects: the base project and the new project snapshot.

To view all the project snapshots in a snapshot set:

1. Select **Projects** .

   The projects list displays a list of projects and project snapshots. The top-level snapshot is the default project snapshot.

2. Click the **Snapshot** icon to display the project snapshots in the snapshot set.

   In the Snapshot view, you can:

   - Create a new project snapshot. To begin, click the Edit icon.
   - Change the default snapshot for a project. Click the Edit icon and click Make Default.
   - Edit the project snapshot definition just like you would for a standard project.

Starting a job for the default project snapshot

Use the Quick Start icon or the Start Project page to start the default project snapshot.

To start the default project snapshot by using the Quick Start icon:

1. Select **Projects** .

Document # SC23-7681-08

2. In the projects list, click the **Quick Start** icon.

   The job runs using the default values for the selector, class, tag format, and environment.

To start the default project snapshot by using the Start Project page:

1. Select **Jobs** → **Start** .
2. In the projects list, click the project name of the default snapshot to display its Start Project page.

   On the Start Project page, you can change environment variables, the selector, the class, and the tag format.

   If the selector has selector snapshots, the field below the Snapshot field lists the selector snapshots that you can select.

   ### Note

   To quickly select the default selector snapshot, select Default Snapshot. The Default Snapshot maps to the name of the default selector snapshot.

3. Click **Execute** .

Starting a job for a nondefault project snapshot

Use the Quick Start icon or the Start Project page to start a nondefault project snapshot.

To start the non-default project snapshot by using the Quick Start icon:

1. Select **Projects** .
2. In the projects list, click the **Snapshot** icon for the base snapshot.

   The Snapshot view displays the project snapshots in the set.

3. Click the **Quick Start** icon beside the nondefault project snapshot.

   The job runs using the default values for the selector, class, tag format, and environment.

To start the non-default project snapshot by using the Start Project page:

1. Select **Jobs** → **Start** .
2. In the projects list, click the **Snapshot** icon for the base snapshot.

   The Snapshot view displays the project snapshots in the set.

3. Click the project name of the nondefault project snapshot to display its Start Project page.

   On the Start Project page, you can change environment variables, the selector, the class, and tag format.

Document # SC23-7681-08

If the selector has selector snapshots, the field below the Snapshot field lists the selector snapshots that you can select.

Note

To quickly select the default selector snapshot, select Default Snapshot. The Default Snapshot maps to the name of the default selector snapshot.

4. Click **Execute** .

Deleting a project snapshot

Delete a project snapshot by using the Delete Project or Clobber Project options.

- The Delete Project option is available if there are no jobs for the project snapshot and if the project snapshot is not referenced by other objects, either by classes or by other projects as an inline or chain project.
- The Clobber option deletes a project and its jobs from the Build Forge database and removes any references to the project by other objects.

To delete a project snapshot:

1. Select **Projects** .
2. In the projects list, click the **Snapshot** icon for the base snapshot.

   The Snapshot view displays the project snapshots in the set.
3. Click the **Edit** icon beside the project snapshot to be deleted.
4. Click **Clobber** or **Delete Project** .

Document # SC23-7681-08

Working with steps

This topic describes how to create and manage steps in the Management Console.

About steps

A step is a component of a project. When the project is run as a job, each step is executed in order. A step contains one or more commands and has step properties that affect its behavior.

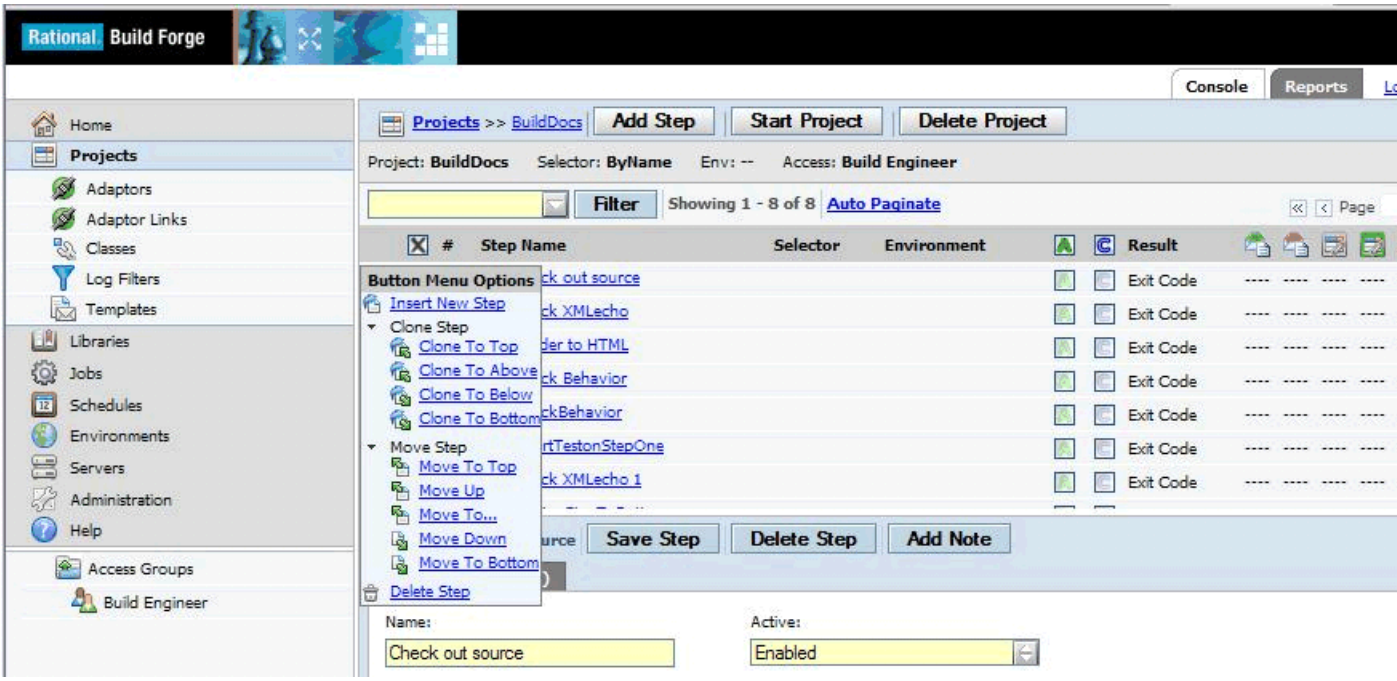
To work with steps, select **Projects** , then click on a project name. The system displays the list of steps for the selected project. When the list is displayed you can add, edit, copy, and move steps.

- **Editing a step**: Click the step name. The system displays the step's properties in the lower pane. Make your changes, then click **Save step** .
- **Disabling a step**: Click the check box in front of the step name. When a step is checked ☒, it has the following effects on the project:
  - The step is not executed when you run the project as a job.
  - The step is greyed out in the list of steps when you start a job normally: Select **Jobs** → **Start** , then click the job name, then click **Job Steps** . The step is visible but cannot be made active for the job you are starting.

  You can also disable a step by setting the **Active** property to Disabled in the **Details** tab for the step. When you save the step, the check box is checked.
- **Adding a step to the end of the project**: Click **Add Step** at the top of the main pane. The system displays an empty step details form. Enter values for the properties (you must enter a **Name** and **Command** at least), then click **Save Step** .
- **Other step operations**: Click the Actions icon in front of the step name to display additional options:
  - **Insert New Step** :
  - **Clone Step** : Copy a step and all of its properties. The name is changed to add a number at the end of the step. Copying a step named Build results in a new step named Build 1. The number is set automatically. You can copy to these locations:
    - Top: start of the list
    - Above: immediately before the current step
    - Below: immediately after the current step
    - Bottom: bottom of the list
  - **Move Step** : move a step to a different position in the list. You can move to these locations:

Document # SC23-7681-08

- Top: start of the list
- Up: move up one position
- Move to...: a dialog requests a step number. The step is moved to that position and other step numbers are adjusted as needed.
- Down: move down one position
- Bottom: bottom of the list



Details tab: step properties

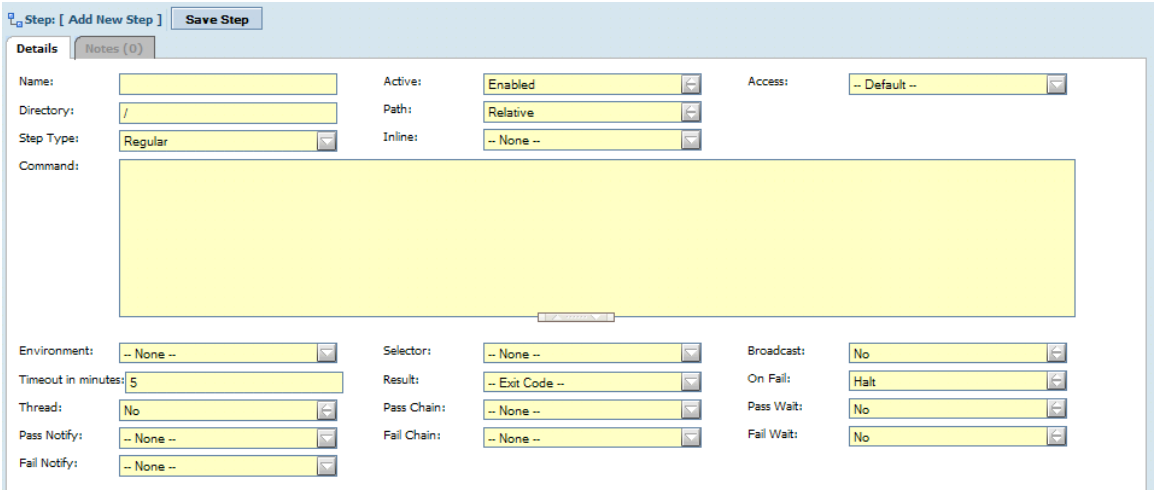
Step properties specify how to execute a step, handle its output, and what to do when the step completes. A step can also execute another project or library.

To view step properties, select a step within a project. The Details tab is shown by default. It displays the step properties.

If a step property is not set explicitly, its value is inherited from the project. Step properties set for a step override inherited values.

Document # SC23-7681-08

Step Details tab, showing step properties



Step properties include:

Name	The name of the step. It is used as a label for the step in the system and the log.
Active	Specifies whether the step is run. By default a step is Enabled. Select Disabled to prevent the step from running. A disabled step is not available to be run in a job.
Directory	Sets the location where step commands execute. The system automatically creates a unique directory for every job. The **Directory** field provides a convenient way to execute commands in directories your project has constructed during a job. (Build Forge does not construct directories mentioned in the **Directory** field.)
Path	Specifies whether **Directory** is an absolute or a relative path. • **Relative** : Step commands are executed in a path found by adding together the server, project, job, and step directories. • **Absolute** : Step commands are executed in a path found by adding together the server and step directories. This option allows you to access directories that are not in the project directory structure. Example: It can be used to launch applications permanently installed on the server.
Step Type	Determines how the step is executed. Step execution includes the execution of the contents of Command and execution of the project specified in Inline, if any. • Regular: The step is executed once.

Document # SC23-7681-08

	• Conditional: The step is executed once if the expression in the Condition property evaluates to true. Selecting conditional causes the Condition, Else Inline, and Else Command properties to be shown. If the Condition property evaluates to false, then the Command and Inline are not executed. Instead, the Else Command and Else Inline are executed if they are specified. • While Loop: The step can be executed multiple times. It is executed until the expression in the Condition property is false or until the maximum number of iterations is reached. Selecting While Loop causes the Condition and Max Iterations properties to be shown.
Inline	Specifies a project or library to run inline. The steps from the project or library are executed using the environment and properties of the current project. The behavior is as if the steps in the specified project are copied after the current step.
Access	Choose an access group to define which users are allowed to use the step. You can use this property to restrict access to specific steps within a project. When a user who is not a member of the access group for a step launches the project that contains the step, the step is skipped. Choosing Project Default causes the step to inherit the access properties of the project.
Max Iterations	Shown only if Step Type is While Loop. Specifies the maximum number of iterations that the step can be executed in a loop. The system-imposed default is 100. The step is shown as completed successfully (passed) in the step log. Use **Fail step if max reached** , to make the step fail when Max Iterations is reached. When jobs are executing, the read-only variable BF_ITERATION contains the number of iterations entered successfully. If a job is stopped and then restarted, it is restarted at the iteration in BF_ITERATION.
Fail step if max reached	If Yes, a While Loop step fails if Max Iterations is reached. If No, the step passes.
Else Inline	Shown only if Step Type is conditional. Specifies a project to be run inline if the specified condition is false. Default is No.
Command	Enter one or more operating system commands, subject to the following guidelines:

Document # SC23-7681-08

- Multiple commands: you can execute more than one command in an individual step. Separate individual commands by placing them on separate lines.

  ### Note

  When you use the default Exit Code setting for the **Result** property of your step, the success or failure of the entire step is based on the exit code returned by the last command in the step. To detect failure in any of the commands, create a Log Filter and specify its use in the **Result** property.

- Threading: All commands in a single step are executed on one server. If you want to be able to thread commands across multiple servers, the commands must be in separate steps.

- Shell specification: You can use the #! directive to specify the shell to be used to execute the command(s). This works on Windows® as well as Linux® and UNIX® systems (the Windows agent handles passing the commands to the specified interpreter). To send the commands from your step to a copy of Perl in c:\perl\bin on Windows, use #!c:\perl\bin\perl.exe. If you use the Windows agent with Cygwin, but need to direct a command to the Windows shell cmd.exe, you can use the following line, which takes advantage of Windows implicit paths:

  ```
  #!cmd.exe /C
  ```

  Note that the /C option is required for cmd.exe as otherwise it waits for additional commands after your step commands are delivered to it. You might use the `#!/bin/perl` command on a UNIX or Linux machine.

  ### Note

  When you use the `#!` command on Linux or UNIX systems, the system does not change to the standard default directory (the path constructed from a combination of the server path, project, name, and step path field) because it cannot predict the required syntax; you must include your own directory-changing command. Use special environment variables created by the system, such as BF_SERVER_ROOT and BF_PROJECTNAME_PHYS, to do this.

Document # SC23-7681-08

Condition	Shown only if you have selected a step type of Conditional or While Loop. • Conditional: the command is executed if the condition evaluates to true. • While Loop: the command can be executed multiple times as long as the condition evaluates to true. You can set the limit using Max Iterations. A condition can be an function or a command to be run on the selected server resource. • A *function*, if used, must be used at the beginning of the Condition field. It is evaluated by the Build Forge engine. It is not sent to the server resource. For a list of the functions and instructions on how to use them, see “Condition functions” on page 108. • A *command* is run on the selected server. Any command used here must be valid in the agent's shell environment. The return code from execution determines whether the condition passes or fails. Build Forge variables for the project are available to use in a condition expression. See “Calling variables in steps” on page 47 for more information on how variables can be expressed and how they are evaluated.
Else command	Shown only if you have selected a step type of Conditional. Specifies a command to execute if condition evaluates to false.
Environment	Specifies an environment to apply before executing the commands. Values in this environment override any values inherited from the server environment, project environment, and step variables. Choosing **Project default** causes the step to inherit the environment of the project.
Selector	Specifies a selector to use to choose a server for this step. If left as **Default** , the step executes on the server determined by the project's selector.
Broadcast	If checked, runs the step on **every** server matching the current selector (the step selector if specified, otherwise the project selector). At run time, the system replaces a broadcast step with a series of steps, one for each server, and executes them serially or in parallel, depending on the broadcast step's **Thread** property.

Document # SC23-7681-08

Timeout	Specifies how many minutes the system should wait for the current command to produce output (default is 5 minutes). If the timeout value is reached, the system fails the step. The project also fails unless the step is set to Continue on Fail.
Result	The Result property determines how the system judges whether a step succeeded or failed. Use the default value of Exit Code to determine success based on an exit code returned by the command shell. You may also choose a Log Filter that examines the command output. To select a Log Filter, you must first create it.
On Fail	Specifies whether to halt or continue the job if the step fails. By default, the system halts the job.
Thread	If Yes, runs this step in parallel with other steps. Set this property to Yes to allow threading of this step (running the step in parallel with other steps). Set the property to No to avoid threading. Set the property to Join to separate threaded blocks of steps. The first set of steps must complete before the next set of threaded steps following the Join step can start.
Pass Notify	Specifies the access group to be notified if the step passes.
Pass Chain	Specifies a project to launch if the current step passes. The steps for the chained project are indented in the log.
Pass Wait	If checked, the system suspends the current project until the pass chained project completes. If this step (or its project) is cancelled, the chained project is also cancelled.
Fail Notify	Specifies the access group to be notified if the step fails.
Fail Chain	Specifies a project to launch on the failure of the current step. The system displays the chained steps as indented steps in the Jobs reports.
Fail Wait	If checked, the system suspends the current project until the fail chain project completes. If this step (or its project) is cancelled, the chained project is also cancelled.

Notes tab: adding notes to steps

The **Notes** tab contains a time-stamped list of notes made about the step. You create notes manually. It does not automatically record edits to the step itself. The tab shows the current number of notes, for example **Notes (2)** .

To add a note:

- Click the **Notes** tab.

Document # SC23-7681-08

- Write the new note in the text field.
- Click **Submit** .

To edit a note:

- Click the **Notes** tab.
- Click the  icon next to the note you want to edit. Make your edits.
- Click **Submit** .

To delete a note:

- Click the **Notes** tab.
- Click the  icon. A prompt asks if you are sure that you want to delete the note.
- Click **OK** .

Controlling execution flow

Within steps there are several features available to control execution flow within a project:

- **Inline**: Use the Inline property of a step to specify a project or library. The steps of the project or library are run inline immediately after the command for this step.
- Pass and fail chains: a step can have its own Pass Chain and Fail Chain distinct from the chains specified for the project.
- **Threading**: You can execute steps marked for threading in parallel. Use the Thread property of a step to mark it for threading.
- **Broadcasting**: You can execute steps marked for broadcasting on multiple servers. Use the Broadcast property of a step.
- **Conditional**: You can set a step to run only if a condition is true. You can set an alternate set of commands and an inline project or library to run if the condition is false. Set the Step Type property to conditional and use the related Condition and Else properties to use this feature.
- **While loop**: You can run a step in a loop each time a condition is evaluated as true. Set the Step Type property to While Loop and use the related properties to use this feature.
- **Dot commands**: The .run and .runwait commands launch a library or project from the command for a step.

A common use of complex execution flow is *job optimization*, that is, executing steps only where needed.

In a software build engineering environment, job optimization can mean building only parts of an application as needed rather than the entire application. A job can check the source status against

Document # SC23-7681-08

the last-compiled binaries and run a compile only if there have been source changes. For complex applications, execution flow can respond to module dependencies as well as source status.

Threading: running steps in parallel

Threading enables steps to run in parallel, either on the same server or on different servers. Threading is controlled by the Thread property setting for a step. By default, the Thread property is set to No. Threading helps reduce project execution time when there are parts of a project that can be run independently of each other.

When the Thread property for multiple adjacent steps is set to Yes, the system attempts to run the step in parallel. Such steps are considered *thread-enabled*, and each step can be run separately while the rest of the job continues. Threading follows these rules.

- At least two steps in sequence must have the Thread property set to **Yes** for threading to occur. A set of threaded steps in sequence is called a *thread block*.
- A thread block is terminated by a step whose Thread property is set to **Join** . At that point step execution becomes sequential again.
- When the system encounters a thread-enabled step, it attempts to start the step. If the following step is also threaded, the system attempts to start that step and continues on to the next step, repeating until there are no more thread-enabled steps or the job limit is reached. If the selector for the project specifies a server pool, the job limit conceptually is the sum of the job limits of servers in the pool.

  ### Note

  The start time of a threaded steps depends on the availability of the server it is supposed to run on. If a step cannot be started, the system waits and tries again. You cannot explicitly control which steps start first.

- Steps may end up running simultaneously on one server (depending on the capacity of that server) or on several servers, depending on how many servers match the selector.
- To force all steps to run on a single server, use the **Sticky** property for the project.
- If there are multiple thread blocks, the first thread block must complete before the next thread block can start.

  In the following example, steps 2, 3, and 4 must complete before steps 5 and 6 can start.

Project	Thread property for step
Step 1	No
Step 2	Yes
Step 3	Yes
Step 4	Join
Step 5	Yes

Document # SC23-7681-08

Project	Thread property for step
Step 6	Yes
Step 7	No

- Use the **Max Threads** property for the project to limit the number of threads that can run at the same time. Each thread-enabled step and its inline project, if any, can result in parallel processes. All processes are counted until the maximum for the parent project is reached. The system stops launching new parallel processes when it reaches the **Max Threads** value. It waits until the number of parallel processes for the project drops below the **Max Threads** value before continuing.

Broadcasting a step to multiple servers

When you have an activity that can be usefully performed on many servers, you can use the broadcast feature to repeat the same step on many servers.

Normally, a step runs on only one server. However, each step has a **Broadcast** check box. When a step's **Broadcast** box is checked, at run time the system replaces the step with a set of non-broadcast steps, one for each server that matches the step's selector.

Note

If the selector for the step matches only one server, then the step runs only once.

Potential uses for broadcasting include:

- Rebooting a group of servers.
- Running a test on a group of servers.
- Copying the same set of files to a whole group of servers.
- Checking out the same set of source code to multiple servers, preparing them for later individual tasks with a single, easy-to-maintain step.

Threading in broadcast steps

When it creates replacement steps for a broadcast step at run time, the system threads steps as follows:

- If the broadcast step's **Thread** property is set to **No** the replacement steps get the same **Thread** value, and thus all execute in series. Each step must complete before the next one can start.
- If the broadcast step's **Thread** property is set to **Yes** the replacement steps also get the same **Thread** values. This results in a set of steps that execute in parallel with each other and with any threaded steps that precede or follow the broadcast step.
- If the broadcast step's **Thread** property is set to **Join** the system creates the replacement steps with **Thread** set to **Yes** except the last step, which is marked **Join** . The result is a set of steps

Document # SC23-7681-08

that execute in parallel with each other, and with any threaded steps that precede them, but the whole set must complete before the step following the broadcast step can start.

Launching other projects from a broadcast step

You can broadcast a step that includes an inline project or that chains a project on the step's passage or failure (**Pass Chain** / **Fail Chain**). When you broadcast a step that launches (chains) another project, be aware that the broadcast step does not override the launched project's selector. In general, use a library project (one that has no selector of its own) when launching a project from a broadcast step, if your intent is to launch the project on every server matching the broadcast step's selector.

If you do not use a library project, each copy of the broadcast step runs on a different server, but the inlined or chained project obeys its own selector, which may not choose the same server as the copy of the broadcast step. You can end up with each broadcast step running on a different server, while all of the steps from an inlined project run on the same server, multiple times.

Note

If your intent is to use **Broadcast** to launch a library once on each server that matches a selector, be sure to also set the **Sticky** option on the library, so that all of its steps (that use the default project server) run on the same server.

Conditional step execution

Conditional execution implements if-then-else branching for a step.

Simple If-Then execution:

- Set the step type to Conditional.
- Set the condition to an expression that can be evaluated.
- Fill in the command to be run.
- If desired, specify an Inline chain to run. (Command can be blank if Inline is set.)

If Condition evaluates to true when the job runs, the step is executed. If specified, the Inline project or library is also executed. If the expression evaluates to false, it is skipped and job execution proceeds to the next step.

If-Then-Else execution:

If you wish to run a different command and/or inline if Condition returns false, fill in additional properties:

- Fill in the Else command to run.
- If desired, specify an Else Inline project or library to run. (Else Command can be blank if an Else Inline is set).

Document # SC23-7681-08

During job execution, the step result is marked Pass if the condition is evaluated successfully and the commands in Command or Else command are executed successfully. To determine which path was taken, you must look at the log.

See also Condition functions.

While loop execution

While loop execution allows you to repeat a step based on a condition.

To implement a step as a while loop:

- Set the step type to While Loop.
- Set Condition to a command or an expression that can be evaluated.
- Fill in the command to be run.
- If desired, specify an Inline chain to run. (You can leave Command blank if Inline is set).
- If desired, set Max Iterations to the maximum number of times you want the step to execute. Use this limit during development until you are satisfied that the condition you specify works correctly. The default is 100.
- If you want the step to fail if Max Iterations is reached, set **Fail step if max reached** to Yes. Otherwise the step passes when Max Iterations is reached.

Each iteration of the step is recorded in the log. Each iteration result is set to Pass or Fail according to the Result criteria.

See also Condition functions.

Condition functions

Condition functions are used in the Condition step property and in the condition attribute in adaptor XML elements.

- For steps using the Condition property: If the following functions are used at the beginning of the Condition field, they are evaluated *by the engine* and no information is sent to the selected server unless the condition evaluates to true. The step is executed on the selected server if the condition evaluates to true.

  ### Important

  Do not attempt to use the functions on variables that are set in the shell environment of the server resource. The evaluation takes place on the Build Forge engine, so they work only on variables that are defined in the Build Forge environment for the step.

- For adaptor templates: the following functions are available to use in adaptor XML elements that have a condition attribute. They are used to specify how the adaptor executes.

Document # SC23-7681-08

The following functions are available:

true(*expression*)	Returns true if *expression* is true.
false(*expression*)	Returns true if the expression is false.
contains(*a*,*b*)	Returns true if string *a* is found in string *b*. The *a* and *b* parameters can be literal strings or variables. Literal strings should not be quoted. If literal strings are quoted, the quotes become part of the string that is evaluated.
hastext(*var*)	Returns true if the variable is not empty.*Var* is a variable set within Build Forge.
isempty(*var*)	Returns true if the variable is empty. *Var* is a variable set within Build Forge.
a eq *b*	Returns true if *a* is equal to *b*. The *a* and *b* parameters can be variables set within Build Forge or literal values. Character and numeric types can be used. Use a space between the parameters and the operator.
a ne*b*	Returns true if *a* is not equal to *b*. The *a* and *b* parameters can be variables set within Build Forge or literal values. Character and numeric types can be used. Character and numeric types can be used. Use a space between the parameters and the operator.
a contains *b*	Returns true if string *b* is found in string *a*. Literal strings should not be quoted. If literal strings are quoted, the quotes become part of the string that is evaluated. Character and numeric types can be used. Use a space between the parameters and the operator.

Expressions in functions

The *expression* parameter of the true() and false() functions can use the following operators:

a==*b*	Tests equality. Parameters can be strings or numbers. Parameters can be literals or variables defined in Build Forge.
a eq *b*	Tests equality. Parameters can be strings or numbers. Parameters can be literals or variables defined in Build Forge. Use a space between the parameters and the operator.
a!=*b*	Tests inequality. Parameters can be strings or numbers. Parameters can be literals or variables defined in Build Forge.
a ne *b*	Tests inequality. Parameters can be strings or numbers. Parameters can be literals or variables defined in Build Forge. Use a space between the parameters and the operator.

Document # SC23-7681-08

a>*b*	Tests that *a* is greater than *b*. *Parameters must be numeric*. Parameters can be literals or variables defined in Build Forge. Literals can use arithmetic operators, for example 2+2.
a<*b*	Tests that *a* is not greater than *b*. *Parameters must be numeric*. Parameters can be literals or variables defined in Build Forge. Literals can use arithmetic operators, for example 2+2.
a contains *b*	Tests that string *b* is found in string *a*. Parameters can be literals or variables defined in Build Forge. Literal strings should not be quoted.

Examples of Condition functions

In the examples in the table below, variables are set as follows:

- `$AVAL` contains the value `String`
- `$BVAL` contains the value `3`.

Condition	Evaluates To	Notes
A String contains $AVAL	TRUE	String comparison
A String contains "String"	FALSE	The quotes around String become part of the comparison.
true(A String contains $AVAL)	TRUE	String comparison
$AVAL contains String	TRUE	String comparison
$AVAL contains "String"	FALSE	The quotes around String become part of the comparison.
contains(A String,$AVAL)	TRUE	String comparison
true(A String contains "$AVAL")	FALSE	The quotes around $AVAL become part of the comparison, "A String" does not have a quotes around the "String" part.
A String != $AVAL	TRUE	String comparison
A String ne $AVAL	TRUE	String comparison
false("Not Here" contains $AVAL)	TRUE	Test string comparison
true(2+1 == $BVAL)	TRUE	Numeric expression for equality
false(2+2 < $BVAL)	TRUE	Numeric expression for inequality
$AVAL eq $AVAL	TRUE	Test string comparison
true($AVAL ne Linus)	TRUE	Test string comparison
true($BVAL > 2+2)	FALSE	(3 > 2+2) is not true
contains(Not Here, $AVAL)	FALSE	Test string comparison

Document # SC23-7681-08

Launching projects from steps

Inlining and dot commands allow you to run a project or library from inside a step.

To launch a project from a step, use one of the following features:

- Inline: Specify the project or library you want to launch as the Inline property of a step. The steps of the project or library are executed in the context of the calling project immediately after the calling step. The inlined steps use the environment of the calling step.
- Dot commands: Use the .run or .runwait command to launch a project. A project launched this way acts like a chain. The project runs using its own selector and environment.

Customizing log output

These topics show some ways to customize log output using command features in a step.

Labelling log output for a step

Create a label to have step output listed in its own category in the step log.

This task assumes that you have already created a selector, server, and project. It assumes that you are using a server with an operating system that accepts the echo command (for example, Windows, Linux, or UNIX).

You can include an uppercase label at the beginning of any line of output. When a line begins with a label, Build Forge repeats the output in the category with the same name as the label, or creates a new category if the category does not already exist. The system recognizes a label when it follows these rules:

- A line in the Command property consists of an `echo` command that uses label syntax.
- Label syntax: a label is identified when the first argument of the echo command consists of uppercase letters and ends with a colon.
- Labels cannot contain spaces, punctuation, or lowercase letters. `"SPACESHIPS:"` is valid. `"Space Ships:"` is not.
- Label length: a label has a minimum length of 3 characters.

When the command processor encounters a line that qualifies as a label, it uses that label at the beginning of all output lines. The label is used until the step ends or a new label is encountered. As the example shows, you can set the label to an existing output label name as well as new names.

1. Create a new step in a project.

   The project in this example is named `Say_hi`.

2. Name the step `LabeledLogOutput`.

Document # SC23-7681-08

3. Enter the following text in the Command field:

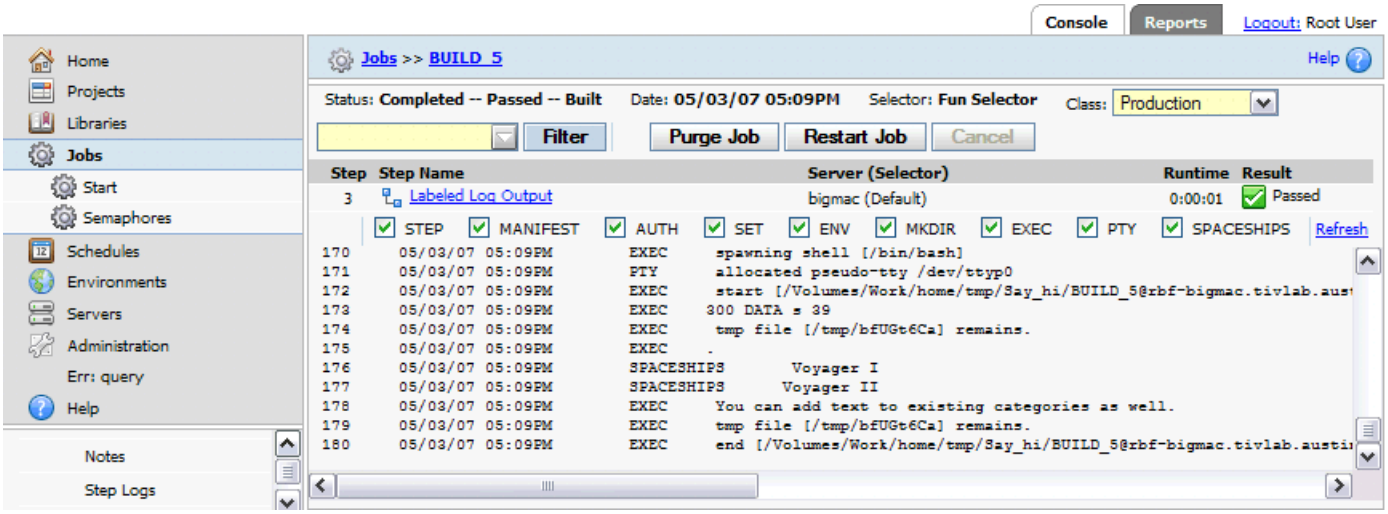
```
echo SPACESHIPS: Voyager I
echo Voyager II
echo EXEC: (You can add text to existing categories as well)
```

4. Run the project.

5. When the job completes, view the log.

Note the SPACESHIPS checkbox in the category header and output lines 176 and 177, which are labeled SPACESHIPS.

Labeled log output with SPACESHIPS category



Note

You can select or deselect the SPACESHIPS check box to show or hide the category.

Highlighting step output as a color or active link

The log viewer recognizes the [STATUS] and [URL] commands in output text. The commands are not case-sensitive. You can use them in an `echo` command or any command that produces output. The start and end tags must appear on the same line of the output.

- [STATUS=*condition*] and [/STATUS] tags mark text to highlight. The *condition* sets the highlight color as follows:
  - PASS - green
  - WARN - yellow
  - FAIL - red
  - RUN - blue

Document # SC23-7681-08

- [URL] and [/URL] tags mark text as an active hyperlink.

Example using [status]:

```
echo [STATUS=WARN]Access to source control timed out[/STATUS]
```

Example using [url]:

```
echo See the support forums at [url]http://www.ibm.com[/url]
```

The log displays the text as an active hyperlink to the indicated URL. Clicking on it opens a new browser window or tab and shows the page.

Working with job data

These topics shows some ways to use command features to modify projects and jobs.

Embedding build numbers in project files

You can use the .strsub command to swap one string for another in files; a common use is to replace a standard token with a system variable such as the $B variable that provides the current job number.

You can use the .strsub dot command to embed build or version numbers in code files. By placing a .strsub command early in your project, a later step can compile files that contain the updated information.

For example, the following steps set up a project to embed build numbers:

1. Add a unique string such as `_BUILD_` to a file in your project. For example, modify a file `README.TXT` and change the version declaration as follows:

   ```
   Application version 5.0.123
   Application version 5.0._BUILD_
   ```

2. An early step in your project should check out the files to be worked on. Add a step after `README.TXT` is checked out that replaces `_BUILD_` with the $B system variable. For the command, use the following:

   ```
   .strsub _BUILD_ $B README.TXT
   ```

3. Run the project and verify that the README.TXT file contains the current job number. For the third run of the project, the README.TXT file should contain this line:

   ```
   Application version 5.0.3
   ```

Enhancements

You can improve this practice in the following ways:

Document # SC23-7681-08

- Use additional environment variables. For example, create variables named $MAJORVERSION and $MINORVERSION and use them as follows:

```
.strsub _MAJORVERSION_ $MAJORVERSION README.TXT
.srsub _MINORVERSION_ $MINORVERSION README.TXT
```

- Update your environment variables when you start a project. By selecting **Jobs → Start** to start your projects, you can see the current environment variables and edit their values before you launch the project. You might include a comment in your jobs, for example, as a variable. Use the Project Action **Must Change** on the comment variable to force users to enter a new value when they run the project.

Changing the build tag during a job

You can set the value of a tag to a completely new value during the job by using the .retag command, which has the following syntax:

```
.retag <new tag value>
```

Here is an example of simple usage:

```
.retag MyProject
```

More complex usage is possible:

```
.retag Job_${B}_${BF_D}
```

This example sets the tag to use the run increment and current date system variables. You can use a command to the server's command interpreter to set the result. To use a command within the dot command, enclose the command in backtick or backquote (` `) characters:

```
.retag `hostname`
```

This example sets the tag to the result of running a hostname command on the server running the step.

Note

Do not mix the backtick form and the standard assignment form of the command.

Changing environment variable values during a job

You can use the .set, .bset, and .tset commands to change an environment variable from within a step. These commands change the values of existing environment variables as follows:

- Use the .set command to change the *master record* for an environment. When the system runs a project, it makes a copy of the project environment from the master record, and uses that copy as the project default. This has the following effects:
  - If a .set command modifies the project environment, later steps that use the default environment do *not* see the changes, because the system does not refer back to the master record.

Document # SC23-7681-08

- If you use a .set command to modify an environment and a later step explicitly uses the same environment, that step will see the changes you made. The system goes back to the master record for the environment when the step has a specific environment selected. This works even if the named group is the same as the project default group, so long as the step's environment setting is not “Default.”
- Changes made by a .set command persist after a job is over. Future jobs use the values created by previously-run .set commands.

Use the following basic syntax:

```
.set env <GroupName> "<VariableName>=<DesiredValue>"
```

- Use the .bset command to add or change variable values during job execution. Changes take effect in the step after the step .bset appears in. They are in effect for the remainder of the job.

  ```
  .bset env "<VariableName>=<DesiredValue>"
  ```

  ### Note

  Unlike the .set command, the variable you specify for a .bset command does not have to exist when you set it, so you can use the .bset command to create a new variable during a job. The value of the variable does not persist past the current job.

- Use the .tset command to add or change variable values during job execution. Changes take effect in the current step. They are in effect for any other commands in the step and for any Inline specified for the step.

  ```
  .tset env "<VariableName>=<DesiredValue>"
  ```

  ### Note

  Unlike the .set command, the variable you specify for a .tset command does not have to exist when you set it, so you can use the .bset command to create a new variable during a job. The value of the variable does not persist past the current step.

Setting multiple variables

You can set more than one variable at once with these commands by including additional variable and value pairs, separated by spaces, as in the following examples:

```
.set env MyGroup "X=5" "X2=45"
```

```
.bset env "Y=7" "CompilerVersion=4.511"
```

Using command output to set values

You can generate the value of a variable for a .set or .bset command by sending a command to the server's command interpreter. To use a command within the dot command, enclose the command in backtick characters. For example, the command:

```
.set env SetupGroup “PerlVer=`perl --version`”
```


Document # SC23-7681-08

sets the variable PerlVer to the output of the perl --version command.

The variables can only store 256 characters; if more are assigned to a variable, the value is truncated.

By default, the system assigns the entire output of a command in backticks to the variable, but you can use range commands in brackets to select which lines from the command output you want to assign to your variable. The range numbers specify lines from the output using a 0-index (the first line is numbered zero, the second 1, etc.). In the following example,

```
.set env SetupGroup “WindowsIPinfo[0,5-8]=`ipconfig`”
```

the variable WindowsIPinfo receives the first and sixth through ninth lines of the ipconfig command's output.

The following are all valid range modifiers, selecting single lines, groups of lines or combinations:

```
[5]
```

```
[4-6]
```

```
[1,2,5,8-11]
```

The system combines lines without any separation; no spaces or carriage returns are added.

Note

Do not mix the backtick form and the standard assignment form of the command.

Using Registers

Registers are general-purpose buffers that steps can use for storing persistent data. Ordinary registers can have single-letter names, or multi-character names that begin with letters.

All register names are stored in uppercase. Although you can create a register named Alpha, it is stored as ALPHA and must be referenced in project steps or in the .pop command as ALPHA.

You can include register variables in notification templates; use the ${X} braced form when referencing registers in notification templates. Referencing an empty register returns an empty string.

Use the .push and .pop dot commands to store information in and retrieve it from registers. See also the .poptag command (“.poptag” on page 134), which makes the current job tag equal the contents of a register.

Note

You cannot use registers in commands like variables. You must first pop the value of a register to a file before you can use it.

Document # SC23-7681-08

Special registers

Register	Contains
!	Contains the command output lines that matched Fail filter patterns.
@	Contains the command output lines that matched Pass filter patterns.
=	Specifies the notes database for a job. Allows steps to add data from a file as a note to a job. This register is different from the others: • You can only write (push) to this register; you cannot read from it. • Data pushed to this register is always appended to it, rather than overwriting previous data. • The system supplies a time stamp and user ID with the appended data. This preserves an audit trail of job notes.

Project registers

Project registers are distinct from ordinary registers. They persist across builds and you can create and view them through the Management Console interface, making them an ideal way to store some kinds of configuration information.

For example, you could store an IBM® Rational® ClearCase® config spec as a project register, then have a step use a `.pop -p` command to extract the specification and use it with a cleartool `setcs` command, configuring your build. This allows you to manage the configuration along with the project.

If you have a project register named ALPHA, you could also have an ordinary register named ALPHA, with entirely different contents. Project registers are a separate set of values.

You can create and access project registers in two ways:

- Through dot commands (`.push` and `.pop`), by adding a -p option. When you use the -p option, your command refers to a project register, rather than an ordinary register.

  For example, a command

  ```
  .push -p ALPHA register.txt
  ```

  places the contents of the file register.txt in the project register named ALPHA.

- Through the Management Console interface. Select **Projects,** then click the  icon next to the desired project's name. The project properties appear in the lower pane; click the **Registers** tab to display the project's registers. The tab provides a form for managing registers:

  - To create a new register, enter a name and contents, then click **Create** .
  - To delete a register, click the trash can icon next to the register's name in the list at the right of the form.

Document # SC23-7681-08

- To edit a register, click the  icon next to the register's name in the list. The system populates the register form with the register's contents. Make your changes, then click the **Save Edited Register** button.

Anyone who has access to a project can view and edit its project registers.

Note

Use uppercase to create register names and to reference all register types. Even though you can create registers using lowercase letters, registers are stored and must be referenced using uppercase letters.

Copying files to and from server resources in a step

You can use dot commands to copy files from one server resource to another. This topic describes how to use the .get and .put commands (for single files) and the .rget and .rput commans (to copy whole directory trees).

Note

These two commands return an error if the destination file already exists. You cannot use them to replace files.

Important

The server resources must have file copying enabled. It is not enabled by default.
See“Enabling file copying on a server resource” on page 118

Enabling file copying on a server resource

The default settings for servers do not allow files to be copied by using dot commands. To allow projects to copy files to and from server resources, change the **Files** property for the server.

To change the setting, do the following:

1. Select **Servers** → **<ServerName>** .
2. Select a value other than None for the **Files** property. You can enable copying files from a server, to a server, or both.

Getting a file from a server

To get a copy of a file from a server and place it in a destination relative to the current step's working directory, use the .get command. For example, if you have a server named winbuildserver1 with a file config.txt in its config directory, you can add the following step to your project to copy the file to the current server's config directory:

```
.get winbuildserver1:./config/config.txt ./config/config.txt
```

For more information see the following:

Document # SC23-7681-08

- Reference entry for ".get" on page 128
- Reference entry for ".rget" on page 136
- Description of how paths in steps are resolved, in "Working directories for jobs" on page 161.

Putting a file on a server

To copy a file from your current server to a different server, use the .put command. The following example step assumes that you have a config.txt file in a config directory on the current server, accessible from the current path:

```
.put ./config/config.txt winbuildserver1:./config/config.txt
```

For more information see the following:

- Reference entry for ".put" on page 135
- Reference entry for ".rput" on page 137
- Description of how paths in steps are resolved, in "Working directories for jobs" on page 161.

Troubleshooting step processing

If you encounter problems with step processing, review the information in this topic to see if there is an acceptable workaround or solution.

Job does not process any step commands after an ANT build command

Problem description: The commands in a step after an ANT build command are not processed.
In the following step example, the echo command does not run.

```
<path to ant bin directory> ant -f <path to Java
project>\build.xml build
echo "Ant build complete"
```

Explanation: ANT builds return an error code of 1 whether the ANT build fails or succeeds. In the Command property of a step, if multiple commands are used, only the exit status of the last command executed affects the step result status. When the server executes a command script for a step that contains an ANT build command, the error status of 1 causes any commands following the ANT build to fail.

Solution: Create a step log filter to process the step output produced by the ANT build. The step log filter sets the step result and ensures that the next step in the job is processed.

Document # SC23-7681-08

1. The ANT build should be the only command in the step or the last command in the step.

   Without a log filter, the ANT builds return an error code of 1 and the step result is set to fail.

2. Create a log filter to search step output for the appropriate failure text string (BUILD FAILED) and effectively control step processing.

   If the text string is found, use the Set Fail action to set the step result to fail. When you use a step log filter, if the text string is not found, the step result is always set to pass.

   For details about setting log filters for steps, see “Log filters” on page 76.

Commands in a step after a Windows batch command are not run

Problem Description: The step commands after a Windows batch command are not processed.
In the following step example, the two commands after the first batch file are not run.

```
c:\script1.bat
c:\script2.bat
echo "Performed both batch commands"
```

Explanation: All commands in a step are placed in a Windows batch file to be run by the server. If the step command contains a reference to a batch file, the step exits the batch file executes. Step commands after the batch file reference are not run.

Solution: Use the call command to run batch files within a step. The call commands are executed within the step batch file.

```
call c:\script1.bat
call c:\script2.bat
echo "Performed both batch commands"
```

Dot command reference

You can use dot commands in the Command field of a step. They provide access to special capabilities and functions within the system.

You can mix dot commands with ordinary commands in a step and you can have multiple dot commands in a single step. Do not use more than one .scan command in a single step, however, as the system cannot accurately report the command's results if you do.

Syntax specifications: each dot command description includes a syntax specification, using the following notation:

Document # SC23-7681-08

- User-supplied values are shown in angle brackets: <value>
- Optional text is shown in brackets: [optional text]

You can use environment variables for command parameters except where specifically noted.

There is a separate list of dot commands for use with environement variables. See “Using dot commands in variables” on page 53.

.bom

```
.bom addcategory "<Category Name>"
.bom setcolumn "<Category Name>" "<SectId>" "<Column>" "<Column>" "<Column>"
.bom data <Category Name>" "<SectId>" "<Column=Value>" "<Column=Value>" "<Column=Value>"
```

The .bom command adds data to the Bill of Materials (BOM) for a build. With it, you can add categories, sections, and data.

Categories	A category is a header printed in the BOM.
Sections	A section defines columns of data within a category. Section names are not printed. The set of column headers for the section is printed at the beginning of a section. You can nest sections.
Data	Data populates the columns defined in a section.

The command has the following options:

- Categories: use the `addcategory` option to create a new category in the BOM. The following example creates a category named Spaceships.:

  ```
  .bom addcategory "Spaceships"
  ```

- Sections: use the `setcolumn` option to specify sections and columns of data in a category. You specify a section ID and then column headers. The following example creates a section named Section1 with three columns:

  ```
  .bom setcolumn "Spaceships" "Section1" "ShipName" "WarpSpeed" "Tonnage"
  ```

  To nest a section within another, create a section and use -p to indicate its parent section. The following example creates a nested section named Subsection1 within the section named Section1 under category Spaceships:

  ```
  .bom setcolumn "Spaceships" "Subsection1" -p "Section1" "ShippingDate" "ShippingManifest"
  ```

- Use the `data` option to write a row of data. For example:

  ```
  .bom data "Spaceships" "Section1" "ShipName=SpaceShipOne" "WarpSpeed=9" "Tonnage=10000"
  ```

  specifies that SpaceShipOne's warp speed is 9 and its tonnage is 10,000.

As with other dot commands, you can use environment variables in the command. A command like

```
.bom data "Spaceships" "${SECTION}" "ShipName=${NAME}" "WarpSpeed=${SPEED}"
"Tonnage=${TONNAGE}"
```

populates the BOM with data loaded into environment variables by earlier commands.

You can create any number of columns, but the system does not write a line to the BOM until the last column is populated. Also, if you omit a column from a data line, the system uses the value from the previous row. Using the Spaceships example, if you added a line

```
.bom data "Spaceships" "Section1" "ShipName=Freighter" "Tonnage=20000"
```

the system would repeat the WarpSpeed value, giving rows like these:

```
ShipName      WarpSpeed   Tonnage
SpaceShipOne      9        10000
Freighter         9       20000
```

.bomexport

Description

The .bomexport dot command exports the BOM for the job to an XML file. After collecting the BOM information, .bomexport saves it to the file and location you specify.

The path and file name are optional. By default, Build Forge saves the BOM report to the step's working directory on the server and uses the tag name as the file name (<*build_tagname*>.xml).

Specify the .bomexport command as the last step in the project.

Syntax

```
.bomexport [path_name][file_name]
```

Options

Option	Description
path_name	An optional path name. If provided, the path must be relative to the step's working directory on the Build Forge server. If omitted, the file is saved to the step's working directory.
file_name	An optional file name. The BOM for the job is saved to the file in XML format. If a file name is not provided, a file name is constructed from the build tag name and the string _BOM: <*build_tagname*>_BOM.xml.

Examples

```
.bomexport
```

```
.bomexport myproj.xml
```

```
.bomexport path/to/myproj.xml
```


Document # SC23-7681-08

```
.bomexport /path/to/myproj.xml
```

.break

```
.break [<notification_group_name>]
```

Use the .break command to make a job halt until you restart it. When the system encounters a step with a .break command, the run completes with a result of Stop. Use the **Restart** icon to restart the job, continuing with the step after the .break step.

If the .break command occurs within a chained job, the system stops the chained job, but returns control to the calling job, which continues processing steps.

You can include an access group as an optional argument to the command; if you do, the system sends an e-mail message to the specified access group when it stops the job.

.bset

```
.bset env "<VarName>=<Value>" ["<VarName>=<Value>"]

.bset selector <SelectorName>
.bset server <ServerName>
.bset buildserver <ServerName>
```

The .bset command changes project settings temporarily during a job. The command has several options:

- **env** changes the value of one or more project environment variables for a running job. For more information on using the command this way, see “Changing the build tag during a job” on page 114. You can set a variable that does not yet exist. Values set by the .bset command are written to the job record. If you set a new value for a variable that is also defined in a project or step environment, the new value is in effect only during the job. The environment for the project or step is not changed.

- **selector** changes the project selector during a job. The new selector takes effect in the next step. The new selector is used only for steps that do not have an explicit selector setting (the selector for the step is set to Project Default). The project's **Sticky** option must be set or the **server** option has no effect. The new server setting takes effect in the next step. Steps that have an explicit selector set are not affected.

- **server** changes the default project server during a job. Steps use the specified server only; they do not have an explicit selector setting.

  Backtick syntax: you can use backticks to set the server name to the output of a command. For example, the command

  ```
  .bset server `SelectAServer.sh`
  ```

  runs the SelectAServer.sh script and provides its output as the server name for the .bset server command.

Document # SC23-7681-08

- **buildserver** changes the default project server during a job. The new server setting takes effect in the next step. It differs from the **server** option because it applies to inlined steps as well as project-level steps.

The .bset changes do not take effect until the current step completes. For example, if your step includes multiple commands as in the following example, the second line operates in an environment unmodified by the .bset command:

```
.bset env "CompilerVersion=1.1"
compile driverset
```

To make the compile command use the changed environment variable, move the compile command to the following step.

Note

Avoid using multiple .bset commands in threaded steps.

.buildstatus

```
.buildstatus <state>
```

This command forces the build to be in a specified <state> after the build has finished, regardless of the results of the builds steps.

.date

```
.date <conversion_specifier>
```

Use the .date command with one or more conversion specifier characters as arguments to generate current date-time information when a project runs.

The .date command and its arguments must be defined as an environment variable in an environment. You can then assign the environment to a project or a step.

For example, in an environment, define an environment variable DayofWeek and assign it a value of .date %A. Assign the environment to a project or step. If the project is run on a Wednesday, the job assigns the text Wednesday to the environment variable DayofWeek.

Important

The .date command cannot be referenced directly in the step Command field.

The .date command is built on the POSIX strftime function and accepts conversion specifiers identified by the ANSI C89 standard. Date-time values for conversion specifiers are provided in the following table.

Note

Not all conversion specifiers are portable across locales and operating systems. Test the .date command results on the server operating systems and locales in which you plan to use it.

Document # SC23-7681-08

Date Conversion Specifier	Description
%a	The abbreviated weekday name according to the current locale.
%A	The full weekday name according to the current locale.
%b	The abbreviated month name according to the current locale.
%B	The full month name according to the current locale.
%c	The preferred date and time representation for the current locale.
%d	The day of the month as a decimal number (range 01 to 31).
%H	The hour as a decimal number using a 24-hour clock (range 00 to 23).
%I	The hour as a decimal number using a 12-hour clock (range 01 to 12).
%j	The day of the year as a decimal number (range 001 to 366).
%m	The month as a decimal number (range 01 to 12).
%M	The minute as a decimal number (range 00 to 59).
%p	Either “AM” or “PM” according to the given time value, or the corresponding strings for the current locale. Noon is treated as “pm” and midnight as “am”.
%S	The second as a decimal number (range 00 to 61).
%U	The week number of the current year as a decimal number, range 00 to 53, starting with the first Sunday as the first day of week 01.
%w	The day of the week as a decimal, range 0 to 6, Sunday being 0.
%W	The week number of the current year as a decimal number, range 00 to 53, starting with the first Monday as the first day of week 01.
%y	The year as a decimal number without a century (range 00 to 99).
%Y	The year as a decimal number including the century.
%Z	The time zone or name or abbreviation.
%%	A literal “%” character.

.defect

Description Use the .defect command to add an adaptor for a defect tracking application to a project step. A defect adaptor is a Build Forge object; it is based on the adaptor template for a defect tracking application. The adaptor code for the step is executed when the project runs.

Syntax `.defect <adaptor_name> [entry_name]`

The *<adaptor_name>* is required; it is the name assigned to the adaptor in the Management Console. The *<adaptor_name>* case should match the case used in the console.
If your adaptor template has multiple interface functions, specify the one that you want to execute by using the *entry_name* option. The *entry_name* must match the name attribute specified for interface element in your adaptor template. In the following example, the entry name is DefectFunction.

Document # SC23-7681-08

If you are using an adaptor link, the adaptor is called automatically and the first interface function in the adaptor template is executed. To execute a different interface, in the adaptor template, set the default attribute to true (default="true") on the interface that you do want to execute.

Examples

```
.defect MyClearCaseQuestAdaptor

.defect MyClearCaseQuestAdaptor DefectFunction
```

Notes To create an adaptor or view a list of adaptors, select **Projects → Adaptors** . The adaptor templates provided with the Build Forge product are located in:

```
<bf-install>/interface
```

.drill

```
.drill [through] <"var1,var2,var3"|$EnvVar|-r[p] Register> [gr[ouped by] "{}"] [sep[arated
 by] ","] [exec] "Command $1 $2"
```

The .drill command allows you to loop over a command, executing the command once for each member of a series of values. You can specify the values on the command line, or draw them from an environment variable or register. When it executes a .drill command, the system uses the .drill syntax to construct a series of command lines and sends them to the agent for execution.

For example, the command `.drill "A,B,C,D" "echo value $1"` creates the following commands:

```
echo value A
echo value B
echo value C
echo value D
```

Grouping

You can group the values and reference multiple values in each group using the $1, $2, etc. syntax. $1 refers to the first value in the group, $2 to the second value in the group. For example, `.drill through "(A,B,C,D,E),(B,C,D,E,F),(C,D,E,F,G)" grouped by "()" separated by "," exec "echo 1[$1] 2[$2] 3[$3] 4[$4] 5[$5]"` creates these commands:

```
echo 1[A] 2[B] 3[C] 4[D] 5[E]
echo 1[B] 2[C] 3[D] 4[E] 5[F]
echo 1[C] 2[D] 3[E] 4[F] 5[G]
```

Note

There is no default grouping character. There is a default separator character, the comma. If you do not specify `grouped by`, the system looks through the supplied values as separated by the separator character and considers each such string a single value. For example, the command `.drill "(A,B),(C,D)" "echo $1 $2"` resolves to the following commands:

```
echo (A 2
echo B) 2
echo (C 2
echo D) 2
```


Document # SC23-7681-08

Data sources

You have several options for where the .drill command gets the data that it loops through. The first parameter for the command is the data source. You can include the optional command word "through" to indicate the data source.

- You can explicitly list the data in the command line, as in the following command, which loops over the values one, two, and three:

  ```
  .drill through "one,two,three" exec "echo $1"
  ```

- You can draw the data from an environment variable. The following command assumes that the environment variable FILENAMES is a comma-separated list of files, and uses a DOS command to delete all the files in the list:

  ```
  .drill through $FILENAMES exec "del $1"
  ```

- You can draw the data from a register or a project register. If RegisterA contains a comma-separated list of filenames, then the following command issued to a Linux system writes out the content of each file:

  ```
  .drill -r RegisterA exec "cat $1"
  ```

  while the following example does the same but uses a project register:

  ```
  .drill -rp ProjectRegisterA exec "cat $1"
  ```

.edit

`.edit /<search_expression>/<replace_expression/ [<relative_path>/]file [file ...]`

Use the .edit command to search and replace text strings in one or more files. The .edit command replaces the first instance of the string (search_expression) on each and every line in each file specified. Files are assumed to reside in the step's working directory unless you specify a relative path.

The .edit command implements standard POSIX regular expressions for matching and replacement, including the use of () substring selection and \N substitution in the replacement pattern.

The .edit command replaces the first instance of the string on each and every line in each of the files.

The .edit command uses POSIX Extended Regular Expression syntax by default. **if the agent has been compiled with Perl Compatible Regular Expression support**, then the substitution expression may be followed with a "p" character specify PCRE syntax. .

In both cases, the expression is interpreted twice by the agent processing, so four backslashes should be used anywhere that a single backslash would normally be used. For example, use four backslashes (\\\\) instead of a single backslash (\) to match a literal period character.

Document # SC23-7681-08

Note

You must explicitly list one or more file names, without wildcards.

Example: the following command replaces strings such as winXPdriver and win2000driver in the file called drivermakefile.

```
.edit /win.*driver/linuxdriver/drivermakefile
```

The .edit command is similar to the .strsub command; differences include:

- The .strsub command is faster than .edit when performing replacements in large text files or many files.
- The .edit command can perform regular expression searches and replaces.
- The .edit command replaces the first instance only of the string (search_expression) on each and every line in each file.
- The .strsub command replaces every instance of the string (source) on each and every line in each file.

.email

```
.email <to>
```

This command sends an email to <to>, using the step_email mail template. Ideally, the specified addressee is a Build Forge user, but .email will also send to any valid email address.

.export

```
.export [path_name][file_name]
```

The .export command saves the project definition for the calling project to an XML file located in the working directory of the step. The XML file describes the project, current build, and steps. It does not describe other associated objects, such as the server.

The exported XML file can be used to import the project definition to the Management console.

The .export command can take an optional path and/or file name. The path must be a relative path. It is applied from the step's working directory.

If no file name is provided, the file name is constructed from the current project tag: $BF_TAG.xml.

.get

```
.get server:[[<relative_path>/]file/]file [[[<relative_path>/]file/]file]
```

Use the .get command to transfer a file from one logical server to another. The .get operation executes from the current server/path and retrieves the file from the specified server/path. The destination path name is relative to the current step working directory. The source path name is relative to the specified server base path. Server must specify a logical server that allows the .get

Document # SC23-7681-08

operation for files (see “Enabling file copying on a server resource” on page 118). Only single files may be transferred.

The path specification can include environment variables. This capability allows you to specify files relative to the path used by a specific job. See the description of paths used by jobs in “Working directories for jobs” on page 161.

If the server name you are using has spaces in it, put the name in quotes.

The transfer is not fast, so you may want to choose a different method to transfer large files. Expect speeds of no more than 40 KB per second; a 70 MB file could take 45 minutes to an hour to transfer.

Note

If the destination file already exists, it is overwritten without warning.

.load

```
.load [-o] [-e] [-v] [-j] [<relative_path>/]<filename>
.load -r|-p <registername>
.load -s `<command name>`
```

The .load command loads a project from an XML file and adds the steps of the loaded project to the current project, *after* the step that executed the .load command, allowing a project to dynamically create and load steps at run time. Using options, you can cause the .load command to draw its data from a register or from the output of a command.

To write an XML file for a .load command, start with an export file from an existing project to give you the appropriate basic structure. You can also create a project within the system, then export it to use it in a .load command. This topic includes sample XML code.

The steps loaded by a .load command can contain references to inlined or chained projects. By default, the system looks for the definitions of inlined projects within the XML file, and loads their steps; see the -e option later in this topic for a way to have the system get the inlined project definition from the database. For pass chained or fail chained projects, the system always looks for the project definition in the database.

Note

For JPO steps executed from a .load, the else-inline identified project or library must be a project or library already existing within the system, otherwise the inline will not be executed.

Multiple projects in XML files

Because the system exports inlined projects along with their calling projects, an XML file may contain several projects. The .load command executes the project that is labelled primary in the file. This project has attribute `primary="1"` on its `<project>` element.

Document # SC23-7681-08

Command options and parameters

The simplest command form is `.load <filename>`. You can include an optional path name (relative to the job directory) in front of the filename. For example, the command

```
.load ../../project.xml
```

loads the file project.xml from the server directory (the directory that contains the project and job directories), assuming that the step's path property is "/" (the default).

Note

When a normal step launches an inlined project, the system goes to the database to the get the current definition for that project; when a step that is imported by .load command launches an inlined project, the system looks inside the XML file for the definition of the inlined project. See the description of the -e option below for a way to avoid this situation.

The command has the following options:

-r or -p — These options cause the system to load steps from a register. Use the command line with these options.

```
.load -r|-p <registername>
```

The -r option loads steps from an ordinary register, while the -p option loads steps from a project register. You can build up data in a register in earlier steps in your project, then load the steps from the register with this command.

-s — This option causes the system to run a command and use the output of that command as the data to load. Use the command line

```
.load -s `<command name>`
```

-e — When the -e option is set, the system gets inlined projects from the database instead of from the loaded XML file. It treats the value of `chainID` as a reference to a project ID within the database. This enables your XML file to reference the latest version of an inlined project, instead of the one in the XML file, or to reference a project that is not included in the XML file.

-o — Use the -o option to disable inlined projects within the XML file. When this option is used, the system ignores any inlined projects within the main project. A step that contains a reference to an inlined project executes its command but then ignores its inline.

-j — Use the -j option if the last set of steps in the XML file are threaded and the steps following the .load command are also threaded. The -j option turns the last threaded step into a join step. Otherwise, the threaded steps become part of the threaded block of steps following the .load command.

-v — Sends the contents of the XML file that is loaded to the display terminal (stdout) for viewing.

Document # SC23-7681-08

Example 1. Sample XML

The following example shows an XML file to use with the .load command. The XML was created by exporting a project named HelloWorldPlusInline.

Note the following details of the example XML:

- The XML contains two `<project>` elements.
- The first project in the XML is the primary project; it has the attributes `name="HelloWorldPlusInline"` and `primary="1"` .
- The second project in the XML is called Sleepytime and has the attribute `primary="0"` to indicate that it is not primary.
- The first step of the HelloWorldPlusInline is a step named EchoHelloWorld which contains an echo command and a `chainID` attribute. The `chainID` attribute has a value of 2, indicating that the system should inline the project with the ID 2, which is the Sleepytime project.

  ### Note

  Ignore the step attribute `inline`; it is a deprecated attribute that is no longer used. All steps have this attribute with a value of N. To determine if a step has an inlined project, look for the attribute `chainID` . The value of `chainID` refers to the ID of a project. By default, the system looks for the inlined project within the XML file, but if you use the -e option in your .load command, the system treats the value as project ID within the database. This allows you to create your own .load files without having to include inlined projects within them.

- Each project has an `id` attribute. This ID value is the same as the project's ID in the database. You can get a list of project IDs by executing the following command from your installation directory.

  ```
  bfexport -l
  ```

```
<?xml version="1.0" encoding="UTF-8"?>

<buildforge schema="7.000301" comment="">
  <project access="6" active="Y" name="HelloWorldPlusInline" primary="1"
selectorId="Choose_local" maxthread="0" increment="Y" tagsync="0" buildclass="Production"
 sticky="N" envId="0" tag="BUILD_$B" id="19" exclusive="0">
    <tagvar autoincrement="Y" name="B" id="1">2</tagvar>
    <step absolute="N" failwait="N" selectorId="" dir="/" broadcast="N" timeout="300"
id="1" passwait="N" inline="N" threadable="N" chainId="2" access="6" active="Y"
passnotify="0" description="EchoHelloWorld" onfail=" " failnotify="0" envId="0">
      <command>echo Hello World</command>
    </step>
    <step absolute="N" failwait="N" selectorId="" dir="/" broadcast="N" timeout="300"
id="2" passwait="N" inline="N" threadable="N" access="6" active="Y" passnotify="0"
description="export proj to build and server folders" onfail=" " failnotify="0" envId="0">

      <command>.export $BF_PROJECTNAME_PHYS.xml
copy /Y $BF_PROJECTNAME_PHYS.xml ..\..</command>
    </step>
```


Document # SC23-7681-08

```
  </project>
  <project access="6" active="Y" name="Sleepytime" primary="0" selectorId="Choose_local"
maxthread="0" increment="Y" tagsync="0" buildclass="Production" sticky="N" envId="0"
tag="SLEEP_$B" id="2" exclusive="0">
    <tagvar autoincrement="Y" name="B" id="1">21</tagvar>
    <step absolute="N" failwait="N" selectorId="" dir="/" broadcast="N" timeout="300"
id="1" passwait="N" inline="N" threadable="N" access="6" active="Y" passnotify="0"
description="Sleep, perchance to dream" onfail=" " failnotify="0" envId="0">
      <command>.sleep 0</command>
    </step>
  </project>
  <class maxdays="0" access="1" entranceprojectId="1" name="Production" keepfiles="B"
deletechangedata="N" purgeprojectId="2" exitProjectId="5" candidates="AnyBuild "
maxbuilds="0"></class>
  <selector operator="" required="" access="6" value="" name="Choose_local" selectorId=""
 property=""></selector>
</buildforge>
```

.lock

```
.lock
```

The .lock command causes the system to lock a job after it completes. This prevents the job from being automatically deleted based on the properties of its class; also, a locked run is not listed on the **Jobs → Completed** tab, appearing instead on the **Locked** tab. The command takes no parameters; it locks the job that it is used in.

.mkdir

```
.mkdir <relative_path>
```

The .mkdir command creates a directory. The `<relative_path>` parameter is interpreted as a relative path from the current step directory. If directories in the path name specification do not exist, they are created. Absolute paths and paths including a drive letter (such as `C:\`) are not allowed.

.monitor

```
.monitor [-c] [-w] <interval> [<relative_path>/]<filename>
```

The .monitor command causes the system to halt the project while it watches a file to see when the file size stops changing. When a step issues this command, the system checks the indicated file; it then rechecks the file every <interval> seconds. When the file size fails to change between two intervals, the system continues to the next step.

If you use the -c option, the system writes the contents of the monitored file out to the step log after it determines that it has stopped changing; then it continues on to the next step.

If the file does not exist, then the system does not wait but continues immediately after the first interval. Use the -w option to force the system to wait for the file to be created before starting the monitoring process.

Document # SC23-7681-08

.pack

Description Use the .pack command to add an adaptor for a packaging application to a project step. A packaging adaptor is a Build Forge object; it is based on the adaptor template for a packaging application. The adaptor code for the step is executed when the project runs.

Syntax `.pack <adaptor_name> [entry_name]`

The *<adaptor_name>* is required; it is the name assigned to the adaptor in the Management Console. The *<adaptor_name>* case should match the case used in the console.
If your adaptor template has multiple interface functions, specify the one that you want to execute by using the *entry_name* option. The *entry_name* must match the name attribute specified for interface element in your adaptor template. In the following example, the entry name is PackageFunction.
If you are using an adaptor link, the adaptor is called automatically and the first interface function in the adaptor template is executed. To execute a different interface, in the adaptor template, set the default attribute to true (default="true") on the interface that you do want to execute.

Examples
```
.pack MyPackagingAdaptor

.pack MyPackagingAdaptor PackageFunction
```

Notes To create an adaptor or view a list of adaptors, select **Projects → Adaptors** .
The adaptor templates provided with the Build Forge product are located in:

`<bf-install>/interface`

.pop

```
.pop [-p] <register_name> [+][<relative_pathname>|-]
.pop [-p] <register_name> [>|>>]<register_name>
```

Write the contents of a register to a file, to the step log, or to another register.

The optional -p parameter makes the command refer to a project register. Project registers are separate from ordinary registers, and project registers persist after a job ends.

The following examples show the variety of uses for a .pop command:

- `.pop A data.txt` - writes register A to the file data.txt, located in the step's working directory.
- `.pop ver +data.txt` - appends the contents of register ver to the file data.txt.
- `.pop Alpha` - writes the contents of register Alpha to the step's log.
- `.pop ALPHA > BETA` - gives register BETA the same contents as register ALPHA.
- `.pop A >> B` - appends the contents of register A to register B.

Document # SC23-7681-08

Note

Popping a register does not empty it. To change the contents of the register, push a new value into it using the `.push` command.

.poptag

```
.poptag [-p]<registername>
```

The .poptag command changes the current tag, replacing it with the contents of the specified register.

The optional -p parameter makes the command refer to a project register. Project registers are separate from ordinary registers, and project registers persist after a job ends.

.purge

```
.purge
```

Use the .purge command to set a flag that causes the job to be purged immediately after the job completes. A .lock command that executes after the .purge command causes the job to be saved instead. You can use this command to create jobs that are saved only if they successfully complete all of their steps, by making a .purge command the first step in the project, and by making a .lock command the last step.

.push

```
.push [-p] [+]<register_name> [<relative_pathname> | -]
```

Put the contents of <*relative_pathname*> into register <*register_name*>. The current contents of <*register_name*> are replaced.

To append rather than replace, put a plus sign (+) in front of <*register_name*>.

To clear the register, use a hyphen in place of <*relative_pathname*>.

The optional -p parameter makes the command refer to a project register. Project registers are separate from ordinary registers, and project registers persist after a job ends.

The <*relative_pathname*> is relative to the project or tag path unless the Absolute property for the step is enabled.

The following examples assume that the Absolute property is not enabled for the step:

- `.push Alpha data.txt`

  The contents of the file `data.txt` in the step working directory are put in register Alpha.

- `.push +B ..\newdata.txt`

  The contents of the file `newdata.txt` in the parent directory of the stepworking directory is appended to register B.

Document # SC23-7681-08

- `.push ALPHA -`

  Register ALPHA is cleared.

.put

`.put [<relative_path>/]file server:[[<relative_path>/]file]`

Use the .put command to transfer a file from one logical server to another. The .put operation executes from the current server/path and sends the specified file to the remote server. The destination path name is relative to the target server's base path. The source path name is relative to the step's current working directory. The remote server must specify a logical server that allows the .put operation for files (see “Enabling file copying on a server resource” on page 118). Only single files may be transferred.

The path specification can include environment variables. This capability allows you to specify files relative to the path used by a specific job. See the description of paths used by jobs in “Working directories for jobs” on page 161.

If the server name you are using has spaces in it, put the name in quotes.

The transfer is not fast, so you may want to choose a different method to transfer large files. Expect speeds of no more than 40 KB per second; a 70 MB file could take 45 minutes to an hour to transfer.

Note

If the destination file already exists, it is overwritten without warning.

.rem

Description

The .rem dot command is for comments. It is useful for making notes about the commands in a step.

Example

```
.rem This is some comment text here
.rem all text after .rem is ignored.
```

.report

Add report output for a report that you create in Quick Report to a job BOM. Quick Report is a separately licensed feature of Rational Build Forge.

Description	Use the .report command to add the report to a project. The .report command runs the report and adds its output to the job BOM. Whenever the job runs, the BOM displays report results based on current data.

Document # SC23-7681-08

Syntax `.report <report_name>`

The restrictions for the <*report name*> are as follows:

- The <*report name*> is required.
  - The name is case sensitive
  - If the name includes spaces, enclose it in quotes. For example, "my_report".
- The <*report_name*> you specify must be a saved as a public report.
- The BOM report type is not supported.

.retag

`.retag <new_tag>`

Use the .retag command in a step to change the tag for a job during the job. You can use variables or commands as the new tag value.

.retry

`.retry <count> <command>`

Use the .retry command to allow a command to be retried on failure. The .retry command takes a single count argument that specifies the number of times to retry the command. The command to execute is taken as the remainder of the arguments and thus the .retry command must be the final dot command for a step. For example, the command

```
.retry 3 myscript.sh arg1 arg2 arg3
```

executes “myscript.sh arg1 arg2 arg3” up to 3 times before failing the step. The first invocation of the command that returns a successful status stops the retry process.

.rget

`.rget server:[<path>] [<path>]`

The .rget command works similarly to the .get command, but copies an entire directory tree, recursively. You must supply directory names as the parameters. For example, the command

```
.rget winbuildserver1:config myconfig
```

copies the contents of the directory config on the server winbuildserver into the myconfig directory on the current server.

Note

You cannot use environment variables in this command.

Document # SC23-7681-08

.rmdir

```
.rmdir <relative_path>
```

The .rmdir command removes a directory specified by <relative_path>. The system removes the base directory specified by the path name, including all contents and descendants.

.rput

```
.rput [<relative_path>] server:[<relative_path>]
```

The .rput command works similarly to the .put command, but copies an entire directory tree, recursively. The relative paths you supply must be directories, not files. For example, the command

```
.rput myconfig linuxserver5:feb2005
```

copies the contents of a directory myconfig from the current server to the feb2005 directory on server linuxserver5.

Note

The source path is relative to the working directory of the step, so it includes or does not include the project and tag directories based on the value of the step's Absolute property. The destination path is relative only to the Path property of the destination server. See “Working directories for jobs” on page 161 for more information on how the system constructs paths.

Note

You cannot use environment variables in this command.

.run and .runwait

```
.run [-c "<condition>"] "<ProjectName>"
```

```
.runwait [-c "<condition>"] "<ProjectName>"
```

You can use the .run and .runwait commands to launch a chained project from a step command, providing a more flexible means of launching chains, including the evaluation of environment variables to determine whether a chain is launched.

The commands differ in how they behave after they launch a project:

- The .run command launches the specified project as a chain, following the rules for environment variable inheritance for chained projects.
- The .runwait command launches the specified project; then the launching step waits while the launched build completes. When the launched build completes, the system sets the result value of the launching step to the pass or fail status of the launched build (the only values that can result are pass or fail).

Document # SC23-7681-08

Important

A project that includes a step with .runwait consumes two job slots when it runs. If there are insufficient job slots available, the step fails with an error.

Conditional launches

You can use the optional -c parameter to make the launch depend on a condition. You can use environment variables in the condition. The condition can be of several forms:

String comparison	You can use equal (=) or not equal (!=) operators to evaluate strings. The chain is launched if the comparison evaluates true.
Numeric comparison	You can use <, >, <>, ><, or = operators to compare two numeric values.
Command success	You can use a command enclosed in backticks as the value of the -c parameter. The system runs the command; if it succeeds, the chain is launched.

Example 2. Examples

```
.run "BuildWindowsDriver"
```

The system launches the BuildWindowsDriver project. The launching project continues with the next step immediately.

```
.runwait "BuildWindowsDriver"
```

The system launches the BuildWindowsDriver project. The system pauses the launching project at the .runwait step. When the BuildWindowsDriver project completes and passes, the .runwait step's status is set to pass.

```
.run -c "$HOMEDRIVE=C:" "Simple Echo"
```

The system runs the project Simple Echo if and only if the HOMEDRIVE variable has the value C.

This command produces log output such as the following (in the EXEC section of the step log):

- When HOMEDRIVE is C:

```
.run Condition: 'C:' = 'C:' satisfied.

Queueing Project "Simple Echo" on server [WinBox].
Queued Build 'BUILD_202' of project 'Simple Echo'.
```

- When HOMEDRIVE is not C:

```
.run -c "$HOMEDRIVE=C:" "Simple Echo"

.run Condition: 'D:' = 'C:' unsatisfied, no project queued.
```


Document # SC23-7681-08

The system can numerically compare strings if they contain numbers. For example, it handles the following cases as shown:

.runwait -c "a12b<c42d" "Simple Echo"

```
.run Condition: '12' < '42' satisfied.
Queueing Project "Simple Echo" on server [WinBox].
Waiting for .run build (4411) to complete.
.run build is now running.
.run build has finished.
Build 'BUILD_203' of project 'Simple Echo' completed.
```

.runwait -c "f43g<>h43i" "Simple Echo"

```
.run Condition: '43' <> '43' unsatisfied, no project
queued.
```

The following examples show how to use commands as conditions. Note that the command must be enclosed in both quotes and backticks:

.run -c "`exit 1`" "Simple Echo"

```
Env .run encountered an error during variable expansion,
parameter [`exit1`] expanded to [].
Expansion returned non-zero exit, project will not be
queued.
```

.run -c "`exit 0`" "Simple Echo"

```
Expansion returned zero exit, project will be queued.
Queueing Project "Simple Echo" on server [WinBox].
Queued Build 'BUILD_204' of project 'Simple Echo'.
```

When you use .runwait and a build fails, the log looks similar to the following:

.runwait "Fail Build"

```
Queueing Project "Fail Build" on server [WinBox].
Waiting for .run build (4413) to complete.
.run build is now running.
.run build has finished.
Build 'BUILD_3' of project 'Fail Build' Failed, setting step
 status to fail.
```

.scan

`.scan [-v][-i <ignorepattern>] baseline | checkpoint`

Use the .scan command to enhance the data stored in the BOM for the job. It tracks the files in the step's working directory, along with MD5 values for each file.

`.scan baseline`

Stores a list of all files in the step's working directory. The system displays the list as a category in the BOM for the job. You can have multiple baseline commands in a job, but each one resets the list to the state of the step working directory when the command executes. The final BOM displays only one baseline category.

`.scan checkpoint`

Stores a list of all new, changed, and deleted files since the last `.scan baseline` or `.scan checkpoint` command in the job.

Document # SC23-7681-08

The system displays the list in the BOM. Each checkpoint command creates a new category in the BOM.

You must use a `.scan baseline` command before the first `.scan checkpoint` command in your job. A `.scan checkpoint` command that precedes a `.scan baseline` command is ignored.

Command options:

-v Record a copy of the change information in the job log.

-i Ignore directories that match the supplied pattern. The pattern can match the beginning, end, or any directory part of the path. You can use this option to eliminate source control directories from change listings.

Example for CVS:

```
.scan -i CVS checkpoint
```

The example command keeps CVS directories out of the reports.

Example for Subversion:

```
.scan -i .svn baseline
```

If `-v` is used with `-i`, the system logs changes to the source control directories but the changes are not included in the BOM.

Note

Do not use more than one `.scan` command in a single step. The system cannot provide accurate output for the `.scan` commands if you use more than one in a single step.

For more information on using these commands, see “Adding baselines and checkpoints with the .scan command” on page 153.

.semget

```
.semget <semaphorename>
```

When a step issues this command, the system checks whether a semaphore with the stated name exists.

- If no such semaphore exists, the system creates one and assigns it to the step's job. Then execution continues with the next step.
- If some other job has already claimed this semaphore name, the job hangs on the .semget step until the other project releases the semaphore.

See “semaphore” on page 294 for more information on using this command.

Document # SC23-7681-08

.semput

```
.semput <semaphorename>
```

Use the .semput command to release the semaphore with the name <*semaphore_name*>. See “semaphore” on page 294 for more information on using this command.

.set

```
.set env <EnvironGroup> "<VariableName>=<DesiredValue>"
```

The `.set` command assigns a value to an environment variable.

Note

Variables set by this command must already exist.

Use the `.set` command to change the *master record* for an environment. When the system runs a project, it makes a copy of the project environment from the master record, stores the copy in the job records, and uses that copy as the project default.

When a step runs, it uses the job copy of the environment, not the master record. Therefore, using `.set` has the following effects:

- When a `.set` command executes in a step, later steps that use the *default* step environment do *not* see the changes. The system uses the job copy of the default environment for the step.
- When a `.set` command executes on a *specified* environment, then later steps that specify that environment see the changes you made. The system reads the master record for the environment when the step specifies an environment. This is true even if the specified step environment is the same environment as the project default.
- Changes made by a `.set` command persist after a job is over. Future runs use the values created by previously-run `.set` commands.

For more information on using this command, see “Working with job data” on page 113. Also see the similar command “.bset” on page 123.

.sleep

```
.sleep <seconds>
```

Use the .sleep command to specify a number of seconds for which the step will be paused. Because the Management Console processes this command, no connection to the remote server is created. You can also use a “.sleep 0” command as a platform-independent null command.

.snapshot

Use the .snapshot command to create a new instance of the calling project and store the instance as a project snapshot in the database. A project snapshot is an executable project.

Document # SC23-7681-08

Description The .snapshot command creates a snapshot of a project and its associated objects that you also choose to snapshot or copy.
Use the .snapshot options to specify the objects to snapshot or copy, as described in the following table. If you do not specify any options, only the project definition and steps, project tags, and notes are included.
The snapshot name is required and must be unique to the project snapshot set. The snapshot name is also assigned to the other objects that you snapshot.
After the project that runs .snapshot is completed, the project snapshot is displayed in the UI as a child of the calling project. The other snapshot objects are also displayed in the UI as children of their base snapshot or parent object.

Syntax
```
.snapshot -v <"snapshot_name"> [-c <"comment">][-e[f]][-s[f]][-pI][-pC]
 [-a][-t][-r][-g]
```

Option	Description
- v <"*snapshot_name*">	The name of the project snapshot is required. The snapshot name must be unique to the project. You must quote the name.
- c <"*comment*">	Saves an optional comment as part of the snapshot. You must quote the comment.
-e -ef	Snapshots the project and step environments when the project snapshot is created. Adding the f option also snapshots any environments that the snapshot environments include with the Include environment variable type.
-s -sf	Snapshots the project and step selectors when the project snapshot is created. Adding the f option also snapshots any selectors that the snapshot selectors include with the Include selector property type.
-pI	Snapshots the inline projects or libraries and their steps when the project snapshot is created. Inlined projects or libraries are triggered by a step and run after the step completes.
-pC	Snapshots the chained projects or libraries and their steps when the project snapshot is created. Projects or libraries can be triggered by a project pass/fail condition or a step pass/fail condition. Snapshots are created for both types of conditionally chained projects or libraries.
-a	Copies the adaptor link when the project snapshot is created. The adaptor link adds an adaptor to the project and runs the adaptor code.
-t	Copies notification templates for pass and fail notification events that are set at the project level and step level.
-r	Copies the project registers when the project snapshot is created.

Document # SC23-7681-08

Option	Description
-g	Copies the tag values for the project tag variables. Tag variables are automatically copied, but their values are not. If you do not copy the tag values, they are reset to 1.

.source

Description Use the .source command to add an adaptor for a source code application to a project step. A source code adaptor is a Build Forge object; it is based on the adaptor template for a source code application. The adaptor code for the step is executed when the project runs.

Syntax `.source <adaptor_name> [entry_name]`

The <*adaptor_name*> is required; it is the name assigned to the adaptor in the Management Console. The <*adaptor_name*> case should match the case used in the console.
If your adaptor template has multiple interface functions, specify the one that you want to execute by using the *entry_name* option. The *entry_name* option must match the name attribute specified for interface element in your adaptor template. In the following example, the entry name is By Date.
If you are using an adaptor link, the adaptor is called automatically and the first interface function in the adaptor template is executed. To execute a different interface, in the adaptor template, set the default attribute to true (default="true") on the interface that you do want to execute.

Examples
```
.source MyClearCaseAdaptor

.source MyClearCaseAdaptor "By Date"
```

Notes To create an adaptor or view a list of adaptors, select **Projects** → **Adaptors** .
The adaptor templates provided with the Build Forge product are located in:

`<bf-install>/interface`

.stop

```
.stop <state of build>
```

.stop forces the build to immediately stop processing. Use this command to terminate a build.

.strsub

`.strsub <source> <replacement> file [file ...]`

Use the .strsub command to perform basic string replacement in one or more text files. The system scans the target file(s) for the <*source*> string; where a match is found, the system replaces the

Document # SC23-7681-08

<*source*> string with the <*replacement*>. The .strsub command replaces every instance of the string (source) on each and every line in each file.

The .strsub command works across operating systems, without depending on any specific commands being available on the server.

To replace a string _VERSION_ in a file about.c, use a command such as:

```
.strsub _VERSION_ 2.34 about.c
```

You must specify one or more filenames exactly, without using wildcards. For example, a command like the following will fail:

```
.strsub _VERSION_ 2.34 *.txt
```

However, you can use variables in the command, so a command such as the following will work if the VERSION and FILENAME variables have been defined in the environment.

```
.strsub _VERSION_ ${VERSION} ${FILENAME}
```

Note

Use spaces to separate parameters in the command. Do not quote the parameters.

The .strsub command is similar to the .edit command; their differences include:

- The .strsub command is faster than .edit when performing replacements in large text files or many files.
- The .edit command can perform regular expression searches and replaces.
- The .edit command replaces the first instance only of the string (search_expression) on each and every line in each file.
- The .strsub command replaces every instance of the string (source) on each and every line in each file.

The .edit command uses POSIX Extended Regular Expression syntax by default. If the agent has been compiled with Perl Compatible Regular Expression support, then the substitution expression may be followed with a "p" character (to indicate that PCRE syntax should be used, instead).

In both cases, the expression is interpreted twice by the agent processing. Therefore four backslashes should be used anywhere that a single backslash would normally be used. For example:

```
Four slashes escape a literal period:
\\\\.
```

Alternatively, you may use the /x flag to suppress backslashes:

```
.strsub/x
```


Document # SC23-7681-08

.test

Description Use the .test command to add an adaptor for a test application to a project step. A test adaptor is a Build Forge object; it is based on the adaptor template for a test application. The adaptor code for the step is executed when the project runs.

Syntax `.test <adaptor_name> [entry_name]`

The *<adaptor_name>* is required; it is the name assigned to the adaptor in the Management Console. The *<adaptor_name>* case should match the case used in the console.
If your adaptor template has multiple interface functions, specify the one that you want to run by using the *entry_name* option. The *entry_name* must match the name attribute specified for interface element in your adaptor template. In the following example, the entry name is TestFunction.
If you are using an adaptor link, the adaptor is called automatically and the first interface function in the adaptor template is run. To execute a different interface in the adaptor template, set the default attribute to true (default="true") on the interface that you want to run.

Examples
```
.test MyTestAdaptor
```
```
.test MyTestAdaptor TestFunction
```

Notes To create an adaptor or view a list of adaptors, select **Projects → Adaptors** .
The adaptor templates provided with the Build Forge product are located in:

```
<bf-install>/interface
```

.tset

```
.tset env <EnvironGroup> "<VariableName>=<DesiredValue>"
```

The .tset command changes project settings temporarily during a step.

- Use the *env* paramet to specify the environment in which to add or change variable values. You can use .tset to set a variable that does not yet exist. The values set by the .tset command are written to the job record. They do not update the database record for the environment set. Later jobs are not affected by the changes.

The .tset changes take effect in the current step. It takes effect for all commands in the step. It is also in effect for any Inline specified for the step.

.unlock

```
.unlock
```

.unlock causes the system to release a job that was locked using the .lock command. The unlocked run is now listed on the Jobs > Completed tab, and is de-listed from the Locked tab.

Document # SC23-7681-08

Working with jobs

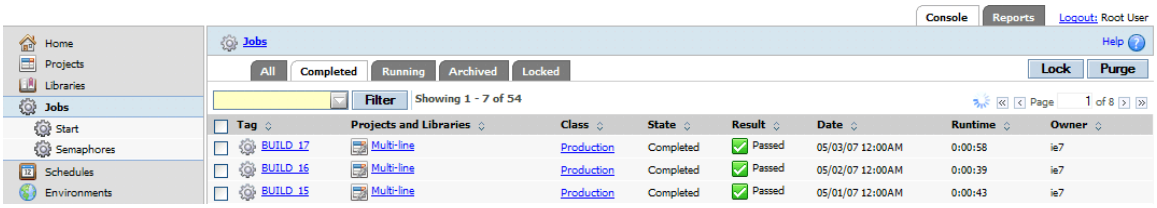
This topic describes how to run, view, and manage jobs in the Management Console.

About jobs

A job is a project that is executing or has finished executing.

Use the Jobs panel to work with running jobs and job results.

The Jobs module



Jobs tabs

The Jobs tabs let you to view lists of jobs by status: All, Completed, Running, Archived, or Locked.

Click a tab to view jobs by type. Each tab has different options for managing jobs in the list.

All	All running jobs are presented first, sorted by their start time. All completed jobs follow, sorted by their completion time. The check boxes are greyed. You can click on any of the links. No special actions are available.
Completed	All completed jobs are presented, sorted by their start time. Running, archived, and locked jobs are not presented. You can purge or lock completed jobs. Select the check boxes for the desired jobs, then click **Purge** or **Lock** . When a completed job is purged, it is moved to **Archived** if it is not completely deleted.
Running	All running jobs are presented, sorted by their start time. You can cancel one or more running jobs. Select the check boxes for the desired jobs, then click **Cancel** .
Archived	All archived jobs are presented, sorted by their start time. An archived job is one for which some information has been deleted, typically from an automatic purge specified in the class definition for the job's class. You can purge archived jobs. Select the check boxes for the desired jobs, then click **Purge** .
Locked	All locked jobs are presented, sorted by their start time. You can unlock one or more locked jobs. Select the check boxes for the desired jobs, then click **Unlock** .

Document # SC23-7681-08

Note

You cannot deleted locked jobs directly. To delete them, first unlock them, then go to the **Completed** tab and use **Purge** .

The list of jobs in any tab is presented by completion time, most recent job first. For each job listing, the following information is shown in columns:

1. **Tag** , the build tag, shown as a link. Click it to see the logs for the job steps.
2. **Projects and Libraries** , shown as a link. Click it to see the project used for the job.
3. **Class** , shown as a link. Click it to see the class definition of the class used for the job.
4. **State** , one of Running, Completed, Archived, or Locked.
5. **Result** , one of Passed, Failed, Warning, or Cancelled.
6. **Date** , the date and time the job was started.
7. **Runtime** : for running jobs, the current elapsed time. For completed jobs, the total elapsed time.
8. **Owner** : the user who started the job.

You can control what jobs are presented in the jobs list in any tab:

- *Filter*: you can filter the list of jobs to view only those that contain a specified string. The filter is matched if the string exists in any column.
  - New filter: Type the string in the box, then click **Filter** .
  - Existing filter: click the arrow next to the box, then select the filter string to use.
- *Sort*: you can sort the jobs list by any single column's values. Click the arrows next to the column name. If you click it multiple times, the sort cycles through the following sort orders.
  1. ascending (up arrow is highlighted in the column header)
  2. descending (down arrow is highlighted in the column header)
  3. no filter

Jobs list fields

Fields show information for each job.

The list of jobs in any tab is presented by completion time, most recent job first. For each job, the following information is shown in columns:

1. **Tag** , the build tag, shown as a link. Click it to see the logs for the job steps.

Document # SC23-7681-08

2. **Projects and Libraries** , shown as a link. Click it to see the project used for the job.
3. **Class** , shown as a link. Click it to see the class definition of the class used for the job.
4. **State** , one of Running, Completed, Archived, or Locked.
5. **Result** , one of Passed, Failed, Warning, or Cancelled.
6. **Date** , the date and time the job was started.
7. **Runtime** : for running jobs, the current elapsed time. For completed jobs, the total elapsed time.
8. **Owner** : the user who started the job.

Filtering and sorting the jobs list

You can filter and sort the jobs list.

Filtering limits the jobs list to those that match a string you specify. Sorting orders the list by the values in any one column.

- *Filter*: you can filter the list of jobs to view only those that contain a specified string. The filter is matched if the string exists in any column.
  - New filter: Type the string in the box, then click **Filter** .
  - Existing filter: Click the arrow next to the box, then select the filter string to use.
- *Sort*: You can sort the jobs list by any single column's values. Click the arrows next to the column name. If you click it multiple times, the sort cycles through the following sort orders.
  1. ascending (up arrow is highlighted in the column header)
  2. descending (down arrow is highlighted in the column header)
  3. no filter

Running jobs and viewing results

You can run, schedule, cancel, and restart jobs and view job results.

Starting jobs

There are several ways to start a job.

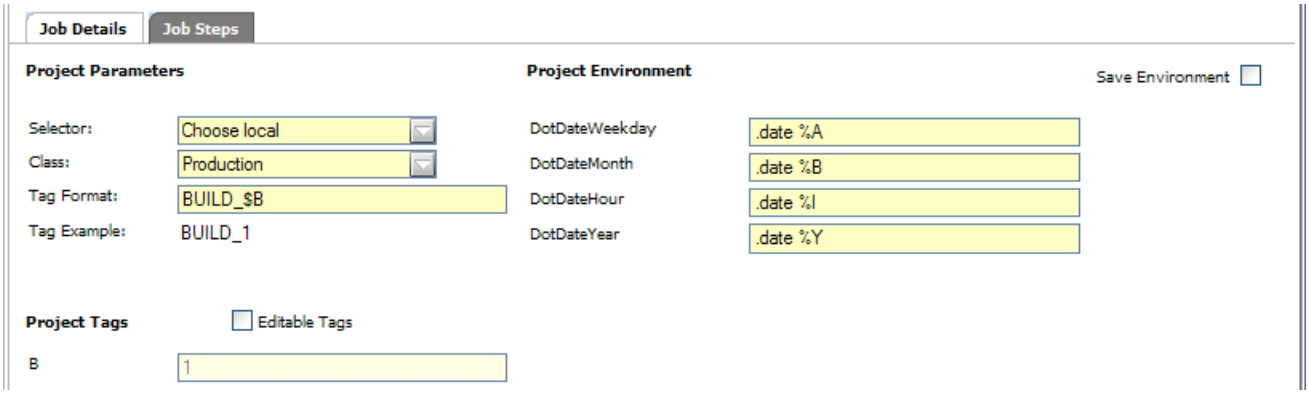
A project must have an environment and a selector defined in order to run as a job.

- *From the Jobs panel*: Select **Jobs** → **Start** and click the project name.

  As when you click **Start Project** , this method displays the Start Project page.

Document # SC23-7681-08

Start Project Page



- *From the Projects panel*: Click the Fast Start icon.

  The following conditions are checked. If the check passes, the project starts immediately.

  - The project contains one or more steps.
  - No variable in the environments for the project is set to Must Change in the On Project variable property.

  If the check fails, then the Fast Start icon is greyed.

  Running a project this way uses its default values for selector, class, tags, and environment variables.

  If the Enable Quickstart system setting is set to YES, the Projects page checks *all* projects to determine if they are eligible for Quick Start. If a large number of projects are defined, the Projects page can take a long time to display the projects list. A different icon indicates that a project can be started: . This was the default behavior until version 7.1.1.1.

  For both cases, the Must Change Variable Check Depth system setting controls how deeply the project environments are checked. The default is 5, indicating that up to five levels of include are checked. Variables of type Include are used to include the variables of a specified environment.

- *From the Steps panel*: While viewing a 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.

Document # SC23-7681-08

- 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.

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.

Viewing job results

You can view job results in all job tabs.

1. Click **Jobs** .

   The **Completed** tab is selected.

2. Click the tab you want.

3. Click the tag of the job you want to view.

   The job results are displayed initially as follows:

   - A list of steps is shown in the main panel. For each step there are columns of information.
     - Step: The step number
     - Step Name: The step name. A graphic in front of the step name indicates the step type: step, threaded, broadcast, join. Click the step name to see the step log.
     - Server (Selector): The server name where the step ran, plus the selector that chose the server.
     - Runtime: Elapsed time of the step (hours:minutes:seconds)
     - Result: The step result, Passed, Failed, Cancelled
     - Chains: If a step started another job, a link to that job is shown.
   - A menu appears at the bottom of the left-pane menu, showing the following items:
     - Results: Shows the list of steps in the main pane.
     - Bill of Materials: Shows the bill of materials (BOM) in the main panel. The BOM contains links to show Job Steps, Step Manifests, and Checkpoints (if checkpoints were used)
     - Notes: Shows the notes entered for this job.
     - Step Logs (this item is open and the step names are shown): shows the list of steps. Click a step to see the step log.
     - Chains: Shows the job numbers of any jobs launched as a chain.

Document # SC23-7681-08

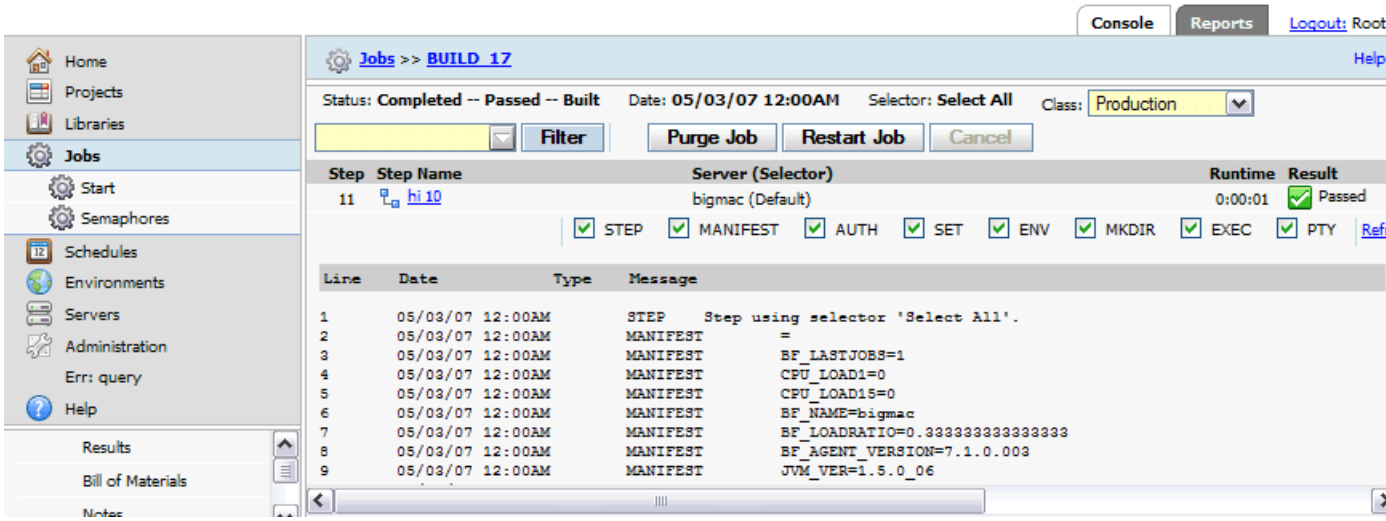
4. To view a step log: Click a step name from any list of steps.

   Initially, all categories are selected.

   If you are viewing a step in a Running job and the step has not completed, you see a partial log and you may not see all categories. To update the log view for a running step, click the build tag (at the top, shown as **Jobs >>** *Tag*), then click the step.

5. To filter the step log: Check or uncheck categories, then click **Refresh** .

Sample step log



Restarting failed jobs

You can *restart* a job if it fails. Restarting starts a new run under the same tag. It continues from the point where it failed.

To restart a job, click the job tag in the list of builds (on the **Jobs** → **Completed** tab). The system displays information about the build, and includes a **Restart** button at the top of the main pane.

1. Click **Restart** ; the system displays a Restart page.

2. Select options. You can click the **Steps** tab to view the steps in the build. On the Steps tab, select the **Sync Cmds** check box if you want the system to get any updates to the commands in its steps from the project record; if you do not select it, the commands are run exactly as they were when the job was originally started.

3. Click the Restart page's **Restart** button.

A restarted job differs from a new job in the following ways:

- It uses the same tag number as the failed run, and replaces the failed run in the Completed list.

Document # SC23-7681-08

- By default, it starts from the failed step, and does not repeat any of the steps that passed in the previous run. However, you can choose which steps actually run when you restart the job.
- By default, the system supplies the same environment variable values that you supplied on the previous run; however, you can change these before restarting the job.
- The system evaluates the success of the job based only on the steps it runs during the restarted job. Failures in the prior run do not affect the status of the restarted job.

Restarting while loops

In the case of steps of type While Loop, the step is restarted at the iteration where the step failed, based on the value of the BF_ITERATION system variable. Example job flow:

1. Job enters step of type While Loop
2. Condition evaluates to true
3. BF_ITERATION is set to 1
4. The Command and Inline are executed successfully
5. The step loops
6. Condition evaluates to true
7. BF_ITERATION is set to 2
8. The job is stopped during execution of the Command or Inline

If the job above is restarted, it restarts at Iteration 2. It attempts to execute the Command and Inline for the step.

Using the Bill of Materials

The system generates a Bill of Materials (BOM) after each job. The BOM contains information about the steps in the job and the changes to files that resulted from it. The BOM can be provided to consumers of the job, such as your quality assurance department, for help in understanding the contents of a new job. It can serve as an audit solution for your build and release process. With the BOM, you get complete documentation of a job's contents. It can include the results, notes, environments, lists of files, and code changes. Information about job stops, starts, and restarts is logged under the "Job Audit Log." You can use this to compare and summarize the state of builds across the enterprise.

The system generates a BOM for each job automatically, but you can use dot commands in steps to cause the system to store additional information about the state of your files before and after the build.

Document # SC23-7681-08

Displaying the Bill of Materials (BOM)

When you view a completed build (**Jobs** → **Completed**), the system displays the **Steps** tab by default. Click the **BOM** tab to display the Bill of Materials.

Click the + next to any category to expand it. The actual categories you see depend on the project and how your system is configured:

- The Project Steps category is displayed in every job and provides information about the steps that ran for this job.
- The Source Changes category is displayed only if your system includes a source code adaptor and the project has a link to the adaptor. See “Adaptors and job results” on page 239 for details. You can change the format and even the name of the Source Changes category when you configure your adaptor.
- Baseline and checkpoint sections are displayed only if your project included .scan commands.

Adding baselines and checkpoints with the .scan command

You can use the .scan command to add more information to the BOM. When the .scan command is executed, the system stores information about the state of the files in the step's working directory. This section shows an example of how to use it. See also the reference information for “.scan” on page 139.

The command has two forms.

```
.scan baseline
```

Stores a list of all files in the step's working directory tree, with MD5 values for each. The system displays the list in the BOM for the job. You might want to issue this command after performing some setup steps and checking out an appropriate set of files. You can have multiple baseline commands in a project, but each one resets the list to the state of the step's working directory when the .baseline command executes.

```
.scan checkpoint
```

Stores a list of all new, changed, and deleted files since the last .scan baseline or .scan checkpoint in the project, with MD5 values for each file. As with the .scan baseline command, the system displays the list in the BOM. You must issue a .scan baseline command before the first .scan checkpoint command in your project. A .scan checkpoint command that precedes a .scan baseline command will be ignored.

The following example shows how .scan baseline and checkpoint commands work together:

Number	Step	Files after step	BOM data
1	Check out initial files	`config.c` `execute.c`	
2	Baseline	`config.c` `execute.c`	Baseline: `config.c`

Document # SC23-7681-08

Number	Step	Files after step	BOM data
			`execute.c`

3	Add data file	`config.c`
		`execute.c`
		`data.txt`	

4	Checkpoint 1	`config.c`
		`execute.c`
		`data.txt`
			Checkpoint 1:
			Added `data.txt`

5	Add more data files	`config.c`
		`execute.c`
		`data.txt`
		`data2.txt`
		`data3.txt`	

6	Delete data.txt	`config.c`
		`execute.c`
		`data2.txt`
		`data3.txt`	

7	Checkpoint 2	`config.c`
		`execute.c`
		`data.txt`
		`data2.txt`
		`data3.txt`
			Checkpoint 2:
			Added `data2.txt`, `data3.txt`
			Deleted `data.txt`

Exporting the BOM as an XML file

This topic describes the syntax, usage, and option descriptions for the bfbomexport command.

Build Forge commands are located in the Build Forge installation directory on Windows and in the <*bf-install*> /Platform directory on UNIX or Linux.

Description

Use the bfbomexport command to export the Bill of Materials (BOM) for a job to an XML file. After collecting BOM information, bfbomexport saves it to the location and file name you specify.

To identify which BOM you want to save to an XML file, you must identify the project and the build for the job.

Document # SC23-7681-08

You must run the bfbomexport command from the Build Forge installation directory for the Management Console and from the /platform directory on UNIX or Linux.

Syntax

```
bfbomexport [-f filename] [-p projectID | -P projectName] [-b buildID | -t buildTag] [-H]
```

Options

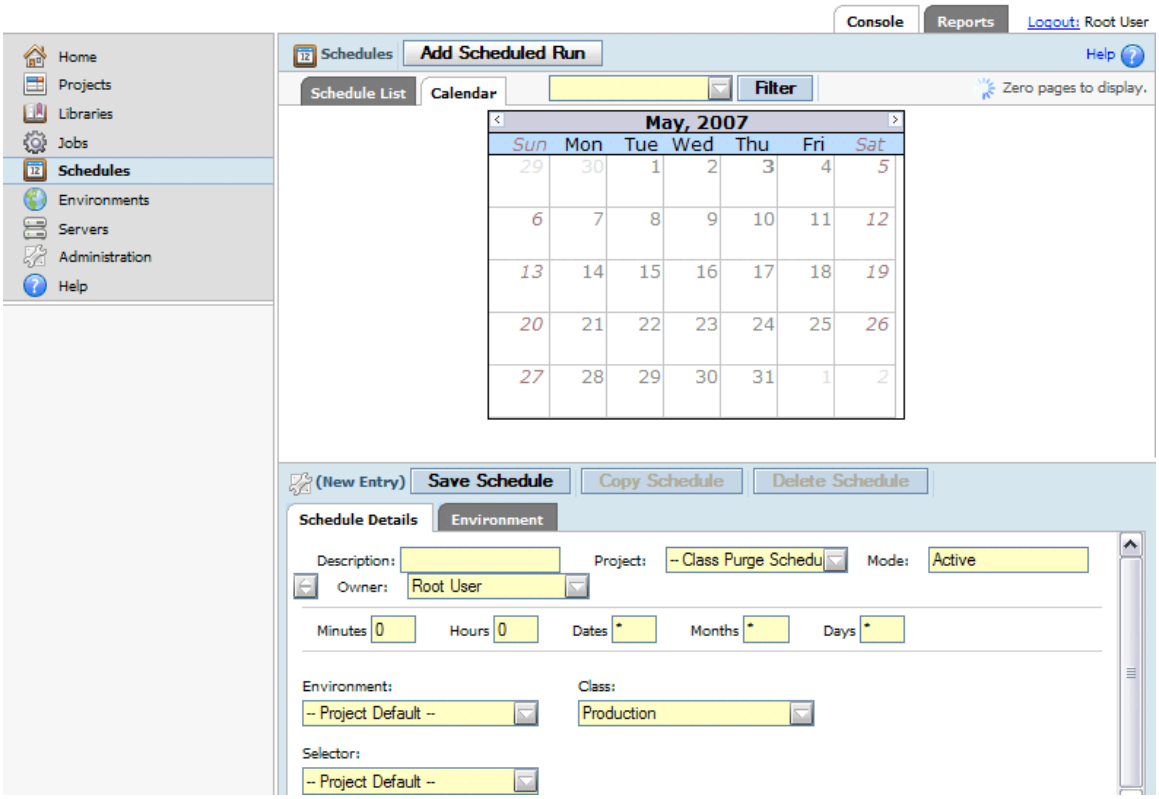
Option	Description
-f filename	An optional path and/or file name. The BOM for the job is saved in XML format. If a filename is not provided, the BOM is written to standard output (stdout). If a pathname is not provided, the current working directory is used.
-p projectID	The project ID for the job. (The bfexport command with the -l option lists project IDs.)
-P projectName	The name of the project.
-b buildID	The build ID.
-t buildTag	The build tag name.

Scheduling jobs

After you create a project, you can schedule it to run at a future time, or at regular, repeated intervals. For example, you can set up a project to run every hour or every day.

Document # SC23-7681-08

The Schedule module



Note

If one or more jobs of a project are already running when the schedule activates, the system checks the **Run Limit** property of the project. The system does not launch the scheduled job if the number of running copies of the project equals the **Run Limit** . Set the **Run Limit** value to 1 if you want the system to skip a run if the prior run has not yet completed.

1. Click **Add Scheduled Run** .
2. Enter a description for the schedule entry.
3. Select the project (in this case Hello World) from the Project list.
4. You can leave the **Owner** , **Environment** , **Selector** , and **Mode** at their default values for this example, but note the following:
   - The scheduled job runs as if manually launched by the Owner.
   - If you specify an environment here, you can also set starting values just as if you manually started the project.
   - If you specify an environment or selector, your selections override the project settings for environment or selector for the scheduled run only.

Document # SC23-7681-08

- If you specify an environment for the project being run by the schedule, a copy of the environment is made. This copy can only be accessed through the schedule. To apply your changes to the schedule's original environment, click **Resync Environment** .
- The Mode field options are Active, Inactive, and Once. If you select Once, the schedule runs only on the next occasion that matches the time parameters, instead of repeating for every time that matches.

5. Change the Hour field to an asterisk.
6. Change the Minute field to `*/5`. This specifies every 5 minutes. (The schedule parameters work like the UNIX® tool cron.)
7. Click **Save Schedule** to save the schedule entry.

After you add the schedule entry, the system starts to calculate the next runtime for the project; the Next Run column shows “Calculating” until the next scheduled time can be displayed.

When the system has computed the next runtime for the project, it displays it in the Next Run column.

The system displays a dynamic calendar on the **Schedules** page, as well as the form displayed when modifying a schedule. The calendar shows the number of projects scheduled for a given day, for two months (the current and upcoming months) and you can mouse over individual days to see the names and schedule parameters of all the projects scheduled for a given day. If you have more than one project scheduled, the system displays a dropdown box to allow you to filter the calendar by project.

Use the Refresh link to refresh the page in a few seconds to see the next run time. Wait until the schedule time has arrived and then refresh the **Jobs → Completed** page to see the executed job.

If you end up with many extraneous runs after you experiment with scheduling, you can delete them by changing the properties of the project's class. You can control other features of project types using classes; see the topic on classes for more information.

You can disable a schedule temporarily, or configure it to run once. When you create a schedule, the system displays a green circle next to it to indicate that it is enabled. Click the green circle to change it to a blue one, which indicates the schedule will run once. Click it again to change it to a red circle, indicating the schedule is disabled; while disabled, the project does not run.

You can also use schedules to denote when purges should be performed, so that the system does not try to run purge jobs when the system is otherwise occupied.

Schedule parameters

This section describes the parameters that you can use to specify when to execute a job.

You use a series of fields to specify the times for a job to execute.

Document # SC23-7681-08

Field	Description	Range
Minutes	The number of minutes.	0-59
Hours	The hour in the day.	0-23
Dates (Day of Month)	The day of the month.	1-31
Months	The month of the year.	1-12
Days (Weekday)	The day of the week (Sunday = 0).	0-6

The numeric values you enter in the fields can be represented as follows:

- Use an asterisk (*) to denote all valid values in the range. An asterisk can be followed by a forward slash (/) and a step value. For example, a value of */2 in the Hours field executes the job every 2 hours.

  ### Note

  Asterisk is interpreted specially in the Dates and Days fields . It matches its meaning in the UNIX/Linux cron facility. If one field has a literal value and the other has an asterisk, then only the frequency for the literal value is used. For example, if Month is *, Dates is *, and Days is 1, the execution occurs every Monday. See also the Day/Date examples below.

- Use a range of numbers separated by a hyphen. For example, 8-11 in the Hours field specifies hours 8, 9, 10 and 11. You can follow a range by a forward slash (/) and a step value. For example, 0-23/2 in the Hours field executes the job every other hour.

- Use a comma-separated list of a set of numbers (or ranges) separated by commas. For example, 1, 2, 3-5, 8.

The schedule is constructed from the values specified for the fields. Examples:

Values Desired schedule	Minutes	Hours	Dates	Months	Days
Run job daily, at 5 p.m.	0	17	*	*	*
Run job weekly, every Monday at 4:30 p.m.	30	16	*	*	1
Run job every half hour on every week day, skip weekends	*/30	*	*	*	1-5
Run job at 12:30 a.m., every other day	30	0	*/2	*	*

The Days and Dates fields use asterisk in a special way.

- If one has an asterisk value but the other has a literal value, the job runs according to the field having a literal setting.

- If both have non-asterisk values, then the job runs when *either* condition occurs.

- If both have asterisk values, then the job runs every day.

Document # SC23-7681-08

Day/Date examples:

Values Desired schedule	Minutes	Hours	Dates	Months	Days
Run job at 1:01 a.m. on the first day of each month. Date uses a literal value, Day uses asterisk.	1	1	1	*	*
Run job at 1:01 a.m. on every Monday in the month. Date uses asterisk, Days uses a literal value.	1	1	*	*	1
Run job at 1:01 a.m. on every Monday in the month and on the first day of the month regardless of whether that day is a Monday. Both Date and Day use a literal value.	1	1	1	*	1
Run job at 1:01 a.m. every day. Both Date and Day use an asterisk value.	1	1	*	*	*

Scheduling purges for classes of job

You can control when the system conducts purges of old jobs, by creating schedules for classes of jobs. You create these schedules just as you would create a schedule to launch a project, but you select the “Class Purge Schedule” option instead of a project. When you do this, the system checks for jobs to purge of the class you select at the time you select in the schedule. For each job that qualifies, the system creates a purge job and puts the job in the waiting queue.

By default, the system checks for jobs that should be purged (based on the class properties that define rules for automatic deletion) at intervals set by the Purge Check Time system setting (which defaults to every 15 minutes). This behavior can lead to purges competing for system resources with ordinary jobs.

If you create a schedule for a class of job, the system only checks for jobs to purge when the schedule activates. If no schedule exists for a particular class, the system uses the default behavior for the jobs of that class. If you want to restrict all purges to specific times, you must create at least one schedule for each class.

To define a purge schedule, create a schedule as usual, but select “Class Purge Schedule” in the Project field. Then select a class in the Class field.

The following example shows how to set up a purge schedule to purge Production-class jobs on Saturdays at noon. With such a schedule in effect, on Saturdays at noon the system checks all the Production jobs, and schedules purge jobs for all the jobs that meet the Production class's deletion criteria.

Administering jobs

You can lock, archive, and delete jobs.

Locking jobs

You can lock a project to prevent it from being automatically deleted.

Document # SC23-7681-08

You can lock a job by selecting it in any of the **Jobs** tabs, and then clicking **Lock** . To unlock a job, visit the **Locked** tab in the **Jobs** module, and select a locked build; then click **Unlock** .

The system will not purge a locked job automatically (such as when the class purge properties for the run call for it to be deleted). You can still delete a locked job manually.

Use the **Locked** tab to view jobs that are in the locked state. Such projects do not appear in the **Completed** tab.

Deleting jobs

The following topics describe several ways to delete jobs.

Deleting a job from the Completed tab

You can manually delete jobs from the Completed tab of the **Jobs** module.

You can manually delete one or more jobs from this list. When you do this, the system launches a purge job based on the class of each run. The process is the same as if an automatic deletion had been triggered by the class properties. The system archives the job (if it is not to be completely deleted) and deletes the data specified for jobs of that class. See “Using classes” on page 80.

To delete jobs:

1. Select the **Jobs** → **Completed** page to display the completed run list.
2. Click one or more check boxes on the right end of the table to select builds to be deleted.
3. From the list box at the bottom of the list, select the Purge option.
4. Click the Go button.

If the class is set to delete output files but retain console data, then deleting the run from the **Completed** list deletes the output files and moves the job's entry from the **Completed** list to the **Archived** list.

Completely deleting a job from the archive list

Select **Jobs** → **Archived** to display the archive list. This list displays data about jobs whose files have been deleted. You can delete jobs here just as you would from the **Jobs** → **Completed** list. Deleting a job from the archive list removes all traces of the job from the database, and drops it from statistics reported by the application.

Automatically deleting jobs

The system automatically deletes a job when the class properties for the job determine that it should be deleted. You can use this feature to prevent data from piling up, and to delete groups of jobs.

If you create a schedule for a class, the system only checks for jobs to purge when the schedule activates. See “Scheduling purges for classes of job” on page 159.

Document # SC23-7681-08

To make sure a job is deleted when you want it to be, check the following settings before you launch or schedule the job:

1. Set the deletion properties of one or more of your classes to allow the system to delete the jobs after a certain number of days or after a certain number of jobs have accumulated, or both.

2. Set the **Class** property of the project you are working with to an appropriate class.

If you generate a number of jobs and then want to delete them, you can temporarily change the deletion properties of the relevant class. Or you can select multiple builds on the **Completed** tab and delete them (click the check box next to each job to select it, then click the **Purge** button.

For example, if you generated many Production jobs on the previous day, you would use the following process to delete them:

1. Note the current settings for the Production class.

2. Change the Days property of the Production Class to 1, then click the **Save Class** button. After a 15 minute delay, the system begins deleting jobs that are older than one day.

3. When the jobs are deleted, change the properties of the Production Class back to their original setting.

Working directories for jobs

The system constructs working directories for each job, so that each run has a labeled and isolated area in which to work. It makes the working directory names using the values provided for the server path, project name, and tag.

When it executes a command, the system starts the command in the directory specified for the step. By default, that directory is the job's working directory, but you can also specify some other directory relative to the server's **Path** property. The topics in this section describe how to create the path and directory in more detail.

Note

When it executes a project, the system constructs the project directory (if it does not already exist) and the job directory. It does not construct the server directory (specified in the server's **Path** property) or the step directory (the step's **Dir** property).

Working directory names for jobs

The following example shows how the system uses several values to construct a job directory, in a job that occurs on a single server:

System Values	Directory Created
Server's Path field: c:\BuildForge	C:\BuildForge**My_Project\Job_5**

Document # SC23-7681-08

System Values	Directory Created
Project name: My Project Tag: Job 5	The system creates only the **bold** portions of the path. You must create the server directory before running the project, or it will fail.

Note

When it creates a project directory, the system changes characters specified in the **Invalid Relative Dir Characters** system setting into underscore characters. By default, the setting contains a space and a backtick character, so that a project named My Project receives a project directory named My_Project.

If a job runs on more than one server, the system makes a job directory on each server. Because each step in a project can specify a different server, the system can potentially create many directories. The following example describes a project that uses two servers:

System Values	Directories Created
Project server: ServerA, with **Path** value of c:\BuildForge Third step in project specifies ServerB, with **Path** value of c:\deployments. (All other steps use the default (project) server, ServerA). Project name: My Project Tag: Job_6	On ServerA: C:\BuildForge**My_Project\Job_6** On ServerB: c:\deployments**My_Project\Job_6** The system creates only the **bold** portion of the path.

In the above example, you can expect any output files from a step to be created by default in the c:\BuildForge\My_Project\Job_6 directory, except for the third step, which uses the c:\deployments\My_Project\job_6 directory.

Constructing directory paths for steps

When the system executes a step, it can start from the directory it constructed for the job, or it can use the **Absolute** option to ignore the project and tag directories.

- When the **Absolute** box is checked, the system constructs the path for the step by adding together the **server path** and the **step's Dir field**. The value in the **Dir** field is a path relative to the server's working directory.

Values for step	Resulting path for command
Server's **Path** field: c:/BuildForge Step's **Dir** field: /bin **Absolute** check box: Checked	C:\BuildForge\bin Use this form to access directories located in the server directory.
Server's **Path** field: c:/BuildForge Step's **Dir** field: / (the default value)	c:\BuildForge

Document # SC23-7681-08

Values for step	Resulting path for command
Absolute check box: Checked	
Server's **Path** field: c:/BuildForge Step's **Dir** field: c:/temp **Absolute** check box: Checked	c:\BuildForge\c:\temp (This example will cause an error; the step will fail.)

Note

You can enter path values with backslashes or forward slashes. The system stores them with backslashes, and changes them to forward slashes as needed on Windows® machines.

- When the **Absolute** box is *not* checked, the system constructs the path for the step by adding together the *server path*, the *project name*, the *tag*, and the step's *Dir field*. The value in the **Dir** field becomes the path relative to the job's working directory.

Values for step	Resulting path for command
Server's **Path** field: c:/BuildForge Project name: My Project Tag: Job_5 Step's **Dir** field: /bin **Absolute** check box: Not checked	C:\BuildForge**My_Project\Job_5**\bin **Bold** portions of the path are constructed by the system if they do not already exist. **Note** When it creates a project directory, the system changes characters specified in the **Invalid Relative Dir Characters** system setting into underscore characters. By default, the setting contains a space and a backtick character, so that a project named “My Project” receives a project directory named “My_Project”.

If the directories specified in a server's **Path** field or a step's **Dir** field do not exist, the step fails; the system does not create these directories. The portion of the path specified by the Step's Dir field must be explicitly created during the project by a preceding step.

Typically, steps early in a project check out a tree of directories from source code control, and following steps act on those directories.

When you add new steps, the system remembers the last setting you chose for this check box, and uses that as the default on new steps.

Slashes in paths

When you enter a path in a **Path** or **Dir** field, the system converts any back slashes to forward slashes. When the system creates the actual path, if the server being used is a Windows® machine,

Document # SC23-7681-08

the system converts any forward slashes to back slashes. Therefore, you can use either type of slash in your paths.

Note

The system does not modify slashes in the command field of a step. Use back slashes or forward slashes as needed for the target server.

Semaphores

Semaphores are global signal flags in the system that set up mutually exclusive (mutex) resources. Use them to make some processes wait for other processes to finish.

Use **Jobs → Semaphores** to view job semaphores that are in use. You can also clear semaphores, which may be necessary when a hung or cancelled job fails to release its semaphore.

You implement semaphores through a pair of dot commands: the .semget and .semput commands. Use the .semget command to “grab” a label: after a step gets a label, any other steps (in any project) that try to get the same label must wait until the original requester releases it.

Note

A step that contains a .semget command *waits* until the semaphore is released. If a job fails and leaves its semaphore active, the semaphore must be cleared manually before any job that uses the semaphore can run again.

For example, suppose you have a program that creates a printer driver, and you want the program to be used by only one process at any one time. Within each project that calls the program, set up three steps with the following command lines:

Step	Command Line
Get semaphore	`.semget $BF_PROJECTNAME_PHYS`
Execute driver creator	`printdrivermaker.exe windows`
Release semaphore	`.semput $BF_PROJECTNAME_PHYS`

You can establish semaphores for key resources in your organization, such as a heavily loaded server or a software program with a single-user license. Every step that uses the resource you want to protect should be wrapped with .semget and .semput commands.

Semaphores obey the following rules:

- When used, a semaphore dot command must be the only dot command for the step.
- The system uses the label as-is; avoid using special characters in a label, or trailing whitespace, as you may confuse the label parser.
- Semaphores are global and can be used to synchronize separate projects in addition to thread blocks.

Document # SC23-7681-08

- Semaphores are created the first time they are accessed.
- Semaphores obtained by a project are automatically released when that project terminates.
- If two steps request a semaphore at the same time, there is no guaranteed order as to which step gets the lock.

Clearing Semaphores Manually

The system automatically releases any semaphores created by a project when the project terminates. However, if an abnormal project termination leads to a semaphore not being released, you can manually clear it.

1. Select **Jobs** → **Semaphores** to display the Semaphores list.
2. Select the semaphore you want to clear.
3. Click **Clear**

Document # SC23-7681-08

Working with reports

This topic describes the built-in reports provided with Rational Build Forge. It also describes how to create and run reports using Quick Report, a separately licensed reporting tool.

About reports

There are two report categories: the built-in reports that you access from the Performance and Queries modules and the reports that you create using the Quick Report tool.

To get started, click the **Reports** tab to display module choices for reporting:

- Select the **Performance** module to view standard reports.
- Select the **Queries** module to view and run predefined query reports.
- Select the **Quick Report** module if you want to create your own reports. Quick Report is a licensed option of Rational Build Forge.

Performance and Queries modules

The Performance module uses Build Forge data to create several standard reports. When you open a performance report, results are updated automatically so that you are always looking at current data.

The Performance module also has a Queries module that includes predefined query reports. Run these reports by providing required data; then click the Quick Start  icon. Query reports are run against current data. Simply rerun a report to see results based on the latest data.

The reports in the Performance and Queries modules cannot be modified. By design, you cannot modify fields or change the report presentation.

Quick Report tool

Quick Report is a reporting tool based on BIRT, an Eclipse-based open source reporting system. With Quick Report, you can create your own reports designs. You can use a provided report type as a starting point; then choose the report fields that you want. All report types use data in the Rational Build Forge database as their data source.

For any report that you design in Quick Report, you can choose from several report formats, such as tables or charts, and you can use grouping and filtering functions to control report presentation.

Prerequisites for displaying data in report output

All reports, including sample reports, display data from the Build Forge database. To populate the database, you must first create projects and run jobs.

Document # SC23-7681-08

Note

If you create and run reports before the database contains data, report results are empty. In addition, you might experience report errors.

Access permissions for displaying data in reports

The access permissions of the user running the report determines what project data is included and visible in the report results.

The project access permissions of the user determine the project data that is included in the report results, as follows:

- For standard reports in the Performance module, the permissions used are those of the user who opens the report page.
- For the predefined reports in the Queries module, the permissions used are those of the user who runs the report.
- For Quick Report, the permissions used are those of the user who runs the report not the permission of the report designer.

Other access permissions enable users to modify report designs and run reports. See Report group permissions for Quick Report.

Exporting results for built-in reports to a CSV file

Several built-in reports provide the capability to export report results to a CSV file. Use this capability if you want to save results or import results to another report application.

For any built-in report that has the Save to CSV link, you can export report results.

Note

This capability is not provided for Quick Report.

To export report results to a CSV file format:

1. Run a report to generate results:
   - Select the **Analysis Report** link to open the Step and server performance report (**Reports > Performance**).
   - Run any report in the Queries module (**Reports > Queries**).
2. From the report results page, click **Save to CSV** .
3. At the File Download dialog, click **Open** or **Save** :
   - Open the CSV file using any application that supports the CSV file format. For example, Microsoft Excel or your favorite text-based editor.

Document # SC23-7681-08

- Save the file to your local workstation or another location on your network.

Standard reports for the Performance module

Standard reports display results based on current data when you open a standard report page.

The Performance module uses current project and job data to create several standard reports:

- a job performance report for all projects
- a job duration report by project
- a step and server analysis report by project

Viewing job performance statics for projects

The Performance module features a job performance report that summarizes high-level job statistics for all projects.

- For the last job, the report shows the completion date-time and the job duration.
- For the total jobs, statistics include the number of jobs passed, failed, or passed with warnings.

Data source: Rational Build Forge database

Report format: Table

Display options: Create a filter to control which projects are displayed in the list.

Fields:

- Confidence Interval: A range of values (in seconds) on either side (+/-) of the job average duration. This range of values is 95% likely to contain the next job duration. For example, if the confidence interval is 5.88 seconds, then the range of values that is 95% likely to contain the next job duration is 5.88 seconds on either side of the job average duration.

To view the job performance statistics report:

1. Open the **Reports** tab.
2. Click the **Performance** module.

   The job performance report for all projects is displayed as a table of results.

Viewing job duration times for a project

The Performance module includes a job duration report that graphs the durations of all the project's jobs.

Data source: Rational Build Forge database

Report format: Graph

Document # SC23-7681-08

To view the project jobs duration report:

1. Open the **Reports** tab.
2. Click the **Performance** module.
3. Click a project name in the list.

   The project jobs duration report is displayed as a graph of job times.

Viewing step and server performance by project

The Performance module includes a step and server analysis report by project. It also includes a critical path analysis report for the steps in the project.

For each project step, the step and server analysis report lists the server that ran the step and lists the times required to perform the step for different runs, for example, the fastest or slowest run. Step times are categorized as pass, fail, or warning. The categories prevent you from comparing times incorrectly, for example comparing pass times to fail times.

The critical path analysis report for steps lists some key metrics for passed jobs.

Data source: Rational Build Forge database

Report format: Table

Key: The -/-/- key provides two pieces of step information: the step result and the step duration in seconds, as shown in the following example:

14/-/- Pass/Warn/Fail	The step passed and the total run time is 14 seconds.
-/-/1 Pass/Warn/Fail	The step failed and the total run time is 1 second.
-/1/- Pass/Warn/Fail	The step passed with warnings and the total run time is 1 second.

Fields:

- Passed Steps: The number of times that the step passed for all jobs.
- Average Run: The average job duration of the passed jobs.
- Confidence Interval: A range of values (in seconds) on either side (+/-) of the job average duration. This range of values is 95% likely to contain the next job duration. For example, if the confidence interval is 5.88 seconds, then the range of values that is 95% likely to contain the next job duration is 5.88 seconds on either side of the job average duration.
- Probability Longest Duration: The likelihood that the next job will equal the longest duration of the passed jobs.

Document # SC23-7681-08

- Probability Shortest Duration: The likelihood that the next job will equal the shortest duration of the passed jobs.

To view the analysis reports, do the following:

1. Open the **Reports** tab.
2. Click the **Performance** module.
3. Choose a project in the list; then click **Analyze**.

   The analysis reports for steps and servers and the critical path analysis report are displayed as tables of results.

Predefined query reports for the Queries module

Provide a date range or other required data to run a predefined report using current project and job data.

The Queries module uses current data to create several predefined reports:

- selector utilization history for projects and steps
- current server manifest for all servers
- job results pass/fail/warning history
- server utilization history for jobs in a date range
- search for a job file based on its MD5 value

Viewing selector utilization history

A report that shows selector utilization for all projects.

The Queries module includes a report that shows selector utilization for all projects. Use this report to see which selector is assigned to projects and steps. This report is especially useful if you are using selectors to dynamically select a server to run a project or step.

Data source: Rational Build Forge database

Report format: Table

Display options: Select **Flatten report output** to expand the tree hierarchy and display all report results in a list.

To view the server utilization report for your steps and jobs:

1. Open the **Reports** tab.
2. Click the **Queries** module.

Document # SC23-7681-08

3. Click the Quick Start icon for the project selectors and step servers report.

   The selector utilization report is displayed as a table of results.

Viewing the current server manifests by server

A report that lists the server manifest properties and values for all servers.

The Queries module includes a report that lists the server manifest properties and values for all servers. Use this report to compare server properties, identify the servers that might be chosen by the same selector, or to identify a needed property for a server.

This report displays all the manifest properties assigned to the server by the collector assigned to the server. To include the special manifest properties that are automatically assigned to all servers, select **Show BF_properties** .

Data source: Rational Build Forge database

Report format: Table

Display options: Select **Flatten report output** to expand the tree hierarchy and display all report results in a list.

To view the report of the current server manifests for your servers:

1. Open the **Reports** tab.
2. Click the **Queries** module.
3. (Optional) Select **Show BF_ properties** to include the special manifest properties in the report results.
4. Click the Quick Start icon for the current server manifest report.

   The current server manifest report is displayed as a table of results.

Viewing job pass/fail/warning results

A report that lists the pass/fail/warning results for the jobs completed over a specified date-time range.

The Queries module includes a report that lists the pass/fail/warning results for the jobs completed over a specified date-time range. This report applies to all jobs for all projects. Use this report to get a quick health check of completed jobs.

Data source: Rational Build Forge database

Report format: Table

Display options: Select **Flatten report output** to expand the tree hierarchy and display all report results in a list.

To view the job results report for your projects:

Document # SC23-7681-08

1. Open the **Reports** tab.
2. Click the **Queries** module.
3. Use the default time range or specify a time range for the report results in the following format:

   `MM:DD:YY HH:MM AM/PM`

   The day starts at midnight (00:00) and ends at midnight the next day. For example, to return all the jobs completed on July 21, 2008, enter the following date range:

From:	`07/21/08 00:00 AM`
To:	`07/22/08 00:00 AM`

4. Click the Quick Start icon for the job results report.

   The job results report is displayed as a table of results.

Viewing server utilization for jobs in a date range

A server utilization report that lists the servers for all jobs of the project.

The Queries module includes a server utilization report that lists the servers for all jobs of the project. For each step, the report identifies which server ran the step and lists the step duration. Use this report to compare step durations for different servers.

Data source: Rational Build Forge database

Report format: Table

Display options: Select **Flatten report output** to expand the tree hierarchy and display all report results in a list.

To view the server usage report for steps:

1. Open the **Reports** tab.
2. Click the **Queries** module.
3. Use the default time range or specify a time range for the report results in the following format:

   `MM:DD:YY HH:MM AM/PM`

   The day starts at midnight (00:00) and ends at midnight the next day. For example, to return all the jobs completed on July 21, 2008, enter the following date range:

From:	`07/21/08 00:00 AM`
To:	`07/22/08 00:00 AM`

4. Click the Quick Start icon for the server utilization report.

Document # SC23-7681-08

The server utilization report is displayed as a table of results.

Searching for a job file using its MD5 value

A report that searches the job BOMs for a file that matches an MD5 value. The MD5 hash value is a digital fingerprint of a file

The Queries module includes a report that searches the job BOMs for a file that matches an MD5 value and, if a match is found, lists the job tag and the file location. Use this report if you need to search for a specific version of a file and know its MD5 value.

As a prerequisite, your projects must include the .scan baseline or .scan checkpoint commands to store MD5 values. For details, see the reference information for ".scan" on page 139. The .scan command completes the following tasks:

- creates a list of files in a steps's working directory
- generates the MD5 values for the files in the working directory and stores the MD5 values in the job BOM

Data source: Rational Build Forge database

Report format: Table

Display options: Select **Flatten report output** to expand the tree hierarchy and display all report results in a list.

To search the job BOM for an MD5 value:

1. Open the **Reports** tab.
2. Click the **Queries** module.
3. At **MD5 search value** , enter the MD5 value that you want to find.
4. Click the Quick Start  icon for the MD5 files report.

   The MD5 files report is displayed as a table of results.

Creating reports with Quick Report

Create your own report designs using report types provided by Quick Report. Quick Report is a licensed option of Rational Build Forge.

In Quick Report, you create, save, edit, and run report designs. Save reports to a public directory to share them with other users.

Report types in Quick Report use data in the Build Forge database to create reports. Different report types include different report fields and have names that describe their function, for example, Capacity and Build.

Document # SC23-7681-08

When creating reports, choose from several report formats, such as tables or charts, and use grouping and sorting functions to control report presentation.

You can optionally run and view report results in the job BOM. See the reference information for ".report" on page 135.

To get started, click the **Reports** tab to display module choices for reporting; then select **Quick Report**.

Report group permissions for Quick Report

You store reports in a public or private repository to control who can run, modify, save, and delete reports created in Quick Report.

Private reports can be accessed only by the Rational Build Forge user who creates and saves the report.

Public reports can be accessed by any user who belongs to an access group with the appropriate permissions. By default, users in the Build Engineer group have read/write/edit access to reports you save as public reports.

To grant access, you assign one or more of the following reports permissions to a user's access group.

Read Public Reports	Permission to run reports saved to the public report repository in Quick Report.
Save, Edit, or Delete Reports	Permission to save, or edit, or delete reports saved to the public repository in Quick Report.

The following table lists the permissions of the Reports permission group and identifies the default permissions assigned to access groups.

To manage permissions, in the Console UI, select **Administration > Permissions** .

	Build Engineer	Developer	Guest	Operator	Security	System Manager
Read Public Reports	Yes	Yes			Yes	Yes
Save Public Reports	Yes					
Edit Public Reports	Yes					
Delete Public Reports	Yes					

Report type reference for Quick Report

Report types have functional names to describe their content, for example, Capacity and Build. For each report type, this topic describes the report purpose, the report fields, and report examples.

Analytic **Description**: Use the analytic report type to report on job performance at the step level using step duration and step execution order.
Report Examples:

Document # SC23-7681-08

- Create a table report to show step duration and step sequence for each step included in a job. Group the jobs by project name
- Create a line chart report to show the step duration for each build tag. Group the build tags by step name.

Field Descriptions:

Field Name	Description
Build Tag	The job tag is a unique identifier based on the project tag format.
Project Name	The user-assigned project name.
Step Duration	The total step run time in seconds.
Step Name	The user-assigned step name.
Step Sequence	A number that identifies the step run order.

Build

Description: Use the build report type to report on job performance at the project level.
Report Examples:

- Create a table report to show the build results, start time, and duration for each build tag. Group the build tags by project.
- Create a bar chart report to show the build count for each project. Group by project.

Field Descriptions:

Field Name	Description
Build Count (aggregate field)	The total job count, including completed and failed jobs.
Build Duration	The total job run time in seconds.
Build Result	The job result: passed, passed with warnings, or failed.
Build Start Time	The job start time.
Build State	The build state: running, complete, archived, or locked.
Build Tag	The job tag is a unique identifier based on the project tag format.
Project Name	The user-assigned project name.
Selector Name	The user-assigned selector name.
User Login	The Build Forge login or user name of the user who started the job.
User Name	The name of the user who started the job.

Capacity

Description: Use the capacity report type to report on job performance by project.
Report Examples:

- Create a table report to show the build start, build duration, build average duration, and build results for each build tag. Group the build tags by project.

Document # SC23-7681-08

- Create a chart report to show the average build time for each project. Group the projects by selector.

Field Descriptions:

Field Name	Description
Build Average Duration (aggregate field)	The average job run time based on the total number of jobs, both completed and failed.
Build Duration	The total job run time in seconds.
Build Result	The job result: passed, passed with warnings, or failed.
Build Start Time	The job start time.
Build Tag	The job tag is a unique identifier based on the project tag format.
Last Build Duration (aggregate field)	The run time for the last job. The total time (in seconds) to complete the last job.
Project Name	The user-assigned project name.
Selector Name	The user-assigned selector name.

Project

Description: Use the project report to report on server, environment, and step usage by project and to report on step performance by project.

Report Examples:

- Create a table report to show the project, class name, and project environment. Group projects by server.
- Create a table report to show the step, step result, step sequence, step environment, and server. Group steps by project and sort by step sequence.
- Create a bar chart report to show the step count for each step. Group the steps by step result.

Field Descriptions:

Field Name	Description
Class Name	The user-assigned class for the project, for example, production or test.
Failed Step Count (aggregate field)	The number of steps that failed for the group field that you select; for example, project or server or other field name.
Passed Step Count (aggregate field)	The number of steps that passed for the group field that you select; for example, project or server or other field name.
Project Environment Name	The name of the project environment used to define project environment variables.
Project Name	The user-assigned project name.

Document # SC23-7681-08

Field Name	Description
Server Name	The user-assigned server name.
Step Count (aggregate field)	The total number of steps for the group field that you select, for example, project or server.
Step Environment Name	The name of the step environment used to define environment variables for the step.
Step Name	The user-assigned step name.
Step Result	The step result: passed, passed with warnings, or failed.
Step Sequence	A number that identifies the step run order.

Step Metrics

Description: Use the step metrics report to report on step success and failure statistics by project.
Report Examples:

- Create a table report to show the step name, step count, and the percentage of steps that passed and failed. Group the steps by project.
- Create a line chart report to show the step duration by build tag. Group build tags by step name.

Field Descriptions:

Field Name	Description
Build Tag	The job tag is a unique identifier based on the project tag format.
Percent Steps Failed (aggregate field)	The percentage of failed steps out of the total step count. The total step count is for the group field you select, for example, projects.
Percent Steps Passed (aggregate field)	The percentage of passed steps out of the total step count. The total step count is for the group field you select, for example, projects.
Project Name	The user-assigned project name.
Server Name	The user-assigned server name.
Step Average Duration (aggregate field)	The average step run time based on the total number of steps, both completed and failed.
Step Count (aggregate field)	The total step count, including completed and failed steps. The total step count is for the group field you select, for example, projects.
Step Duration	The total step run time in seconds.
Step Name	The user-assigned step name.
Step Result	The step result: passed, passed with warnings, or failed.

Document # SC23-7681-08

Field Name	Description
Step Sequence	A number that identifies the step run order.
Step Start Time	The step start time.

Quality

Description: Use the quality report to report on job success and failure statistics by project.
Report Examples:

- Create a table report to show the build results by build tag. Group the build tags by project.
- Create a table report to show the percentage of all builds that passed and failed by project. Group by project.
- Create a bar chart report to show the build count by project. Group the projects by build result.

Field Descriptions:

Field Name	Description
Build Count (aggregate field)	The total job count, including completed and failed jobs.
Build Result	The job result or status: passed, passed with warnings, or failed.
Build Start Time	The job start time.
Build Tag	The job tag is a unique identifier based on the project tag format.
Percent Builds Failed (aggregate field)	The percentage of failed builds out of the total build count. The total build count is for the group field you select, for example, projects.
Percent Builds Passed (aggregate field)	The percentage of passed builds out of the total build count. The total build count is for the group field you select, for example, projects.
Project Name	The user-assigned project name.

Resource

Description: Use the resource report to report on step and job performance by projects and servers.
Report Examples:

- Create a table report to show step run times by server. Select the step sequence, step name, server name, step start time, and step duration. Sort by step sequence and start time.
- Create a table report to show the job run times by server. Select the build tag, server name, build start time, build duration, and build result. Group the build tags by project and sort by build start time.

Document # SC23-7681-08

Field Descriptions:

Field Name	Description
Build Duration	The total job run time in seconds.
Build Result	The job result: passed, passed with warnings, or failed.
Build Start Time	The job start time.
Build Tag	The job tag is a unique identifier based on the project tag format.
Project Name	The user-assigned project name.
Selector Name	The user-assigned selector name.
Server Name	The user-assigned server name.
Step Duration	The total step run time in seconds.
Step Sequence	A number that identifies the step run order.
Step Start Time	The step start time.

BOM

Description:
Use the BOM report to create a report using information in the job BOM. The BOM report can run against any of the following data sets:

- all projects in the database (the default)
- a single project
- one or more builds in a single project

The report fields that you can select vary by data set. Report fields might include step manifest properties, output from the .scan command, output logged by adaptors, and user-defined columns specified using the .bom command.
Report Examples:

- Create a table report to show the output of the .scan command across multiple projects. Select the build tag, BOM data, BOM path, and BOM type fields. Group the build tags by project.
- Create a table report show the number of times that a filter action is invoked across multiple projects. Select the step name, filter event count type, and filter count. Group the steps by project.

Field Descriptions:

Field Name	Description
BOM Data	If you include the .scan command in a project, for the files scanned, the BOM Data field displays MD5 values. .
BOM Path	If you include the .scan command in a project, for the files scanned, the BOM Path field displays the file path.

Document # SC23-7681-08

Field Name	Description
BOM Type	If you include the .scan command in a project, for the files scanned, the BOM Type field indicates whether the path is D (a directory), F (a file), or S (a symbolic link).
Build Tag	The job tag is a unique identifier based on the project tag format.
Filter Event Count	The number of times that the filter action was invoked by the filter as a result of finding a pattern match in step output. .
Filter Event Count Type	The filter action that was invoked by the filter when a pattern match in step output was found.
Project Name	The user-assigned project name.
Step Name	The user-assigned step name.
Selector Name	The user-assigned selector name.
Server Name	The user-assigned server name.
Step Duration	The total step run time in seconds.
Step Result	The step result: passed, passed with warnings, or failed.

Report format and presentation reference for Quick Report

Quick Report provides several common report formats: tables, bar charts, line charts, and pie charts.

Note

The sample reports provided with Quick Report do not contain any data. To see report data, you must first create projects and run jobs in the Management Console.

Table report format

Samples

To see a sample table report, in Quick Report, select the **SampleAnalytic-StepDuration** report.

To see presentation options, select the **Edit** icon for the SampleAnalytic-StepDuration report.

Requirements

For tables, observe the following requirements:

- Select at least one report field.
- If you select an aggregate report field, you must select a grouping field. See Selection requirements for report formats and aggregate report fields.

Report fields

In the Report Field, select one or more fields to be table columns in the report. In the selection list, order matters. The first field will be the first table column and the last field will be the last table column. The first table column is also used to group report results.

Document # SC23-7681-08

To experiment with report fields, select the **Edit** icon for the Sample-Analytic-StepDuration report. In the selection list, change the field order, save your selections, and run the report to see the change in the table column order.

Group fields

In the Group field, optionally select a field to group table rows. The group you select is added as a tree control to the first table column, which is used to group report results. You can select multiple group fields, and you can select a group field that is not displayed as a table column.

To experiment with grouping, select the Edit icon for the SampleAnalytic-StepDuration report. Select the Project Name as the group field, save your selections, and run the report. A tree node for the Project Name is added to the Build Tag column.

Sort fields

In the Sort field, optionally select the table columns with data to be sorted. Use arrows to specify sort direction. The up arrow sorts data in ascending order (lowest to highest), and the down arrow

sorts data in descending order (highest to lowest).

To experiment with sorting, select the Edit icon for the SampleAnalytic-StepDuration report. Select the Project Name as the sort field, use arrows to specify sort direction, save your selections, and run the report.

Bar chart report format

Samples

To see a sample bar chart, in Quick Report, select the **SampleCapacity-RunTimeByProject** report.

To see presentation options, select the **Edit** icon for the SampleCapacity-RunTimeByProject report.

Vertical chart element (y-axis)

Select one field to be the data value (y-axis) that you want to compare for a set of data elements (x-axis). The field name and the data value units are displayed on the chart's vertical or y-axis. Only one field can be selected as the vertical chart element.

Horizontal chart element (x-axis)

Select one field to be the data element set (x-axis) that you want to compare using the data value (y-axis). The field name is displayed as the x-axis label.

Data values are displayed on bar columns and values equal the bar column height. Only one field can be selected as the horizontal chart element.

Group fields

You can optionally group data elements on the chart's horizontal or x-axis by selecting a group field.

Requirements

For charts, observe the following requirements:

Document # SC23-7681-08

- Select both an x-axis and a y-axis report field.
- Select a different report field for the x-axis and the y-axis.
- Select a grouping field for the x-axis data.
- If you select an aggregate report field, you must select a grouping field. See Selection requirements for report formats and aggregate report fields.

Examples:
To experiment with bar charts, try creating the following reports:

- For the Step report type, for each Step Name (x-axis), compare the Percent Steps Failed (y-axis) and group results by the Build Tag.
- For the Capacity report type, for each Project (x-axis), compare the Last Build Duration (y-axis) and group results by Project Name.
- For the Resource report type, compare the Build Duration Times (y-axis) for each Build Tag (x-axis) and group the Build Tags by Project Name.

Line chart report format

Description
A line chart shows a progression of data over time or for a sequence of events. The default grouping field is the x-axis report field. If you select multiple grouping fields, multiple lines are displayed in the report.
Samples
To see a sample line chart, in Quick Report, select the **SampleResource-DurationOverTime** report.
To see presentation options, select the **Edit** icon for the SampleResource-DurationOverTime report.
Vertical (y-axis) chart element
Select one field to be the data value (y-axis) that you want to compare for a set of data elements (x-axis). The field name and data value units are displayed on the chart's y-axis. Only one field can be selected as the vertical chart element.
Horizontal (x-axis) chart element
Select one field to be the data element set (x-axis) that you want to compare using the data value (y-axis). The field name is displayed as the label for the chart's x-axis.
Data values for x-axis data elements are displayed beside a dot equaling their value on the y-axis. A continuous line connects the dots to form the line chart. Only one field can be selected as the horizontal chart element.
Group fields
You can optionally group data elements on the chart's horizontal or x-axis by selecting a group field. The default grouping field

Document # SC23-7681-08

is the x-axis report field. If you select multiple grouping fields, multiple lines are displayed in the report.

Requirements

For charts, observe the following guidelines:

- Select both an x-axis and a y-axis report field.
- Select a different report field for the x-axis and the y-axis.
- Select a grouping field for the x-axis data.
- If you select an aggregate report field, you must select a grouping field. See Selection requirements for report formats and aggregate report fields.

Examples

To experiment with line charts, try creating the following reports:

- For the Step report type, for each Build Tag (x-axis), show the Step Duration (y-axis) and group results by the Step Name.
- For the Build report, for each Build Start Time (x-axis), show the Duration (y-axis) and group results by Project Name.
- For the Capacity report type, for each Build Tag (x-axis), show the Build Duration (y-axis) times and group results by Server Name.

Pie chart report format

Y-series element

Select one field to be the data value of the pie chart. The wedge size represents the data value. The field name and data value units are displayed on the pie chart. Only one field can be selected as the y-series element.

X-series element

Select one field to be the data element set (x-series) that you want to evaluate using the data value (y-series). The number of wedges represents the number of data elements. Only one field can be selected as the x-series element.

Group fields

A group is required for the pie chart. The x-series element is used as the default group. Selecting a different group generates another pie chart for the data value (y-series).

Requirements

For charts, observe the following guidelines:

- Select both an x-series and a y-series report field.
- Select a different report field for the x-series and the y-series.
- Select a grouping field for the x-axis data.

Document # SC23-7681-08

- If you select an aggregate report field, you must select a grouping field. See Selection requirements for report formats and aggregate report fields.

Examples

To experiment with pie charts, try creating the following reports:

- For the Project report type, select the Failed Step Count (y-series) and the Step Name as the (x-series).
- For the Capacity report type, select the Build Average Duration (y-series) and the Project Name (x-series).
- For the Analytic report type, select the Step Duration (y-series) and the Step Name (x-series) and group by Step Name.

Selection requirements for report formats and aggregate report fields

To produce meaningful reports, observe some basic requirement for tables and charts and aggregate report fields.

Aggregate report fields

Requirements

If you include an aggregate report field in a table or chart, you must include a grouping field.

Definition

An aggregate report field contains data that is derived from one or more original data fields in the Build Forge database. The data for aggregate report fields is not stored in the database. The following report fields are aggregate fields.

Analytic: Last Build Duration	Project: Failed Step Count	Project: Passed Step Count
Project: Step Count	Step Metrics: Percent Steps Failed	Step Metrics: Percent Steps Passed
Step Metrics: Step Average Duration	Step Metrics: Step Count	Quality: Build Count
Quality: Percent Builds Failed	Quality: Percent Builds Passed	Build: Build Count

Table report format

Requirements

For tables, observe the following requirements:

- Select at least one report field.
- If you select an aggregate report field, you must select a group field.

Document # SC23-7681-08

Chart report formats

Requirements
For bar charts, line charts, and pie charts observe the following requirements:

- Select both an x-series and a y-series report field.
- Select a different report field for the x-series and the y-series.
- Select a grouping field for the x-axis data.
- If you select an aggregate report field, you must select a group field.

Sample reports reference

Sample reports are examples of reports that you can create with the provided report types.

Note

Sample reports use data in the Rational Build Forge database to create report output. Results display only if you have already created projects and run jobs.

To display a list of sample reports, click **Quick Report** .

To see a sample report's fields and format options, select the report's **Edit** icon.

To copy a sample report, select the report's **Edit** icon and click **Copy Report** .

To run a sample report, select the report name.

Sample report definitions are provided in the following table.

Sample Report Name	Description
SampleAnalytic-StepDuration	Description: Creates a table that reports on step details for builds Report type: Analytic Report format: Table Report fields: Build Tag, Project Name, Step Name, Step Duration Report options: No grouping or sorting options
SampleBuild-BuildsByState	Description: Creates a table that lists builds and the build state by project Report type: Build Report format: Table

Document # SC23-7681-08

Sample Report Name	Description
	Report fields: Build Tag, Project Name, Build Count, Build State Report options: Group by Project Name
SampleCapacity-RuntimeByProject	Description: Creates a bar chart that shows the average build duration for jobs by project Report type: Capacity Report format: Bar chart Report fields: Build Average Duration (y-axis) and Project Name (x-axis) Report options: Group by Project Name
SampleProject-TotalsByProject	Description: Creates a table that reports on the number of steps passed and failed by project Report type: Project Report format: Table Report fields: Project Name, Passed Step Count, Failed Step Count, Step Count Report options: Group by Project Name
SampleQuality-PercentSuccess	Description: Creates a table that reports on the percentage of jobs passed and failed by project Report type: Quality Report format: Table Report fields: Project Name, Build Count, Percentage Builds Failed, Percentage Builds Passed Report options: Group by Project Name
SampleResource-DurationOverTime	Description: Creates a chart that shows build start times and build durations for each build Report type: Resource Report format: Table Report fields: Build Duration (y-axis) and Build Start Time (x-axis) Report options: Ascending sort sequence

Document # SC23-7681-08

Creating a report using a provided report type

Create your own report by simply selecting the report type, report format, and report fields that you want to use.

To create a report using a provided report type:

1. Select the **Reports** tab; then select **Quick Report** .
2. At **Report Name** , enter a unique name. A report name is required. The report name is used to save the report in the database and must be unique.
3. At **Report Title** , enter a descriptive title for the report. A report title is required. The title is displayed at the top of the report.
4. At **Visibility** , select Public or Private.

   Private reports cannot be shared. Public reports can be shared with users who have required access. For details, see Report group permissions for Quick Report.
5. At **Report Format** , select a table or one of the chart formats: bar, line, or pie. For details about reports formats, see Report format and presentation reference for Quick Report.
6. At **Report Type** , choose a report type.

   The report type determines the content that you include in your report. For details and examples, see Report type reference for Quick Report.

   - Select the report fields to display in your table or chart report.
   - Select the group and sort options to control report presentation.

   ## Important

   If you select the BOM report type, the Project box is displayed. To create a BOM report for a specific project, first select the project to display the complete list of report fields, then select the fields to include in the report.
7. (BOM report type only) At **Projects** , select one of the following options:

   - Select All Projects to display the standard BOM report fields only. (All Projects is the default setting.)

     The report output contains BOM information for all the projects in the database.
   - Select one project to display its project-specific BOM fields in addition to the standard BOM report fields.

     The report output contains BOM information for a single project only.
8. Click **Save Report** to save your selections. The report is displayed in the report list.

Document # SC23-7681-08

Adding report output to the job BOM

Optionally use the .report command to add report results to the job BOM.

To add report output to the job BOM:

1. Use the Quick Report tool to create a report. See Creating a report using a provided report type.

   ### Note

   This feature is not supported for the BOM report type or for private reports.

2. Use the .report command to add the report to the job BOM. See the reference information for ".report" on page 135.

Modifying and managing reports in Quick Report

In Quick Report, you can run a report, view report results, and edit your report design.

Reports can be edited to change the report design. Filters are part of the report definition and can be added to any report to filter output.

Running reports

To run any report you create in Quick Report, click the report name.

Report results are displayed in Quick Report results view.

To return to the reports list, click the Back arrow of your Web browser.

Editing reports

To edit any report you create in Quick Report, click the **Edit** icon beside the report name.

The report selections are displayed in the Report details. After you are done making changes, click **Save Report** to save your changes.

Copying reports

Copying a sample report or other report copies its report fields and formatting options to a new report design and assigns the report a unique name.

A unique report name is created by adding Copy to the report name, for example, <*Report_Name*> Copy.

If you make multiple copies of a report, numbers are appended using the following syntax: <*Report_Name*> Copy <*Copy_Number*>. For example, <*Report_Name*> Copy 2.

To copy a report:

Document # SC23-7681-08

1. Click the **Edit** icon for the report to be copied.
2. Click **Copy Report** .

   The copied report is displayed in the report list.

Creating a filter for report output

You can create a report filter to control what information is displayed in the report output.

Before creating a filter, review the following requirements and restrictions:

- Create a report first; filtering uses report definitions to provide filter options.
- Report filters are saved as part of the report definition and apply to a single report only.
- After you create a report filter, it is applied to every run of that report until you change or delete the filter.
- For the BOM report type, you can filter by project, filter by build, and specify filter criteria for specific report fields.
- For report types other than BOM, you must specify filter criteria for specific report fields.

To create a report filter:

1. In the reports list, select a report and select the **Edit** icon for the report.
2. (Required for the BOM report type only) At **Projects** , select one of the following options:
   - Select All Projects to display the standard BOM report fields only. (All Projects is the default setting.)

     The report output contains BOM information for all the projects in the database.
   - Select one project to display its project-specific BOM fields in addition to the standard BOM report fields.

     The report output contains BOM information for a single project only.
3. Select the **Filters** tab.
4. (Optional for the BOM report type only) Click **Show Build Filter** , and select from the following options to specify what build information to include in the report:
   - To include data for all current and future builds, make no selections.
   - To include data for all current builds only, click **All current builds only** .
   - To include data for all current and future builds for selected builds only, select the build tags in the list.

Document # SC23-7681-08

5. (Required for all other report types) Click **Add Filter** to select a report field from the report type selected on the **Report Details** tab to use as a filter.

   To create the filter expression:

   a. At **Filter Field** , select the report field to use to filter report data.

   b. At **Filter Operator** , select the relational operator.

   c. At **Filter Value** , enter the report field value.

6. Click **Save Report** to save your report filter selections.

Troubleshooting common problems for Quick Report

If you encounter problems when using Rational Software Analyzer, review the information in this topic to see if there is an acceptable workaround or solution.

Port Conflict	Quick Report uses the application web server that you specified during installation to display reports. If you experience a port conflict, you must configure an unassigned port for the application web server.

Document # SC23-7681-08

Working with utilities

This topic describes how to set up and run command-line utilities provided by Rational Build Forge.

Accessing and running command-line utilities

The command-line utilities are located in the Build Forge installation directory. On Windows, <*bf-install*> or on UNIX/Linux, <*bf-install*>/Platform.

You must properly configure your environment (or the system-level environment) for engine-level shell commands to work. For example, on Oracle and Unix, ORACLE_HOME, TNS_ADMIN, LD_LIBRARY_PATH, must be manually set, or they will not execute.

When you use command-line utilities such as bfexport or bfimport, the command needs to be able to find the buildforge.conf file to access the database, so you must execute the command either from your installation directory, or set the environment variable BF_CONFIG_FILE to the full path of the buildforge.conf file.

Exporting projects

You can export a project and other Build Forge objects to an XML file by using the bfexport command or the .export dot command.

An exported project is stored in an XML file. The exported project can be imported back into Build Forge.

Example: you can add a step at the end of a project that runs bfexport to save the project configuration data. You can use it as a backup for the project definition. You could also use it to move the project to another Build Forge installation.

bfexport reference

Use the bfexport command to export project data to a named XML file or to send project data to the display terminal (stdout) for viewing. An export file contains project configuration data for a single project or project snapshot.

Syntax

```
bfexport

bfexport [-l]

bfexport [-l] <project_name>

bfexport [-c "<comment>"] [-f <file_name>] [-g] [-s] [-C] [-L ] [-n ]
<project_name> | <project_name> <snapshot_name> | <project_id>
```

Usage

To complete common project export tasks, use these command options:

Document # SC23-7681-08

- To display command syntax, use bfexport with no options.
- To list the project names and project IDs that are stored in the Build Forge database, use bfexport -l.

  Snapshot names are appended to the project name in the command output: <*project_ID*>: <*project_name*> <*snapshot_name*>
- To send project data to an XML file, bfexport -f <*file_name*>. You must specify the -f <*file_name*> option to generate a file that can be used to import project data.
- On the z/Linux platform, you must run the command as bfexport.pl. On all other platforms, the command does not require an extension.

Prerequisites and restrictions

Find the bfexport utility in your Build Forge installation directory.

Server authorization passwords for servers are not included in the export file; after import, you must manually enter server authorization passwords in the UI.

The bfexport command must be able to find the buildforge.conf file and access the Build Forge database. Run bfexport from the directory where buildforge.conf is located. On Windows, <*bf-install*>, or on UNIX/Linux, <*bf-install*>/Platform.

Examples

To write output to a file, use the -f <*file_name*> option. In the following example, helloworld is the output file name and the project ID is used instead of the project name.

```
bfexport -c "Saving a copy of project before making changes"
 -f helloworld 675B57CC-8366-11DD-B2E0-043C04E44E1A
```

To export the default project snapshot only, use the <*project_name*>.

```
bfexport -f helloworld test_project
```

To export one snapshot of a project, use the <*project_name*> <*snapshot_name*>.

```
bfexport -f helloworld test_project snapshot_1
```

If the parent project snapshot is not the default project, you must specify the <*project_name*> followed by the parent keyword to export the parent project snapshot.

```
bfexport -f helloworld test_project parent
```


Document # SC23-7681-08

Option descriptions

Option	Description
<*project_name*>	The name of the project to export. The project name or the project ID is required. If the project name contains spaces, you must quote the name. Specify the project name after the command options.
<*snapshot_name*>	The name of the project snapshot to export; the project name is required, as shown in the following syntax: `<project_name> <snapshot_name>` Specify the project name and snapshot name after the command options. If the project or snapshot name contains spaces, you must quote the name.
<*project_id*>	The identifier for the project to export. The project ID is a UUID. The project ID or project name is required. Specify the project ID after the command options.
-f <*file_name*>	An XML file name for bfexport output. If you do not provide a path name, the current working directory is used. If the file name contains spaces, you must quote the name. If you do not provide a file name, bfexport output is sent to stdout. **Note** Use stdout for viewing only. Do not redirect stdout to a file; the resulting file includes logging messages and cannot be used as an import file for the bfimport command or the UI import utility.
parent	A keyword that is required to export a parent project snapshot if the parent is not the default project snapshot. Specify the parent keyword after the project snapshot name: `bfexport -f helloworld test_project parent`
-l	Lists the projects in the database by name and project ID. You cannot use the -l option with other options.
- c "<*comment*>"	Includes a comment. You must quote the comment (for example, "my project version 50"). The comment is added to the <buildforge> XML element.
-g	Saves to the XML file the users who are members of the access groups designated to receive notifications. Users and their properties are listed in the <user> XML element.
-s	Saves to the XML file the servers defined in the Management Console. Servers and their properties are listed in the <server> XML element, along with any associated <auth> and <collector> information.

Document # SC23-7681-08

Option	Description
-L	Saves to the XML file the LDAP domain controllers defined in the UI. LDAP domain controllers and their properties are listed in the <ldap> XML element.
-n	Saves to the XML file the notification templates assigned to the project and steps. The notification templates and their properties are listed in the <mail-template> XML element.
-C	Saves to the XML file the collectors assigned to the servers for the project. Collectors and their properties are listed in the <collector> XML element.

Using .export

You can use .export to export a project from a step in the project.

The .export command does not give you the option of exporting any other object data. To export other Build Forge objects you must use the bfexport command.

See ".export" on page 128.

Importing projects

You can import a previously exported project and other Build Forge objects by using the bfimport command or the Import facility in the console.

A range of options lets you select the objects to install.

You have options for how to apply access groups to the imported objects. They are set through the **Import with Secure Access** system setting.

Importing projects and other objects using the Import utility

You can use the UI Import utility to import the object definitions for projects and other objects that have been saved to an export file. The Import utility allows you to select objects to import from the export file.

1. Select **Administration → Import** .
2. Click **Browse** to locate the export XML file for the project.

   You must create the export file by using the bfexport command or the .export dot command.
3. Select the project and the other objects to import to the UI from the export XML file.

Note

If you import server objects, you must manually enter their server authorization passwords in the UI after importing them. The bfexport and .export commands do not save server authorization passwords to the export file.

Document # SC23-7681-08

4. Select **Replace Entities** or **Rename Entities** (the default option) to specify whether the import utility replaces or overwrites objects with the same name or renames them.

Important

To understand how rename and replace work, see Renaming and replacing objects on import.

bfimport reference

Use the bfimport utility to import definitions for projects and other objects to the UI that were previously exported to a XML file. You can also use the Import utility to import selected objects from the XML file.

This topic describes the syntax of the bfimport command and provides usage details.

Syntax

```
bfimport

bfimport [-L] <file_name>

bfimport [-p -I -s -S -e -c -C -u -T -f -d -r ] <file_name.xml>


bfimport [-L | [-p -I -s -S -e -c -C -u -T -f -d -r ]] <file_name.xml>
```

Restrictions and considerations

Server authorization passwords for servers are not included in the export file; you must manually enter server authorization passwords.

By default, on import if an object exists with the same name as an imported object, the object being imported is renamed to prevent the database object from being overwritten. Alternatively, you can choose to replace objects with the -r option if an object with the same name exists.

By default, objects are renamed by bfimport and the following naming convention is used:

<*object_name*>_IMPORT_<*number*>

For rename, snapshot objects lose their snapshot name and are imported as a new base or parent-level snapshot, even if the snapshot object is a child of a parent snapshot.

For details about rename, see Renaming and replacing objects on import.

To replace objects, you must specify the -r option. The replace option overwrites existing objects. For 7.0.2 and earlier export files, snapshot objects are not replaced, instead, they are renamed using the <*object_name*>_IMPORT_<*number*> convention. For 7.1 export files, snapshot objects are replaced.

Prerequisites

An export XML file that was created by the bfexport command or the .export dot command.

Document # SC23-7681-08

Find the bfimport utility in your Build Forge installation directory.

The bfimport command must be able to find the buildforge.conf file and access the Build Forge database. Run bfimport from the directory where buildforge.conf is located. On Windows, <*bf-install*> or on UNIX/Linux, <*bf-install*>/Platform.

Usage

To complete common import tasks, use these command options:

- To display command syntax, use bfimport with no options.
- To display a summary list of the Build Forge objects in the XML file and their names, use bfimport -L <*file_name.xml*>
- On the zLinux platform, you must run the command as bfexport.pl. For all other platforms, the command does not require an extension.
- If you specify no options, no objects are imported. You must specify options to import individual objects.
- If you specify options for objects that do not exist, the import utility skips the objects that are not in the XML file and imports the objects that are in the file.

Examples

To list the Build Forge objects in the XML file, specify the -L option and the XML file name only. The following example displays partial command output.

```
                                c:\Program Files\IBM\Build Forge>bfimport -L
samples\projects\basic.xml
10/07/2008 5:31:55 PM: Import: 7624: CRRBF20081I: Importing export
file from a 7.0.10025 version console.
Project : [Basic Sample]
 Tag Variable : [MAJ]
 Tag Variable : [MIN]
 Step : [Checkout Source]
 Step : [Update Applet Version]
 Step : [Create Baseline]
Environment : [Basic Environment]
Class : [Production]
Filter : []
Selector : [Web Server]
Selector : [Local Server]
```

To import all the objects in an XML file, specify the options for the objects to be imported, as shown in the following example. Objects are renamed on import. A success statement displays if the import is successful.

```
c:\Program Files\IBM\Build Forge>bfimport p -I -s -S -e -c -C -u -T -d -f
"samples\projects\basic.xml"
```


Document # SC23-7681-08

Option descriptions

Option	Description
<*file_name.xml*>	The name of the export XML file that contains the Build Forge objects to be imported. The XML file must be created by using the bfexport command or the .export dot command. The XML file name is required and you must provide the path name if the XML file is not in the current directory, the directory from which you execute the bfexport command. If the file name contains spaces, you must quote the name.
-L	Lists the objects in the export XML file and their object names. Use this option by itself; do not specify it with any other bfimport options. Output from the -L option can be sent to stdout or redirected to an XML or text file.
-p	Imports project configuration data from the XML file. Project configuration data includes step and project definition data, including tag variables. On rename, a project name is imported to the UI as <*project_name*>_IMPORT_<*number*>.
-I	Imports chained projects or libraries that are referenced at the project or step level. On rename, a chained project or library is imported to the UI as <*project_or_library_name*>_IMPORT_<*number*> in the UI.
-S	Imports the selector objects that are defined in the UI. On rename, a selector is imported to the UI as <*selector_name*>_IMPORT_<*number*>.
-s	Imports the server objects that are defined in the Management Console, if the -s option is specified for bfexport. On rename, a server is imported to the UI as <*server_name*>_IMPORT_<*number*>. Server authorization passwords for servers are not included in the export XML file; you must manually enter server authorization passwords.
-e	Imports environments and their variables that are referenced at the project or step level. On rename, an environment is imported as <*environment_name*>_IMPORT_<*number*>.
-c	Imports classes that are referenced by projects. On rename, a class is imported as <*class_name*>_IMPORT_<*number*>.

Document # SC23-7681-08

Option	Description
-C	Imports collectors that are assigned to the servers for the project, if the -C option is specified for bfexport. On rename, a collector is imported as <*collector_name*>_IMPORT_<*number*>.
-u	Imports users who are members of the access groups designated to receive e-mail notifications, if the -g option is specified for bfexport. On rename, information for users is imported as <*users*>_IMPORT_<*number*>
-T	Imports the user-created notification templates that are assigned to projects and steps, if the -n option is specified for bfexport. On rename, a notification template is imported as <*template_name*>_IMPORT_<*number*>.
-f	Imports log filters that are assigned to project steps, if the -n option is specified for bfexport. On rename, the log filters are imported as <*filter_name*>_IMPORT_<*number*>.
-d	Imports the LDAP domain controllers defined in the UI, if the -L option is specified for bfexport. On rename, the log filters are imported as <*LDAP_domain_controller*>_IMPORT_<*number*>.
-r	Replaces the imported objects instead of renaming them. By default, imported objects are renamed and the following naming convention is used: `<object_name>_IMPORT_<number>` For 7.1 objects, if you specify the replace option, the bfimport command overwrites objects in the UI for 7.1 objects. For 7.0.2 and earlier objects, snapshot objects are not replaced. They are renamed by using the following naming convention: `<object_name>_IMPORT_<number>` For details, see Renaming and replacing objects on import.

Applying access groups to imported objects

You can import a project or other Build Forge object that was previously exported. Perform an import from the UI (**Administration** > **Import**) or use the bfimport command.

To apply access groups to imported Build Forge objects, the import function uses the **Import with Secure Access** system setting (**Administration** > **System**), as follows:

Document # SC23-7681-08

- If **Import with Secure Access** is set to Yes (the default), the import function ignores access groups assigned to objects in the import file.

  Instead, objects in the import file are assigned the access group specified by the **Import Default Secure Access Group** setting. Build Engineer is the default.

- If **Import with Secure Access** is set to No, the import function accepts the access groups that are already assigned to objects in the import file.

  For objects without access groups, the group specified by the **Import Insecure Default Access Group** setting is used. Developer is the default access group.

  If an access group for **Import Insecure Default Access Group** is not specified, the import function assigns the most recently created access group to objects without access groups. If no access groups have been created, the Default access group is assigned.

Renaming and replacing objects on import

The bfimport command and the Import utility rename an imported object if an object with the same name already exists in the database. Renaming objects on import is the default behavior.

To change this behavior and replace existing objects on import, you must specify the -r option for bfimport or select the Replace Entities option in the UI.

The following topics describe the naming conventions that the bfimport command and Import utility use when renaming and replacing imported objects.

Snapshot objects (projects, selectors, and environments) retain their snapshot name, if they have one, or are assigned a default snapshot name on import.

Renaming objects in 7.1 or earlier export files

For objects in 7.1 and earlier export files, the bfimport command and the Import utility rename objects in the UI by using the following naming conventions.

Object status	**UI object name**	**UI snapshot name (applies to snapshot objects only)**
New, not in database	<*new_object_name*>	<*snapshot_name*>
Exists in database	<*existing_object_name*>_IMPORT_<*n*>	<*snapshot_name*> | Base Snapshot

Snapshot objects are imported as a new parent-level snapshot, even if it was a child of a parent snapshot. Only projects, selectors, and environments can be snapshot objects. If a snapshot name exists, it is retained. Otherwise, the default Base Snapshot name is assigned, as shown in the table.

Document # SC23-7681-08

Replacing objects in pre-7.1 export files

For objects in pre-7.1 export files, the bfimport command and the Import utility replace objects in the UI by using the following naming conventions. All objects are replaced except existing snapshot objects. Existing snapshot objects are renamed.

Object status	**UI object name**	**UI snapshot name (applies to snapshot objects only)**
New, not in database	<*new_object_name*>	Base Snapshot
Exists in database, non-snapshot object	<*existing_object_name*	n/a
Exists in database, snapshot object	<*existing_object_name*>_IMPORT_<*n*>	Base Snapshot

For snapshot objects, the snapshot object is imported as a new parent-level snapshot, even if it was a child of a parent snapshot. Only projects, selectors, and environments can be snapshot objects. Pre-7.1 export files cannot contain objects with snapshot names, so the default Base Snapshot name is assigned, as shown in the table.

Replacing objects in 7.1 export files

For objects in 7.1 export files, the bfimport command and the Import utility replace objects in the UI by using the following naming conventions.

Object status	**UI object name**	**UI snapshot name (applies to snapshot objects only)**
New, not in database	<*new_object_name*>	<*snapshot_name*> | Base Snapshot
Exists in database	<*new_object_name*>	<*snapshot_name*> | Base Snapshot

For snapshot objects, the snapshot object is imported as a new parent-level snapshot, even if it was a child of a parent snapshot. Only projects, selectors, and environments can be snapshot objects. All 7.1 objects either have a unique snapshot name or use the default Base Snapshot name.

Document # SC23-7681-08

Administration

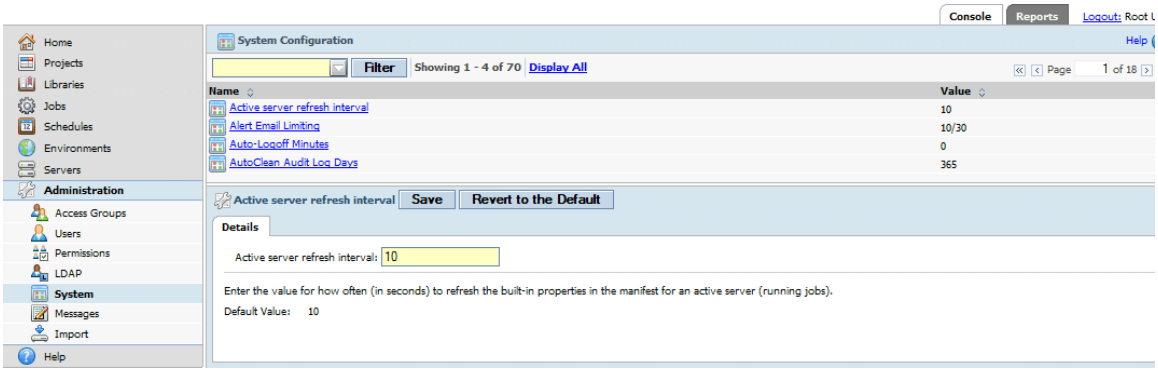
This topic describes administrative operations for the Build Forge® system.

About administration

Use the Administration module to manage configurations and preferences.

Within the Administration module you can work with users, security privileges, notification settings, collectors, and system settings to configure your Management Console.

Administration module, system settings



Access groups

An access group is a collection of users that the system uses to control permissions.

Use the **Administration → Access Groups** page to create new access groups, to add/remove users, and to modify group properties. The page displays a list of existing access groups. Click the name of a group to select it and display its properties in the lower pane.

- To create a new group, click **Add Group** to clear the fields in the lower pane (if needed). Then give the group a **Name** and select an **Owner** group (a group to control access to the new group. To edit a group, a user must be a member of the **Owner** group). If you are using LDAP authentication, fill out the LDAP Group DNs field to tell the system which LDAP groups to map to your group. For example, you might map the Developer group to an LDAP group named SoftwareEngineers, and then assign permissions to the Developer group to provide the type of access you want your software engineers to have.
    - In the LDAP Group DNs field, list the Distinguished Names of all the LDAP groups whose members should receive the Management Console security privileges associated with this access group.
    - You can map multiple LDAP groups to any access group. You can use the asterisk (*) character in this property to give all LDAP users membership in this access group. You can list multiple LDAP groups and separate them with semicolons.

- To delete an access group, select the group, then click **Delete** .

  ### Note

  You cannot delete access groups that are assigned as an **Access** property elsewhere. For example, if you create an access group, then set the **Access** property of a project to use it, you cannot delete the group. You must first edit the project to use another access group.

- To add or remove users, select the group, then click the **Users** tab. The system displays a list of nonmembers on the left and members on the right. Select users and use **Add** and **Remove** to move them from one list to another.

- To nest groups, by add a group as a subgroup of another group. When you do this, all the permissions that apply to the containing group apply to all the users in member groups as well. To make one group a subgroup of another, select the desired parent group, then click the **Subgroups** tab in the lower pane. Select the groups you want to make into subgroups, and click **Add** . You can recursively nest groups, for example, add a parent to a child so that group A contains group B which contains group A. If you do this, the system treats all members of group A as members of group B, and vice versa.

- To manage the permissions for a group, choose the group, then click the **Permissions** tab. You can view the current permissions for the group and add or remove permissions.

Access overview

The system manages users in its database, and allows you to control the privileges of users (through the groups you assign them to). You assign privileges to groups, then make each user a member of appropriate groups.

This is a role-based system—a group represents a role that a user can have in your organization. Roles have privileges. A user's privileges are the sum of the groups the user belongs to. You cannot assign privileges to individual users directly, only to groups.

The system also uses access groups for notification. When you ask the system to send notification messages, the target of the messages must be an access group. See “Setting up notification” on page 83.

Security privileges, or *permissions*, define what a group can do and see. They can serve as a filter of the group's experience of the system. For example, a user who is a member of the Guest group (and no other groups) sees only Projects that have the Guest group assigned as their Access property. That user can only launch projects with Guest access. If the user was also a member of the Developer group, he would see all the projects whose Access properties were either Guest or Developer.

Note

You can use an existing LDAP database for user authentication, instead of the database. When you do this, instead of defining users in the system, you allow some or all of the

Document # SC23-7681-08

users from your LDAP database to access the system. You can also map access groups to LDAP groups. For details on setting up LDAP, see “About LDAP integration” on page 209.

The activities and resources that you can control with access groups are Permissions, Servers, Projects, Steps, and Access Groups.

- To extend access to a resource (Server, Project, or Step) to a particular group, select that resource and change its Access field to that group's name. For example, to give the Developer group access to a particular server named Win234, set the server's Access property to Developer.
- To allow members of one access group to edit another access group, set one group as the Control Group of the other. For example, to allow Group A members to add members to Group B, make Group A the Control Group for Group B.
- To extend a global privilege to a group, use the **Administration** → **Permissions** page to enable a particular permission for that group.

This flexible model allows you to securely give one privilege (such as the ability to run builds) to some types of users, while restricting others (such as the right to edit projects or use certain servers).

Access example: giving a group the ability to run jobs

You can use the security features to extend the ability to run certain jobs to one of your access groups. For example, you might have a group of device driver programmers and you want to allow them to run jobs that are relevant to their work, without cluttering their view of the system with many other jobs, and without allowing them to edit the jobs you create. To create this scenario:

- Create an access group for this role in your organization (for example, DeviceDriverDevs).
- Assign the new access group as the **Access** property of all the projects you want the users to be able to run.
- Make sure the steps of the projects have appropriate **Access** properties also. Any steps that the users do not have access to are skipped when the job runs.
- Assign the permission Execute Builds to the group.
- Create all the users who need to launch these builds to the new DeviceDriverDevs group. You may also need to make administrators of the system members of the group as well. When you change the Access property of the projects, users who are not members of the DeviceDriverDevs group lose the ability to view, execute, or edit the project.

Note that users can be members of many groups, and permissions are cumulative. You could have a group for another project team (for example, PlatformDevs) and a user who was a member of both groups would be able to view and launch projects that had either group set as the Access property.

Document # SC23-7681-08

Team and project security plan

If you have many users who work on different projects, the following general plan provides you with the ability to manage them so that individual users get the permissions they need, but only see the projects and other resources that they need to interact with:

- Create role-based access groups for the various activities people perform. For example, create Build Manager and Developer groups. Assign permissions to these groups as appropriate for their jobs. Build Managers might have most of the available permissions, while Developers might have only permissions related to executing jobs.
- Create additional groups for each cross-functional team in your organization. You might have an IDE team, a PrinterDriver team, and others.
- Set the Access properties of projects, servers, and other resources to team groups. All the projects that are relevant to the PrinterDriver team should have the PrinterDriver access group as their access properties.
- When you add a user to the system, assign a user to all the appropriate access groups. All users must be assigned to at least one role group and at least one team group.

If you follow these guidelines, users see only the projects that are relevant to them, and have permissions appropriate to their roles in those projects. Also, you can easily change user permissions as their jobs within your organization change.

Managing access properties

When you assign access properties to data objects such as projects, servers, or steps keep in mind that a user can only assign a project or server to an access group to which they are a member.

If you are not a member of the Admin group, you cannot assign a project to that group. The list of access groups is restricted to those of which you are a member.

For example, steps inherit their access group properties from their parent project. The step creator can change the access group property for a step so that it has a different access group property than the project. Users who are not members of the access group specified for the step cannot execute the step. This allows you to prevent users from running specific steps in a project.

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 to test security features.

Select **Administration → Users** to display a list of current users, with a user form below it. The system displays the Name, Login, Email, Limit, Activity (elapsed time since last user activity), and Time Zone of each user.

Document # SC23-7681-08

- 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, start entering 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 log in as a user without using their password, first log in as root. Click the user's name, then click **Switch to User** .
- To free up a fixed license seat, first log in as root. Click the user's name, then click **Purge Seat** . The console removes the user from the list of IDs counted 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 to make room for another user. Purging a seat does not delete the user from the console. If the user logs back in, the fixed license count is increased. If used on a floating-license user, **Purge Seat** has the same effect as **Logout User** .
- To copy a user, click the user's name and then click **Copy** . The new user's name appended with the word "Copy" appears in the list.

  ## Note

  The password of a copied user is reset to `password`. You may manually change this at any time.

- To delete a user account, click the user's name and then click **Delete**.

  If Delete is disabled, a scheduled job is owned by the user account and the user account cannot be deleted. If you want to delete a user account that has scheduled jobs, you must first delete the scheduled jobs.

The user record sets default properties for the user's experience of the system and controls the user's login name, password, and password expiration. The data for a user record can be entered into the system through the Management Console, or it 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 must change these properties in the source database.

To add a new user, click **Add User**, edit the form, then click **Update**.

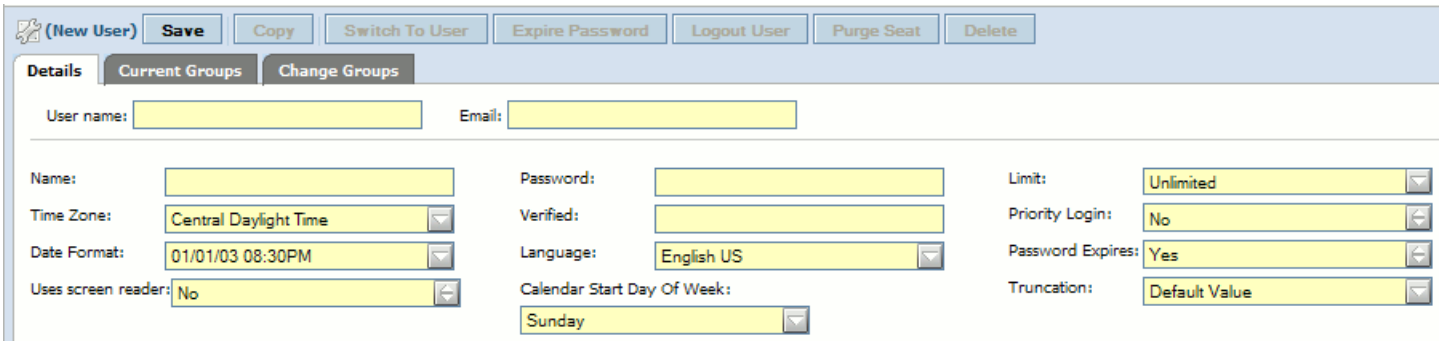
When you display a user record, three tabs are available:

- **Details** : Use this tab to edit most of the user properties. The available properties are described below.

Document # SC23-7681-08

- **Current Groups** : Displays the access groups the user is a member of, either directly or through a direct group being a member of another group.
- **Change Groups** : Displays the groups the user is a direct member of, and allows you to add the user to groups or remove the user from them.

User details form



For each user, you can set these properties on the **Details** tab:

Name — The display name and label for the user.

Email — The e-mail address where the system can send e-mail notifications for this user.

Note

Emails are sent only to users explicitly selected for notification.

User Name — The name used to log on to the Management Console.

Password — The password used to log on to the Management Console. The field is not displayed for the user currently logged on. Use this field to enter a new password or change the existing one. Enter the same password in the **Verified** field.

Limit — Sets the maximum number of jobs the user can launch in a day. When the limit is reached, the system displays messages indicating that the user's run quota has been exceeded. If the value is 0, the system allows the user to run any number of builds.

Time Zone — 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 the time zone assigned to them.

Document # SC23-7681-08

Verified	Repeat the password here to verify it.
Priority Login	If this option is checked, the user becomes 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 to make room for the priority user. The root user is always a priority user.
Date Format	Selects the user's preferred display format.
Language	Selects the user's language.
Password Expires	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	If set to Yes, the interface is enabled to support screen reader features for vision impaired users 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.
Truncation	The Truncation field controls how many characters displays in the lists and pulldown menus. For example, if set to 20, a project name will only display the first 20 characters of the name.

Root user

The root user (the user whose login name is **root**) has special characteristics within the system:

- **Created upon installation**: The root user is the only default user created by the installation program. 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, any other 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.
- **All permissions**: The root user has all available permissions and can edit other user's 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.

Document # SC23-7681-08

- **Log in as any user**: The root user can log in as a user without using a password by clicking **Switch to User** on the **Administration → Users → <UserName>** page.
- **Logging out current users**: The root user can log out users by clicking **Logout User** on the **Administration → Users → <UserName>** page.
- 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.

API users

The API uses a standard console login to access the system. A script that uses the API to log in to the Management Console must have a valid console username and password.

Set up a special user for the API to use. Build Forge allows only one login session per user. If the same user logs in a second time, the first session is terminated.

Permissions

Permissions define what a user can do within the system. You assign permissions to access groups; you do not assign them directly to users.

To work with permissions, select **Administration → Permissions** .

To *assign permissions* to a group, choose a permission and make sure the desired group is listed as having that permission. Select **Administration → Permissions** , select a permission from the list, and then work with the lower pane. Groups on the left side lack the permission; groups on the right have the permission. To give a group a permission, select it in the group on the left and then click **Add** .

Permissions exercise

In this example, you give the user Jane Doe exclusive rights to add and edit servers by giving those permissions to a new access group and assigning her to that group.

1. Log in as root.
2. Make a new access group called Server Admin.
3. Add Jane Doe to the new access group.
4. Go to **Administration → Permissions** .
5. Scroll to and click the **Add New Servers** permission.
6. In the **Details** tab, use **Add** and **Remove** to make Server Admin the only access group that has this permission.
7. Click **Update** . Now only Jane Doe has permission to add servers to the system.

Document # SC23-7681-08

Note

The root user always has all permissions. You cannot remove any access privileges from the root user.

Notice that only Jane can add a server. However, she does not have the ability to enter the server's login information. You can do this by adding the Server Admin group to the **Edit Server Authentication** permission.

LDAP integration

You can set up Build Forge to work with an LDAP server to log in users. This allows users to use the same login names and passwords to log in to Build Forge that they use elsewhere in your organization. When you use LDAP, you do not have to manually create users in Build Forge. Each user is created in Build Forge when they log in to Build Forge for the first time.

You have the option to map LDAP groups to Build Forge access groups. This allows you to manage groups in LDAP and have user permissions updated automatically when a user logs in.

You may still manually create and maintain users who in the Build Forge system. Their access must be managed manually.

To enable this integration, you create entries in **Administration → LDAP**

Note

Only the root user may create and edit LDAP Domain entries in Build Forge.

About LDAP integration

When a user logs in to Build Forge for the first time using LDAP credentials, the user is authenticated and set up within Build Forge as follows:

1. The user sees a **Domain** field on the login panel. If more than one domain is configured, the field is a pull-down list. The user selects the domain and logs in.
2. Build Forge checks for the account on the LDAP server. You can configure Build Forge to use a normal user or an administrative user to perform the check.
3. If the user name is found, Build Forge then attempts to log in to LDAP using the credentials the user supplied at the Build Forge login panel (or from a login from a program using an API client).
   - If the credentials do not match or the user name is not found, the login fails.
   - If the credentials match, login proceeds.
4. If the user has not logged on before, Build Forge automatically creates a user in its user list. A user who logs in through LDAP has its **User Name** , **Password** , **Login** , **Confirm** , and **Email** fields disabled, because that information is provided by LDAP.

Document # SC23-7681-08

Note

The system assigns LDAP users to the *root user's* time zone on first login, because it does not get time zone information from LDAP. You can manually set the time zone afterward.

5. Build Forge applies access groups to the user.

   - If LDAP group mapping is enabled, the specified access groups are applied. The default Build Forge access groups are also applied. Enabling group mapping requires configuration in the Build Forge LDAP domain properties.

     ### Note

     Group mapping is performed each time the user logs in. This keeps Build Forge synchronized with group membership changes in LDAP.

   - If LDAP group mapping is not enabled, Build Forge default access groups are applied. Access group membership can then be managed manually.

     ### Important

     If you intend to use group mapping, enable LDAP group mapping *before* users log in.

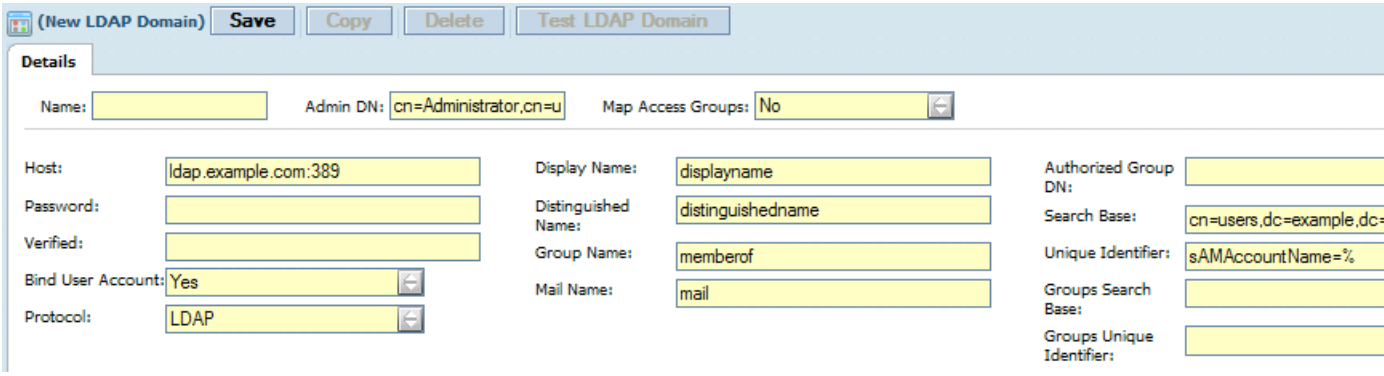
     If group mapping is disabled, users log in, and you later enable group mapping, the mapping is not performed on the existing users. Delete the users from the Build Forge Users list and have them log on again.

LDAP domain properties

To edit properties of a created LDAP domain:

1. Select **Administration → LDAP <Domain Name>**
2. Select the domain to edit. Properties are shown in the LDAP domain properties panel.

LDAP domain properties panel



Document # SC23-7681-08

3. Edit the values for any of the fields, then click **Save** . The following fields are required:

    - Name
    - Host
    - Bind User Account
    - Protocol
    - Display Name
    - Distinguished Name
    - Mail Name
    - Unique Identifier

Name	Required. Name for the LDAP domain within Build Forge. If there is at least one LDAP domain configured, the Build Forge login form lists them by this name.
Admin DN	Account to use to provide search access to the LDAP server database. If your server allows an anonymous bind for searching the database, leave this field blank. Some LDAP servers require an administrative bind in order to search the database. This setting allows you to specify the DN of the administrator account, as shown in the following example. `cn=Administrator,cn=users,dc=example,dc=com` Specify the password for the Admin DN account in the **Password** and **Verify Password** fields.
Map Access Groups	Determines whether to map group information from the LDAP server to access groups in the Management Console. The default is No. Each access group in Build Forge must have its **LDAP Group DNs** property set to the correct group name in LDAP. • If **No**, then LDAP groups are not mapped to Build Forge access groups. You can assign users to access groups in Build Forge after they have logged in at least once. Using this option implies that you manage access groups for users within Build Forge. Default access groups are applied when the user first logs in and has a user name created in Build Forge. • If **Yes**, the Build Forge refreshes group membership information from the LDAP server for a user every time the user logs in to Build Forge. Any changes to access group

Document # SC23-7681-08

membership made for the user within Build Forge since the last login are overwritten. Using this option implies that you manage all group memberships in LDAP. The LDAP group memberships are automatically mapped (added or removed) to access groups in Build Forge. Group properties are used as follows to determine group membership for a user:

1. If **Group Name** is not blank, query for the value of the keyword specified. Use the values returned as the groups for the user.

2. If **Group Name** is blank *or its query does not return a value*, then use **Groups Search Base** and **Groups Unique Identifier** to query for LDAP groups that the user belongs to.

3. If no group information is returned in (1) and (2), the user is allowed to log in and is assigned membership in the access groups that are specified as default access groups for new users.

Default LDAP Domain

If checked, this domain is displayed first in the login form and searched first during login. If there is more than one domain defined, the domains are presented as a pull-down menu.

Host

Required. Host name and port of the LDAP server. Examples:

```
ldapserver.mycompanyname.com
ldap.mycompany.com:9000
```

Password

Password for the Admin DN account. Required if **Admin DN** is specified.

Verified

Repeat entry of the Admin DN password.

Bind User Account

Required. Determines whether the Build Forge attempts to validate user credentials against LDAP at login time. The default is Yes.

- If **Yes**, Build Forge checks the user name and password supplied at login with the LDAP server.

- If **No**, Build Forge accepts the username without validation. This setting is used when an external password validation is implemented for Build Forge, such as Single Sign-on (SSO).

Protocol

Required. Identifies the protocol Build Forge uses to read and write data from the directory service for the purpose of authenticating Build Forge users. The default is LDAP. Enter

Document # SC23-7681-08

	LDAPS if you use LDAP over SSL (LDAPS). Additional setup is required for this option. See “Enabling secure LDAP (LDAPS)” on page 214
Display Name	Required. Enter the keyname that specifies the full name of the user.
Distinguished Name	Required. Enter the keyname that specifies the Distinguished Name for a user account.
Mail Name	Required. Enter the keyname that specifies an email address for the user.
Group Name	Enter the keyname in the LDAP schema that holds the list of groups the user is a member of. Used only when **Map Access Groups** is Yes.
Authorized Group DN	Distinguished Name of an LDAP group. If set, then only members if the specified group are allowed to log in. If blank, then *any* valid LDAP user can log in to the console.
Search Base	Required. Search string used to query LDAP records for users. Example: `cn=users,dc=buildforge,dc=com`
Unique Identifier	Required. Identifies the field in the LDAP database to compare with user name a user enters at login. Use a % character for the login name entered by the user. Example: `(sAMAccountName=%)`
Groups Search Base	Requires **Groups Unique Identifier** . Used only when **Map Access Groups** is Yes. Search string used to query LDAP records for group data. Needed if your LDAP database stores group membership in a database that is separate from the database used to store user records. Example: `cn=groups,dc=buildforge,dc=com`
Groups Unique Identifier	Requires **Groups Search Base** . Used only when **Map Access Groups** is Yes. Identifies the field in the LDAP user database to use to obtain group membership information. The filter can use any of the data fields for a user account as a key into the groups table. Use the `%fieldname%` syntax to identify the field.The following example works if your groups table uses the `sAMAccountname` field as a key for users. `sAMAccountName=%sAMAccountname%`

Document # SC23-7681-08

Tasks

These topics identify tasks for working with LDAP domains.

Creating LDAP domain entries

You can create as many LDAP domain entries as you like. When a user attempts to log in, the Domain must be specified. For an API client login, the domain must be specified in the login call or the domain to use must be configured in `bfclient.conf`.

To add a domain entry:

1. Select **Administration → LDAP** .
2. Click **Add LDAP Domain** .
3. Fill in or change the properties for the domain. The **Name** property is internal to Build Forge. The values provided by default are designed to work with LDAP or a standard Microsoft® Active Directory server.
4. Click **Save** .

Testing an LDAP domain entry

To verify that your LDAP domain is set up correctly, do the following:

1. Select **Administration → LDAP**
2. Select a domain from the list.
3. Click **Test LDAP Domain**

The system queries the LDAP server using the properties in the LDAP domain entry.

Enabling secure LDAP (LDAPS)

If your LDAP server supports LDAP over SSL (LDAPS), you can configure Build Forge LDAP domain entries to use LDAPS as well. Strict SSL is configured by default. Strict SSL requires server certification.

1. Create an LDAP domain entry in Build Forge.
2. Set the **Protocol** property to LDAPS. This will enable an encryption-only method of LDAPS.
3. Set the **Host** to the fully qualified domain name and SSL port of your LDAP server. Port 636 is the defined default for strict secure LDAP. Example: `myldap.mycompany.com:636`.
4. Get a signer certificate from the LDAP server and add it to the Build Forge truststore. Outbound LDAP is configured by default to use the following settings in **Administration Security** :
    - SSL panel: Default JSSE Outbound SSL

Document # SC23-7681-08

- Keystore panel: Default JSSE Trust Store. This trust store is set to use `<bfinstall>/keystore/buildForgeTrustStore.p12` by default. Place the signer certificate here.

5. Restart Build Forge.
6. Go to **Administration Security** and select your secure LDAP configuration.
7. Click **Test Connection** .

Note

Strict LDAPS SSL is set in Build Forge by default. The strict configuration requires server certificate validation. If you do not want to use strict LDAP, do the following:

1. Set Tomcat system property `-Dcom.buildforge.services.server.ldap.strict=false` in the JAVA_OPTS environment variable. Tomcat scripts read this variable and apply any system properties specified to the Tomcat process.
2. Restart Build Forge.

In this configuration you do not have to add the LDAP server certificate to the Build Forge truststore. However, this configuration is a weak implementation of the SSL protocol design. Build Forge does not verify the LDAP server's identity during communication with it.

Changing LDAPS SSL configuration

The SSL configuration used by outbound LDAP requests is set up by default. You can change two aspects of it:

- SSL configuration. You need to do this if your LDAP server cannot communicate with Build Forge using the default protocol or handshake.
- Keystore configuration. Strict SSL requires that you place a signer certificate in the truststore used by the client (Build Forge) to communicate securely with the LDAP server. If you want to use a different truststore or place it in a different location, you need to create a new keystore configuration in Build Forge for the truststore.

These instructions assume that you have already enabled secure LDAPS for Build Forge and that you have not enabled SSL for Build Forge components.

To change the LDAPS SSL configuration, do the following:

1. If you are changing the location or name of the truststore, place it on the Build Forge host in the desired location. Add the LDAP server's signer certificate to it.
2. Create a truststore configuration in **Administration Security Keystore** if needed. The truststore configuration includes properties for the location and name of the truststore.

Document # SC23-7681-08

3. Create an SSL configuration in **Administration Security SSL** if needed. Configure it to use the new truststore configuration (if you created one). Make other adjustments to the configuration as needed.
4. In **Administration Security** , set **SSL Enabled** to Yes if it is not already set. Additional fields appear.
5. Select the SSL configuration you created in the **Outbound LDAP** list. Do not change the other settings.
6. Click **Save** .
7. Click **Update Master BFClient.conf** .
8. If SSL was not enabled before, do the following:
    a. Click **SSL Enabled** to No.
    b. Click **Save** .
    c. Click **Update Master BFClient.conf** .
9. Restart Build Forge.
10. In **Administration LDAP** , select your LDAP configuration.
11. Click **Test Connection** .

Turning off LDAP/Active Directory support

If you want to discontinue authentication with LDAP or Active Directory, do the following

1. Select **Administration → LDAP**
2. Delete all domain entries. Click the trash can icon next to a domain entry to delete it.

When no domains exist, only users who were added manually Build Forge can log in.

System configuration settings

You can use a variety of settings to configure your Management Console. You can find these settings on the **Administration → System** page.

When you click **Administration → System** , the system displays a list of settings. Click the name of a system setting to display an edit form for the setting.

Note

For system settings that take numeric values, the Management Console accepts any value comprised of one or more integers (0 through 9). Numeric grouping characters are not supported; for example, commas (,), decimals (.), and other noninteger separators.

Document # SC23-7681-08

The form includes the following buttons:

- **Save:** Saves any changes you make to the setting's value.
- **Revert to the Default:** Resets the setting to its default value.

The following table describes the available settings.

Setting	Description
Active server refresh interval	Sets the time, in seconds, between updates of Built-in and Set Value properties for server manifests on active servers: servers that are currently running projects.
Alert Email Limiting	Sets the maximum number of alert e-mails the system sends over some period of time. This value should be expressed as <number of e-mails>/<time in minutes>. For example, the value 10/60 sets the maximum to 10 messages per hour. The default value of 0/0 is interpreted by the system as no limit on messages. The system maintains a counter of messages during the limit period, and stops sending messages when it reaches the limit, until the end of the limit period, when it resets the counter.
Apply inlined steps container environment	Default: No. If Yes, applies the environment of the project or library that contains an inlined step.
Apply server environment last	Default: No. If Yes, applies the server environment for the step last. The server environment is applied after the step environment or project environment, if these environments are specified.
Auto-Logoff Minutes	The system can automatically log off users who are idle. This setting specifies the number of minutes of idle time that must pass before the system logs off a user. When the setting is 0, the system does not attempt to automatically log off users.
AutoClean Audit Log Days, AutoClean Error Log Days, AutoClean Info Log Days, AutoClean Warning Log Days	These values set a maximum number of days that each category of entry remains in the audit log; older entries are automatically deleted. If the value is 0, the system never deletes entries of that category. Since string values evaluate to 0 as integers, you can use a value such as “Never” instead of 0.
Build Cancel Check Frequency	Specifies how often the system checks for build cancellation requests, in terms of seconds between the checks.
Bump Active Users	When set to N, if an already logged in user (non-root) tries to log in from a new session, Build Forge rejects the login attempt. When set to Y (the default), if an already logged in user (non-root) tries to log in from a new session, Build Forge logs the user out of the previous session, allowing them to log in to the new session.

Document # SC23-7681-08

Setting	Description
	Note Regardless of how this setting is defined, a root user who logs in to a new session always bumps another active root user. Only one user can be logged in as user root.
Console Host	Hostname or IP address where the console is running.
Console Port	Port number that the web server uses to listen for Build Forge requests.
Console URL	URL that the web server uses to listen for Build Forge requests. Must be set if the console is running on a port other than 80. If set, overrides the default console URL with the value. It takes the form `<protocol>://<hostname>[:<port>]`. Example: `http://myHost:81`.
Create Missing Paths	Default: No. If yes, creates paths for projects if the path is not already present.
Default _AGE	Sets the interval, in seconds, between updates of Run Command properties for server manifests. This value can be overridden by setting a value for _AGE in the server's collector.
Default Agent Port	Sets the default port number used for making connections to agents.
Default Import Class	Class to use if an imported project has no defined class or it has a defined class that does not exist. Default: Production.
Enable Quickstart	Default is no: all projects show the following icon:  . When you click the project, all variables included for the project are checked for variables of type Must Change. The project is started if it does not contain Must Change variable. If the project contains a Must Change variable, then the project does not start, a dialog describes why, and the icon changes to this icon:  . If set to YES, the Projects page checks all environments for all projects on the page to determine if any variable is of type Must Change. Projects that are eligible to be started immediately indicate it with this icon:  . This was the default behavior until version 7.1.1.1. See also Must Change Variable Check Depth.
Expandable Step Log	Determines whether your system uses the standard expanding/contracting view of logs (if you use the default value, Yes) or the new flat log view (if you set it to No) as the default view when displaying step logs.
Hard Run Limit	Default: No. If Yes, the scheduler includes waiting builds in its run limit calculation.
Import Default Secure Access Group	Specifies a default access group for imported projects when the **Import with Secure Access** setting is set to Y.
Import Insecure Default Access Group	Specifies a default access group for imported projects when the **Import with Secure Access** setting is set to N. The default group is used only when the import file lacks an access group.

Document # SC23-7681-08

Setting	Description
Import with Secure Access	When set to Y, the system assigns the default access group listed in the preceding setting to imported data objects. This overrides any access group specified in the XML file you are importing, so that users cannot override security by importing data. When set to N, the system honors any access group settings in imported files.
Inactive server refresh interval	Sets the time, in seconds, between updates of Built-in and Set Value properties for server manifests on inactive servers: servers that are not currently running projects.
Inherit Tag	When set to Yes, causes jobs that are launched via a chain to use the same job tag as their caller. If BUILD_15 of project MasterProject calls project ComponentProject, then the job tag (and the job directory name) for that run of ComponentProject becomes BUILD_15. **Note** The called project always inherits the original tag of the caller; if the caller's tag changes during the run, as a result of a .retag command for example, the called project still gets the tag that the caller started with.
Invalid Relative Dir Characters	Sets the characters that the system will change into underscores if used in project names.
LASTRUN Format	Enter the value for the format for the BF_LASTRUN environment variable, using date format characters as defined for the .date command (see “.date” on page 124).
License Server	The license server hostname. It is set during installation. Example: `myhost.mycompany.com`. The value may include a port number. Example: `myhost.mycompany.com:80`. To change the license server, see "Changing the License Server to Use" in the Installation Guide.
Link Debug Mode	When set to Yes, projects that have source links defined for them run a test of the link instead of running an actual job. The job output for such a run has a single step with output from the adaptor, which you can inspect when troubleshooting your adaptor interfaces.
Link Manual Jobs	Determines whether the system checks source code links when you run a project manually. When set to Y, the system checks the source code link for the project when you launch a job, and may produce change output in the BOM for the job. The system also displays a “Run Link” check box on the Start page for the project, so that you can choose whether to check the source code link when you manually start the job.
Lock Manager Address	Reserved for internal use: IP address and port to use to connect to the lock manager.
Log All Changes	Logs all changes made to all parts of the system. Off is the default. Using this setting has a significant effect on system loading and on database growth. Be sure that you consider its impact when sizing hardware and database capabilities.

Document # SC23-7681-08

Setting	Description
Manifest check interval	Determines how often the system checks for servers whose manifests need to be refreshed, by specifying the number of seconds between checks.
Max Console Procs	Sets the maximum number of processes the console runs at one time. Use as a general throttle on console activity. The system manages processes by storing an ID for each process in the database, and checking the total before launching a new external process. Make sure this value is greater than your **Run Queue Size** setting by at least 5; otherwise the system cannot run enough processes to support the run queue.
Max Inline Depth	Controls the number of levels the system allows for inlining of projects, so that projects cannot be infinitely nested. The default value is 32. If the value is set to 0, the system uses 32. When the system reaches the inline limit, an inlined project that would exceed the limit does not get run, and its steps do not get inserted in the containing project. A message is written to the system messages list: "inline abandoned."
Max simultaneous server tests	Specifies how many server tests can be run at once. Depending on your system resources, running too many server tests at one time can severely slow or lock up the console.
Maximum Refreshes	Maximum number of times that a page refreshes automatically. Default: 50
Must Change Variable Check Depth	The number of .include environments checked for, to determine whether the project can be quick started or not. Default setting is 5. **Note** Increasing this value can cause major performance degradation if Enable Quickstart is set to YES. If your environments tend to not be nested very deeply, decreasing this value can help optimize performance.
Override Class when Chaining	Determines whether the system replaces a chained project's class with the class of its caller. The default value of Y causes the system to override the chained project's class and use the caller's class instead.
Password Expiration Days	Sets the number of days before users whose passwords are set to expire have to change their passwords. When this time expires, the relevant users are required to change their passwords on next login.
Password Format	Specifies the requirements for user passwords using a format string of six fields separated by periods: `length.req_types.upper.lower.numeric.special` The first two fields specify the following:

Document # SC23-7681-08

Setting	Description
	• Minimum password length (characters) • Minimum number of character types that must be used (an integer ranging from 1 to 4) The remaining fields describe different character type specifications. Each field includes a type and a number. • Type: one of u (uppercase), l (lowercase), n (number), or s (special) • Case: Uppercase (U, L, N, S) indicates that the character is required. Lower case (u, l, n, s) indicates that the character is optional. • Number: for required, indicates the number of characters of this type that are required. For optional, indicates the number of characters of this type that are required *if any are used*. The types are as follows: • U or u to indicate uppercase characters. • L or l to indicate lowercase characters. • N or n to indicate numeric characters. • S or s to indicate special characters. Example: the string 5.2.u1.l1.n1.s1 indicates the following password requirements: • At least 5 characters long • Must include characters from a minimum of two of the four categories (uppercase, lowercase, numeric, special). • For each type, one character of the type qualifies as a match to count toward the requirement. Passwords such as abC1x and Abc2% would qualify.
Pause Build Forge Engine	When set to Y, the system completes any current jobs and then pauses the engine. Set it to N to return to normal operation.
Public Hostname	When set, the system substitutes the value of this setting for the server host name in the CONSOLEHOST variable in notification templates
Purge Check Time	Sets the frequency with which the system checks for jobs to purge, in terms of minutes between checks.

Document # SC23-7681-08

Setting	Description
QuickReport Public dir	The file system location of the public report designs. 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) 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.
QuickReport User sub directory for custom data sources	User's saved data sources. It is a relative path from <*bf-install*>/Apache/tomcat/reports/users/<*username*>. Default: `Datasource`
QuickReport User sub directory for saved reports	User's saved report designs. It is a relative path from <*bf-install*>/Apache/tomcat/reports/users/<*username*>. Default: `reports`
QuickReport User sub directory for uploaded XML files	User's uploaded XML files. It is a relative path from <*bf-install*>/Apache/tomcat/reports/users/<*username*>. Default: `XML`
QuickReport Users dir	The file system location of the private report designs. In 7.1, use this system setting to specify the fully-qualified location to private reports. Your report designs must be 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.
Reload Language Packs	Default: No. If set to Yes, the console reloads its language packs upon restart and resets this value to No. No longer necessary starting in version 7.0.1.
Reset Interface Templates	Use this setting to reset the adaptor templates (to copy changes from an update into your configuration). To use it, set the value to “Yes”, then wait one minute. The system resets the templates and then sets the value back to No.
Reset Server Job-Count	Use this setting (Yes) to simultaneously reset the job count (BF_JOBS) for all servers to zero. The reset occurs when the manifest check interval runs. (The default time is every 10 seconds.)

Document # SC23-7681-08

Setting	Description
	After BF_JOBS has been reset for all servers, the Reset Server Job-Count value reverts back to No (the default).
Restart Report Migration	Default: No. In 7.1, if you want to start migration without restarting the services layer component, set this value to Yes.
Run Queue Size	This value limits the number of jobs the system attempts to run at once. When the number of runs in the queue equals or exceeds this number, the system stops moving runs from the Wait queue to the Run queue until the number of jobs drops below this value. If you change your Run Queue Size, check the Max Console Pros setting, which should be greater than the Run Queue Size by at least 5.
Save Start Environ	Controls the default value of the “Save Env” check box on the manual start page for a project. When this setting is Y, the box defaults to checked; otherwise, the box defaults to unchecked. The “Save Env” check box, when checked, causes any changes you make to the environment variables on the Start Page to be saved to the environment records in the database, so that future runs default to those values.
Secondary Site Timeout	If a secondary server fails to respond in this amount of time, the primary console marks it down. The down condition may trigger failover if failover is enabled. Default: 600 seconds.
Server Connect Timeout	Sets the number of seconds the system allows for an attempted connection to a server before giving up.
Server Env Before Chain	Determines whether the system sets a chained project's project environment variables before (Y) or after (N) it sets the calling step's server environment. The default value is N; set this setting to Y if you want the server environment to override the called project's environment.
Server Retries	Sets how many times the system tries to allocate a step to a server before it gives up and fails the step. If 0, it never gives up.
Server Self-heal time	Sets how long, in seconds, the system waits after an error before trying to use a particular server again.
Server Wait Time	Sets how long, in seconds, the system waits (after it has found no server for a step) before checking again to see if a server is available for the step.
Services layer authentication servlet URL	When set, will override the programmatically constructed URL to the services layer authentication servlet. If you are using an alias or a non-default port, then this setting will need to be updated using the following format: http://server:port/rbf-services/AuthServlet
SMTP Server	Sets the machine to use as an SMTP server when sending e-mail notifications. The default setting is localhost.
SSO Remote User	Default: No. If Yes, allows single sign-on remote user connections via standard web server authentication.

Document # SC23-7681-08

Setting	Description
Stack BuildForge Env Variables	The system normally changes the name of BF_ variables that are passed down to a chained project to BF_CALLER_; this setting determines whether the system *stacks* the naming when chaining goes more than one level deep. The default value is N. When the setting is changed to Y, the BF_TAG variable derived from a calling project two levels deep receives the name BF_CALLER_CALLER_TAG.
Step Log Page Size	Defines how many lines of data to place on each page when the system divides large logs into pages.
Store User Authentication Locally	Determines whether the system caches LDAP/Active Directory user authentication information (in encrypted form). The default value is Yes. The system is only relevant when you use LDAP/Active Directory authentication. When the setting is Yes, the system caches user authentication information in encrypted form, and can use it with the _USE_BFCREDS special variable (which applies user authentication to servers). You may choose to turn off caching by changing the value to No; however, if you do, the system cannot use the _USE_BFCREDS special variable to use the user's credentials when logging into a server.
System Alert Email	The system sends alert e-mail messages to the address defined by this setting. The default is root@localhost.
System Alert Source	When the system sends alert e-mail messages, it uses the address defined in this setting as the sender. The default is root@localhost.
System Wide Login Message	Allows you to define a message to be displayed above the login form.
System Wide User Message	Allows you to define a message to be displayed at the top of each page, just below the navigation buttons.
Tab Stop	Defines how many spaces the system should use to substitute for tabs when it creates change data reports in BOMs and step logs.
Tag: Date Format	Defines the format used to display the date in the BF_D tag variable. Use the characters y, m, and d as variables for the year, month, and day to show the desired format, along with any desired special characters as separators. For example, for the date September 21, 2005: Format string....Output ymd...050921 m/d/y...09/21/05
Tag: Time Format	Defines the format used to display the date in the BF_T tag variable. Works such as the Tag: Date format setting, but uses the characters h, m, and s to stand for hours, minutes, and seconds. The setting h:m:s produces output such as 12:53:42.

Document # SC23-7681-08

Setting	Description
Tail Log Amount for Mail Template	Sets the number of lines from the end of a log that are displayed in a notification when the TAILNORMALLOG variable is used in the notification template.
UI Page Size	Number of entries to display in a list at one time (for example, the list of jobs or the list of projects). The default is 15.
Update Time Vars on Restart	This setting controls whether the system updates system variables when you restart a job.
Use System Language for Agent Communication	Default is Yes. When Yes, the console uses the agent host system language to communicate with the agent. When No, the console uses the user's language to communicate with the agent.

Messages

The system messages page displays a log of system and user actions that you can review.

Select **Administration** → **Messages** to display the list of messages. These messages form an audit trail of actions.

You can use the **Severity** box to filter the list by the severity of the messages, and you can limit the list to recent messages by changing the value in the **Last** box.

To display audit messages, select **Audit** in the **Severity** list.

Subscribing to RSS data feed for jobs status

You can now track and filter the status of individual jobs using RSS data feeds. The Build Forge RSS data feed for jobs displays the same information as the server status in the Build Forge Management Console.

To subscribe to the RSS data feed for jobs status, do the following:

1. In the Build Forge Management console, select **Jobs** .

   The Web browser detects the RSS feed and displays an RSS icon in the browser address bar.

2. In the RSS aggregator tool, load the Build Forge RSS data feed.

   For example, copy the URL to add it to the list of RSS data feeds or drag-and-drop the RSS icon to add the URL to the list of RSS data feeds.

3. Subscribe to the RSS data feed to save the URL and be notified when updates occur.

Note

- For information about loading URLs and subscribing to RSS data feeds, see the documentation for your RSS aggregator tool.

Document # SC23-7681-08

- To view Build Forge jobs status, system messages, or server status through an RSS data feed in languages other than English, your RSS aggregator tool must support UTF-8 multibyte character encoding.

Filtering the RSS data feed for system messages

You can filter the messages displayed in the RSS data feed. To do this, use the filter properties in the **Administration → Messages** page in the Management Console.

For example, change the Severity property to Audit or change the Last property to 12 Hours. When you change a filter property, the URL for the RSS icon on the Messages page is automatically updated.

To load the updated URL into the RSS aggregator tool, right-click the RSS icon on the Messages page to copy the link or drag-and-drop the updated RSS icon to the list of RSS data feeds in the tool.

Note

For details about loading URLs and subscribing to RSS data feeds, consult the documentation for your RSS aggregator tool.

Security panel

The Security panel allows you to enable security services:

- SSL: enabling SSL in this panel is only one part of enabling the SSL security feature throughout the system. Additional work is needed before turning it on and after.
- Password encryption: enabling password encryption is only one part of enabling password encryption throughout the system.

Important

Only some of the setup is done in the Security panel. See the Installation Guide, **Configuring additional features in Management Console Security Features** .

Enabling SSL

Prerequisites in installation: During installation you specify two things that are used by the SSL configuration:

- **SSL port** , specified in the Web and Application Server panel. That port must match the port specified in the configurations you choose below. The default during installation and in the configurations is port 8443. This port is used by the authentication servlet on Apache Tomcat during login to encode or encrypt user login credentials.
- **Certificate** , specified in the Web and Application Server panel. You either provided your own or allowed the installer to create a self-signed certificate for you. The certificate is stored in the default keystore. The keystore location is defined in named SSL configurations.

Document # SC23-7681-08

- Set **SSL Enabled** to Yes. Additional properties are shown:
  - **LDAP Outbound** : specifies the configuration used for outbound communication through LDAP. The default is Default JSSE Outbound SSL.
  - **Engine to Agent Default Outbound** : specifies the configuration used for communications from the engine component to agents. The default is Default OpenSSL Outbound SSL.
  - **Services Layer Inbound** : specifies the configuration used by the Services Layer component to accept communications from the web interface component and engine component. The default is Default JSSE Inbound SSL.
  - **Services Layer Outbound (JSSE)** : specifies the JSSE configuration used by the engine component and web interface component Services Layer component to communicate with databases. The default is Default JSSE Outbound SSL.
  - **Services Layer Outbound (OpenSSL)** : specifies the OpenSSL configuration used by the engine component and web interface component Services Layer component to communicate with databases. The default is Default OpenSSL Outbound SSL.
- Click **Save** .
- Click **Update Master BFClient.conf** . This step edits the BFClient.conf file using these property settings.
- Restart Build Forge. Secure communications are not in effect until the system starts using these settings.

The configurations you select are defined in the **SSL** panel.

Requirements after SSL is enabled in this panel:

- Certificate distribution: certificates must be installed in keystores on agent hosts, the database host, and any additional Build Forge installations that are running (redundant configuration).
- Agent SSL enablement: if you intend to use SSL for communication between the engine component and agents, each agent must be configured to use SSL.
- API client enablement: all API clients must configure SSL in order to communicate with the services layer component.

Enabling password encryption

Prerequisites:

- **SSL port** , specified in the Web and Application Server panel. That port must match the port specified in the configurations you choose below. The default during installation and in the configurations is port 8443. This port is used by the authentication servlet on Apache Tomcat during login to encode or encrypt user login credentials.

Document # SC23-7681-08

- **Certificate** , specified in the Web and Application Server panel. You either provided your own or allowed the installer to create a self-signed certificate for you. The certificate is stored in the default keystore. The keystore location is defined in named SSL configurations.
- Set **Password Encryption Enabled** to Yes.
- Click **Save** .
- Click **Update Master BFClient.conf** . This step edits the BFClient.conf file using these property settings.
- Restart Build Forge. Password encryption is not in effect until the system starts using these settings.
- Once it is enabled, any new passwords entered at the console are encrypted, including Server Auth passwords and user passwords for users created at the console.

Additional Requirements:

Once you have enabled encryption, you need to do the following

- Enable encryption on all agents. Export the key and use it to update the server auth password in each agent's configuration. The password must be manually updated in `BFAgent.conf`.
- Enable an encrypted password for database access. Export the key and use it to update the database password that Build Forge uses to log on to the database. The password must be manually updated in `buildforge.conf`.

Keystore panel

The Keystore panel contains configurations for individual keystores. When you edit an SSL configuration in the **Administration → Security → SSL** panel, you can select these individual configurations to be part of an SSL configuration.

You can create configurations or use the configurations that are provided:

- Default JSSE Key Store
- Default JSSE Trust Store
- Default OpenSSL CA Store (certificate authority)
- Default OpenSSL Cert Store (for certificates)
- Default OpenSSL Key Store

Each keystore has the following properties:

Name

Access The access group that defines what users can edit or delete this keystore.

Document # SC23-7681-08

Location	The location of the keystore file. The default keystores all use the default location for Build Forge keystores: `<bfinstall>/keystore`. If you are using Websphere Application Server as the application server rather than the provided Tomcat application server, specify an absolute path.
Keystore Type	A keystore must be one of the following types: • JKS • PKCS12 • JCEKS • PEM
Password	Specifies a password that must be used when accessing the keystore.
Verified	Specify the password again here to verify it.

Keystores and Websphere Application Server

If you use Websphere Application Server rather than the provided Tomcat application server, additional requirements apply to configuring keystores:

- Location field: you must provide an absolute path, rather than a relative path.
- Multiple services components: if you install multiple Build Forge services components, they are installed on different hosts. You configure security for each services component. The keystore path specified in the Location field must be identical for each services component.

SSL panel

The SSL panel contains individual configurations of SSL. When you select **SSL Enabled** to Yes in the **Administration → Security** panel, you can select these individual configurations to be part of the SSL enablement.

You can create your own configurations or use the ones provided:

- Default JSSE Inbound SSL
- Default JSSE Outbound SSL
- Default OpenSSL Inbound SSL
- Default OpenSSL Outbound SSL

Each configuration has the following properties:

Name	Name for this configuration.
Access	The access group that defines what users can edit or delete this keystore.

Document # SC23-7681-08

Type	Select JSSE or OpenSSL.
Client Authentication	Select one of the following: • Never • Supported • Required
Server Certificate Alias	Enter the alias for the server certificate.
Client Certificate Alias	Enter the alias for the client certficate.
Keystore Configuration	Select one of the Keystore configurations. They are configured in the Keystore panel.
Truststore Configuration	Select one of the Keystore configurations. They are configured in the Keystore panel.
Handshake Protocol	One of the following: • SSLv2 • SSLv3 • SSL • TLSv1 • TLS • SSL_TLS
Cipher Suite Group	One of High, Medium, Low, or All. Higher-order ciphers are more secure but entail slower performance.
Cipher Override List	

SSO panel

The SSO panel contains configurations for Single Sign-On (SSO). You can create new ones or use the configurations that are provided:

- Form SSO Interceptor
- SPNEGO SSO Interceptor

Enabling a single sign-on service requires additional setup. It is not done completely from this panel. See the Installation Guide, **Configuring additional features in Management Console Security Features Implementing Single Sign-on**

Document # SC23-7681-08

Each configuration has the following properties:

Name	The name for the SSO configuration.
Active	Select Yes to make this interceptor active.
Access	The access group that defines what users can edit or delete this configuration.

Java Class	The Java class that implements this SSO configuration. Two are provided:
	• Form interceptor:
	`com.buildforge.services.server.sso.form.FORMSSOInterceptor`
	• SPNEGO interceptor:
	`com.buildforge.services.server.sso.spnego.SPNEGOSSOInterceptor`

Keystore Type	A keystore must be one of the following types: • JKS • PKCS12 • JCEKS • PEM
Password	Specifies a password that must be used when accessing the keystore.
Verified	Specify the password again here to verify it.

Managing licenses

When you create a user, the system stores data about the user in the database. After the user logs in to the system, the system assigns a license to the user. You can have two types of users: authorized users and floating users. Authorized users can always log in, but always consume a license. Floating users only consume a license when they are logged in, but may not be able to log in if all the floating licenses are in use.

If you create more users than you have licenses, the new users cannot log in until you purchase additional licenses or use the root user account to remove licenses from some users (or delete some users from the system).

A user can only be logged in once. If you are logged in on one machine, and you log in from another machine under the same name, the original session is invalidated.

Note

If you purchased your system before version 3.8, your license scheme may differ from the one described here. Contact customer support if you have questions about your licensing.

Document # SC23-7681-08

Entering a new license key

To change your license key, select **Administration → System** and then locate the License Key setting in the list of settings. (Type "license" in the Filter box to quickly display this setting without paging through the list).

Click the License Key item in the list, and the system displays a tab at the bottom of the content pane with an editable **License Key** field. After you edit the field, click the **Save Config** button.

Managing the engine

This topic describes how to pause, start, or stop the engine (`bfengine.exe` / `bfengine.pl`).

Note

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. On Windows, select **Start → All Programs → IBM Rational Build Forge → Start Engine (Foreground)** to run the engine in the foreground, which causes it to display output in a console window. When the engine is running in the foreground, you can stop it by closing its console window.

Pausing the engine

If you want to act directly on the database, for example, to back it up or restore it, then you need to pause the engine.

To pause the engine, select **Administration → System** and locate the **Pause Build Forge Engine** property (use the **Filter** box). The system displays a form for editing the **Pause Build Forge Engine** setting. Change the property to Y and click the **Save Config** button.

The system pauses the engine, but any currently-running projects continue. You must wait for any currently active jobs to complete to ensure that no data is being written to the database.

To reactivate the engine, change the value of the property back to N, and click **Save Config** again.

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 on Windows®, Linux® or Solaris.

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**

Document # SC23-7681-08

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, close its console window.

Starting and stopping the engine on Linux or Solaris 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>}
```

Managing the database

This topic describes issues that are important when setting up the Management Console database, especially if you want to change default configurations.

Deleting the database log file

Delete the database log file regularly.

The system stores database debugging information in a `db.log` file in the Management Console installation directory. You should check the size of this file on a monthly basis, and delete it if you need to free space on the console machine.

Error messages

This topic describes error messages you might encounter while using the Management Console.

No active steps

When this message appears in the Next Run column of a schedule entry, all of the steps in the associated project have been disabled. Display the list of steps for the project and enable some of them by clicking the red circle next to each disabled step.

Document # SC23-7681-08

License key is invalid *or* Build Forge license key is corrupt or missing.

Your license key is either expired or invalid for the product version you have installed. Enter a new license key. See “Entering a new license key” on page 232.

A database license is required

```
LicMgr: 5140: A database license is required.
```

If the above message appears in your console output (which can be viewed when you run the console in the foreground, or if you view the console log on Linux® or UNIX® systems), you have tried to use an advanced database feature without the benefit of an Enterprise license. For example, you may be trying to use the system with a non-DB2® database. If you want to upgrade your license, contact support.

Document # SC23-7681-08

Linking to Web resources

Use the **UI Config** tab to add tabs to the Build Forge UI.

You can add tabs to the Build Forge user interface by using the **UI Config** tab. Each new tab contains a URL. You can use the tabs to link to the following:

- External resources. This could be information about your applications, operating systems, servers, or users.

Note

Do not create tabs that link to internal Build Forge URLs.

To add a tab, do the following:

1. Select the **UI Config** tab.
2. Click **Add Tab** .
3. At **Name** , enter the title or name of the tab.
4. At **Link** , you can enter any of the following:

Option	Usage	Example
URL with protocol	• required if the protocol is not http • optional if it is http	http://www.ibm.com
URL without protocol	uses the default http protocol	www.ibm.com

5. At **Target** , select one option:
   - Internal (the default): Select to open the link in the existing browser window.
   - External: Select to open the link in a new browser window.
6. At **Enabled** , select one option:
   - Enabled (the default): Select to enable the tab. If enabled, a connection to the URL is attempted when you select the tab.
   - Disabled: Select to disable the tab. If disabled, a connection to the URL is not attempted when you select the tab.
7. At **Visible** , select one option:
   - True (the default): Select to show the tab in the UI.

Document # SC23-7681-08

- False: Select to hide the tab in the UI.

8. At **Root Only** , select one option:

   - True (the default): Allows only the root user or a Build Forge user with root access to select the tab and connect to the URL link.
   - False: Allows any Build Forge user to select the tab and connect to the URL link.

Note

After saving a new tab, the tab title will appear in the list. If the tab itself does not, refresh the page.

Document # SC23-7681-08

Adaptor integrations

This topic describes how to set up and use adaptors with the Management Console. Sample adaptors are provided for integration with IBM Rational ClearCase and IBM Rational ClearQuest. The Adaptor Toolkit, which allows you to write custom adaptors, is a separately licensed feature.

Adaptor concepts

This topic gives you some general information about adaptors. It also describes how adaptors interact with other Build Forge objects and features.

Get started with adaptors by reviewing the information provided in this section.

Adaptor

An adaptor is an interface to an external application. Adaptors allow a Build Forge project to exchange information with an external application to accomplish a goal.

For example, the adaptor for a source code application checks a repository for source code changes as a prerequisite to running a Build Forge project. If source code files have changed, the project runs. If there are no changes, the project does not run.

Adaptor template

An adaptor is an instance of an adaptor template. When you create an adaptor, you assign it a unique name and associate it with a template.

The template is an XML file. The XML contains application commands to gather information, instructions for analyzing information, and format details for displaying results in the BOM report.

The templates provided by Build Forge are designed to be used without modification. But, you can modify templates or use templates as a model for creating a new adaptor template.

The adaptor templates are installed in the following directory:

```
bf-install\interface
```

Adaptors and projects

To execute adaptor code and interface with an external application, the adaptor must be added to a Build Forge project.

Adaptors are added to projects using a dot command or an adaptor link.

Any adaptor can be added to a project using the dot command for the application type: .source, .defect, .test, or .pack.

Only source code adaptors can be added to a project with an adaptor link. An adaptor link is used instead of the .source command to connect the adaptor to the project.

Document # SC23-7681-08

Adaptors and environment variables

The adaptor requires environment variables to execute application commands. In the adaptor templates, environment variables are listed in the `<env>` elements in the `<template>` section of the XML file.

For example, for the ClearCaseBaseline adaptor, the following environment variables are listed in the ClearCaseBaseline.xml file:

```
<template>
<!-- Template section, these are parsed out of the final xml.
Use the list below to help identify the variables needed to run this interface
if you are integrating it during a regular BuildForge step.
-->
<env name="VIEW" value="my_adaptor_view" />
<env name="VOB_PATH" value="\AdaptorVob" />
<env name="CCSERVER" value="BFServerName" />
</template>
```

In Build Forge, environment variables are stored in environments. Before creating an adaptor, you create an environment for application environment variables.

Adaptor link

An adaptor link connects an adaptor to a project and associates an environment to an adaptor.

Adaptor links work only with source code adaptors. You can connect any adaptor type to a project with a dot command.

The adaptor link has the following features:

- Adds adaptor code to the project as step 0 (tests for source code changes *before* running other project steps).
- Automatically populates an environment with application environment variables.
- Allows you to control whether the adaptor code is executed by selecting a state: active, inactive, debug.
- Allows you to run the adaptor as a manually started job.

The adaptor link has the following restriction:

- An adaptor link is defined for one adaptor and one project only. If you wish to use the same adaptor with another project, you must create another instance of the adaptor.

Adaptors and notification

Most adaptor templates send e-mail notification to users. For example, when the ClearCaseByDate adaptor executes, it sends pass e-mail notification to users who changed source code files. If no files were changed, it sends a fail e-mail notification.

You can optionally modify notification for adaptors:

Document # SC23-7681-08

- In the adaptor template, duplicate the `<adduser>` element to add users to the adaptor notification group.
- In the adaptor template, use the `<notify>` element to add or delete notification messages.
- For adaptor projects, set up project-level notification.
- For adaptor dot-command steps, set up step-level notification.

Important

If the `<notify>` directive fails (for example, if the user the email is addressed to doesn't exist), the XML will fail, and all subsequent notifications will fail.

Adaptors and job execution

Adaptor projects that use an adaptor dot-command can be started as a scheduled job or started using any of the manual start options for projects.

Adaptor-linked projects are typically run on a schedule. But, you can manually start an adaptor-linked project if you complete some additional setup. See Manually starting adaptor-linked projects.

Adaptors and job results

For adaptor dot-command projects, view job results in the step log or in the BOM report, as follows:

- Select **Jobs** → **Completed** . Select the tag for the job to view its step log.
- Select **Jobs** → **Completed** . Select the BOM tab to display the BOM report and view job results by category.

For adaptor-linked projects, view job results in the Source Changes category of the BOM report, as follows:

- Select **Jobs** → **Completed** . Select the BOM tab. In the BOM report, locate the Source Changes category.

Adaptor task overview

This topic covers all of the tasks required to create a source code adaptor, connect it to a project with an adaptor link, and run the adaptor-linked project in test mode.

Creating an adaptor by selecting a template

To create an adaptor by selecting a template:

1. Select **Projects** → **Adaptors.**
2. Click **Add Adaptor** .

3. At Name, enter a unique name for the adaptor. The adaptor name must be unique across the entire set of adaptors.
4. At Type, select the adaptor type.
5. At Template, select the template. The list contains the adaptor templates installed with the Build Forge product. ClearCase and ClearQuest adaptors do not require a separate license key. Other adaptors are separately licensed through the Adaptor Toolkit.
6. At Access, select an access group. The ability to view or edit the adaptor is restricted to these group members.
7. Click **Save Adaptor** .

Creating an empty environment

To create an empty environment:

1. Select **Projects → Environments.**
2. Click **Add Environment**.
3. At Name, enter the environment name. Assign a name that describes the purpose of the environment.
4. At Access, select an access group. The ability to view or edit the environment is restricted to these group members.
5. Click **Save Environment**.

Adding adaptors to projects

To add the adaptor to a project:

1. Select **Projects → Adaptor Links.**
2. Click **Add Adaptor Link**.
3. At Adaptor, select the adaptor (and adaptor template) that you created.
4. At Project, select the project. The list displays the projects that are not already linked to an adaptor.
5. At State, select **Active**.
6. At Environment, select the empty environment that you created for the adaptor link.
7. At Populate Environment, select **Yes**. The application environment variables in the adaptor template are added to the environment.

Document # SC23-7681-08

8. Click **Save** to link the adaptor to the project. The adaptor and the project are added to the list of adaptor links.

Editing environment variables

To edit environment variables:

1. Select **Environments.**
2. For the environment you created, click the  icon. The main content pane displays the adaptor environment variables automatically added to the environment.
3. Review the default values for the environment variables provided by the adaptor template.
4. Change the default values for your source code application as necessary to run the adaptor project.

Condition attribute

The condition attribute allows conditions to be applied to some adapter properties using Perl comparison operators. String literals, numbers, or variables can be used for the comparison.

The syntax for the condition attribute is : condition="<type>(<lvalue> <comparison operator> <rvalue>)"

The condition attribute allows for four types for the test results:

1. true - uses an expression to evaluate whether the lvalue and rvalue are equal.
2. false - uses an expression to evaluate whether the lvalue and rvalue are not equal.
3. hastext - looks at the size of a variable, or entry and returns true if larger than 0.
4. isempty - same as above but checks to see if the length is 0.

The lvalue and rvalue can be strings, numbers, or variables containing strings or numbers. The condition operator is any Perl compatible condition operator. There are string and number condition operators. You must use the appropriate operator or you will receive unpredictable results.

String Operators	Numeric Counterparts
eq	==
ne	!=
gt	>
lt	<
ge	>=
le	<=

Document # SC23-7681-08

Using numeric operators with strings will not return correct results, and the same holds true for using string comparison operators on numeric values.

For example: condition="true("PASS"=="FAIL")

The above condition will always return true, which is incorrect.

Examples of conditions:

- condition="true($BF_SERVER eq "TEST_BOX") - Runs the item only if the build server variable contains TEST_BOX.
- condition="false($BF_BID <=141) - Runs the build only if the build tag is greater than 141, or not less than 141.

Adapter properties which support conditions:

- adduser
- bom
- run
- setenv

Double check the DTD for your current install of Build Forge for up to date information on which properties support the condition attribute.

You will see an entry similar to the following for properties supporting conditions:

<!ATTLIST adduser condition CDATA #IMPLIED>

The adapter DTD is located in %BF_HOME%\interface for Windows and $BF_HOME/Platform/interface for UNIX as the file interface.dtd.

Running adaptors in test mode

To run the adapter in test mode:

1. Select **Administration → System.**
2. In the list of system configuration parameters, select **Link Debug Mode** .
3. At Link Debug Mode, select **Yes** .
4. Click **Save**.
5. Select **Jobs → Start.**
6. In the list of projects, select the adaptor-linked project you created from the Start Project page.
7. Click **Execute**.

Document # SC23-7681-08

View job status and logs

To view the job status and log information for the adaptor project:

1. Open **Jobs** .
2. In the list of projects, locate the adaptor-linked project to view job pass/fail status.
3. To view job logs:
    - Select the Tag Name for the adaptor project to access job log information.
    - Select the Bill of Materials to access the BOM report.

Core adaptor tasks

This topic gives you information about creating and configuring adaptors:

- **Selecting a Template**: describes each of the adaptor application templates.
- **Creating an Environment**: describes options for associating an environment with an adaptor.
- **Creating an Adaptor**: describes how to create an adaptor and associate it with a template.
- **Adding an Adaptor to a Project**: describes options for adding an adaptor to a project.
- **Testing the Adaptor**: describes how to test the adaptor configuration only.

This topic also gives you information about other adaptor tasks:

- **Setting the Adaptor Log Level**: describes how to control the quantity of information logged for the adaptor.
- **Quick Start for an Adaptor-Linked Project**: describes setup required for manual start.
- **Resetting Adaptor Templates**: describes when reset is required to update template information.

Sample adaptor templates

Sample adaptor templates are provided for several application types: source code, defect tracking, testing, and packaging. Adaptors are classified by the type of application they support.

The Build Forge product provides sample adaptor templates for the applications in the following table. The templates for ClearCase and ClearQuest do not require a separate license key. Other application templates are licensed through the Build Forge Adaptor Toolkit.

Adaptor templates XML files are installed in the following directory. All adaptor templates have an .xml file extension.

```
<bf-install>/interface
```


Document # SC23-7681-08

Adaptor template descriptions

Adaptor template name	Description	Type
ClearCaseBaseline	Scans a directory in a ClearCase view. Writes branch and version information reported by ClearCase to the BOM report.	source
ClearCaseByBaselineActivities	Creates a new baseline from the contents of a ClearCase view. Compares the new baseline and the baseline from the previous adaptor execution to identify change activity. For each change activity, writes the following information to the BOM report: activity, files changed, user, date, comments, and version. For each changed file, writes change details (from diff command output) to the BOM report.	source
ClearCaseByBaselineVersions	Creates a new baseline from the contents of a ClearCase view. Compares the new baseline and the baseline from the previous adaptor execution to identify changed files. For each changed file, writes the following information to the BOM report: file name, version, date, user, and comments. For each changed file, writes change details (from diff command output) to the BOM report.	source
ClearCaseByDate	Queries a ClearCase view for changes between two dates. The default dates are the current timestamp and the timestamp of the previous adaptor execution. For each changed file, writes the following information to the BOM report: file name, version, date, user, and comments. For each changed file, writes change details (from diff command output) to the BOM report.	source

Document # SC23-7681-08

Adaptor template name	Description	Type
ClearCaseByLabel	Creates and applies a new label to the contents of a ClearCase view. Compares the new label and the label from the previous adaptor execution to identify changed files. For each changed file, writes the following information to the BOM report: file name, version, date, user, and comments. For each changed file, writes change details (from diff command output) to the BOM report.	source
ClearQuestClearCaseByActivity	Finds ClearQuest defect records associated with a list of ClearCase activities. For each defect record found, it adds job information to resolve the defect record within ClearQuest if the ClearQuest status allows it to be resolved. Writes the following information to the BOM report: files associated with ClearCase activity IDs and the ClearQuest defect status.	defect
ClearQuestClearCaseByDate	Queries a ClearCase view for changes between two dates. The default dates are the current timestamp and the timestamp of the previous adaptor execution. For each changed file, looks for a CrmRequest hyperlink attribute that identifies a ClearQuest change ID. Attempts to resolve the change ID by adding job information to resolve the defect record in ClearQuest if the ClearQuest status allows it to be resolved. For each changed file, writes the following information to the BOM report: the file name, defect ID, defect status, and any ClearQuest errors.	defect
CVSv1Baseline	Scans a CVS directory on a Build Forge agent looking for changed files. Writes the following information to the BOM report: changed file name, status, working version, repository version, and sticky tag.	source
CVSv1ByDate	Queries a CVS view for changes between two dates. The default dates are the current timestamp and the timestamp of the previous adaptor execution. Writes the following information to the BOM report: change type, date, user name, version, and file name. For each changed file, writes change details (from diff command output) to the BOM report.	source

Document # SC23-7681-08

Adaptor template name	Description	Type
CVSv1ByTag	Applies a new tag to a CVS module. Compares the differences between the newly tagged module and a module tagged during the previous adaptor execution. Writes the following information to the BOM report: file name, revision, state, date, time, change author, and commit comments. For each changed file, writes change details (from diff command output) to the BOM report.	source
CVSv2ByDate	Queries a CVS view for changes between two dates. The default dates are the current timestamp and the timestamp of the previous adaptor execution. Writes the following information to the BOM report: change type, date, user name, version, and file name. For each changed file, writes change details (from diff command output) to the BOM report.	source
JazzSCM	Queries Rational Team Concert for changed source files. Writes the following information to the BOM report: source file changed, size of changed file, component changed, and changeset.	source
PerforceByDate	Queries a Perforce client for changes that occurred since the adaptor execution. Writes the following information to the BOM report: change, date, time, user, Perforce client, and comments. Writes change details (from diff command output) to the BOM report.	source
PerforceByRev	Queries a Perforce client for changes that occurred since the last repository revision. Writes the following information to the BOM report: change, date, time, user, Perforce client, and comments. Writes change details (from diff command output) to the BOM report.	source

Document # SC23-7681-08

Adaptor template name	Description	Type
Quota	Queries a Windows folder to determine if any of its subfolders exceed a specified threshold size. For each subfolder, writes the following information to the BOM report: folder size, owner, and last modified date. Writes to the BOM report a list of subfolders that exceeded the threshold size. **Important** Notification is sent to the users who own the exceeded directory in this order: first, by name within the Build Forge system; and afterwards, by user name. If the notified user does not exist, <notify> will fail.	source
StarTeamBaseline	Queries the folder for a StarTeam view to gather information about files. Writes the following information to the BOM report: file name, status, revision, and branch.	source
StarTeamByDate	Uses the StarTeam API to query a StarTeam view to identify changes between the current date and the previous adaptor execution. Writes the following information to the BOM report: changed files and directories, user, version, date, and change comments. Writes change details (from diff command output) to the BOM report.	source
SubversionByDate	Queries Subversion for repository changes that occurred between a past date and the current date. Writes the following information to the BOM report: change type, revision, user, file or directory, and change date. Writes the following information to the BOM report: file name, status, revision, and branch.	source

Document # SC23-7681-08

Adaptor template name	Description	Type
SubversionByRev	Queries Subversion for changes to a repository that occurred between the current revision and an earlier revision. For each change, writes the following information to the BOM report: revision, user, change type, file or directory path, and change date. Writes change details (from diff command output) to the BOM report.	source
VSSByDate	Queries a Visual Source Safe directory for changes between an earlier date and the current date. Writes change information for projects and files to the BOM report: project or file, version, user, date, time, project activity, file project and action. Writes change details (from diff command output) to the BOM report.	source

Creating an environment for the adaptor

Adaptors require environment variables to execute application commands. In the adaptor templates, environment variables are listed in the `<env>` element in the `<template>` section of the XML file.

Do not edit environment variables in the adaptor template files. In the Build Forge product, you define environment variables for an adaptor using an environment.

Use an existing environment or create an environment exclusively for adaptor use. Creating one is recommended because it allows you to assign a specific use and descriptive name to the environment; it also simplifies troubleshooting.

This topic describes how to associate an environment with an adaptor:

- through an adaptor link.
- in the step associated with an adaptor dot command.

Creating an environment for an adaptor link

Use this method if you are linking the source code adaptor to the project using an adaptor link.

1. Select **Projects** → **Environments** .
2. Click **Add Environment** .
3. At Name, enter the environment name. Assign a name that describes the purpose of the environment.

Document # SC23-7681-08

4. At Access, select an access group. The ability to view or edit the environment is restricted to group members only.

5. Click **Save Environment** . Do not add environment variables to the group at this time. They are automatically populated from the adaptor template when you create an adaptor link.

Creating an environment for adaptor dot commands

Use this method if you are adding the adaptor to a project using an adaptor dot command.

For this task, you need the environment variables for your adaptor.

In the <*adaptor_name*>.xml file, external environment variables are listed in the `<template>/<env>` elements.

Locate adaptor templates in the following directory:

`<bf-install>\interface`

1. Select **Projects** → **Environments** .

2. Click **Add Environment** .

3. At Name, enter the environment name. Assign a name that describes the purpose of the environment.

4. At Access, select an access group. The ability to view or edit the environment is restricted to group members only.

5. Click **Save Environment** .

6. Click **Add Environment Variable** .

7. At Name, enter the environment variable name as it appears in the XML `<env>` element.

8. At Value, change the substitution variable in the XML `<env>` element to a real value for your application (If you do not know the correct value, you can enter it later).

9. At Action, select **Set** .

10. At On Project, select **Normal** .

Creating an adaptor from a template

Every adaptor is a based on an adaptor template. Decide which template you want to use before creating an adaptor.

To create an adaptor, do the following:

1. Select **Projects** → **Adaptors** .

2. Click **Add Adaptor** .

3. At Name, enter a unique name for the adaptor. The adaptor name must be unique across the entire set of adaptors.

4. At Type, select the adaptor type.

5. At Template, select the template. The list contains the adaptor templates installed with the Build Forge product. ClearCase and ClearQuest adaptors do not require a separate license key. Other adaptors are separately licensed through the Adaptor Toolkit.

6. At Access, select an access group. The ability to view or edit the adaptor is restricted to these group members.

7. Click **Save Adaptor** .

Adding an adaptor to a project

To execute the adaptor code, you must add the adaptor to a project. Create a new project or add the adaptor to an existing project.

This section tells you how to add an adaptor to a project by using the following methods:

- Source code adaptors may be added to a project using an adaptor link.
- Any adaptor (including source code adaptors) can be added to a project using adaptor dot-commands.

Adding a source code adaptor with an adaptor link

The Adaptor link connects the adaptor to a project and connects application environment variables to the adaptor.

Before starting this task, create a project and an environment for the adaptor.

After completing this task, open the environment and provide real values for application environment variables if you have not already done so.

1. Select **Projects** → **Adaptor Links** .

2. Click **Add Adaptor Link** .

3. At State, select the state:

State	Description
Active	Runs the adaptor code when the project runs.
Inactive	Skips the adaptor code when the project runs.
Debug	Runs the adaptor code only; skips the other steps when the project runs.

4. At Adaptor, select the adaptor template. The list displays the adaptor templates installed with the Build Forge product.

Document # SC23-7681-08

5. At Project, select the project. The list displays the projects not already linked to an adaptor.
6. Click **Save** to link the adaptor to the project. The adaptor name is added to the list.
7. At Environment, select the environment for the adaptor link.
8. At Populate Environment, select **Yes** . The application environment variables in the adaptor template are added to the environment.
9. Click **Save** to save the adaptor link.

Adding an adaptor with a dot command

Any adaptor can be added to a project using an adaptor dot command. The dot commands calls the <*adaptor_name*>.xml file when the step runs.

Before starting this task, create the project and the environment for the adaptor.

After completing this task, open the environment and provide real values for the application environment variables, if you have not already done so.

To add the adaptor dot command to the project as a step, do the following:

1. Select **Libraries** .
2. In the list, select the project.
3. Click **Add Step**.
4. At Name, enter a step name.
5. At Command, enter the adaptor dot command for the application type: .source, .defect, .test, .pack.
6. At Environment, select the environment created for the adaptor.
7. Click **Save Step**.

Testing the adaptor

Run the adaptor project to test the adaptor configuration. To verify that the adaptor can interact with the external application and return the expected results, run the adaptor code in isolation from the other project steps.

This topic tells you how to test adaptors that are:

- added to a project through an adaptor link.
- added to a project with an adaptor dot-command.

Document # SC23-7681-08

Testing a linked adaptor

For source code adaptors linked to the project through an adaptor link, use this procedure to test the adaptor configuration.

The general procedure is as follows:

- make changes to your source files
- run the Build Forge project with the linked adaptor
- check the BOM report for information about changed source files
- check e-mail for pass or fail notification

To test a linked adaptor, do the following:

1. Select **Projects** → **Adaptor Links** .
2. In the list, select the linked adaptor and project.
3. At State, select **Debug** .
4. Click **Save** .
5. In your source code application, make changes to one or more source files. Submit changes to update the source code repository.
6. Run the adaptor-linked project as follows:
    a. Select **Administration** → **System** .
    b. In the list, select **Link Debug Mode** .
    c. At Link Debug Mode, select **Yes** .
    d. Click **Save** .
    e. Select **Jobs** → **Start.**
    f. In the list of projects, select the adaptor-linked project from the Start Project page.
    g. Click **Execute**.
7. Review the BOM report for the job:
    a. Open **Jobs** .
    b. Select the **Completed** tab, then select the **BOM** tab.

Document # SC23-7681-08

Testing an adaptor dot command

For adaptors added to a project with a dot command, use this optional procedure to test the adaptor configuration.

The general procedure is as follows:

- make changes to your source files
- run the Build Forge project with the adaptor dot command
- check the BOM report for information about changed source files
- check e-mail for pass or fail notification

To test the adaptor dot-command, do the following:

1. Select **Jobs** → **Start** .
2. In the list, select the project.
3. Open the **Job Steps** tab.
4. Use the Step Name check box to clear checks for everything except the adaptor dot-command.
5. Click **Execute** to run the project.
6. Review the BOM report for the job:
    a. Select **Jobs** .
    b. Select the **Completed** tab, then select the **BOM** tab.

Setting the adaptor log level

To control how much information is written to the step log for the adaptor, use the _InterfaceLoggingLevel environment variable.

1. Add _InterfaceLoggingLevel to the adaptor's environment.
    - Level 8 logs the most information and level 0 logs the least.
    - Logging levels are inclusive. For example, level 2 includes information from levels 1 and 0.
    - Level 7 is the default logging level.
2. Assign a log level as the value for the _InterfaceLoggingLevel variable:

0: Exec line plus server connection errors or cancel notification; nothing else

1: Parsed commands (commands as they will be sent to the server)

2: Unparsed commands (commands prior to having their local variables set)

3: Build and environment variable SET lines

4: Temp and internal variable SET lines

5: Environment evaluations, e-mail group additions, BOM text logging lines

6: Block and sub-block start/end lines

7: (Default logging level) Agent output that will be checked against match patterns, plus the lines that matched the patterns

8: All agent output

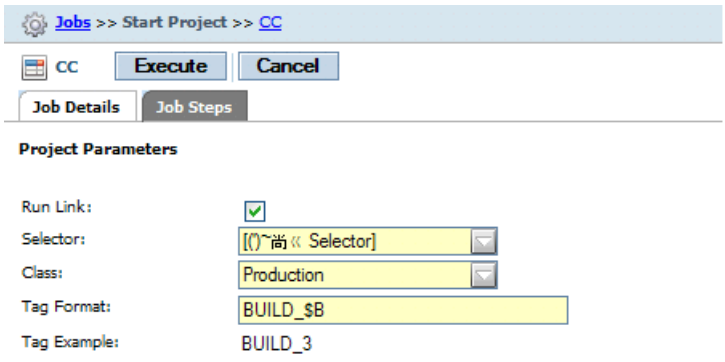
Manually starting adaptor-linked projects

Adaptor-linked projects can be run on a schedule, run using the quick-start option or started manually if you check the Run Link check box. If you do not check Run Link, the project runs without the adaptor step.

1. Select **Jobs → Start.**
2. In the list of projects, select the name of the adaptor-linked project.

   The Start Project page opens.
3. Check the **Run Link** check box.
4. Click **Execute** from the Start Project page.

Run Link Check Box



Quick start for adaptor-linked projects

Adaptor-linked projects can be run on a schedule, started manually by selecting the project name from the Start Project page (Jobs → Start), or run using the quick start icon if you complete some additional setup:

1. Select **Administration → System.**

2. In the list, select **Link Manual Jobs.**
3. At Link Manual Jobs, select **Yes.**
4. Click **Save** .
5. Select **Jobs** → **Start.**
6. In the list of projects, select the Quick Start icon for the adaptor-linked project to run it immediately. To verify that the job is running, select the Jobs > Running tab.

Resetting adaptor templates

Resetting adaptor templates copies the latest adaptor templates from the `<bf-install>\interface` directory to the Build Forge database.

Reset adaptor templates whenever you:

- install a maintenance release or a new product version
- modify a template by editing the version in the interface directory (not in the Management Console)
- create a new adaptor template

To reset the adaptor templates, do the following:

1. Select **Administration** → **System** .
2. In the list, select **Reset Interface Templates** .
3. At Reset Interface Templates, select **Yes** .
4. Click **Save** .

Updating ClearQuest build records

You can integrate the system with Rational ClearQuest to automatically create and update build records within a ClearQuest database.

The system can automatically create build records in your IBM Rational® ClearQuest® database, with links to the log data. Furthermore, when a job passes, the system can update the ClearQuest database, noting that the job is complete, recording the end time and a summary of the steps that were accomplished. This feature requires Rational ClearQuest version 7.0 or later.

Note

To successfully create the build record, and populate it with information about the build, you must first open the ClearQuest Maintenance Tool and set up a connection to the database.

Document # SC23-7681-08

When you configure a project to update a ClearQuest database, the system performs the following:

Job start — When the system launches a job, the system creates a ClearQuest build record. The build record is in the Submitted state and includes the job log URL, the start time, release name, and ID as well as a log entry indicating "Build XYZ started." If a source control adaptor cancels the job (because no source changes are found, for example), no ClearQuest build record is created.

Note

If a project chains another project, the new project gets its own unique ClearQuest build ID.

Job pass/fail — When a job passes or fails, the system changes the build state within ClearQuest to Completed or Failed, sets the build end time, and stores a summary of the job's steps in the ClearQuest build log. The summary includes the name, result status, and server for each step.

Job restart — When a job is restarted, the system changes the build state within ClearQuest to Submitted and creates a ClearQuest build log entry indicating “Build XYZ restarted.”

You configure the ClearQuest integration through special environment variables. To link a project to a ClearQuest database, make sure the following variables are included in the project's environment.

Note

These variables must be present in the project environment. Adding them to a step is not sufficient. You can use a variable that is set to type Include that includes these variables through another environment. Also, because the CQ_RELEASE_NAME value is the only one likely to vary per project, you may want to create an environment that contains the other variables, then use a variable of type Include to include that environment in the project environment, where you can also specify CQ_RELEASE_NAME as a project-specific environment variable.

Document # SC23-7681-08

Environment variables required for Rational ClearQuest integration

Variable	Description
CQ_DBNAME	Name of the ClearQuest database you want to update.
CQ_INTERACTION	If your project environment has the correct environment variables defined to enable the creation of a ClearQuest build record, but you do not want to create the build record, set this variable to OFF to disable build record creation. To enable build record creation, set this environment variable to ON. **Note** If you are using one of the ClearQuest adaptors, set this environment variable to OFF.
CQ_USER	Username to use when logging into the ClearQuest database.
CQ_RELEASE_NAME	The name of the release within the ClearQuest database that you want to update.
CQ_PASSWORD	Password to use when logging into the ClearQuest database. Not required; defaults to blank.
CQ_DBSET	The ClearQuest database set value. Not required; defaults to blank.

Advanced adaptor tasks

You can modify a Build Forge adaptor template or create a new adaptor template for an external application that you want to interface with a Build Forge project.

Modifying and creating templates are advanced tasks that require a working knowledge of:

- the XML language
- the command language for the external application
- regular expressions

Before working on advanced tasks, read the information in the following sections:

- Core adaptor tasks
-
- Adaptor reference

The section gives you information about the following tasks:

- **Modifying an Adaptor Template**: describes how to modify a template for all adaptors.
- **Modifying a Template for Single Adaptor**: describes how to modify a template for one adaptor.

- **Creating a New Adaptor Template**: describes the general procedure for creating a new template.
- **Example: Adding a User to Adaptor Notification**: describes how to add an access group to e-mail notification.
- **Example: Removing Change Details from a BOM Report**: describes how to remove change details from the BOM report.

Modifying adaptor templates

Use this procedure if you want template modifications to be picked up by all future adaptors created from the adaptor template.

Before you begin, know what you want to modify. For example, you may want to change notifications or modify the BOM report format.

1. Using an XML editor, open the adaptor template that you want to modify. Adaptor templates are located in the following directory:

   ```
   <bf-install>\interface
   ```

2. Enter your template modifications.
3. Save the adaptor template. Do **not** change the template name. Adaptors and adaptor templates must have unique names.

The next time you create an adaptor with the modified adaptor template, the new adaptor will include your template modifications.

Modifying a template for a single adaptor

Use this procedure to modify a template for a single instance of an adaptor only.

Before you begin, know what you want to modify. For example, you may want to change notifications or modify the BOM report format.

If the adaptor you want to modify has been created, make your template modifications through Management Console, as follows:

1. Select **Projects** → **Adaptors** .
2. In the text box, enter your modifications to the template.
3. Click **Save Adaptor** .

Document # SC23-7681-08

Note

Your changes are saved only to the instance of the adaptor template associated with the adaptor in the Build Forge database. Changes are not saved to the adaptor template file in the `interface` directory.

If the adaptor you want to modify has not been created, modify the template before you create the adaptor, as follows:

1. Using an XML editor, open the adaptor template. Adaptor templates are located in the following directory:

   `<BuildForge_directory>\interface`

2. Enter your modifications to the template.
3. Change the adaptor name and then save the adaptor template.

Note

Adaptors and adaptor templates must have unique names.

4. Reset the adaptor templates to pick up your modified template and add it to the list of available templates in the Management Console. See Resetting adaptor templates.

Creating a new adaptor template

Use this procedure to create a new adaptor template for an external application that you want to interface with a Build Forge project.

1. Review the XML structure and elements in the adaptor templates provided by the Build Forge product. Adaptor templates are located in the following directory:

   `<bf-install>\interface`

2. Plan what you want the new adaptor to do:
   - know what commands it will run
   - determine how the commands will be parsed
   - determine what to do with the data gathered from the parsed results
   - know which external environment variables are required
3. Select one of the adaptor templates in the interface directory to use as a model. If possible, select an adaptor based on the same external application. Or, select an adaptor that has a function similar to the one you are creating. Use the XML hierarchy, elements, and element attributes in the model as a guide to create the new template.

4. Using an XML editor, open the model adaptor template. Save the template to the interface directory with a new name.
5. Using your plan, write the XML code for the new adaptor.
6. Save the new adaptor template.
7. Validate the adaptor template using the interface.dtd file in the interface directory.
8. Reset the adaptor templates to pick up your new template and add it to the list of available templates in the Management Console. See Resetting adaptor templates.
9. Create a project for the adaptor.
10. Create an environment for the adaptor. See Creating an environment for the adaptor.
11. Create an adaptor using the new adaptor template. See Creating an adaptor from a template.
12. Add the adaptor to the project. See Adding an adaptor to a project.
13. Run the adaptor project to test the adaptor. See Testing the adaptor.

Creating multiple entry point adaptors

The adaptor templates provided by Build Forge are single entry point adaptors.

For single entry point adaptors, in the Management Console (Projects > Adaptors), you select the template name for the application and function that you want to run. For example, ClearCaseBaseline or ClearCaseByDate.

If you prefer you can create one adaptor template for ClearCase that contains multiple interfaces or functions for ClearCase. For a multiple entry point adaptor, you identify each interface by a name, called an entry point.

To create a multiple entry point adaptor:

1. Create the adaptor template. To create the template, you have the option of using one of the provided templates as a model and modifying the XML as necessary. In the template, you must add the name attribute to the <interface> element to identify the entry point for each interface that you add to the template. The related syntax for the interface element is shown in the following example:

   ```
   <interface name="By Date" default="true">
   </interface>
   ```

2. Create an adaptor with a unique name and associate it with the adaptor template. See Creating an adaptor from a template.
3. Add the adaptor to the project using an adaptor link or an adaptor dot command.

   - The following example uses the .source adaptor command to add the adaptor to a project step which calls the By Date interface function in the ClearCase adaptor:

```
.source ClearCase "By Date"
```

- To use an adaptor link to call a multiple entry point adaptor, take one of the following actions in the adaptor template to specify which interface function gets executed when the project runs:
  - place the interface element definition for the function to execute as the first interface element in the template file
  - set the default attribute for the interface element to execute to true (default="true")

Example: enabling e-mail notification

Adaptor templates can be configured to send e-mail notification to users who cause a change in the external application. The following example shows how to set up two types of notification:

- Notify all users who checked in files for the current build
- Notify all members of a Build Forge access group

The following procedures reference elements in the ClearCaseByDate template. Any adaptor template can use their elements to enable notification.

Notify all users who checked in files for the current build

You can use the `<adduser>` command to dynamically build the group of users who checked in code for the build, then use the `<notify>` command to send notifications to that group.

The ClearCaseByDate template queries ClearCase for a view of all changes between two timestamps. The default timestamps are for the current adaptor run and the last adaptor run. In practical terms, that translates to a list of all changes since the last build that are checked in for the current build.

Assumption: all user names in the view are known to the SMTP server you use for notification by that name. That means ClearCase user names need to align with e-mail user names.

To enable this notification:

1. Open the ClearCaseByDate adaptor template in an XML editor.
2. Find and edit the `<adduser>` to create a group of users, as follows:

   ```
   <adduser group="MyChangers" user="$4">
   ```

   The positional parameter `$4` refers to the user name field that is shown in the ClearCase view generated by the ClearCaseByDate template.
3. Set up notification to send e-mail to this group. The following setup sends e-mail both when the project fails and when the project succeeds. In some environments you may prefer to notify only if the build fails.

   ```
   <!-- Set some notifications for when the build completes -->
     <onproject result="fail">
   ```

```
   <notify group="MyChangers" subject="Build $BF_TAG ($CurDate) Failed."
message="${Changing}${Changes}"/>
  </onproject>
  <onproject result="pass">
   <notify group="MyChangers" subject="Build $BF_TAG ($CurDate) Passed."
message="${Changing}${Changes}"/>
  </onproject>
```

4. Save the adaptor template.

When the adaptor runs, the MyChangers group is built from the user names in the view. E-mail notification is sent to that group when the build project completes.

Notify all users who belong to a Build Forge access group

In this example you want to notify all members of a Build Forge access group. The ClearCaseByDate adaptor template is used for the example. Assumption: all user names in Build Forge correspond to e-mail user names in the SMTP server.

1. Open the ClearCaseByDate adaptor template in an XML editor.
2. Find and edit the `<adduser>` to create a group of users from a Build Forge access group, as follows:

   ```
   <adduser group="Developer_Access_Group" user="Developer">
   ```

3. Set up notification to send e-mail to this group. The following setup sends e-mail both when the project fails and when the project succeeds. In some environments you may prefer to notify only if the build fails.

   ```
   <!-- Set some notifications for when the build completes -->
   <<<<<<< .mine
     <onproject result="fail">
      <notify group="Developer_Access_Group" subject="Build $BF_TAG ($CurDate) Failed."
    message="${Changing}${Changes}"/>
     </onproject>
     <onproject result="pass">
      <notify group="Developer_Access_Group" subject="Build $BF_TAG ($CurDate) Passed."
    message="${Changing}${Changes}"/>
     </onproject>
   ```

4. Save the adaptor template.

When the adaptor runs, the Developer_Access_Group group is built from the user names that belong to the Developers access group. E-mail notification is sent to that group when the build project completes.

Important

If the `<notify>` directive fails (for example, if the user the e-mail is addressed to doesn't exist), the .xml will fail, and all subsequent notifications will fail.

Document # SC23-7681-08

Example: removing change details from a BOM report

Most adaptor templates log change details to the BOM report. (The diff command is used to log change details.)

The following steps reference elements in the ClearCaseByDate template, but they can be used to remove change details for any adaptor template.

To remove change details in the BOM report, do the following:

1. Open the adaptor template in an XML editor.
2. Find the `<run>` element that calls the diff command. Remove the following line:

```
<run command="cc_diff" params="$VIEW $1 $2" server="$CCSERVER" dir="/" timeout="360"/>
```

3. Find the `<command>` element for the diff command. Remove the following lines:

```
<!-- The cc_diff command does a generic clearcase diff, logging the full output
of the diff in the BuildForge BOM -->
<command name="cc_diff">
<execute>
pushd \\view\$1 && cleartool diff -pred -diff_format "$2@@$3"
</execute>
<resultsblock>
<match pattern=".+">
<bom category="Source" section="diff">
<field name="diff" text="$_"/>
</bom>
</match>
<resultsblock/>
</command>
```

4. Find the `<bomformat>` section, then find the `<section>` element for the diff command output. Remove the following lines:

```
<section name="diff">
<field name="diff" title="Change Details"/>
</section>
```

5. Save the adaptor template.

Adaptor reference

Adaptors are designed to be added to Build Forge projects and run without modification. To modify an adaptor or create a new adaptor, you need to understand the XML template structure and elements used in the Build Forge adaptor templates.

Document # SC23-7681-08

Note

This section does not describe the external application commands used in the Build Forge adaptor templates. For information about these commands, consult the documentation for the external application.

Adaptor templates installed with the Build Forge product are located in the following directory:

```
<bf-install>\interface
```

This section provides the following reference information:

- **Adaptor Requirements**: describes general requirements for using adaptors and specific requirements for ClearQuest adaptor templates.
- **Dot Commands for Adaptors**: describes the syntax for the adaptor dot commands.
- **ClearCase and ClearQuest Environment Variables**: describes the environment variables used by the ClearCase and ClearQuest adaptors.
- **Perforce Environment Variables**: describes some additional environment variables required for Perforce.
- **Adaptor Template Structure**: describes the general structure of the Build Forge adaptor template.
- **Adaptor XML Reference**: describes the XML elements used in the Build Forge adaptor templates.

Adaptor requirements

This topic identifies installation and configuration requirements and software requirements for Build Forge adaptors.

- **General Adaptor Requirements**: requirements for all adaptors.
- **ClearQuest Requirements**: requirements for using the ClearQuestClearCaseByActivity and ClearQuestClearCaseByDate adaptor templates.

General adaptor requirements

Verify that your environment meets the requirements for using any adaptor provided with the Build Forge product:

- Install a Build Forge agent on the machine where the external application is running.
- Install a license key for the Build Forge Adaptor Toolkit if you want to use application templates other than ClearCase or ClearQuest.
- Run an external application version that is supported by the Build Forge product.

Document # SC23-7681-08

Application Versions Supported for Adaptors

Application	Version
Rational ClearCase	6.0 or later
Rational ClearQuest	7.0 or later
CVS	1.1, 1.2
Microsoft Visual SourceSafe	6.0
Perforce	2005.1
StarTeam	2005 Release 2
Subversion	1.3.1 and later

ClearQuest adaptor requirements

For ClearQuest adaptors, there are additional installation and configuration requirements:

- Install the ClearQuest application on the Management Console system.
- Install the CQperl utility (ClearQuest Perl API) on the Management Console system and add the CQperl installation directory to the system path statement.
- Add the CQ_INTERACTION environment variable to your project and set it to OFF.
- Open the CQ Maintenance Tool and set up a connection to the database. This ensures that the Build Forge Management Console can successfully create the build record, and populate it with information about the build.

Dot commands for adaptors

Some dot commands allow you to add an adaptor for an external application to a Build Forge project as a project step.

- **.source**: adds the adaptor for a source code application to a project step.
- **.defect**: adds the adaptor for a defect tracking application to a project step.
- **.test**: adds the adaptor for a testing application to a project step.
- **.pack**: adds the adaptor for a packaging application to a project step.

See also “Dot command reference” on page 120.

Rational ClearCase and Rational ClearQuest environment variables

The Rational ClearCase and Rational ClearQuest adaptor templates use the environment variables in the following table to execute cleartool commands.

Not every environment variable in the following table is required for each ClearCase or ClearQuest adaptor template.

In each adaptor template, required environment variables are listed in the `<env>` elements in the `<template>` section.

Before running a ClearCase or ClearQuest adaptor project, supply real values for the required variables or accept defaults. Edit variable values in the environment assigned to the adaptor.

Adaptor templates are located in the following directory:

```
<bf-install>\interface
```

Environment Variables for ClearCase and ClearQuest

Environment variable name	Substitution variable	Description
VIEW	value=my_adaptor_view	Set this variable to the name of the ClearCase view that you want to use with the adaptor.
VOB_PATH	value=\c_vob	Set this value to the name of your component VOB, and optionally, its subdirectories.
PROJECT_VOB	value=\ProjectVob	When you use the ByBaseline adaptor, set this variable to the name of your Project VOB (only used with UCM ClearCase).
CCSERVER	value=BFServerName	Set this variable to the name of a Build Forge server that has the ClearCase client installed and running.
CurDate	value=.date %d-%b-%y.%H:%M:%S	Supplies the current date to the adaptor, using a .date command to generate the date in the format expected by ClearCase. Do not change this value.
LAST_RUN	value=1-Jan-05.00:00:00	For ByDate adaptors, the system uses this value to determine whether any changes have occurred; it's the date of the last successful run. You can manipulate this value when testing the adaptor to force the adaptor to run, by picking a date that you know precedes some changes. If the adaptor allows the run to continue, it automatically updates this value to the current date. The default value is 1-Jan-05.00:00:00.
LABEL	value=BUILD_1	For ByLabel adaptors, when you use your adaptor to generate differences by label (with the ByLabel adaptor), the system uses this value as the label.
BASELINE	value=BUILD_1	For ByBaseline adaptors, when you use your adaptor to generate differences by baseline, the system uses this value as the baseline.
ACTIVITIES	value=SAMPL0001@\ProjectVob	For the ClearQuestClearCaseByActivity adaptor, a space-delimited set of activity IDs.

Document # SC23-7681-08

Perforce environment variables

Add the following Perforce environment variables to the Build Forge environment group that is assigned to the Perforce adaptor:

- P4USER
- P4PASSWD

To access the Perforce server, Build Forge requires a valid user name and password. In the step log, the Perforce user name and password are written in plain text.

You cannot use the Assign Hidden property for environment variables to encrypt Perforce authentication information.

Adaptor template structure

This topic describes the general XML structure or element hierarchy in the Build Forge adaptor templates.

The adaptor template is made up of the following section elements: `<template>`, `<interface>`, `<command>`, and `<bomformat>`. Each of these sections contains child elements.

For element descriptions, see the “Adaptor XML reference” on page 268.

```
<PROJECT_INTERFACE>
<template>
<env/>
</template>

<interface>
<setenv/>
<run/>
<ontempenv>
<step/>
</ontempenv>
<onproject>
<notify/>
</onproject>
</interface>

<command>
<execute> or <command>
command line
</execute> or </command>
<resultsblock>
<match>
<bom>
<field/>
</bom>
<adduser/>
<setenv/>
<run/>
</match>
</resultsblock>
</command>
```


Document # SC23-7681-08

```
<bomformat>
<section>
<field/>
</section>
</bomformat>
</PROJECT_INTERFACE>
```

Adaptor XML reference

This section lists adaptor XML elements in alphabetical order. It is a reference for the elements used in the adaptor XML language. Some examples and pseudocode are included in the descriptions.

adduser

Use the <adduser> element within a <match> element to add users to a temporary group based on the output of change commands, so that the adaptor can send notifications to users who caused changes. The system does not add a user to a group if the user is already a member of the group, preventing multiple notifications. The <adduser> element is an empty element. The group attribute specifies a temporary group created during the adaptor logic run; you must reference the same group in a <notify> element to cause the actual notifications to be sent.

```
<adduser group="MyChangers" user="$4"/>
```

Use the conditional attribute to control whether the <adduser> element adds users who caused changes to a temporary access group. The value of the conditional attribute is an expression that evaluates to true or false. If the expression evaluates to true, the temporary access group is created; otherwise, it is not created.

bom

A `<bom>` element defines information to be logged to the Bill of Materials (BOM) for the job; it should be enclosed in a `<match>` element. A `<bom>` element must specify a category and section within the BOM, and defines which numbered variables ($1...$n) collected by the `<match>` element should be converted to fields for the BOM data.

```
<bom category="Source" section="changesets" >
        <field name="Change" text="$1"/>
        <field name="Date" text="$2"/>
        <field name="User" text="$4"/>
</bom>
```

Use the conditional attribute to control whether the `<bom>` element is written to the BOM report. The value of the conditional attribute is an expression that evaluates to true or false. If the expression evaluates to true, the information in the `<bom>` element is written to the BOM; otherwise, it is not written.

bomformat

Use the <bomformat> element to define how to display the data collected in earlier <bom> elements. It takes a category attribute that specifies the logical name of a BOM category, and a title attribute

Document # SC23-7681-08

to specify the display name for the category. A structure of <section> elements which contain <field> elements defines the layout, as in the following example:

```
<bomformat category="Detail" title="Change Details">
        <section name="descriptions" parent="section name" expandable="yes">
        <field name="Description" title="Change Description"/>
        </section>
        <section name="diff">
        <field name="Diff" title="Differences"/>
        </section>
```

command

An adaptor XML file can contain several `<command>` elements; each defines a named command that can be referenced by `<run>` elements within `<interface>` elements. The `<command>` elements are specified outside the <interface> elements so that multiple interfaces within an XML file can reuse the same commands.

Commands can call other commands by embedding a `<run>` command in the `match` element in the `<resultsblock>` elements.

The <command> element wraps a structure of `<execute>` and `<resultsblock>` elements as shown below:

```
<command name="p4_changes">
   <execute>
       command line
   </execute>
   <resultsblock>
      Has its own structure.
   </resultsblock>
</command>
```

Alternatively, you can replace the `<execute>` element in the block with an `<integrate>` element.

Use the mode attribute to identified the mode for the `command` element. The mode attribute takes the following values:

- **conjoined**: all calls to the command are grouped in one call for server processing.
- **parallel**: calls are processed individually as server slots become available.
- **exec**: commands are started and immediately processed by the server.

env

The <env> element is used inside a element to define environment variables (with initial values) that can be copied into the environment used with a project link. Each <env> element should include *name* and *value* properties. The value provides an initial value for the variable.

```
<env name="FILESPEC" value="//depot..." />
```


Document # SC23-7681-08

execute

Use the <execute> element inside the <command> element to specify commands. The contents of the element are one or more lines of text to be sent to the server used by the adaptor. You cannot use dot commands in the <execute> element. When a <run> element calls a <command> element, the system replaces any positional parameters in the <execute> element's content with the parameters specified in the calling the <run> element. A $1 parameter in the <execute> element's content is replaced by the first parameter, a $2 parameter is replaced with the second parameter, and so on.

Use the conditional attribute to control the execution of the commands in the <execute> element. The value of the conditional attribute is an expression that evaluates to true or false. If the expression evaluates to true, the system executes the commands; if the express evaluates to false, the commands are not executed.

```
<execute>
p4 changes -s submitted -t -i $2@$1,@now
</execute>
```

field

Use the <field> element within either the <bom> or <section> elements to specify a field.

When used in a <bom> element, specify the name and text; the text attribute defines which variable is used to populate the field with data.

When used in a <bomformat> <section> element, specify the name and title. The name specifies the logical name, while the title is used for display. If there is more than one field in a <section>, include an order attribute.

```
<section name="changesets">
        <field order="1" name="Change" title="Change ID"/>
        <field order="2" name="Date" title="Date"/>
        <field order="3" name="Time" title="Time"/>
        <field order="4" name="User" title="User ID"/>
        <field order="5" name="Client" title="Client"/>
        <field order="6" name="Comment" title="Comment"/>
</section>
```

Use the conditional attribute to control whether the <field> element is written to the BOM report. The value of the conditional attribute is an expression that evaluates to true or false. If the expression evaluates to true, the information in the <field> element is written to the BOM. If the expression evaluates to false, the information is not written to the BOM.

Use the template attribute to define the text format for the <field> element. For example, if the text is a string, the template value might be "Hello $VALUE". When the field is written to the BOM report, $VALUE is replaced with the field text.

integrate

The <integrate> element is similar to the <execute> element. You can use the <integrate> element in place of the <execute> element. Like the <execute> element, the <integrate> element specifies a command line to be run. It has the following differences:

- The command line is executed on the Management Console system, not the server that executes the adaptor.
- The command line uses the `\integration` directory (a subdirectory of the installation directory) as its current directory.

The <integrate> element is useful for executing applications or scripts located on the Management Console machine, especially in the `\integration` directory.

When a <run> element calls the <command> element which contains the <integrate> element, the system replaces any positional variables in the <integrate> element with the parameters specified in the calling <run> element. A $1 in the <integrate> element is replaced by the first parameter, a $2 with the second parameter, and so on.

As with the <execute> element, you cannot use dot commands in an <integrate> element.

The following example, from the IBM Rational ClearQuest adaptor, sends data to ClearQuest by running the CQperl command (a ClearQuest program for executing Perl code) and feeding it the name of a Perl script located in the `\integration` directory. The example assumes ClearQuest is installed on the Management Console system.

```
<integrate>
cqperl bfcqresolve.pl $2 Fixed "Fixed in build $BF_TAG"
</integrate>
```

interface

The <interface> element is a container for an entry point into the adaptor. Elements in it define the program logic of the adaptor. It contains <setenv>, <run>, <ontempenv>, and <onproject> elements.

An adaptor template can have multiple entry points. If you create an adaptor template that has multiple entry points, do the following:

- Use the name attribute to identify each entry point.
- Use the default attribute to identify the one to execute if the adaptor template is called without a name specification.

To specify an entry point, use the entry point name as a parameter on a .source call to the template.

Example: create an adaptor template named MyAdaptorTemplate. In it, place the following code in it to define a named entry point and makes it the default:

```
<interface name="By Date" default="true">
</interface>
```

To call this interface by name, use the following command in a project step:

```
.source MyAdaptorTemplate "By Date"
```

Attributes:

Document # SC23-7681-08

name Optional. The name for this interface. When an adaptor is called using a name as a parameter, the interface whose name matches that parameter is used.

default Optional. Set to either Yes or No. If Yes, the interface is used when the adaptor is called without a name parameter.

match

A <match> element is used within a <resultsblock> element to process lines of output. The <match> element takes a pattern attribute that defines matching lines. The pattern is a Perl regular expression.

The match pattern can include parenthetical expressions, which are stored in the variables $1...$n.

```
<match pattern="^Change (\d+) on (.*?) (.*?) by (.*?)@(.*?) '(.*?)'$">
```

The <match> element uses <bom> and <field> as subelements. See the reference information for"resultsblock" on page 274 to see a more extensive example.

notify

The <notify> element specifies an email and distribution list. It is typically used in an <onproject> element to specify notification based on an execution result.

Attributes:

group The recipient of the email. It is a group of users defined in the adaptor.

message The content of the body of the email. You specify the text.

subject The subject of the email. You specify the text.

onproject

The `<onproject>` element defines notification actions that are performed by the system after the system executes the project steps. The element takes a required result attribute which specifies whether the actions are performed for a passing or failing job. Typically, an adaptor XML file contains two `<onproject>` elements, one for the pass case and one for the failure case. The following example shows a pair of `<onproject>` elements that use notify elements to send different messages depending on whether the project passes or fails:

```
<onproject result="fail">
      <notify group="MyChangers"
      subject="Run $BF_TAG ($CurDate) Failed." message="$Changing$Changes"/>
</onproject>

<onproject result="pass">
     <notify group="MyChangers"
     subject="Run $BF_TAG ($CurDate) Passed." message="$Changing$Changes"/>
</onproject>
```

ontempenv

The <ontempenv> element is used within an <interface> element and acts similar to an if-then statement. Use this element to return a pass or fail value to the project; a pass indicates that the system should continue and run the rest of the project, while a fail indicates it should stop. This is normally used to indicate whether the interface found relevant changes that merit a new run of the project.

After the system executes any commands specified in <run> elements, it processes the <ontempenv> element. Use the name attribute of this element to specify a temporary environment variable, and the state attribute to specify a value.

The <ontempenv> wraps a <step> element, which is executed only if the temporary environment variable name and state exists after the <run> element commands are executed.

```
<ontempenv name="Changes" state="empty">
  <step result="FAIL"/>
</ontempenv>
```

PROJECT_INTERFACE

The <PROJECT_INTERFACE> element wraps all the other tags in the adaptor template. It takes one attribute, IFTYPE, which indicates the adaptor type. Valid types include Source, Test, and Defect.

```
<PROJECT_INTERFACE IFTYPE="Source">
...all other elements...
</PROJECT_INTERFACE>
```

relate

The <relate> specifies a relationship between an artifact and a user. It is used in conjunction with a log filter of type Notify Changers that is defined in a project and used in the Result attribute of a step that runs immediately before the step that calls the adaptor. The Notify Changers filter typically specifies an expected log line that indicates success or failure. When the step runs, the filter is compared to text specified as the artifact in the <relate> element. If there is a match, email is sent to the user associated with the artifact.

See also the description of the Notify Changers filter in “Filter actions” on page 78.

Attributes:

artifact	Text that is matched to text searched for by the log filter.
user	The user associated with the artifact. When there is a match, the system sends email to this user.
text	The text to log to the BOM when there is a match between the log filter and this relationship.

resultsblock

The <resultsblock> element defines how the system should process the results of the command lines executed from the related <execute> element. The <resultsblock> element is only used within a <command> element. The <resultblock> element can be nested to partition results.

The <resultsblock> element can have optional beginpattern and endpattern attributes that use Perl regular expressions to define a range of output lines to process. You can use this to have some portions of the output processed by different <resultsblock> elements. The following pseudocode shows the structure of a <resultsblock>.

```
<resultsblock startpattern="" endpattern="" >
        <match>
        <bom>
         <field/>
        </bom>
        <adduser/>
        <setenv/>
        <run/> (The <run> element can be used to run commands within other commands)
        </match>
        <setenv/>
      </resultsblock>
```

The following example shows how the <resultsblock>, <match>, and <bom> elements work together:

```
<resultsblock
beginpattern="^Change (\d+) by (.*?)@(.*?) on (.*?) (.*?)$"
endpattern="^Differences ...$">
        <match pattern="(?=^(?:(?!Differences ...).)*$).?">
        <bom category="Detail" section="descriptions">
        <field name="Description" text="$_"/>
        </bom>
        </match>
     </resultsblock>
```

run

A <run> element is used within an <interface> element to specify a named command to run. The command is defined later in the same XML file. The <run> element is an empty element. The following attributes are required:

```
<run command="UpdateEnv" params="" server="" dir="/" timeout="360"/>
```

Attributes:

condition — Optional. It specifies condition for the run command. The value of the condition attribute is a function or expression that evaluates to true or false. If it evaluates to true, the command is run. If it evaluates to false, it is not run. See “Condition functions” on page 108.

command — Required. It specifies name of a defined command to run. The command is named and defined in a <command> element.

Document # SC23-7681-08

dir — Required. It specifies the directory in which to run the command. The dir is interpreted as an extension of the path set in the server resource that the command runs on.

mode — Optional. It specifies the run mode for the run command. It can be one of the following:

- **conjoined**: all calls to the command are grouped in one call for server processing.
- **parallel**: calls are processed individually as server slots become available.
- **exec**: commands are started and immediately processed by the server.

params — Required. It specifies parameters to pass to the command. Use spaces to separate the parameters.

server — Required. It specifies the server resource to run the command on. If it is set to null, the command is run on the server used by the step that runs the adaptor. Use `server=""` to set the server to null.

Example:

```
<run command="p4_changes" params="$LAST_RUN $FILESPEC $LAST_VER"
server="$P4CLIENT" dir="/" timeout="360"/>
```

section

Use the `<section>` element to define how to display a portion of a BOM category. It takes a name attribute. You can use the `<section>` element only within `<bomformat>` elements.

setenv

Use the `<setenv>` element to initialize environment variable values within `<interface>` or `<match>` elements. The `<setenv>` element does not contain other elements.

The element can be used in three different ways:

- When you specify a group name, it works similar to the .set command. It sets the variable value in the master record in the database, not the copy used by the current step. The change is not seen by the adaptor running in the current step. You cannot create new variables this way.
- When you do not specify a group name, it works like the .bset command. It sets the variable value in the environment of the running job. The change is available to all the steps in the job. You can create new variables this way.
- When you do not specify the group name *and* specify a temporary variable (type="temp"), it sets up a temporary variable for the use of the adaptor logic only. The variable does not persist after the adaptor step runs. You can create new variables this way.

Attributes:

condition — Optional. It specifies condition for the run command. The value of the condition attribute is a function or expression that evaluates to true or false. If it evaluates to true, the command is run. If it evaluates to false, it is not run. See "Condition functions" on page 108.

eval — Optional. Set to True or False. If true, the adaptor attempts to evaluate the value attribute expression and store the results

group — Optional. It specifies the Build Forge environment that the variable is defined in. When you specify a environment name, you must refer to an existing variable within the specified environment.

If you specify [ADAPTOR] as the value, then the value is set at run time. It is set to the environment of the step or adaptor link that calls the adaptor.

Build Forge allows variables of the same name in multiple environments. The precedence of environment inheritance and environment inclusion can affect how to determine the value to assign to a variable at run time.

name — Required. It specifies the name of the variable to set. The value can be a variable. In that case, the variable name is not set until run time.

type — Optional. It specifies the method of setting the variable. It takes one of the following values:

- append *text*: place the specified value after any existing value. If the optional *text* is specified, that text is placed between the values.
- once: the variable should be set only if it is not already set.
- prepend *text*: place the value before any existing value.
- temp: the variable should be set only in the context of the adaptor. If the optional *text* is specified, that text is placed between the values. See examples below

value — Required. It is the value to set in the variable. It can be an expression to be evaluated if the eval attribute is also specified. The result of the evaluation is stored as the value.

Examples:

The following example evaluates the expression in the value attribute and stores the result in the variable LAST_VER. It is set to the greater of $LAST_VER or the value in the $1 variable.

```
<setenv group="Adaptor" name="LAST_VER"
   value="$LAST_VER>$1?$LAST_VER:$1" eval="true" />
```

The following example inserts a newline character (\n) before appending data to the Changes variable:

```
<setenv name="Changes" value="$4 - $1 - $6" type="temp append\n" />
```

The following example inserts a colon after the value it prepends to variable INFOPATH:

```
<setenv name="INFOPATH" value="/usr/local" type="temp prepend:" />
```

step

The <step> element is used only within the <ontempenv> element. It specifies the outcome of the special adaptor step. It is an empty element. The follow examples show the two forms of the <step> element.

```
<step result="FAIL"/>
```

```
<step result="pass"/>
```


Document # SC23-7681-08

IDE integrations

This section describes how to install and use plug-ins that allow you to access Management Console features from Integrated Development Environments (IDEs)

About IDE integrations

Developers can use integrated development environment (IDE) plug-ins to connect to a Management Console

The provided plug-ins allow developers to do the following directly from their IDE:

- View projects
- Run jobs
- Inspect job results

Other capabilities vary with each plug-in.

Each developer connects to the console with a user name that Build Forge recognizes from its users list. Access to projects is controlled by user name membership in access groups. Access to steps within projects is also controlled by access groups. A step can specify an access group explicitly. If it does not, it inherits the access group of the project.

The plug-ins do not provide the capability to edit or delete projects and steps.

Each user who accesses Build Forge through an IDE consumes a license, just as users consume a license through a browser client session.

Plug-ins are provided for the following IDEs:

- Eclipse™
- Rational® Application Developer, an IBM IDE built on Eclipse™
- Rational® Team Concert, the IBM distribution of Jazz.net

Special variables for test projects

When you run a test build of a project using a plugin, you can use some special environment variables to specify commands to run before and after files from your system are copied to the server.

All commands are run in the project directory:

- Use PRECMD variables to run a command on directories and files that are copied from the developer's machine to the server running the build. The command runs before the project step. Example: You can use this command t check out files from a source control system before they are copied.

- Use POSTCMD variables to run a command on directories and files after a project step has executed. Example: You could use this command to free a checked-out virtual directory (in a source control system that uses such a concept, such as Rational ClearCase).

You run commands on directories and files marked in a Reflector plug-in as Build Forge Project Artifacts. The commands are applied as the directory tree for the Reflector plugin is traversed.

Note

Traversal of the directory tree is breadth-first downward for PRECMD commands and reversed for POSTCMD commands. Commands for directories and commands for files are run as appropriate during traversal.

_PRISM_DIR_PRECMD	Specifies a command to be run on directories as they are encountered during tree traversal. The command is run once for every directory that contains at least one file. If the `$1` token is used in the command, it is replaced the name of the directory the command is running on. Only the first occurrence of `$1` is handled this way.
_PRISM_FILE_PRECMD	Specifies a command to be run on files as they are encountered during tree traversal. The command is run once for every file. If the `$1` token is used in the command, it is replaced the name of the file the command is running on. Only the first occurrence of `$1` is handled this way.
_PRISM_DIR_POSTCMD	Specifies a command to be run on directories as they are encountered during tree traversal. is run once for every directory that contains at least one file. The command is run once for every directory in the project that contains at least one file. If the `$1` token is used in the command, it is replaced the name of the directory the command is running on. Only the first occurrence of `$1` is handled this way.
_PRISM_FILE_POSTCMD	Specifies a command to be run on files as they are encountered during tree traversal. The command is run once for every file. The system replaces the first `$1` in the command with the file name.

Plug-ins for Eclipse and Rational Application Developer

Plug-ins provide access to Management Console features from within Eclipse™ or Rational® Application Developer IDEs.

The following plug-ins are available for the Eclipse and Rational Application Developer environments:

Available Plug-ins

Frequency The Frequency plug-in allows a developer to do the following:

- Access one or more Management Consoles to view projects
- Launch jobs
- View job status
- View build logs of running and completed jobs

Reflector	The Reflector plug-in run jobs using files in a local environment. These jobs are commonly run to test new code before checking it into source control for other developers or production builds to use.

Eclipse plug-in users have the option to override values for project environment variables. When you start a Build Forge project, the Job Settings pop-up is displayed . Changes to environment variables apply to the job only. Default variables values for the project are not changed.

Using the Plug-ins in Eclipse or Rational Application Developer

After you install the plug-ins, you can activate them in the following ways:

- To access Management Consoles to launch jobs and view project logs, use the Frequency plug-in. *Within your IDE*, select **Window → Open Perspective → Other** . Your IDE displays a dialog box with a list of perspective types; select the Build Forge perspective. The system displays the Console Explorer, Build Info, and Build Log windows. Right-click the **Console Explorer** and select **New Console** to configure a connection to a Management Console. For more information on using Frequency, see the on-line help provided with the plug-in.

  ### Note

  If you need to configure access to an LDAP/Active Directory domain, make sure you use the Build Forge system name for the LDAP domain object, not the actual name of the domain.

- To run test builds, use the Reflector plug-in. Within your IDE, configure Reflector by selecting your project and right-clicking. Select **Properties** from the pop-up menu. In the **Properties** dialog's list of properties options, select **Build Forge Project Artifacts** . Configure the dialog with the Build Forge project you want your project to work with, and select files to be uploaded to the system. For more information, see the on-line help provided with the plug-in.

  ### Note

  The Reflector plug-in requires the Frequency plug-in.

Installing the plug-ins for Eclipse or Rational Application Developer

Install the plug-ins for your IDE environment from the Build Forge server.

Prerequisites:

- Eclipse version 3.0.2 or later or Rational Application Developer version 7.0 or later

- Java 2 SE version 5.0
- The Build Forge system must be running

To install the plug-ins, perform these steps from within your IDE.

1. Select **Help** → **Software Updates** → **Find and Install** .
2. Click the **Search for new features to install** radio button, then click **Next** .

   The system displays the **Update Sites to Visit** dialog.
3. Click the **New Remote Site** button.

   The system displays the **New Remote Site** dialog.

   a. Enter “Build Forge Update Site” in the name field.

   b. Enter the following update site URL in the **URL** field, using the hostname of your Management Console machine:
   `http://<console_host_name>/prism/eclipse/updateSite/site.xml`.

      - If you are running Eclipse 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. For example:
      `http://myhostname:11812/prism/eclipse/updateSite/site.xml`

   c. Click **OK** .
4. In the **Update sites to visit** dialog, select the **Build Forge Update Site** check box, then click **Finish** .
5. The system displays a list of available plug-ins in the **Search Results** dialog. Select all the offered plug-ins, 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 **New 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 Eclipse to make the changes take effect. Click **Yes** .

Alternate Installation when SSL is enabled

Use an alternate installation method when SSL is enabled on the Build Forge system.

The current versions of Eclipse and Rational Application Developer are not enabled for SSL. Therefore the Eclipse and Rational Application Developer plug-ins 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.

To package the files, do the following:

- Create a directory named `prism`.
- In `prism`, create a directory named `eclipse`.
- In `eclipse`, create a directory named `updateSite`.

Copy the following files from `<bfinstall>/webroot/public/prism/eclipse/updateSite` to the `updateSite` directory you created:

- `features` directory
- `plugins` directory
- `site.xml` file

Once you have made the `prism` directory available, users perform these steps from within their IDE:

1. Select **Help** → **Software Updates** → **Find and Install** .
2. Click the **Search for new features to install** radio button, 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 the **New Remote Site** button. The system displays the **New Remote Site** dialog.

     b. Enter “Build Forge Update Site” in the name field.

     c. Enter the location of the files:

     `http://host/path/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 the **New Local Site** button. The system displays the **New Local Site** dialog.

  b. Enter “Build Forge 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 Update Site** check box, then click **Finish** .

5. The system displays a list of available plug-ins in the **Search Results** dialog. Select all the offered plug-ins, 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 **New 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 Eclipse to make the changes take effect. Click **Yes** .

Using plug-ins for Eclipse and Rational Application Developer

To access launch jobs and view project logs (Frequency plug-in), do the following:

1. *Within your IDE*, select **Window → Open Perspective → Other**

2. Select the **Build Forge** perspective. The perspective includes these windows:

   - Console Explorer
   - Build Info

- Build Log

3. Right-click **Console Explorer** and select **New Console** to configure a connection to a Management Console. You provide the URL.

   - If Build Forge is running without SSL, the URL is typically `http://hostname/`. However, it can be installed in other ways. If a port other than 80 is used, you must also specify the port. The URL must conform to the **Console URL** system setting. If you cannot access that setting in **Administration → System** , contact an administrator.

   - If Build Forge is running with SSL, the URL is typically `https://hostname/`. If a port other than 443 is used with SSL, you must also specify the port. The URL must conform to the **Console Url** system setting. If you cannot access that setting in **Administration → System** , contact an administrator.

When a connection is established, the Build Info window is populated with jobs that are available for you to run. To run jobs using local files, configure the job to use and what files to use.

1. *Within your IDE*, connect to Build Forge.

2. In the **Console Explorer** window, right-click a job, then select **Properties** .

3. In **Properties** , select **Build Forge Project Artifacts** .

4. In **Build Forge Project Artifacts** , select the project to work with and select the local files to use.

For more information, see the on-line help provided with the plug-ins.

Plug-in for Rational Team Concert

The plug-in for the Rational Team Concert client is one component of the integration of Rational Team Concert and Build Forge. A server extension and adaptor template are also required. When integration with Rational Team Concert 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

For instructions on setting up the integration, see the Installation Guide: **Configuring Additional Features in Management Console → Integration with Other Products → Rational Team Concert Integration** .

Using the Rational Team Concert plug-in

These instructions assume that the Rational Team Concert integration has been set up and the plug-in has been installed in your Rational Team Concert client.

For instructions on setting up the integration, see the Installation Guide: **Configuring Additional Features in Management Console → Integration with Other Products → Rational Team Concert Integration** .

To set up a build definition and run a build, do the following:

1. Set up a build definition.
    a. In the Team Artifacts view, expand the project folder.
    b. Right-click **Builds** , then click **New Build Definition** .
    c. In New Build Definition, select **Create a new build** , then click **Next** .
    d. In General Information, Enter the build ID and description. Select **Rational Build Forge** in the Available Templates list. Click **Next** .
    e. In Additional Configuration, select both General and Properties, then click **Finish** . A tab is created labeled with the Build ID you entered.
    f. Click the **Build Forge** tab.
2. Select a project for the build definition.
    a. Click the Build Forge tab. Enter the information necessary to connect to Build Forge:
        - Hostname - host name of the host where Build Forge runs. This must match the **Console URL** system setting if it is set. If you cannot access that setting in **Administration → System** , contact an administrator.
        - Port - port used to communicate with Build Forge. Port 3966 is the default. If **Connect securely to Build Forge** is selected, Port 49150 is shown by default. If the port number for your installation is different, enter it. This must match the port set in the **Console URL** system setting if it is set. If you cannot access that setting in **Administration → System** , contact an administrator.
        - User name - the user name for connecting to Build Forge. The user must already exist in Build Forge.
        - Password - password for the user name
        - Confirm password - password for the user name
    b. Click **Get Projects**
    c. Select the project for this build definition in the **Build Forge Projects** list that appears.
3. Request a build.
    a. Right-click the build definition, then select **Request build** .
    b. Specify the Build Options and Build Properties you want, then click **Submit** .

4. Check build results.
    a. After the build finishes, select it in the list in the **Builds** tab. A window is shown.
    b. Under External Links, click the **Build Forge Results** link.
    c. A login panel for Build Forge is shown. Log in.
    d. Go to **Home** → **Completed Runs** , then select your build from the list.
    e. The build steps and results are shown. Click on a step link to see the log for the step.
    f. When done, click **Logout** and close the window.

Troubleshooting the Rational Team Concert plug-in

This section describes known problems and how to work around them.

Jobs with status Overdue

If a job stays has the status Overdue for more than a few minutes, then it may be hung.

Workaround

Stop and restart the Build Forge engine. See Starting and stopping the engine.

Working with APIs

Build Forge provides a Java client API and a Perl client API.

The Build Forge client APIs are included with the Enterprise editions.

Client files are stored in `<bf-install>/webroot/public/clients/`.

You can access the files on a running Management Console. The Client download directory is at the following URL: `http://<hostname>:<portnumber>/clients/`

API access to Build Forge

Programs using the APIs communicate with the Services Layer directly. The Services Layer is an application on Apache Tomcat. During installation the Apache Tomcat server is configured to listen on particular ports. Those ports must be open in order for the APIs to communicate with Build Forge. By default the ports are set to the following:

- 3966 (non-secure)
- 49150 (secure - SSL enabled)

See Build Forge Installation Guide for more information SSL configuration. SSL configuration for API clients is in `bfclient.conf`, which must be included with the client. When SSL is enabled, the client must have a keystore and certificates in order to communicate with Build Forge.

Creating a Build Forge user for API programs

Create a user on the Management Console for programs to use to authenticate.

Create a user for API programs to use for logging into the Management Console. Log in to test the user to verify that it works.

Each time a program accesses the console, it must authenticate itself to the console as a user. Note that if it uses the same login as an existing user, that user is logged out of the system.

Note

Do not use a user provided by LDAP/Active Directory authentication. Create the user in the Management Console.

Java client API

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

Programs created using the Java client API run on a client host and access data on the Management Console. The Java client API consists of a .jar file containing classes that define Management Console objects methods that provide operations on those objects.

JDK 1.5 is required for use with the Java client API.

Document # SC23-7681-08

Documentation is provided in JavaDocs.

Note

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

Getting the Java client API package

You can download the Java client software package from your Management Console host.

To download the Java API, do the following:

1. Access the client download directory.

   In a web browser, access the following URL:

   `http://<hostname>:<portnumber>/clients/`

2. Save the JAR file.

   Under Java Client, right-click the **JAR file** link and choose **Save Link As** . Specify where to save the JAR file.

3. Save the JavaDocs.

   Under Java Client, right-click the **JavaDoc reference ZIP** link and choose **Save Link As** . Specify where to save the JAR file.

You can access the documentation through the Management Console. On the Client download directory page, under Java Client, click **JavaDoc reference** .

Setting up the the Java client API

Place the Java API on a client host and set up the JDK to use it.

The host will act as a client to the Management Console host.

1. Place the .jar file where you want it.

2. Update the CLASSPATH.

   Set the CLASSPATH to include the directory where you placed `rbf-services-client-java.jar`.

Perl client API

Use the Perl client API to write Perl programs that access the Management Console.

The Perl Client is a set of Perl modules that provide access to an abstraction of Management Console data objects and methods.

Document # SC23-7681-08

Documentation for Perl Client modules is included inside the client API package in two forms:

- A file: `apidoc.txt`
- Perl documentation in POD (Plain Old Documentation) format. For more information, see the online documentation at http://www.perl.org [http://www.perl.org/].

To use the Perl Client, you must:

- Get the Perl Client package from your Management Console machine.
- Install the package (along with Perl if it is not already installed).

Note

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

Getting the Perl client API package

You can download the Perl client API from your Management Console host.

To download the Perl client API, do the following:

1. Access the Client download directory.

   In a web browser, access the following URL:

   `http://<hostname>:<portnumber>/clients/`

2. Save the ZIP file.

   Under Perl Client, right-click the **ZIP file** link and choose **Save Link As** . Specify where to save the ZIP file.

3. Save the documentation.

   Under Perl Client, right-click the **PerlDoc reference tar.gz** link and choose **Save Link As** . Specify where to save the ZIP file. Unzip the file to access documentation for each module.

You can access the documentation through the Management Console. On the Client download directory page, under Perl Client, click **PerlDoc reference** .

Setting up the Perl client API

To use the Perl client API, you must set it up on a host where you plan to run your applications.

The host acts as a client to the Management Console host.

1. Install a Perl interpreter on the client host, such as ActiveState's ActivePerl version 5.8.4 or later.

Document # SC23-7681-08

The following Perl prerequisite modules are required (ActivePerl version 5.8.8 includes them):

- `Exporter`
- `LWP::UserAgent`
- `HTTP::Request`

See the Perl documentation for information on installing Perl modules.

2. Uncompress the downloaded Perl client API package to a temporary directory.
3. Install the Perl client API as a standard Perl distribution as described in the `apidoc.txt` file.

   On Windows you need `nmake 1.5`, which is included in Visual Studio or downloadable from the Microsoft web site. It must be installed where it can be found by the PATH environment variable, such as `C:\Windows`).

   In the temporary directory where the Perl Client package was uncompressed, run these commands:

   ```
   perl Makefile.PL
   nmake
   nmake install
   ```

   On UNIX or Linux systems (or in a Cygwin environment on Windows):

   ```
   perl Makefile.PL
   make
   make install
   ```

   Once installed, the top Perl client module is `BuildForge::Services::DBO`. See the PerlDoc for each module for more information.

Document # SC23-7681-08

Glossary

This topic provides definitions for concepts and terms used throughout the system.

access group

A collection of users who share permissions, notifications, and LDAP group properties. You can map an access group to an LDAP group. You can nest groups. Users inherit the permissions of the groups to which they belong.

adaptor

An adaptor is an add-on that allows the Build Forge system to interact with an external system, such as a source control system, debugging database, or testing system. For example, source code adaptors allow the system to monitor and track changes in source control systems such as IBM® Rational® ClearCase®, Perforce, Visual SourceSafe, or CVS, and perform actions based on those changes. You can configure an adaptor to collect information for storage in the Bill of Materials (BOM) or push information back to other information systems.

agent

A component of the Build Forge® system. It must be installed on any computer that you want to define as a server resource in the system. Each agent communicates with the Management Console and executes commands defined in a step. The agent also assembles output resulting from step execution and returns it in a step log.

archive

A list of jobs whose output files have been deleted but that still have data in the database. You view the list in the **Jobs** panel.

BOM

A list of data about a job that has completed. BOM is an acronym for Bill of Materials. When viewing the job, it is shown in the BOM tab, while data about individual step execution is shown in the Steps tab. The BOM contains information about steps in a job and changes to files that resulted from it. One common usage is with source code adaptors in software builds, where auditing changes to source files is desired. The .scan command can be used to set a baseline for changes to the source and then set checkpoints to summarize changes since the last .scan command.

class

A grouping of projects that has global properties. The properties are used to manage completed jobs, typically deleting them periodically or starting another job that performs specific cleanup tasks.

Document # SC23-7681-08

clobber

To delete a project and all of its associated jobs from the database.

collector

An object that determines what information is collected from or assigned to server resources. The information is specified through properties in the collector. The collector assigned to a server serves as a specification for the server's manifest. You define collectors in the **Administration → Collectors** panel.

database

The database stores all the information entered into the Management Console and data created by the system when it runs a project or logs user actions.

dynamic

Pertaining to events that occur at run time or during processing.

engine

A component of the system. The engine uses information entered through the Management Console and stored in the database to control project execution, send notification e-mails, and communicate with agents (running on servers).

environments

An environment is a container for a list variables. An environment can be assigned explicitly to servers, projects, and steps. The environment for a step is constructed by applying the server environment, project environment, and step environment, in that order. If a variable appears in more than one of those environments, it takes the last value specified.

handshake

The exchange of messages at the start of a Secure Sockets Layer session that allows the client to authenticate the server using public key techniques (and, optionally, for the server to authenticate the client) and then allows the client and server to cooperate in creating symmetric keys for encryption, decryption, and detection of tampering.

interceptor

A handler used by a web service to authenticate an incoming message. In Build Forge, interceptors are provided to implement single sign-on.

Document # SC23-7681-08

interface

An interface is an instance of an adaptor template. You must create an interface (and edit it) to use an adaptor. The original adaptor template remains unchanged. Note also that an interface can contain more than one <interface> element, each of which is a separately executable action.

job

An instance of a running project. The system stores data for each completed job, including step logs and BOM data.

library

A library is an executable definition of work. It is made up of steps. Its behavior is controlled through properties. It differs from a project in that it has no selector. A library is called from a step within a project.

Lightweight Directory Access Protocol

An open protocol that uses TCP/IP to provide access to directories that support an X.500 model and that does not incur the resource requirements of the more complex X.500 Directory Access Protocol (DAP). For example, LDAP can be used to locate people, organizations, and other resources in an Internet or intranet directory.

manifest

A list of data about a server that has been gathered by a collector. Manifest data is used by selectors to choose servers. Manifests for servers are updated automatically according to system settings **Active server refresh interval** and **Inactive server refresh interval**. You can also update them manually. Use the **Refresh Server Manifest** button updated while viewing the server in **Servers** → *servername*.

Management Console

A component of the system that is installed on a single machine to coordinate the system. You log in to the Management Console to start or schedule project runs and to view results and reports. The Management Console issues instructions to agents to complete jobs.

notification templates

A notification template defines the content and format of the e-mail sent to an access group on the occurrence of a specific event. The system comes with many default templates. You can edit the templates or create new ones to be specific to a project.

Document # SC23-7681-08

plug-in

A separately installable software module that adds function to an existing program, application, or interface.

project

A project is an executable definition of work. It is made up of steps. Its behavior is controlled through project properties. A project has an associated selector that determines what server (or servers) it can be run on. A project can be assigned its own environment. An executing project is a job. A project that is not assigned a selector is a library.

selector

An object associated with a project or step that selects the server where the project or step is run. Properties in the selector determine how the server is selected. Selectors can use static information. For example, a selector can specify the server name. Servers can also can dynamically use dynamic information. For example, a selector could specify a server that has designated properties, such as CPU type or disk size or current loading. At runtime, the system uses the selector to compile a list of matching servers and assign the project or step to one of those servers. You define in the **Administration → Selectors** panel. At least one selector must be defined before a project can be defined.

semaphore

A global flag in the system that prevents activities from occurring at the same time. Each semaphore is a label that the system manages. Typically projects or steps that require exclusive use of a resource are all set up to obtain a semaphore in able to use it.

You set a semaphore in a step by using the **.semget** command. It is released in a separate step by the **.semput**command. Once you obtain the semaphore, no other step can get it. Steps that attempt to obtain wait until it is released.

When a project completes, the system automatically releases any semaphores that the project used. Under some cases, for example when a job ends because of a system error, the semaphore is not released. In that case it can be manually released.

server

In Build Forge, a server is an object associated with a host. It is also called a *server resource*. A project or step runs on the host. The server to use is defined by the selector associated with the project or step.

To set up a machine to be available as server in Build Forge, you must do the following:

- Install an agent on the machine (see the *Build Forge Installation Guide* for more information).
- Create a server resource using the Management Console.

Document # SC23-7681-08

You define server resources in the **Servers** panel.

services

A component of the system, also called the services layer because it serves as an abstraction layer between clients and the database. Clients include the system itself as well as clients constructed with the provided Java API or Perl API.

snapshot

A record of backup data at a certain point in time.

static

Pertaining to an operation that occurs at a predetermined or fixed time.

step

A step is a component of a project or library. It contains one or several command lines that can be executed. The selector associated with a step is used to determine what server to use. If none is specified, the selector for the project is used. Step properties determine how the step is executed and how output is handled. You define steps when creating or editing projects or libraries.

step log

A list of data about a completed step within a completed job. When viewing the job, the step log is shown in the **Steps** tab. Information about each step is shown in columns. When you select **Jobs** → *jobname*, a list of steps is shown. Click on a step to see its step log.

threading

The process whereby various transactions undergo concurrent execution.

user

A login in the system. The system maintains its own set of users and permission settings. In a production installation LDAP is used for user management and LDAP entries and groups are mapped into the system. Users are associated with access groups, which grant them specific permissions for access to system resources.

Document # SC23-7681-08

Support

This topic describes how to get support for the Build Forge® system.

The following topics are discussed:

- Contacting IBM support
- Determining the Management Console version number

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/.

Determining the Management Console version number

You can find out what version of the Management Console you are working with by placing your mouse cursor over the logo at the top of the page. The system displays the version number in a pop-up tool tip.

Document # SC23-7681-08

Notices for IBM Rational Build Forge Online Help

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

```
IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.
```

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Document # SC23-7681-08

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

Document # SC23-7681-08

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

Document # SC23-7681-08